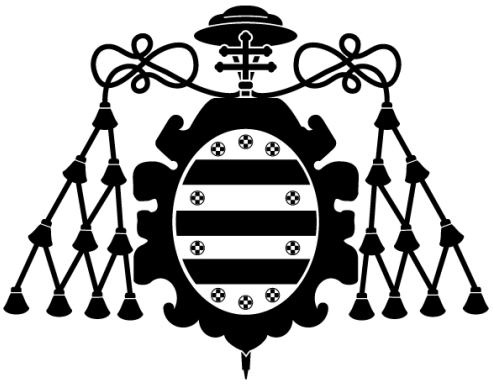


Universidad de Oviedo
Universidá d'Uviéu
University of Oviedo

Aprendizaje profundo para datos tomográficos

Lucía Fernández Álvarez [1]
Aurelio Hierro Rodriguez
Jose Jesús Fernández

Trabajo Fin de Máster

Julio 2023

[1] [redacted]@uniovi.es

Índice general

Introducción

La tomografía ha transformado de manera revolucionaria disciplinas como la medicina, la ciencia de materiales y la biología al brindar una perspectiva no invasiva y no destructiva de las estructuras internas de los objetos de estudio. Esta técnica permite obtener imágenes transversales detalladas que, una vez reconstruidas, revelan información tridimensional muy precisa del interior de objetos, ya sean tejidos humanos, materiales o células.

En el ámbito de las Ciencias Biológicas y la investigación y caracterización de materiales, resulta esencial la observación y análisis de microestructuras a escala microscópica e incluso nanométrica. En este sentido, la tomografía microscópica desempeña un papel relevante al combinar los principios de la microscopía y la tomografía para la obtención de imágenes tridimensionales de alta resolución. En concreto, el uso de la tomografía de microscopía electrónica o tomografía electrónica junto con el procesamiento de imagen y la computación tiene un potencial enorme para el análisis de la arquitectura molecular *in situ* de sistemas biológicos complejos y heterogéneos. Estas estructuras son demasiado pequeñas para ser visualizadas con técnicas convencionales de microscopía óptica. Sin embargo, gracias a la microscopía electrónica, la cual genera imágenes a través de la interacción de haces de electrones con la muestra, se logra superar las limitaciones impuestas por el índice de refracción de la luz y realizar estudios de alta precisión a escala nanométrica.

A lo largo de su historia, el desarrollo de la tomografía ha estado estrechamente relacionado con el avance de la tecnología informática y los ordenadores. Esto se debe a que la tomografía requiere la potencia de cálculo de los ordenadores para llevar a cabo los algoritmos matemáticos necesarios para reconstruir imágenes detalladas del objeto estudiado. Una vez obtenidos los tomogramas, se requieren numerosos refinamientos para mejorar la calidad de las imágenes, como la eliminación de ruido que afecta a la precisión y la claridad de la imagen. Además, otro desafío importante en el campo de la tomografía es la automatización de procesos que todavía dependen

en gran medida de la intervención manual, como la segmentación de estructuras de interés. La segmentación consiste en identificar y delimitar las regiones de interés en los tomogramas y, aunque se han desarrollado algoritmos y enfoques automatizados, aún existen desafíos para lograr una segmentación precisa y fiable.

En las últimas décadas, las redes neuronales artificiales han emergido como una poderosa herramienta en el campo de la visión por ordenador y el procesamiento de imágenes. Su capacidad para aprender y extraer características complejas de los datos ha abierto nuevas posibilidades en áreas como la segmentación tomográfica. Estos modelos neuronales pueden ser entrenados con grandes conjuntos de datos en diversas tareas, ya sea para reconocer y clasificar automáticamente estructuras de interés en los tomogramas o en la reducción de ruido y la mejora de la calidad de las imágenes tomográficas.

En este proyecto, se llevará a cabo un análisis exhaustivo del rendimiento de la arquitectura neuronal U-Net [1] como una herramienta potencial en el proceso de filtrado de ruido y segmentación mitocondrial. Los primeros dos capítulos de esta memoria se centran en proporcionar una explicación detallada sobre la microscopía electrónica y el proceso de procesamiento de datos tomográficos. En el Capítulo 3, se establecerán los fundamentos teóricos del aprendizaje profundo (*Deep Learning*) y las redes neuronales. El Capítulo 4 abordará los avances recientes y revolucionarios en el campo del aprendizaje profundo aplicado a la eliminación de ruido en el procesamiento de imágenes, adaptados específicamente al ámbito de la microscopía electrónica 3D. Posteriormente, el Capítulo 5, dedicado a la segmentación, se explicará el proceso de entrenamiento de diferentes redes neuronales utilizadas y se presentarán los resultados obtenidos. Finalmente, en el Capítulo 6, se realizará un resumen de las principales ideas y conclusiones obtenidas a lo largo del trabajo.

Capítulo 1

Obtención de imágenes y tomogramas

En este capítulo se explicarán los distintos métodos utilizados para conseguir las imágenes de estructuras biológicas que permiten generar tomogramas 3D con resolución nanométrica. Estos tomogramas son de gran utilidad en el estudio de la arquitectura subcelular, es decir, la morfología de los componentes y regiones de las células, su distribución espacial y la interacción entre ellas, lo cual está íntimamente relacionado con su funcionamiento y especialización. El estudio e identificación de anomalías en la arquitectura subcelular pueden reflejar alteraciones funcionales y proporcionar conocimiento sobre ciertas manifestaciones reveladoras de una enfermedad, contribuyendo a la búsqueda de nuevos desarrollos terapéuticos.

Las células humanas son de tamaño microscópico y muy variable. El óvulo (célula sexual femenina), por ejemplo, tiene un diámetro de unas 150 μm, mientras que el diámetro de los eritrocitos (célula sanguínea) es aproximadamente de $7,5\mu$m. En caso de querer estudiar los distintos componentes de las células, la escala disminuye aún más. Las mitocondrias, objeto de estudio en capítulos posteriores, poseen forma cilíndrica con $0{,}5\mu$m de diámetro y 1μm de longitud [2]. Por esta razón, para poder estudiar en detalle estas estructuras se requiere resolución nanométrica, inalcanzable con los microscopios ópticos habituales que poseen un límite de resolución (d_0 [1]) que ronda los 200 nm ($0,2\mu$m), tal como se deduce de la fórmula del poder de resolución (E_0)

$$E_0 = \frac{1}{d_0}\ , \qquad d_0 = \frac{1{,}22\lambda}{2 \cdot n \sin\alpha}\ , \tag{1.1}$$

[1] Límite de difracción de Abbe en un microscopio.

siendo λ longitud de onda, n el índice de refracción y α el ángulo de semiapertura.

La microscopía electrónica (EM) hace uso de un haz de electrones acelerados por un alto voltaje y focalizados por medio de lentes electromagnéticas en lugar de fotones para formar imágenes de objetos en la nanoescala, ya que estos poseen longitudes de onda mucho menores que permiten alcanzar una resolución atómica. La imagen que se forma en EM es el resultado del paso o interacción de los electrones con la materia, por lo tanto, las imágenes generadas por microscopios electrónicos son en blanco y negro al no interaccionar con la luz. Según el tipo de electrones y la interacción registrada para formar la imagen, existen dos tipos de microscopios electrónicos: TEM (*Transmission Electron Microscopy*) y SEM (*Scanning Electron Microscopy*).

Una técnica de microscopía electrónica que ha cobrado especial importancia en las últimas décadas es la tomografía electrónica (TE), posicionándose como una potente técnica para abordar cuestiones fundamentales de la biología molecular y celular [3]. En la TE, la muestra biológica, preparada adecuadamente para soportar cada técnica en específico, se visualiza con un microscopio electrónico y se toman una serie de imágenes que muestran distintas orientaciones o secciones de la misma. Posteriormente, esas imágenes se procesan y combinan para obtener la reconstrucción tridimensional o tomograma y, finalmente, se someten a varios procesos computacionales que facilitan la interpretación del tomograma, como la reducción del ruido, segmentación y análisis de subvolúmenes [3].

Los métodos más comunes para poder obtener la visualización en tres dimensiones de la arquitectura subcelular y la organización macromolecular de células en estado natural a escala nanométrica gracias a la microscopía electrónica son [4]:

- Tomografía electrónica: TEM
- Microscopia electrónica de volumen: FIB-SEM (*Focused Ion Beam Scanning Electron Microscopy*)

La tomografía electrónica proporcionada por TEM, fruto de combinar las proyecciones de una misma muestra para generar una imagen en 3D, facilita el análisis estructural tridimensional de tejidos, células y orgánulos concretos con una resolución de unos pocos nanómetros a partir de muestras de un grosor inferior a media micra. Por otro lado, la microscopia electrónica de volumen (SBF/SEM (*Serial block-face scanning electron microscopy*), FIB/SEM, cryo-FIB/SEM) permiten el análisis de células y tejidos en su entorno celular inalterado, sin restricciones de

grosor y con un nivel de detalle a nanoescala [4].

Una de las principales diferencias de las distintas técnicas empleadas en la obtención de tomogramas es la preparación de muestras, etapa clave en microscopía electrónica 2D y 3D. Las técnicas clásicas de preparación operan a temperatura ambiente y emplean fijadores químicos, los cuales alteran la ultraestructura de la muestra. En los últimos años han surgido las técnicas basadas en crio-fijación que aseguran la preservación de la ultraestructura nativa. En estas técnicas la muestra se congela en cuestión de milisegundos y en condiciones de alta presión (HPF) para evitar la formación de cristales de hielo. La muestra congelada puede visualizarse directamente en condiciones criogénicas en los microscopios preparados para ello. Una opción intermedia de preparación de muestras consiste en crio-sustituir el agua congelada por un solvente orgánico (metanol) , de forma que la ultraestructura nativa se preserva hasta un nivel de interpretación de unos 2-3 nm de resolución, y se puede trabajar a temperatura ambiente. La muestra puede visualizarse directamente en un microscopio FIB-SEM (temperatura ambiente) o crio-FIB-SEM (criogénica). La muestra también puede examinarse por tomografía electrónica (temperatura ambiente) o crio-tomografía electrónica (criogénica) mediante un microscópio TEM, siendo necesario un proceso previo de extracción de secciones de grosor adecuado que permita la transmisión de electrones (menor de 0,5 micras). Una de las técnicas más populares y establecida como estándar en la última década es la de seccionado por un haz de iones focalizado (FIB) [5]. La Figura 1.1 muestra un esquema del proceso de preparación de muestras basado en crio-fijación.

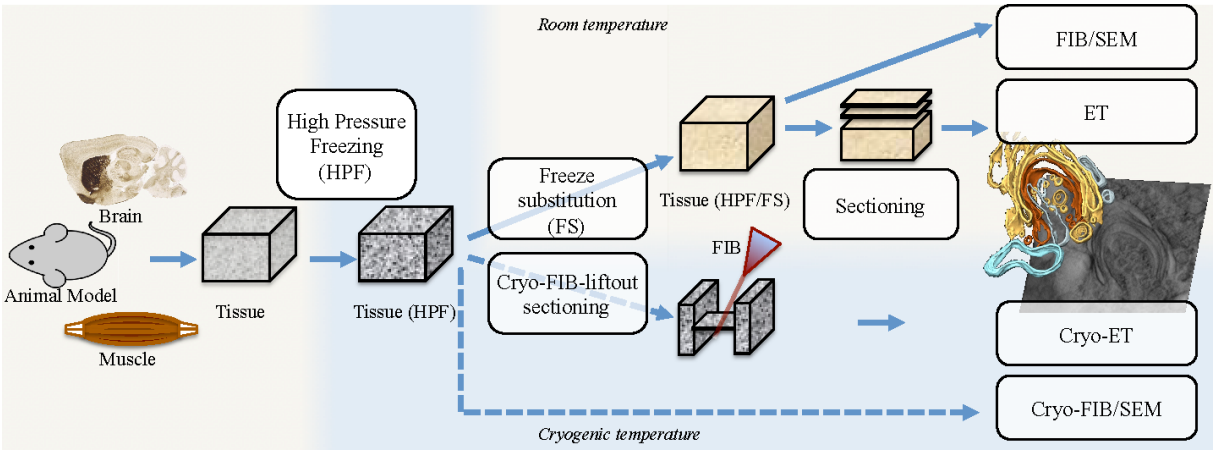


Figura 1.1: Preparación de muestras basada en crio-fijación para microscopía electrónica 3D. Imagen proporcionada por el investigador Jose Jesús Fernández.

1.1. Microscopía electrónica de volumen: FIB-SEM

Las modalidades 3DEM (*3D Electron Microscopy*) de gran volumen, por ejemplo, SBF/SEM (*Serial Block Face*), FIB/SEM o cryo-FIB/SEM, permiten el análisis de células y tejidos sin restricciones de grosor y con un nivel de resolución en torno a 5-10 nm, lo que las hace extremadamente útiles en aplicaciones biomédicas [4].

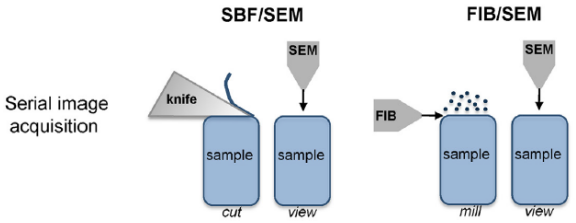


Figura 1.2: Adquisición de imágenes mediante la eliminación iterativa de una capa delgada de la superficie recién expuesta con un SEM. Imagen adaptada de [4]

Estas técnicas hacen uso del microscopio SEM, un tipo de microscopio electrónico que permite examinar muestras en volumen mediante un haz de electrones focalizado. En contraste con el TEM, el SEM no requiere que los electrones atraviesen la muestra para obtener la imagen, ya que esta se obtiene a partir de los electrones secundarios producidos en la interacción del haz de electrones con la superficie de la muestra. Una vez escaneada la superficie, se procede a la eliminación de una capa fina de la muestra que ha sido recién expuesta a los electrones repitiéndose este proceso de escaneo-eliminación de forma iterativa. Como se puede observar en la Figura 1.2, la técnica SBF/SEM utiliza una cuchilla de diamante para cortar la muestra, y el grosor de la capa eliminada suele ser del orden de decenas de nm (por ejemplo, 50-80 nm). En FIB/SEM (Focused Ion Beam/SEM) un haz de iones de galio elimina la capa superior la muestra en pasos mucho más finos que la técnica anterior, 5-10 nm [4].

1.2. Tomografía electrónica (ET)

La criotomografía electrónica es una técnica que permite la visualización y el análisis *in situ* de la organización molecular de la célula. Como cualquier otra técnica de tomografía (p.ej. el TAC en medicina) la ET se basa en la adquisición de una serie de imágenes de proyección de un único espécimen en diferentes vistas [3], por lo que, de acuerdo con esta definición, en esta técnica se empleará un microscopio electrónico de transmisión (TEM), el cual utiliza los electrones que atraviesan la muestra para crear una imagen. Este método de obtención de imágenes proporciona información estructural tridimensional completa con una resolución en el rango de 1-2 nm, y puede aplicarse a células y orgánulos manteniendo un estado lo más parecido al nativo mediante

la vitrificación [6].

De la muestra se adquieren una serie de imágenes a distintas vistas en torno a un eje de inclinación, a partir de las cuales se reconstruirá el volumen 3D mediante una serie de procedimientos descritos en el siguientes capítulo. El rango de inclinación normalmente está limitado en torno a $\pm 60^\circ$ o $\pm 70^\circ$, con un intervalo angular entre $\pm 1^\circ$ y $\pm 5^\circ$. Debido a limitaciones técnicas, no se puede cubrir completamente el rango de inclinación. Como resultado, los volúmenes 3D presentan resolución anisótropa, con un nivel peor de detalle en la dirección del haz de electrones [5]. Estas imágenes se toman de manera automatizada por ordenador, lo cual ha sido crucial para el advenimiento de la TE como técnica estructural en biología celular.

En la actualidad, la crio-tomografía electrónica (cryo-ET, o crioTE) constituye la técnica de imagen con mayor nivel de resolución que permite el estudio en 3D *in situ* de la arquitectura subcelular en condiciones óptimas de preservación estructural (a temperatura criogénica).

El TEM ofrece información valiosa sobre la estructura interna de la muestra, como la estructura cristalina y la morfología, mientras que el SEM proporciona información sobre la superficie de la muestra y su composición. El campo de visión máximo que pueden alcanzar los SEM es mucho mayor que el de los TEM, lo que significa que los usuarios de TEM sólo pueden obtener imágenes de una parte muy pequeña de su muestra. Del mismo modo, la profundidad de campo [2] de los sistemas SEM es mucho mayor que la de los sistemas TEM.

[2] Cantidad de escena que aparece enfocada en la fotografía o imagen.

Capítulo 2

Procesamiento de datos tomográficos

En este capítulo se describen las etapas y aspectos más fundamentales que tienen lugar en el procesamiento de imagen y tomogramas, tal y como se puede ver en la Figura 2.1.

Si bien la palabra tomografía hace referencia a recomponer un volumen a partir de cortes o secciones del mismo, hoy en día este término se asume, en un sentido mucho más amplio, como el conjunto de técnicas dirigidas a visualizar un objeto en tres dimensiones. En esta memoria se trabajará con imágenes tomadas con SEM y TEM, pudiendo considerar ambas técnicas fuente de datos tomográficos. No obstante, las imágenes obtenidas con SEM, dado que no se toman a distintos planos y ángulos, no requieren una reconstrucción tomográfica al uso.

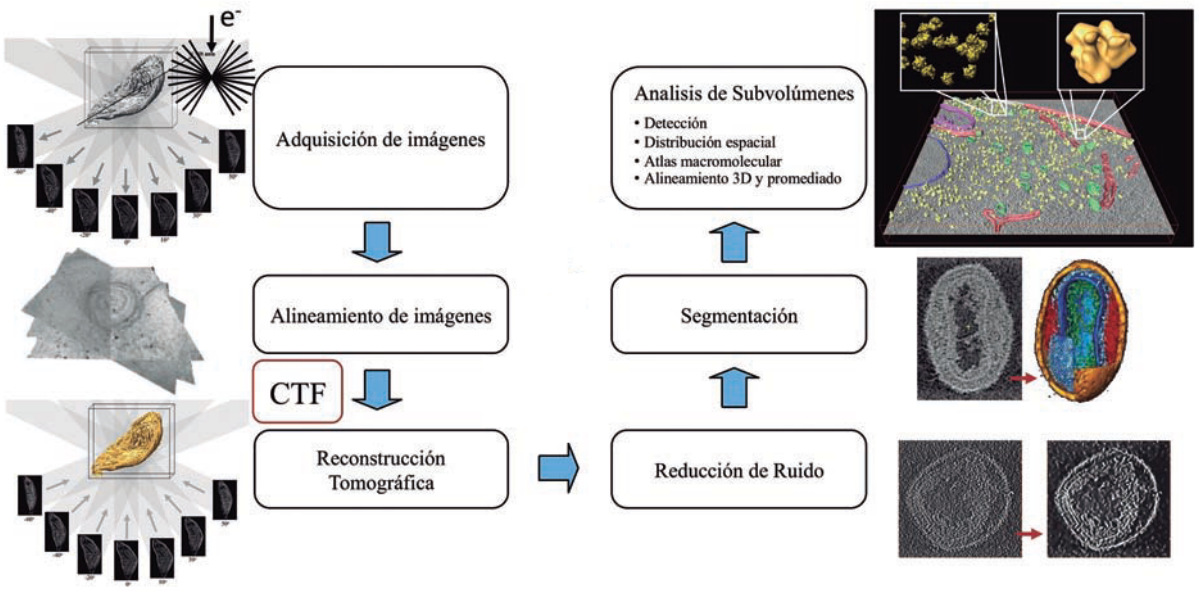


Figura 2.1: Esquema de las etapas en la adquisición y procesamiento de imagen en crioTE. Ilustración adaptada de [5].

2.1. Alineamiento de las imágenes y corrección de la función de transferencia

A pesar de que la adquisición de imágenes es un proceso automatizado por ordenador, debido a imperfecciones mecánicas, las imágenes adquiridas con el microscopio electrónico suelen presentar ligeros desplazamientos y rotaciones. Es importante alinearlas mutuamente para que se correspondan exactamente con la geometría de adquisición de datos tomográficos. Para ello, la estrategia estándar en crioTE consiste utilizar partículas de oro como marcadores fiduciales para su alineamiento. Cuando no es posible esta incubación con oro, en su lugar se intenta emplear como marcadores áreas de la imagen que presenten alguna característica de buen contraste [3]. En el caso de la tomografía FIB-SEM es más común el uso de áreas de referencia en las imágenes en el proceso de alineación.

Por otro lado, las aberraciones en un microscopio electrónico de transmisión (TEM) modifican la imagen de una muestra. Estas modificaciones son descritas matemáticamente a través la función de transferencia de contraste (FTC o CTF). Se caracterizan por la atenuación de algunos detalles en las imágenes y por un cambio de contraste en determinados rangos de resolución espacial, afectando especialmente en los rangos de alta resolución. Su corrección es esencial para poder interpretar las imágenes adecuadamente y calcular una reconstrucción 3D correcta, especialmente si el interés está en los detalles de alta resolución [5].

2.2. Reconstrucción tomográfica

En tomografía FIB-SEM no existe un proceso de reconstrucción tomográfica en sí mismo, ya que la unión del conjunto de imágenes alineadas constituye el volumen 3D o tomograma.

Sin embargo, en tomografía electrónica TEM los los datos de partida son proyecciones bidimensionales del objeto completo o parcial. El proceso de reconstrucción 3D o reconstrucción tomográfica consiste en calcular el volumen 3D o tomograma a partir de este conjunto de imágenes de proyección 2D alineadas. Existen varios métodos de reconstrucción y la selección de un método apropiado en función del problema a analizar tienen una influencia importante tanto en el ruido presente en el tomograma como en su resolución.

2.2.1. Principios de reconstrucción tomográfica

Desde un punto de vista matemático, la tomografía constituye un problema inverso de manera que tenemos que inferir el objeto a partir de mediciones u observaciones de una función del mismo [7].

En el problema de visualizar un objeto a través de un haz, ya sea de luz o de electrones, podemos considerar la radiación que empleamos para iluminar el objeto como un conjunto de rayos que se propagan en línea recta en cualquier medio que atraviesen no necesariamente homogéneo. El hecho de modelizar la radiación como rayos nos permite entender cada punto de la imagen de proyección como una integral de línea, siendo las líneas la trayectoria del rayo a través del objeto.

Las bases matemáticas sobre las que se sostienen las reconstrucciones tomográficas fueron formuladas por Johann Radon en 1917 y consisten en calcular proyecciones de objetos. Un objeto esta representado por una función $f(x, y)$ que asocia una propiedad del mismo a cada punto del plano de proyección. Una proyección paralela en la dirección (cos, sen) esta caracterizada por las integrales de línea sobre todas las rectas perpendiculares a la recta que pasa por el origen de coordenadas y de dirección (cos, sen), con φ fijado, que marca la dirección de los rayos. La intensidad sobre de cada punto de la recta es a lo que vamos a llamar proyección de la función $f(x, y)$, Figura 2.2 a).

La transformada de Radon consiste en la integral de una función sobre el conjunto de rectas en todos los ángulos de proyección y recibe el nombre de senograma o sinograma, puesto que tiene como respuesta característica un seno. En consecuencia, su representación gráfica parece una colección de senos con diferentes fases y amplitudes, tal y como se puede ver en la Figura 2.2.

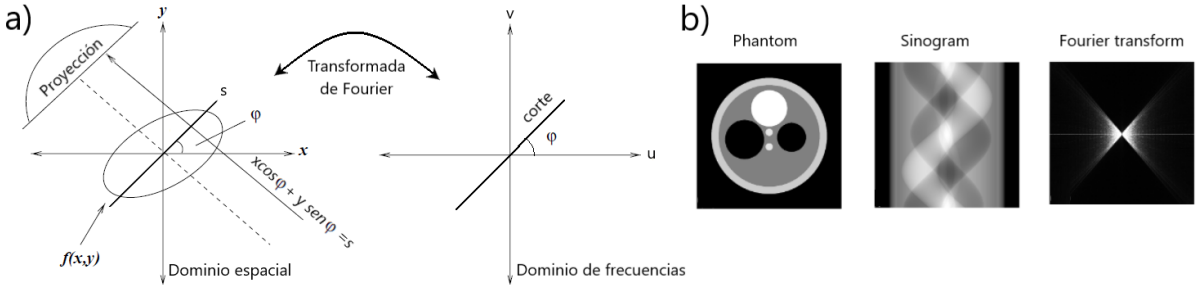


Figura 2.2: a) Integral de línea correspondiente a la proyección y su transformada de Fourier en el dominio de frecuencias b)Phantom sintético modificado de Shepp-Logan [8] [9], su correspondiente transformada de Radón (senograma) y la transformada de Fourier del sinograma.

Una vez se obtienen las proyecciones, se pasa a la reconstrucción del objeto por transformada de Fourier (FT) inversa. Para este propósito repasaremos el teorema de corte de Fourier (o teorema de sección central): La transformada de Fourier $F(u, v)$ de una proyección paralela de una imagen $f(x, y)$ tomada a un ángulo φ nos da un corte de la transformada bidimensional del objeto original. Ahora es posible plantear un sistema de ecuaciones que determine los coeficientes de Fourier de la función objeto en base a los coeficientes de las funciones proyección conocidos. La utilidad del teorema radica en que tomando las proyecciones de un objeto a distintos ángulos y tomando la transformada de Fourier de las mismas, podemos determinar los valores de la transformada bidimensional $F(u, v)$ en líneas radiales del plano uv (dominio de frecuencias espaciales). Si pudiéramos tomar un número infinito de proyecciones, entonces conoceríamos $F(u, v)$ en todo el plano y tomando su transformada inversa encontraríamos la imagen $f(x, y)$ del objeto. Por tanto, la imagen de un objeto está precisa e inequívocamente determinada por el conjunto infinito de todas sus proyecciones[10]. Sin embargo, en los experimentos reales no contamos con un conjunto de proyecciones continuas, por lo que una fórmula matemática idealizada no será aplicable y se tendrá que desarrollar un algoritmo eficiente para llevar a cabo la reconstrucción.

El concepto de proyección para el caso de objeto 3D a su imagen de proyección 2D, se entiende como una sucesión de secciones 2D a las que aplicar la transformada de Radon. Para obtener la estructura 3D de la muestra a partir del conjunto de imágenes alineadas de la serie de inclinación, se siguen los mismos principios matemáticos de la reconstrucción tomográfica basados en el teorema de la sección central, según el cual la transformada de Fourier de una proyección 2D de un objeto 3D es una sección central de la FT 3D del objeto, pudiendo calcular la FT 3D ensamblando las FT 2D de las imágenes de la serie de inclinación [3].

La imposibilidad práctica de obtener funciones continuas de los objetos ha impulsado la búsqueda de nuevos métodos en las ultimas décadas que permitan obtener un modelo del objeto a partir de datos reales incompletos, con ruido y con los artefactos propios de la adquisición de datos experimental. Los métodos de reconstrucción más establecidos son los denominados métodos de Fourier, métodos de retroproyección y métodos algebraicos o iterativos.

Los métodos de Fourier están irremediablemente limitados por la interpolación no trivial en el espacio de Fourier, debido a que la forma de tomar los planos deja inexplorado una parte del espacio recíproco (dejando espacios en forma de cuña entre cada vista), perdiendo la información que ese volumen del espacio de frecuencias contuviera [11]. A demás, debido a que no se puede

cubrir completamente el rango de inclinación, generalmente ±70°, se forman dos cuñas, con una arista común si disponemos de un solo ángulo de giro, sobre las cuales tampoco tenemos información en el espacio real.

La retroproyección ponderada (WBP) es un método estándar de reconstrucción tomográfica. Es equivalente al enfoque de Fourier que acabamos de describir, pero trabajando en el espacio real. WBP asume que las imágenes de proyección representan la cantidad de densidad de masa encontrada por los rayos de imagen. El método simplemente distribuye esa masa del espécimen uniformemente sobre los rayos de retroproyección computados. Cuando este proceso se repite para todas las imágenes de proyección de la serie de inclinación, los rayos de retroproyección de las diferentes imágenes se cruzan y se refuerzan entre sí en los puntos donde se encuentra la masa en la estructura original. Recibe el nombre de retroproyección ponderada debido al filtro paso alto aplicado previamente a las imágenes de proyección para compensar la función de transferencia del proceso de retroproyección (o bien, aunque requiera un mayor costo computacional, aplicado posteriormente a la reconstrucción). Esta ponderación es necesaria para representar adecuadamente la información de alta frecuencia en la reconstrucción. La relevancia de WBP en TE se debe principalmente a su simplicidad computacional. Su desventaja es la sensibilidad a las condiciones encontradas en TE, el ángulo de inclinación limitado y el ruido [11].

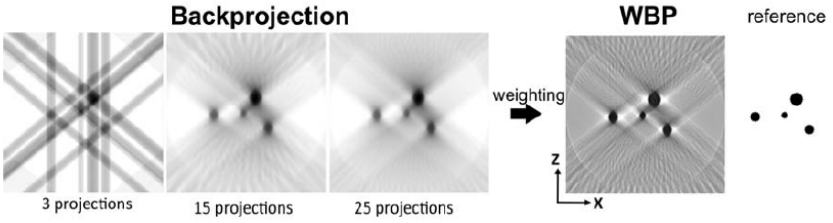


Figura 2.3: Reconstrucción tomográfica con WBP en 2D. Para compensar el emborronamiento causado por el proceso de retroproyección, se requiere una ponderación. El emborronamiento y el alargamiento causados por la cuña que falta son observables en la reconstrucción cuando se compara con los datos de referencia (derecha).

Existen algoritmos alternativos de reconstrucción en espacio real que formulan el problema de reconstrucción 3D como un gran sistema de ecuaciones lineales que deben resolverse mediante métodos iterativos. Son robustos para afrontar las particularidades del rango de inclinación limitado y el ruido en TE. Estos métodos refinan el volumen progresivamente minimizando el error entre las imágenes de proyección experimentales y las proyecciones equivalentes calculadas a partir del volumen reconstruido. Existen muchos algoritmos iterativos que comparten la misma

idea común, pero difieren en la forma de llevar a cabo esta minimización. Un método iterativo muy aceptado en el campo de la TE es SIRT, Técnica de Reconstrucción Iterativa Simultánea, muy utilizado en la ciencia de materiales [11].

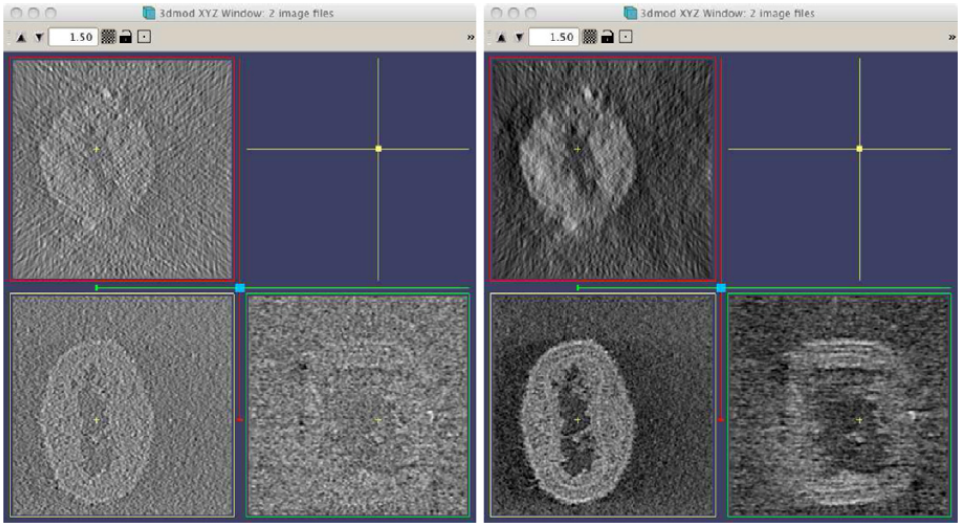


Figura 2.4: Reconstrucción tridimensional del virus Vaccinia preparada y visualizada bajo condiciones cryoET con WBP (izquierda) y 30 iteraciones de SIRT (derecha). Los planos XZ, XY y ZY se muestran en los paneles superior izquierdo, inferior izquierdo e inferior derecho, respectivamente. Los planos que contienen el eje Z muestran claramente el efecto de la cuña que falta, en forma de emborronamiento severo, alargamiento y desvanecimiento de las características [3]. La mejora en el contraste y la definición de características y bordes en SIRT son evidentes.

2.3. Filtrado de ruido: Conceptos básicos del filtrado espacial

Los tomogramas biológicos suelen tener una baja relación señal-ruido debido a la baja dosis electrónica aplicada a los especímenes y al poco tiempo de exposición al haz de electrones. En especial, en la tomografía electrónica generada por TEM, se debe gestionar el daño que la irradiación electrónica inflige a la muestra, dividiendo la dosis electrónica acumulada total uniformemente entre las distintas imágenes adquiridas. En las fases iniciales de la adquisición se toman las imágenes a menor inclinación, que son las que mejor preservan la información estructural porque la muestra ha sufrido menor daño electrónico acumulado [5]. Para reducir el ruido del tomograma y facilitar etapas posteriores de procesado tratando de no afectar a la señal, se emplean filtros que consigan reducir el ruido preservando los detalles estructurales. Cada técnica de mejora de imagen se enfoca principalmente en dos problemas:

- Eliminación del ruido introducido en el proceso de captura.

- Aumento del contraste.

El concepto de filtrado tiene origen en el uso de la transformada de Fourier para el tratamiento de señales en el denominado dominio de la frecuencia [12] (para más información ver el apéndice A). No obstante, una imagen se puede filtrar tanto en el dominio del espacio, trabajando directamente sobre los píxeles de la imagen, como en el dominio de la frecuencia, donde las operaciones se llevan a cabo en la transformada de Fourier de la imagen. En este trabajo, se prestará especial atención a las operaciones de filtrado que se realizan directamente sobre los píxeles de una imagen, puesto que será el dominio en el que trabajarán las redes neuronales utilizadas en capítulos posteriores.

2.3.1. ¿Qué es un filtro o máscara?

Una de las formas de mejorar la calidad de las imágenes reduciendo el ruido de las mismas y resaltando sus detalles es mediante máscaras o *kernels*. Básicamente una máscara es una matriz bidimensional (o tridimensional en caso de tratar volúmenes) pequeña (por ejemplo 3×3) cuyo valor de los elementos o pesos son escogidos para detectar una propiedad de la imagen. A través de los *kernels*, las operaciones espaciales de filtrado utilizan la información contenida en la vecindad del punto a tratar para realizar su transformación.

Aunque se pueden formar máscaras de cualquier tamaño y forma, normalmente es preferible que la máscara posea un tamaño pequeño con un píxel central específico. Esta máscara se desplaza sobre la imagen de forma que el centro de la máscara atraviese todos los píxeles y cambie el valor de gris en cada píxel situado en (x, y) en función de los valores de los píxeles vecinos centrados en (x, y).

Cabe destacar que, debido a que los píxeles originales contribuyen a más de un píxel resultante, la imagen original no puede modificarse antes de que se completen todas las operaciones. El cálculo directo en la propia imagen no suele ser posible para un filtro y, por tanto, se necesita espacio de almacenamiento adicional para la imagen resultante, que posteriormente podría copiarse de nuevo en la imagen de origen si se desea.

Generalmente, los filtros aplicados son lineales. Estos combinan los valores de los píxeles de la región cubierta por el tamaño del *kernel* de forma lineal, es decir, todos los vóxeles [1] del tomograma se someten al mismo *kernel*. Esto se puede llevar a cabo aplicando la operación de

[1] El voxel (del inglés volumetric pixel) es la unidad cúbica que compone un objeto tridimensional.

convolución que explicaremos a continuación. No obstante, dependiendo del resultado final que se desee obtener, el uso exclusivo de filtros lineales acarrea ciertas limitaciones que pueden ser sorteadas haciendo uso de filtros no lineales o anisótropos.

Algunos de los parámetros más importantes a tener en cuenta en el uso de los filtros son el tamaño, la forma y los pesos del *kernel*. El **tamaño** de la región de filtrado especifica cuántos píxeles originales contribuyen a cada valor de píxel resultante y, por tanto, determina la extensión espacial del filtro. La **forma** de la región de filtrado no es necesariamente cuadrada o rectangular. En ocasiones, es preferible una región circular para obtener un efecto de desenfoque isótropo, es decir, igual en todas las direcciones de la imagen. Otra opción es asignar diferentes **pesos** a los píxeles de la región de soporte, por ejemplo, para dar más énfasis a los píxeles que están más cerca del centro de la región. Además, la región de soporte de un filtro no tiene por qué ser contigua y puede incluso no contener el propio píxel original [13].

2.3.2. Operación de convolución

La operación asociada a un filtro lineal se denomina **convolución lineal** [13]. La operación de convolución es un producto interno que generaliza el producto escalar y combina dos funciones de la misma dimensionalidad, ya sean continuas o discretas. Se transforman las dos funciones (f y g) produciendo una tercera función que representa la magnitud en que se superpone la función f a medida que se desplaza sobre la función g. Para el tratamiento de imágenes se debe generalizar para un caso multidimensional. Para funciones 2D discretas I y H, la operación de convolución se define como

$$I'(u,v) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} I(u+i, v+j) \cdot H(i,j) = I * H \ , \tag{2.1}$$

siendo $*$ el operador convolución. El primer término de la convolución, I, se corresponde con la imagen que queremos procesar, mientras que el segundo término, H, es el *kernel* con el que se procesa. Algunas de las propiedades de la convolución lineal son: conmutatividad, linealidad y asociatividad [12].

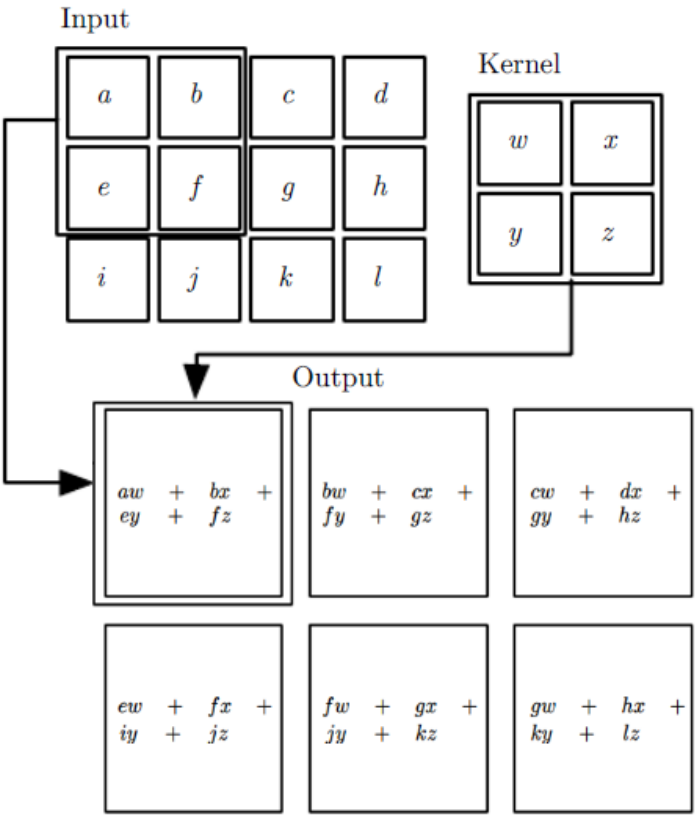


Figura 2.5: Operación de convolución con *kernel* 2×2 sobre una matriz 3×4 que da como resultado de la convolución una matriz 2×3. Imagen obtenida de [14].

Como podemos ver en la Figura 2.5, en general, si el filtro que se aplica sobre la imagen tiene un tamaño $m_1 \times m_2$ y la dimensión de la imagen es $n_1 \times n_2$, el tamaño del *output* será $(n_1 - m_1 + 1) \times (n_2 - m_2 + 1)$, dando lugar a una matriz resultante es más pequeña que la original. En caso de querer obtener una imagen de las mismas dimensiones podemos rodear la imagen por ceros, tantos como nos hagan falta. Este procedimiento se explicará con más detalle en la Sección 3.6.3 bajo el nombre de *padding*.

2.3.3. Suavizado de imágenes

Los filtros de suavizado se utilizan para desenfocar (generalmente en etapas de preprocesamiento para extraer objetos de tamaño relevante) y reducir el ruido [12].

La idea detrás de los filtros de suavizado es relativamente intuitiva: sustituir el valor de cada píxel de una imagen por la media de los niveles de gris de los píxeles recogidos por la máscara. Dado que el ruido aleatorio suele consistir en transiciones bruscas de los niveles de gris, las

técnicas lineales basadas en promedios locales utilizando *kernels* uniformes o de tipo gaussiano[2] consiguen reducir el ruido a costa de difuminar los bordes y las características [12].

Ante la problemática del emborronamiento, surgieron otro tipo de filtros más sofisticados que permitían preservar de una mejor manera las fronteras.

- No-lineales: No se puede implementar a través de la operación convolución. Las técnicas no lineales consiguen no difuminar el borde ni las características ajustando la intensidad del filtrado (es decir, los parámetros/pesos del núcleo) al detalle estructural subyacente. De este modo, el filtrado es fuerte en las zonas homogéneas, mientras que se atenúa o anula en los píxeles con un gradiente elevado, lo que indica que hay un borde o un detalle potencialmente importante. El efecto no deseado de esta estrategia es que los bordes pueden seguir siendo algo ruidosos. El ejemplo más conocido de esta categoría es el filtro de mediana. Sustituye el valor de un píxel por la mediana de los niveles de gris en su vecindad [12].
- Anisótropos: Estos métodos no sólo ajustan la intensidad, sino también la dirección del filtrado, evitando el promediado en la dirección perpendicular a la frontera. El resultado es que los bordes se someten a un proceso de filtrado que discurre paralelo a ellos, o al menos no los atraviesa, con lo que se limpian y realzan. El método más usado en la diferenciación de imágenes es el gradiente. Un método muy conocido es Anisotropic non-linear diffusion ,AND, un procedimiento basado en ecuaciones diferenciales parciales de evolución no lineal que trata de mejorar la calidad de las imágenes eliminando el ruido, conservando los detalles e incluso realzando los bordes. AND consigue preservar y mejorar las características, ya que la fuerza y la dirección del filtrado se ajustan de forma adaptativa a la estructura local alrededor de cada vóxel.

Figura 2.6 se muestran ejemplos ilustrativos de los principales métodos de filtrado de ruido sobre el phantom sintético.

2.3.4. Realce del contraste de las imágenes

Las técnicas de contraste son útiles principalmente para resaltar los bordes en una imagen. En la sección anterior vimos cómo el promedio de puntos en una vecindad tiende a suavizar o

[2]Esencialmente sustituyen cualquier píxel por una media ponderada de sus píxeles vecinos, con pesos decrecientes para los que se encuentran a distancias mayores.

difuminar (emborronar) los detalles de una imagen. Este proceso de promediar es análogo a la integración, por lo que sería natural esperar que la diferenciación tiene el efecto opuesto y por lo tanto constrastará la imagen. Este tipo de filtros son lineales, calculan el gradiente de la misma manera en toda la imagen.

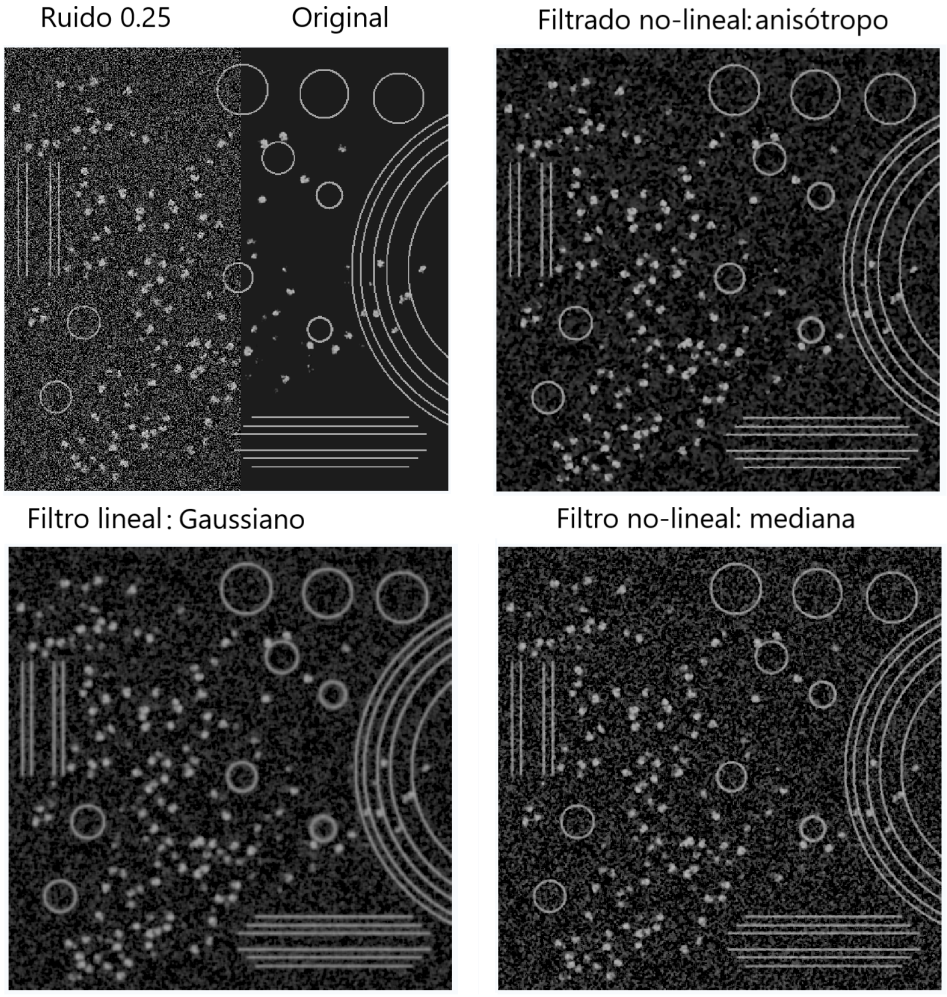


Figura 2.6: Ejemplos de filtrado clásico espacial para un *phantom* sintético con ruido Gaussiano con SNR 0,25 (Ec. 4.8). Se muestran el filtrado Gaussiano, mediana y anisótropo AND. En el filtrado lineal Gaussiano se puede apreciar la difuminación de las fronteras y detalles, mientras que en los otros dos métodos esto no ocurre. En el filtrado no-lineal aún hay ruido en el entorno de las fronteras. En el filtrado anisótropo las fronteras quedan especialmente resaltadas.

2.4. Visualización e interpretación de los tomogramas: Segmentación y análisis de subvolúmenes

La segmentación consiste en extraer los distintos componentes estructurales del tomograma (p.ej. orgánulos celulares) mediante la identificación y agrupamiento de los vóxels que pertenecen a ellos [5]. Debido al bajo contraste y la complejidad de los entornos celulares, las herramientas clásicas de segmentación no han dado buenos resultados por sí mismas, teniendo que combinarse con segmentación manual. Existen técnicas automáticas de reconocimiento de patrones basadas en correlación, normalmente a partir de un volumen de referencia. También se pueden aplicar técnicas de clasificación para identificar distintos tipos de complejos o sus distintas conformaciones. Una vez que los complejos se han detectado, pueden extraerse, alinearse y promediarse en 3D, para así obtener una visión de mayor calidad y mayor resolución del complejo macromolecular en cuestión.

Recientemente, las técnicas basadas en Inteligencia Artificial se han empezado a aplicar con éxito. El sistema es entrenado para reconocer determinadas características estructurales (p.ej. membranas de distintos orgánulos, microtúbulos, ribosomas) a partir de un conjunto relativamente pequeño de datos. Una vez entrenado, el sistema es capaz de segmentar tomogramas complejos directamente y de forma automática [5]. Este tema será abordado en el Capítulo 5, donde se mostrará el resultado del entrenamiento de distintas redes para la segmentación automática de mitocondrias.

Capítulo 3

Deep-Learning

En este capítulo se asentarán las bases teóricas generales sobre las redes neuronales artificiales y el aprendizaje profundo o *Deep Learning*, con el fin de poder facilitar la interpretación de los resultados obtenidos por las redes neuronales utilizadas en capítulos posteriores. Se comenzará explicando el concepto de neurona y cómo entrenar modelos multicapa, abordando los principales problemas que estos presentan y las estrategias más populares para solucionarlos. Finalmente se explicarán las redes neuronales convolucionales (CNN) y Autoencoders convolucionales, presentando la arquitectura específica de la U-net.

3.1. Inteligencia Artificial

Antes de adentrarnos en conceptos y técnicas específicas de *Deep Learning* (DL) y redes neuronales, vamos a comenzar tratando un concepto más general y que nos resulta cada vez más familiar; la inteligencia artificial (IA). Hoy en día la inteligencia artificial está prácticamente en toda la tecnología de la que hacemos uso, desde asistentes de voz, la conducción de coches automáticos o páginas de traducción. IA se define como la disciplina científica que se ocupa de crear programas informáticos que ejecutan operaciones comparables a las que realiza la mente humana, como el aprendizaje o el razonamiento lógico, es decir, es la capacidad de las máquinas de mostrar habilidades y automatizar tareas intelectuales que normalmente realizarían los humanos [15]. Las técnicas o estrategias que se utilizan para mejorar estas capacidades de aprendizaje es lo que se conoce como *Machine Learning*, o aprendizaje automático, y proporcional a la capacidad a los ordenadores de aprender sin ser explícitamente programados a través de

un algoritmo de predicción para cada caso de uso particular. El objetivo de estos algoritmos es aprender patrones o tendencias a partir de datos de entrenamiento y utilizar este conocimiento para poder clasificar o predecir elementos nuevos. Cambia el paradigma de la programación con a la que estamos familiarizados, siendo la máquina la encargada de examinar los datos de entrada y las respuestas correspondientes, y deducir cuáles deben ser las reglas, por lo que los sistemas de aprendizaje automático no se programan explícitamente, sino que se entrenan [16]. Aunque existen numerosas estrategias de predicción en estos algoritmos y no siempre resulta sencillo clasificarlas en categorías bien definidas, en general se pueden agrupar en tres categorías [15]:

- Supervisado: Los datos de entrenamiento están etiquetados con la solución deseada. El modelo o función es aprendido mapeando una entrada a una salida asociada a una etiqueta o *label*. Esta predicción se compara con la respuesta correcta que el algoritmo conoce e iterativamente irá mejorando el modelo comparando sucesivamente.
  - Regresión: En los modelos de regresión, el resultado es continuo. Ejemplos: *Linear Regression*, *Decision Tree* y *Random Forest*.
  - Clasificación: En los modelos de clasificación, la salida es discreta. Algunos de los tipos más comunes de modelos de clasificación son : Regresión logística (similar a la regresión lineal, pero se utiliza para modelizar la probabilidad de un número finito de resultados) y *Support Vector Machine*
- No supervisado: Los datos de entrenamiento no incluyen la etiqueta y el algoritmo deberá clasificar la información por si solo, buscando características y patrones intrínsecos de los propios datos que se parezcan y se puedan agrupar, sin partir con un propósito particular:
  - *Clustering (Agrupamiento o Clusterización)*
  - *Association rules (Reducción dimensional)*
- Aprendizaje por refuerzo: Determinará acciones por prueba y error hasta alcanzar la mejor manera de completar una tarea asignada; aprenderá por si solo gracias a las recompensas y penalizaciones que obtiene de sus acciones (ejem. obtener puntuaciones altas en un juego).

Cabe destacar que, tanto en el aprendizaje por refuerzo como no supervisado, aunque no se proporcionen las soluciones explícitamente al programa, este está mediado por un humano

(asentando las recompensas en el caso del refuerzo o los objetivos en el no supervisado) y por tanto se definen los límites del aprendizaje en ambos casos.

Un caso especial de los algoritmos de *Machine learning* son las *redes neuronales artificiales* (RNA), que permiten aprender la representación de características en múltiples niveles de abstracción sin muchas dependencias de funciones creadas por humanos, realizando un aprendizaje de propósito general. El uso de estas unidades de cálculo, llamadas neuronas artificiales, estructuradas en múltiples capas es lo que actualmente se conoce como *Deep Learning*. Su desarrollo estuvo motivado por la incapacidad de los algoritmos tradicionales para generalizar ciertas tareas de interés como el reconocimiento del habla o el reconocimiento de objetos en imágenes, al igual que por el crecimiento exponencial de dificultad y costes computacionales cuando se trabaja con datos de alta dimensión.

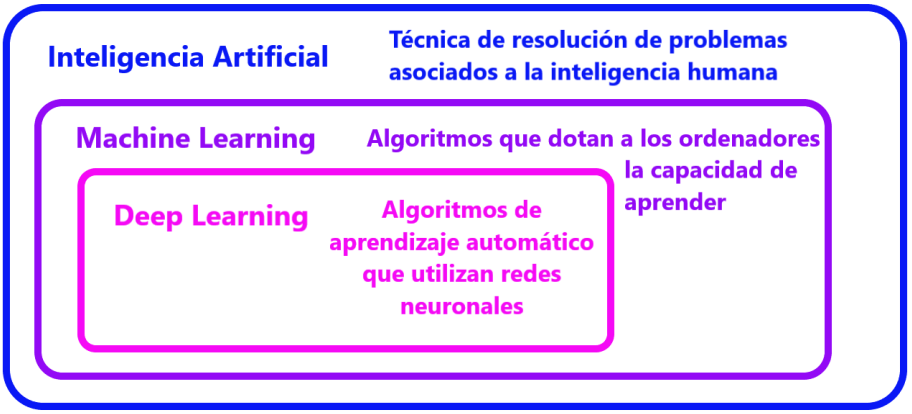


Figura 3.1: Esquema de la relación entre inteligencia artificial, aprendizaje automático y aprendizaje profundo.

Aunque las redes neuronales multicapa son una herramienta muy buena para abordar este tipo de datos más complejos gracias a su capacidad de abstracción, precisamente este nivel de abstracción y falta de especificidad de los algoritmos es lo que a menudo hace que los resultados del modelo o el modelo en sí sea difícil de explicar.

3.2. Redes neuronales artificiales

Como mencionamos anteriormente, en los algoritmos Deep Learning la información se procesa en capas jerárquicas para comprender representaciones y características de datos en niveles crecientes de complejidad. Todos estos algoritmos poseen como unidad de cálculo neuronas arti-

ficiales, encargadas de enseñar a los ordenadores a procesar datos de manera similar al trabajo que realizan las redes neuronales biológicas en el cerebro humano. Estos algoritmos se diferencian entre sí fundamentalmente en su arquitectura, estructura en la que se organizan las capas y neuronas en la red, o en la forma en la que se entrenan. Las neuronas artificiales están conectadas entre sí, permitiendo la transmisión de información sometida a diferentes operaciones al atravesar la red neuronal, dando como resultado final unos valores de salida.

3.2.1. El perceptrón

La teoría de las redes neuronales da su primer gran salto con el desarrollo del perceptrón, la forma más simple que existe de red neuronal capaz de poner en práctica la teoría del reconocimiento de patrones, es decir, extraer información que permite establecer propiedades o reconocer patrones de señales.

Fue inventada en 1957 por Frank Rosenblatt [17], el cual basó sus estudios en el primer modelo matemático de la actividad de una neurona aislada ideado por McCulloch y Pitts en 1943 llamada unidad lógica umbral (TLU *Threshold Logic Unit*), cuyas entradas y salidas eran binarias y sus pesos estaban fijos [18]. En la figura 3.2 podemos observar la idea conceptual detrás una neurona artificial, la cual posee una o varias entradas asociadas a un peso y una salida que puede pasar de estar apagada a encendida en función del valor de las entradas gracias a una función de activación.

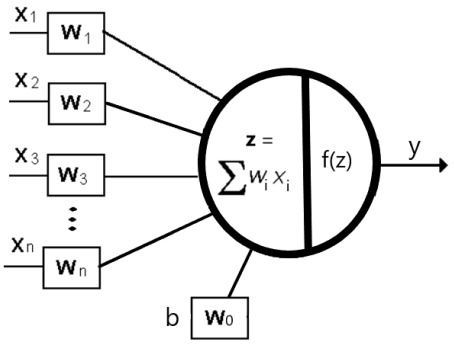


Figura 3.2: Esquema conceptual de una neurona artificial.

Dependiendo de la relación que existan entre los datos y los pesos se pueden distinguir distintos tipos de modelos neuronales[1]. No obstante, el modelo más típico y con el que trabajaremos a lo largo del trabajo es el modelo aditivo. Este expresa la relación lineal entre datos de entrada y salida para una muestra,

$$y = f\big(\sum_{i=1}^{p} w_i x_i + b\big)\ , \tag{3.1}$$

donde y es la etiqueta/salida, x representa las características/datos de entrada, b es el sesgo o

[1] Como puede ser el modelo de nodos multiplicativos [19]

bias, w son los distintos pesos y p son el número de entradas total. Podemos expresar la ecuación anterior en forma matricial haciendo uso de la notación $x_0 = 1$ y $b = w_0$, $z = W^T \cdot X$,

$$y = f\left(W^T \cdot X\right) . \tag{3.2}$$

La función de activación o umbral, f, aplicada a la suma ponderada se encarga de modificar el valor de salida o imponer un límite que se debe sobrepasar para dar una determinada respuesta y poder proseguir a otra neurona. Es la encargada de transmitir la información generada por la combinación lineal de los pesos y las entradas, regulando el flujo de información por las conexiones de salida [16] y otorgando a los modelos la capacidad de resolver problemas no lineales. Como se puede ver en la Tabla 3.1, algunas de las funciones de activación tienen la propiedad de converger asintóticamente a valores de salida constantes para argumentos muy grandes y muy pequeños, pero entre medias poseen una región no lineal, aunque también existe la posibilidad de transmitir la información sin modificaciones mediante la función identidad. En secciones posteriores explicaremos las ventajas e inconvenientes de cada una de ellas y sus usos.

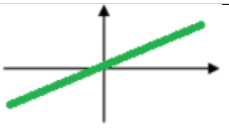
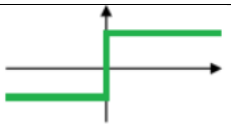
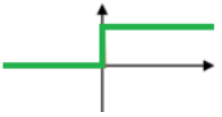
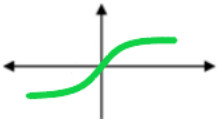
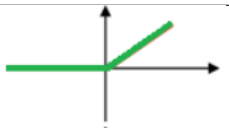
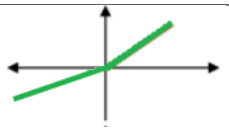
Nombre		Función $f(z)$	Rango	Gráfica
Lineal		$\alpha \cdot z$	$[-\infty, \infty]$	
Escalón	Signo	$\begin{cases} -1 & z \leq 0 \\ 1 & z \geq 0 \end{cases}$	$[-1, 1]$	
	Escalón	$\begin{cases} 0 & z \leq 0 \\ 1 & z \geq 0 \end{cases}$	$[0, 1]$	
Sigmoidales	Logística	$\frac{1}{1+e^{-z}}$	$[0, 1]$	
	Tangente hiperbólica	$\frac{e^z-e^{-z}}{e^z+e^{-z}}$	$[-1, 1]$	
	Softmax	$\frac{e^{z_j}}{\sum_{k=1}^{K} e^z}$	$[0, 1]$	Convertir las salidas en probabilidades que en total suman uno.
ReLU		$max(0, z)$	$[0, \infty]$	
Leaky ReLU		$max(\alpha \cdot z, z)$	$[-\infty, \infty]$	

Tabla 3.1: Ejemplos de funciones de activación típicas para las neuronas artificiales.

Un único perceptrón puede clasificar categorías en un espacio dimensional definido por las entradas y sus pesos asociados, generando un hiperplano ($\sum_{i=1}^{p} w_i x_i + \theta = 0$). Por lo tanto, se puede considerar al perceptrón un clasificador lineal, ya que los patrones deben ser linealmente separables, de manera que se encuentren a ambos lados de un hiperplano. El proceso de entrenamiento de este clasificador lineal consiste en ajustar los pesos y sesgos para conseguir la salida esperada. Al entrenar los perceptrones usando conjuntos de entrenamiento linealmente separables, se obtiene una solución en un número finito de iteraciones, consiguiendo generar un criterio para seleccionar un subgrupo a partir de los patrones del conjunto de entrenamiento. Para automatizar la búsqueda de los pesos correctos partiendo de valores aleatorios, necesitamos

una función de error que permita ir ajustando estos parámetros con un algoritmo que posteriormente llamaremos retropropagación o *backpropagation*. Con la función de error podemos crear una regla que cambie los pesos poco a poco multiplicando el error por la entrada con el peso y un número llamado factor de aprendizaje, generalmente pequeño, que controla el ajuste aplicado al vector de pesos en el iteración n. El proceso llevado a cabo en la n-ésima iteración para ajustar los pesos utilizando como función de error la resta entre el valor predicho y real sería

$$w(n+1) = w(n) + \eta\big[(\hat{y}(n) - y(n)\big]x(n) = w(n) + \Delta w(n) \ , \tag{3.3}$$

donde η es el factor de aprendizaje, $y(n)$ la salida deseada, $\hat{y}(n)$ la predicha por la red y $x(n)$ el dato de entrada de la neurona.

Como podemos imaginar, el modelo con una única capa presenta múltiples limitaciones, entre las que se encuentra la imposibilidad de resolver problemas que no puedan dividirse en hiperplanos. Un ejemplo sencillo es la implementación de la puerta XOR, donde se necesitan dos hiperplanos para diferenciar las salidas [14]. Ante esta limitación es necesario aumentar la capacidad del perceptrón, lo cual es posible gracias a la adicción y concatenación de neuronas, donde la salida de una neurona será la entrada de la siguiente, dando lugar a una estructura en capas [15].

3.2.2. Redes neuronales multicapa

La arquitectura de una RNA es la estructura que tienen las distintas capas de la red y cómo se conectan entre ellas. Los distintos algoritmos se diferenciaran en dicha arquitectura y en ocasiones la forma en la que se entrenan, aunque todos parten de la base común donde las neuronas están interconectadas y organizadas en capas. Una red multicapa tiene una estructura básica con al menos tres capas:

- Capa de entrada: formada por aquellas neuronas que reciben la información de la red.
- Capas ocultas: formada por neuronas cuyas entradas proceden de neuronas de la red y sus salidas permanecen dentro de la red, siendo las entradas a otras neuronas.
- Capas de salida: compuesta por las neuronas cuyas salidas son enviadas al exterior de la red.

Cuando se realiza un aprendizaje supervisado, a través de las etiquetas correspondientes a los datos de entrenamiento, podemos especificar lo que se debe obtener en la capa de salida. Sin embargo, el comportamiento de las otras capas no está especificado directamente, por lo que el algoritmo de aprendizaje debe decidir cómo utilizar esas capas para producir la salida deseada.

La primera capa de cualquier red neuronal toma como entrada los datos, los procesa y extrae la información correspondiente y la pasa a la siguiente capa. Este proceso iterativo acaba cuando los datos llegan a la salida final. Una vez obtenida la predicción final se compara con el dato original y así la red neuronal puede ir ajustando sus parámetros para obtener resultados cada vez mejores. Rumelhart, Hinton y Williams en 1986 desarrollaron los algoritmos base utilizados en el entrenamiento para perceptrones multicapa, usando el método conocido como regla delta generalizada para el aprendizaje por retropropagación [20].

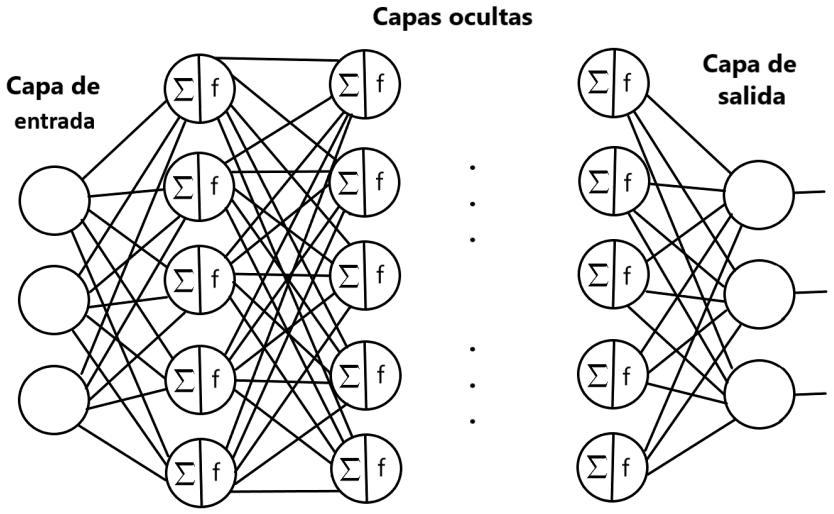


Figura 3.3: Esquema básico de una red neuronal multicapa.

Los algoritmos se pueden clasificar en función de cómo fluyen los datos desde el nodo de entrada hasta el nodo de salida, siendo algunos de los más utilizados hoy en día:

- Perceptrón multicapa o capas densamente conectadas (*feedforward neural networks, or multilayer perceptrons (MLPs)*), donde todas las neuronas están conectadas con todas las neuronas de la capa superior.
- Redes neuronales convolucionales. Son muy efectivas para tareas relacionadas con imágenes, como en la clasificación y segmentación de imágenes o la reducción de ruido, motivo por el cual le dedicaremos especial atención.

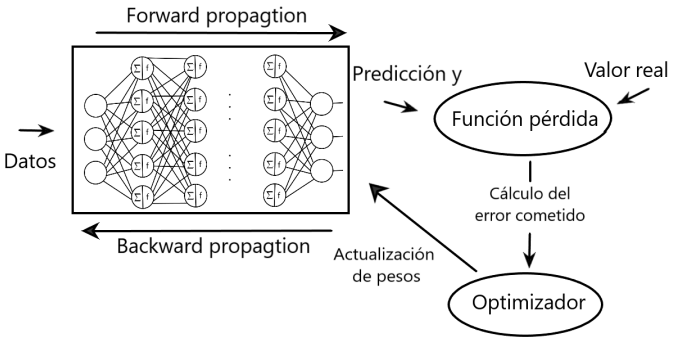
- Redes neuronales recurrentes. Utilizadas para tratar datos secuenciales o datos de series temporales. No tiene una estructura de capas definida, permitiendo conexiones arbitrarias entre las neuronas, incluso pudiendo crear ciclos. Este tipo de redes no se tratarán en esta memoria.

Las redes profundas *feedforward* son los modelos de aprendizaje profundo por excelencia. Se denominan redes *feedforward* a aquellas redes donde la información fluye a través de la capa de entrada hacia las capas ocultas, donde se realizan los cálculos intermedios, conectando cada neurona con las neuronas de la capa siguiente hasta llegar finalmente a la salida. No obstante, las redes neuronales pueden tener otro tipo de conexiones, como laterales o consigo mismas. Por ejemplo, cuando las redes neuronales *feedforward* se amplían para incluir conexiones de retroalimentación, se denominan redes neuronales recurrentes. En las redes multicapa densamente conectadas, todas las salidas de una capa conforman la entrada de la siguiente capa, en cambio, en redes convolucionales se crean conexiones locales que permiten que una capa reciba solo las salidas necesarias de la capa inmediatamente anterior [14].

3.3. Aprendizaje y entrenamiento de las redes neuronales

Hasta el momento hemos ido construyendo el concepto de redes neuronales, las operaciones que realizan para transmitir la información de la entrada a las distintas salidas cuando existen múltiples capas (*feedforward*) e incluso una estrategia de entrenamiento para una única neurona. La pregunta natural ahora es ¿cómo se entrena la red para que realice predicciones correctas cuando contamos con múltiples capas donde no podemos comparar su salida con soluciones deseadas? La cantidad de parámetros a ajustar en redes multicapa es inmensa e inabarcable si se desease hacer un ajuste "manual". Por este motivo el diseño de un buen algoritmo de aprendizaje es crucial.

Como vimos en la sección anterior, cada capa neuronal transforma sus datos de entrada como en la Ec.3.2. Inicialmente, el tensor W correspondiente a los pesos y *bias* contienen números aleatorios que se irán modificando a medida que avance el entrenamiento. Los resultados obtenidos con estos valores iniciales no tendrán sentido, pero serán el punto de partida para iniciar el aprendizaje haciendo uso de los valores conocidos de las etiquetas. Este procedimiento se denomina bucle de entrenamiento [16] y puede estructurarse de la siguiente manera:



- En primer lugar se obtienen las muestras de entrenamiento, x, y los objetivos correspondientes y.
- Se ejecuta el modelo sobre x realizando una propagación *feedforward* para obtener predicciones $\hat{y}$.
- Se calcula el error cometido por el modelo (generalmente en lotes o *batches*) mediante la función de pérdida que mide el desajuste entre $\hat{y}$ e y.
- Finalmente, se realiza una propagación hacia atrás, *backpropagation*, ajustando los pesos de las conexiones entre neuronas con el algoritmo de optimización (optimizador) deseado.

La función perdida o *loss function* relaciona las predicciones de la red y el valor verdadero calculando el error cometido. Es uno de los aspectos más fundamentales a la hora de realizar un buen entrenamiento. En la siguiente tabla se presentarán algunas de las funciones de pérdida más utilizadas en Keras [21].

Tipo	Nombre	Función
Probabilistic	Binary Crossentropy	$L(x) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(\hat{y_i}) + (1-y_i) \cdot log(1-\hat{y_i})$
	Categorical Crossentropy	$\hat{y_i}$: one-hot array
	Sparse Categorical Crossentropy	$\hat{y_i}$: índice de categoría más probable
Regression	*Mean Squared Error (MSE)*	$L(x) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$
	Mean Absolute Error	$L(x) = \frac{1}{N}\sum_{i=1}^{N}\lvert y_i - \hat{y}_i\rvert$
	Mean absolute percentage error	$L(x) = \frac{1}{N}\sum_{i=1}^{N}\left\lvert\frac{y_i - \hat{y}_i}{y_i}\right\rvert$

Tabla 3.2: Ejemplos de funciones de pérdida en función del objetivo final de la red. Tanto la entropía cruzada categórica como la entropía cruzada categórica dispersa tienen la misma función de pérdida, diferenciándose en el formato de salida de la red.

Aprovecharemos la medida de este error como señal de retroalimentación del sistema para ajustar el valor de los parámetros en la dirección que disminuya el error. Ajustar los pesos al error calculado por la función de perdida es el trabajo realizado por el **optimizador**, encargado de implementar la retropropagación o *backpropagation* haciendo uso de la regla de la cadena de Leibniz [16]. Las neuronas de la capa que contribuye directamente a la salida perciben una fracción de la señal total de error proporcional a la contribución relativa que haya aportado cada neurona a la salida original de acuerdo a los pesos de esta. Para poder ajustar los pesos de las neuronas ocultas, no se utilizará únicamente la información proporcionada por la función de coste, sino también por su gradiente, por lo que estas funciones deberán ser diferenciables.

A continuación, presentaremos algunos de los principales optimizadores de Keras [21] en orden cronológico:

- Descenso de gradiente: utiliza gradientes para encontrar el valor mínimo de la función de coste haciendo uso de la regla de la cadena.
    - Descenso de gradiente estocástico (SGD): consiste en la introducción de un comportamiento estocástico (aleatorio) al cálculo de la derivada parcial de la función de coste respecto a uno de los pesos de la red. Limita el cálculo de la derivada a tan solo una observación (por batch), pudiendo quedar estancado en un mínimo local.
    - Descenso de gradiente en lotes: Usamos todo el conjunto de entrenamiento en cada paso del algoritmo de optimización. Dependiendo del volumen de datos tratados, este algoritmo puede requerir bastante memoria y almacenamiento.
    - Descenso de gradiente en minilotes: Si los datos de entrenamiento están bien distribuidos, un pequeño subconjunto de ellos nos debería de bastar para calcular el gradiente de manera fiable. Este procedimiento es más rápido que el descenso de gradiente en lotes y permite reducir el error cometido por SGD.
- Adaptive Gradient Algorithm (AdaGrad): es una modificación de SGD en la que se utilizan diferentes tasas de aprendizaje para las variables teniendo en cuenta su gradiente acumulado. Mejora el algoritmo de descenso de gradiente cuando tenemos varias dimensiones, aunque tiene el riesgo de no llegar al mínimo global [15].
- Adadelta [22] Será el algoritmo de optimización que utilicemos en la parte experimental. Es una extensión de AdaGrad que pretende reducir la brusca reducción de tasa de aprendizaje

que este presenta. En lugar de acumular todos los gradientes pasados al cuadrado, Adadelta restringe la ventana de gradientes acumulados a un tamaño fijo. Puede consultarse más información en el Apéndice C.

- Adam (Adaptive moment estimation). Se puede considerar una extensión de Adadelta, que modifica ciertos ajustes de los hiperparámetros.

Los algoritmos más recientes, como Adadelta o Adam, están construidos en base a sus predecesores. Por esto mismo, para entender a grandes rasgos cómo funcionan estos optimizadores, se explicará el algoritmo de descenso de gradiente de manera genérica, y posteriormente daremos alguna pincelada de las optimizaciones o variaciones del algoritmo que implementan estos optimizadores.

Descenso de gradiente

El descenso de gradiente utiliza las derivadas para minimizar una función, ya que estas nos indican cómo modificar la variable x para lograr una pequeña mejora en y. De este modo, al mover x en pequeños pasos en dirección contraria a la derivada, podemos reducir el valor de la función $f(x)$. Cuando trabajamos con funciones de múltiples entradas, es necesario emplear el concepto de derivadas parciales [14].

Este algoritmo ajusta los pesos realizando pequeñas modificaciones mediante el cálculo de la derivada de la función de pérdida, descendiendo hacia el mínimo global en sentido inverso a la dirección del gradiente. Los pesos son ajustados en orden inverso, propagando hacia todas las neuronas de la capa oculta que contribuyen directamente a la capa de salida una fracción proporcional a su contribución en la seña final. Se calcula en primer lugar las modificaciones a aplicar sobre los pesos de las neuronas de la capa de salida y, a continuación, se aplica la regla de la cadena para calcular las modificaciones de los pesos de las capas inmediatamente anteriores. Se repite este procedimiento de forma sucesiva hasta llegar a al capa de entrada, encadenando la derivada de la función pérdida con las derivadas de cada capa. Este tipo de optimizadores aprovechan el hecho de que las operaciones utilizadas en la red neuronal sean diferenciables, como bien adelantamos en la explicación de las funciones de activación.

El gradiente (primera derivada) siempre apunta hacia el sentido donde se incremente el valor de la función pérdida, por lo que nos moveremos en el sentido contrario para minimizar el error. La actualización de los pesos tras cada iteración n se realizará de la misma manera que en la Ec.

3.3, $w_{ij}(n+1) = w_{ij}(n) + \Delta w_{ij}(n)$, donde los subíndices de ij hacen referencia a la conexión de la neurona j con la neurona i de una capa anterior, η es el factor de aprendizaje y L la función de pérdida,

$$\Delta w_{ij} = -\eta \frac{\partial L}{\partial w_{ij}} \ . \tag{3.4}$$

De esta manera, si los pesos se actualizan en cada iteración, se requiere que los datos de entrenamiento se suministren de forma aleatoria. De no ser aleatoria, en el entrenamiento habría sesgo a favor del último patrón de entrenamiento, ya que su actualización, por realizarse siempre al final, predominaría sobre las restantes. La mayoría de aplicaciones usan un subconjunto de muestras del conjunto de datos de entrenamiento (un hiperpárametro denominado *batch size* que permite pasar subconjuntos de datos a la red de forma simultánea y trataremos más adelante) [23]. En caso de tener P muestras,

$$\Delta w_{ij} = -\frac{\eta}{P} \frac{\partial \sum_{p=1}^{P} L_p}{\partial w_{ij}} \ . \tag{3.5}$$

Alternativamente, se podría ejecutar cada paso en todos los datos disponibles y realizar un descenso de gradiente por lotes. Cada actualización sería más precisa, pero mucho más costosa computacionalmente. Como adelantamos en la explicación del descenso de gradiente en minilotes, una aproximación eficaz intermedia entre estas dos estrategias es dividir en pequeños conjuntos de tamaño razonable los datos de entrenamiento y actualizar los pesos en cada minilote.

Para agilizar el proceso de entrenamiento, se selecciona de forma aleatoria un subconjunto de datos de entrada o *batch* para realizar los cálculos en cada iteración. Denominaremos época o *epoch* a la actualización de todos los pesos de la red tras el paso de todos los datos, normalmente introducidos en la red en forma de batches. Para la obtención del gradiente, el procedimiento consiste en el cálculo de una serie de derivadas parciales sobre parámetros desde la última capa hasta la de la conexión con el peso correspondiente aplicando la regla de la cadena. Este desarrollo puede consultarse en el Apéndice B.

Durante el entrenamiento de la red, el modelo se ajusta inicialmente al conjunto de datos de entrenamiento, empleados para ajustar los pesos. Un aspecto a destacar es la división de la base de datos disponible en un conjunto de entrenamiento, uno de validación y uno de test para ser empleados en diferentes fases. El modelo ajustado tras el entrenamiento se emplea para predecir los resultados de los datos de validación y obtener una evaluación imparcial del desempeño de la red. Esto sirve para guiar el entrenamiento. El objetivo es ajustar la red de manera que ésta

tenga el mejor rendimiento cuando se enfrente a datos nuevos. Por último, el conjunto de test es empleado tras el entrenamiento para evaluar la red ya ajustada sobre datos nuevos nunca vistos por ésta.

3.3.1. Entrenamiento, validación y test: Hiperparámetros

Hemos mencionado la necesidad de dividir el conjunto de datos disponibles en tres lotes: entrenamiento, validación y prueba. ¿Por qué no utilizar únicamente el conjunto de entrenamiento y de prueba? Es razonable pensar que podríamos limitarnos a entrenar con los datos de entrenamiento y se evaluaría con los datos de prueba. No obstante, desarrollar un modelo siempre implica ajustar su configuración, como el número o tamaño de las capas, es decir, ajustar los llamados hiperparámetros del modelo [15]. Estos son variables establecidas por el programador que determinan la estructura de la red o su entrenamiento, a diferencia de los parámetros del modelo, los pesos, que son internos en la red neuronal y se estiman a través del entrenamiento de manera automática. A través de los datos de validación ajustaremos los hiperparámetros de la red, prestando especial atención a dos de los problemas más usuales en el entrenamiento de la red: el *overfitting* y *underfitting*. Puede ocurrir que la precisión del modelo sea muy alta sobre los datos de entrenamiento, pero no ocurra lo mismo sobre los datos de validación. Esto es debido a que en ocasiones la red ha entrenado de forma que ha interiorizado rasgos demasiado específicos de los datos de entrenamiento y no es capaz de generalizarlo a otras muestras. El caso contrario es el de *underfitting*, que ocurre cuando la red ni siquiera es capaz de predecir de forma correcta las etiquetas del conjunto de entrenamiento. Al final de cada época se aplicará el modelo sobre los datos de validación con la finalidad de ir contrastando el error y la precisión del set de entrenamiento frente a unos datos con los que la red no ha sido entrenada, obteniendo una curva de entrenamiento similar a la de la Figura 3.4.

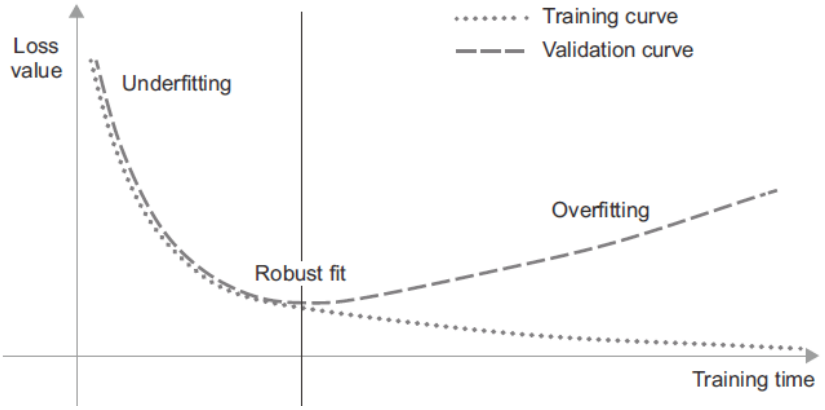
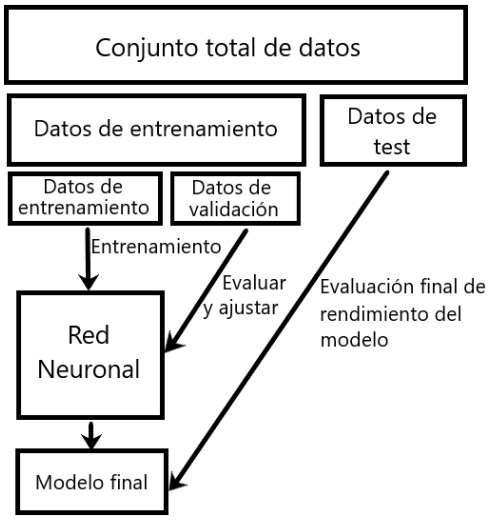


Figura 3.4: Representación de la evolución de *train loss* y *val loss* con las épocas de entrenamiento, donde se pueden diferenciar las zonas de *overfitting* y *underfitting*. Imagen adaptada de [16].

El ajuste de los hiperparámetros se realiza utilizando como señal de retroalimentación el rendimiento del modelo en los datos de validación. En esencia, este ajuste es una forma de aprendizaje, una búsqueda de la configuración óptima de los hiperparámetros [16].



Como resultado, el ajuste de la configuración del modelo basado en el rendimiento sobre el conjunto de validación puede dar lugar a un sobreajuste respecto a los datos de validación, aunque el modelo nunca se haya entrenado directamente con ellos, obteniendo un modelo que funciona artificialmente bien a estos datos en específico. Una vez que el modelo está listo, se prueba por última vez con los datos de prueba, que deben ser lo más parecidos posible a los datos de producción, para poder medir el rendimiento en datos completamente nuevos.

Podemos clasificar los hiperparámetros en dos grupos:

- **Hiperparámetros a nivel de estructura y topología**: Son aquellos parámetros que definen la arquitectura de la red: nº de capas, de neuronas por capa, funciones de activación, recogidas en la Tabla 3.1, o funciones error, recogidas en la Tabla 3.2. Su elección estará condicionada por el tipo de predicción que se desee llevar a cabo. A continuación se muestra una tabla donde se recogen algunos ejemplos.

Tipo de problema	**F. activación**	**F. pérdida**
Clasificación múltiple con *one-hot*	*softmax*	*categorical-crossentropy*
Clasificación múltiple con índice categoía	*softmax*	*sparse-categorical-crossentropy*
Regresión	identidad	MSE
Clasificación binaria *one-hot*	*sigmoid*	*binary-crossentropy*

Tabla 3.3: Elección de hiperparámetros en función del problema abordado por la red. Tabla adaptada de [15].

- **Hiperparámetros a nivel de algoritmo**: Definen el modo de funcionamiento de la fase de aprendizaje con el objetivo de modificar el modo de ejecución del algoritmo de entrenamiento nº de épocas, *batch size*, *learning rate* etc.

  - **Número de epocas**: Nos indica el número de veces que los datos de entrenamiento han pasado por la red neuronal en el proceso de entrenamiento. Un número excesivo o insuficiente de épocas puede dar problemas de *overfitting* o *underfitting* respectivamente. Una buena estrategia a seguir es aumentar el número de épocas hasta que la función de pérdida con los datos de validación comience a decrecer tal y como se puede ver en la Figura 3.4. Cuando se entrena un nuevo modelo, no se puede saber cuántas épocas serán necesarias para llegar a un ajuste óptimo. Una primera estrategia sería entrenar durante suficientes épocas como para empezar a sobreajustar, averiguar el número adecuado de épocas para entrenar y, finalmente, lanzar una nueva ejecución de entrenamiento desde cero utilizando este número óptimo. Una forma mucho mejor de manejar esta situación es detener el entrenamiento cuando se detecte que la pérdida de validación ya no está mejorando. Esto se puede conseguir utilizando *EarlyStopping*, que interrumpe el entrenamiento una vez que la métrica objetivo que se está monitorizando ha dejado de mejorar durante un número fijo de épocas, evitando así tener que volver a entrenar el modelo para un número menor de épocas [16].
  - **Batch size**: El tamaño del lote que se utilizará en cada época en el descenso de gradiente por minilotes, es decir, el número de muestras procesadas antes de la actualización de pesos. El tamaño del lote se ha estudiado en varios artículos, demostrando

que este tiene un efecto crucial en la precisión de la red, sobre todo en el reconocimiento de imágenes [23]. Cuanto mayor sea el valor del parámetro, mayor será la precisión, aunque hay que tener en cuenta que un valor grande conlleva grandes costes computacionales, por lo que depende de la capacidad de memoria del computador.

- **Learning rate y learning decay**: Como vimos en la Ec. 3.4, los algoritmos de descenso de gradiente multiplican la magnitud del gradiente por un escalar conocido como rango de aprendizaje o *learning rate*, η. En general, si es demasiado grande, se corre el riesgo de dar pasos muy grandes con los que se podría saltar el mínimo, entrando en un bucle en el que rebota al azar entorno a este. En cambio, un rango demasiado pequeño podría quedarse estancado en un mínimo local. La mejor tasa de aprendizaje es aquella que desciende a medida que el modelo se acerca a la solución. Para conseguir este modelo podemos hacer uso del *Learning decay*. Este hiperparámetro permite que el aprendizaje avance más rápido al principio (con valores grandes) y cada vez tome valores más pequeños para que el proceso converja al mínimo de la función de pérdida. La tasa de aprendizaje variará en función del optimizador que se esté utilizando [14].
- **Momentum**: Este parámetro es de utilidad en casos donde nos encontramos varios mínimos locales en los que el optimizador puede quedarse atascado fácilmente (ya que una vez que el gradiente sea 0, no podemos salir del mínimo local). Podríamos intentar reiniciar el proceso desde posiciones aleatorias e incrementar la posibilidad de llegar el mínimo global. El factor de *momentum* α, un hiperparámetro con valor entre 0 y 1 que puede incluirse en el ajuste de pesos como $\Delta w_{ij}(n+1) = -\eta \frac{\partial L}{\partial w_{ij}}(n) + \alpha \Delta w_{ij}(n)$, incrementa el valor de los pesos ligeramente, otorgándoles un poco de ímpetu evitando el estancamiento en los mínimos locales. Esto significa actualizar los pesos basándose no sólo en el valor actual del gradiente, sino también en la actualización anterior del parámetro [16].

3.4. Regularización

Uno de los principales inconvenientes que presenta el algoritmo de descenso de gradiente es su inestabilidad, siendo un método propenso a problemas de sobreajuste. Obtener más y mejores datos de entrenamiento es la mejor solución cuando nos encontramos ante este tipo de

problemas, pero cuando esto no es posible, se recurren a técnicas de regularización. Las técnicas de regularización son un conjunto de prácticas que impiden activamente que el modelo se ajuste perfectamente a los datos de entrenamiento, con el objetivo de que funcione mejor durante la validación [16]. Algunas de las técnicas de regularización más usadas son:

- La reducción del tamaño de la red o la inclusión de restricciones al modelo: La forma más sencilla de mitigar el sobreajuste es reducir el tamaño del modelo (el número de pesos, determinado por el número de capas y el número de unidades por capa). Si el modelo tiene recursos de memoria limitados, no podrá memorizar los datos de entrenamiento. Al mismo tiempo, debemos utilizar modelos que tengan suficientes parámetros como para evitar el *underfitting*, intentando encontrar la capacidad óptima de ajuste de la red. Otra técnica utilizada especialmente en redes convolucionales consiste en utilizar los mismos parámetros en diferentes capas o neuronas de la red.

- Disminuir el valor de los pesos añadiendo términos adiciones a la función de coste: Al aumentar el resultado de la función coste, el gradiente seguirá actualizando los pesos de forma que disminuyan sus valores.

- *Dropout*: La idea esencial es desactivar neuronas aleatoriamente, anulando la actividad de ciertos nodos en cada iteración durante el entrenamiento, evitando que la red memorice parte de los datos de entrada. Esto permite aumentar el número de neuronas y, por tanto, su capacidad para ajustar los parámetros y hacer una mejor predicción, evitando el sobreajuste.

3.4.1. Disminuir el valor de los pesos

Una forma común de mitigar el sobreajuste es poner restricciones a la complejidad de un modelo, obligando a que sus pesos tomen sólo valores pequeños, lo que hace que la distribución de los valores de los pesos sea más regular. Esto se denomina regularización de pesos, y una forma de conseguirlo es añadir a la función de pérdida del modelo un coste asociado a tener pesos grandes. Este coste puede ser de dos tipos:

- L2: también conocido como *weight decay* o *Tikhonov regularization*. En este caso el término añadido a la función de pérdida es proporcional al cuadrado del valor de las matrices de

pesos de cada capa

$$loss + \lambda \sum_{l=1}^{n} \left| W^l \right|^2 , \tag{3.6}$$

siendo n el número de capas, W^l la matriz de pesos de la capa l y λ el parámetro de regularización.

- L1: En este caso la norma no se eleva al cuadrado,

$$loss + \lambda \sum_{l=1}^{n} \left| W^l \right| . \tag{3.7}$$

En comparación con la regularización L2, la regularización L1 da como resultado una solución más dispersa. La dispersión en este contexto se refiere al hecho de que algunos parámetros tienen un valor óptimo de cero y por tanto los pesos pueden reducirse hasta anularse [14]. Por lo tanto, haciendo uso de L1 el modelo puede dejar de considerar ciertas características de los datos, pero con L2 todas intervendrán, por poco que sea.

Las regularizaciones del peso se utiliza habitualmente para modelos de aprendizaje profundo más pequeños. Los modelos de gran tamaño tienden a estar tan sobreparametrizados que imponer restricciones a los valores de peso no tiene mucho impacto en la capacidad ni en la generalización del modelo. En estos casos, se prefiere una técnica de regularización como el *dropout* [16].

3.5. Desvanecimiento/Explosión de gradientes

Durante la actualización de los pesos nos podemos encontrar con dos problemas opuestos que dan lugar a un entrenamiento fallido: gradientes evanescentes o *vanishing gradient* y gradiente de explosión o *exploding gradient* [14]. Los pesos más afectados por ambos problemas son los correspondientes a las conexiones entre las neuronas de las primeras capas de la red, debido al uso de *backpropagation* en la actualización de los pesos: cuantas más capas sigan a la de la conexión correspondiente, más factores intervendrán en el cálculo del gradiente.

Los gradientes evanescentes se producen cuando el gradiente de la función de pérdida con respecto a los pesos es demasiado pequeño, por lo que el cambio en los pesos será mínimo, pudiendo ser disminuido aún más al multiplicarlo por la tasa de aprendizaje. Esto se ve acentuado a medida que el algoritmo de retropropagación avanza hacia atrás desde la capa de salida hacia la capa de entrada, disminuyendo cada vez más los gradientes con valores pequeños y aproximando

su valor a cero. Finalmente los pesos de las capas iniciales o inferiores casi no perciben cambios, dificultando la convergencia al mínimo. Los gradientes evanescentes dificultan saber en qué dirección deben moverse los parámetros para mejorar la función de coste, provocando un estado de entrenamiento parado.

Por otra parte, la explosión del gradiente pueden hacer que el aprendizaje sea inestable en casos donde los gradientes siguen aumentando a medida que avanza el algoritmo de retropropagación. En este caso el gradiente es un número grande y los cambios en los pesos serán muy radicales.

Para solucionar estos problemas, se han propuesto ciertas mejoras en los algoritmos, como una inicialización adecuada de los pesos [24] o el refinamiento específico de ciertos hiperparámetros a un problema abordado, como las funciones de activación. A continuación, se explicarán algunas de las medidas más comunes.

3.5.1. Uso de funciones de activación no saturantes

Ciertas funciones de activación como la sigmoide o tanh , Tabla 3.1, tienden a saturar para valores de entradas grandes, ya sean negativas o positivas. Esto puede ser una razón importante en la desaparición de gradientes, por lo que no es recomendable utilizarlas en las capas ocultas de la red.

Para abordar el problema de saturación en las funciones de activación, debemos utilizar otras funciones no saturantes como:

- ReLU *Rectified Linear Unit*. Una de las grandes ventajas es su sencillez en su matemática en comparación con tanh y sigmoides, lo que aumenta su rendimiento computacional. Como ya recogimos en la Tabla 3.1, para entradas positivas posee un comportamiento lineal, permitiendo que los gradientes fluyan bien por las rutas activas de las neuronas y sigan siendo proporcionales a las activaciones de los nodos. Para cualquier entrada negativa se obtiene salida 0. Aunque esta característica confiere a ReLU sus puntos fuertes (a través de la dispersión de la red), se convierte en un problema cuando la mayoría de las entradas a estas neuronas ReLU están en el rango negativo. En el peor de los casos, toda la red muere y se convierte en una función constante.
- LeakyReLU: La cantidad de "fuga" está controlada por el hiperparámetro α, que es la pen-

diente de la función para $z < 0$. Esta pendiente garantiza que las neuronas alimentadas por LeakyReLU nunca mueran, solucionando el problema subyacente de la función ReLU. Ciertas variantes como PReLU (*Parametric Rectified Linear Unit*) permiten el entrenamiento de α. Otra variante podría ser ELU (*Exponential Linear Unit*) o SELU (*Scaled* ELU).

3.5.2. Normalización

Generalmente, las características que definen los datos pueden tomar valores muy extremos o estar definidas en escalas totalmente dispares unas de otras. Tener escalas de datos muy diferentes puede afectar negativamente en el entrenamiento de la red. Además, valores muy altos pueden inducir la explosión del gradiente o aumentar significativamente el coste computacional. Para evitar estos problemas, se han de procesar los datos antes de introducirlos a la red, transformando los datos a la misma escala. Los procesos más comunes son el de normalización y el de estandarización, habitualmente recogidos ambos como normalización.

El primero de ellos consiste en transformar los datos en el intervalo [0,1]. El segundo se basa en centrar los datos en cero restando la media de los datos y establecer una desviación estándar unitaria tras dividir los datos por su desviación estándar [16].

Este proceso normaliza los datos antes de introducirlos en los modelos, pero la normalización de los datos puede ser de interés después de cada transformación operada por la red [16]. Puede ocurrir que algunos pesos sean numéricamente muy elevados por destacar mucho más una propiedad entre todas ellas, llevando consigo una salida de la neurona muy grande. En este caso se debe de aplicar también una estandarización a la salida de la función de activación mediante una normalización por lotes o *Batch Normalization*, un método de reparametrización adaptativa generalmente aplicado en modelos muy profundos [14].

3.6. Redes neuronales convolucionales (CNN)

Las redes neuronales convolucionales han sido especialmente diseñadas para procesar matrices estructuradas como pueden ser señales y secuencias (1D), imágenes y espectrogramas de audio (2D) o para vídeo e imágenes volumétricas (3D) [14].

Las CNNs hacen uso de la operación de convolución en lugar de la multiplicación matricial,

siendo mucho más eficiente en términos de requisitos de memoria [14]. Esta operación otorga a estas redes cierto grado de invariabilidad a los cambios y distorsiones en los datos de entrada gracias al uso de campos receptivos locales[2] y pesos compartidos [25]. Además, la convolución permite trabajar con entradas de tamaño variable y limita la capacidad de la red al compartir el valor de pesos, lo cual reduce el número de parámetros de la red y reduce el sobreaprendizaje.

En las capas de estas redes, cada neurona realiza la operación sobre un conjunto reducido de datos de entrada (*patches*[3]), por lo que no todos los datos de entrada están conectadas con los datos de salida. Esto supone una gran ventaja respecto a las arquitecturas densamente conectadas, donde la topología de la entrada se ignora por completo y las variables de entrada pueden presentarse en cualquier orden fijo sin que ello afecte al resultado del entrenamiento. Cuando tratamos datos como imágenes o series temporales, estas tienen una fuerte estructura local 2D o 1D respectivamente: las variables (o píxeles) espacial o temporalmente cercanas están muy correlacionadas, por lo que extraer y combinar características locales permite reconocer distintos objetos espaciales o temporales de manera mucho más eficaz que mostrando el conjunto de datos al completo.

La operación de convolución se logra haciendo uso de un *kernel* correspondiente a cada neurona. En lugar de aprender un conjunto de parámetros distinto para cada posición, aprende un único conjunto, lo cual reduce los requisitos de almacenamiento y fuerza la replicación de las configuraciones de pesos a través del espacio. Esta forma particular de compartir parámetros y replicar las configuraciones de pesos a través del espacio hace que la capa tenga una propiedad llamada equivarianza[4] a la traslación [14], propiedad de gran utilidad en el procesamiento de imágenes.

La Figura 3.5 muestra el concepto de capas convolucionales formadas por *feature maps* y capas de *pooling* (o submuestreo). Normalmente, en cada capa convolucional tienen lugar fundamentalmente 3 procesos. Primeramente, la capa realiza varias convoluciones en paralelo para producir un conjunto de activaciones lineales. A continuación, cada activación lineal se ejecuta a través de una función de activación no lineal, como la función de activación lineal rectificada (ReLu) y finalmente utilizamos una función de agrupación o *pooling* para modificar y reducir aún más la salida de la capa [14]. En ocasiones, en la salida se sitúan capas *full-connected*.

[2]El campo receptivo local es un área segmentada de los datos de entrada que se introducirá a una neurona dentro de una capa convolucional, es decir, la *sub-imagen* que entra a cada neurona [25]

[3]Subsección de la imagen o mapa de entrada a la cual se le aplica el *kernel*.

[4]Una función es equivariante si al cambiar los datos de entrada, la salida cambia de la misma manera $f(g(x)) = g(f(x))$ [14].

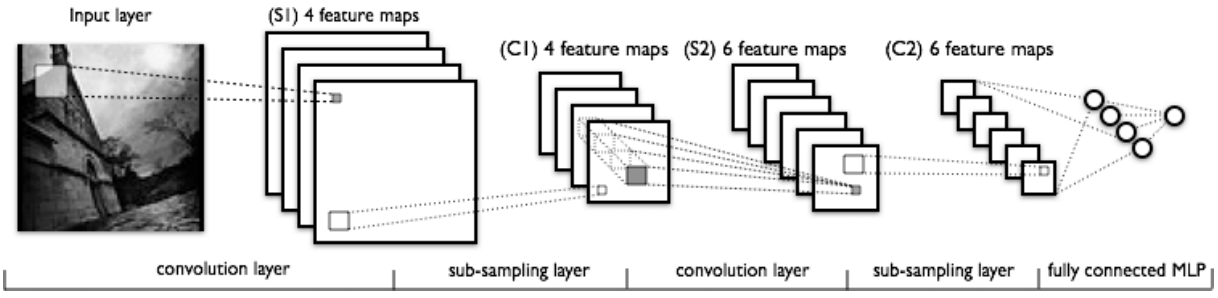


Figura 3.5: Estructura clásica de una red convolucional para clasificación. Figura adaptada de [26].

3.6.1. Capas convolucionales

Las capas convolucionales son simplemente capas neuronales que utilizan la convolución en lugar de la multiplicación matricial [14]. Esta operación, explicada previamente en el apartado 2.3.1, hace uso de *kernels*, una matriz de pequeñas dimensiones, para la extracción de información o características de las imágenes. Estas capas son las encargadas de aprender patrones locales a través de *patches* de dos (o tres) dimensiones de la imagen.

En el caso de la capa inicial, las operaciones de convolución se realizan directamente sobre las imágenes, mientras que en el caso de las capas intermedias los *kernels* se aplican sobre el resultado de la convolución de la capa anterior, denominados mapas de características o de activación (*feature/activation maps* Figura 3.5). La operación de convolución divide el mapa de entrada en parches o *patches* y aplica a cada uno de ellos la misma transformación (*kernel*). Aplicar los *kernels* a un *patch* cada vez en lugar de todo el conjunto de datos permite que los filtros procesen pequeños fragmentos de la imagen para detectar ciertas características (bordes, etc.). La primera capa de convolución aprenderá pequeños patrones locales, como bordes, una segunda capa de convolución aprenderá patrones más grandes formados por las características de las primeras capas, y así sucesivamente. Esto otorga a las redes de convolución la capacidad de aprender de forma eficaz conceptos visuales cada vez más complejos y abstractos de manera jerárquica.

Un filtro definido por una matriz W y un sesgo b solo permite detectar una característica concreta en una imagen. Para poder realizar un reconocimiento de imágenes debemos usar varios filtros a la vez, uno para cada característica que se desee detectar. Por esto mismo, una capa convolucional suele incluir varios filtros organizados en mapas de características.

3.6.2. Capas *pooling*

Seguidamente a las capas convolucionales (tras aplicar o no una activación no lineal), suelen situarse las capas *pooling* o reducción de muestreo. Realizan una simplificación de la información recogida por la capa convolucional, dando lugar a una versión condensada de la información contenida en esa capa reduciendo las dimensiones de cada mapa. Los valores de los píxeles de una determinada zona se agregan empleando una función invariante de permutación que recoja la estadística de los píxeles cercanos: comúnmente el valor máximo (*max pooling*) o el promedio (*average pooling*). Generalmente estas capas se emplean con el fin de disminuir el tamaño de la imagen y tener una menor sobrecarga de cálculo en las capas posteriores, así como para conseguir una reducción del sobreajuste.

Esta operación ayuda a que la representación sea invariante a pequeñas traslaciones de la entrada. La invariancia a la traslación local puede ser una propiedad muy útil si importa más la presencia de alguna característica que dónde está exactamente. Por ejemplo, al determinar si una imagen contiene una cara, no necesitamos conocer la ubicación de los ojos con una precisión de píxeles perfecta, sólo necesitamos saber que hay un ojo en el lado izquierdo de la cara y un ojo en el lado derecho. Cuando esta suposición es correcta, el uso de capas *pooling* puede mejorar mucho la eficacia estadística de la red [14].

Conceptualmente, las capas *pooling* operan de manera similar a la convolución, salvo que en lugar de transformar los parches locales mediante una transformación lineal aprendida (el *kernel* de convolución), se transforman mediante una operación de promedio o valor máximo. Una gran diferencia respecto a la convolución es que el *pooling* se realiza normalmente con ventanas de aplicación no contiguas (lo que más adelante se definirá como *stride*), con el fin de reducir las dimensiones de los mapas de características.

3.6.3. Hiperparámetros propios de las CNNs

Aparte de los hiperparámetros mencionados en la sección 3.3.1, las redes convoluciones cuentan con una serie de hiperparámetros adicionales:

- Tamaño de los *patches* extraídos de las entradas: Elegir su tamaño adecuado es de gran importancia en la detección de ciertos elementos en la imagen. Por ejemplo, si deseamos segmentar ciertos objetos en una imagen, el patch tiene que tener un tamaño apropiado

para poder recoger suficiente información del objeto y poder clasificarlo.

- Profundidad del mapa de características de salida: es el número de filtros calculados por la convolución, generalmente 32, 48 o 64.

- *Padding:* Si se desea obtener un mapa de características de salida con las mismas dimensiones espaciales que el de entrada, se puede utilizar el relleno o *padding*. El relleno consiste en añadir un número adecuado de filas y columnas a cada lado del mapa de características de entrada para que quepan ventanas de convolución centradas alrededor de cada píxel de entrada. Generalmente estas filas y columnas adicionales tienen valor 0, evitando así que tengan influencia en el resultado final [16]. Se utilizará *padding* = "SAME" en las capas de convolución para evitar la reducción espacial rellenando con ceros y "VALID" si no deseamos corregir las dimensiones de salida.

- *Strides*: Hasta ahora hemos asumido que los centros de los *kernels* de convolución eran contiguos. Pero la distancia entre dos *kernels* sucesivos es un parámetro ajustable de la convolución, llamado *stride*, que por defecto es 1. Utilizar *stride* 2 significa que la anchura y la altura del mapa de características se reducen en un factor de 2 (además de los cambios inducidos por los efectos de borde). En general, los *strides* son preferibles a los *max pooling* para cualquier modelo en el que sea relevante la localización de la información ya que tendrá que producir una codificación de la imagen que se pueda utilizar para reconstruir una imagen válida. Por este motivo, convoluciones con *strides* raramente se utilizan en los modelos de clasificación, prefiriendo la operación *max-pooling* para reducir el muestreo de los mapas de características [16].

Las capas de convolución se realizan normalmente con *kernels* de 3×3 y sin *stride* (*stride*= 1). Por otro lado, las capas *pooling* suelen ser matrices 2×2 con *stride*= 2, para reducir los mapas de características en un factor de 2.

En redes CNNs, debemos tener en cuenta el tamaño del *batches* no hará referencia al número de imágenes individuales que se analizan o las *slices* del volumen, sino al número de *patches* procesados antes de la actualización de los pesos.

3.7. Autocodificadores convolucionales (CAE)

El uso típico de las redes convolucionales es en tareas de clasificación, donde la salida correspondiente a cada imagen introducida es una única etiqueta de clase. Se introduce una imagen o cualquier matriz estructurada y se obtiene como salida una probabilidad de distribución de clases. Sin embargo, en muchas tareas visuales, especialmente en el procesamiento de imágenes, como segmentación o filtrado de ruido, la salida deseada debe incluir la localización, es decir, se supone que se asigna una etiqueta de clase a cada píxel [1]. En este tipo de aplicaciones, entran en juego los autocodificadores o *autoencoders* , una red neuronal que se entrena para intentar copiar su entrada, aprendiendo a codificar la entrada en una versión comprimida para después tratar de reconstruirla en su salida [14]. Dependiendo del tipo de problema que se desee abordar, se pueden utilizar diferentes tipos de redes neuronales. Por ejemplo, al trabajar con series temporales, se utilizarán redes neuronales recurrentes (RNN) para codificar y decodificar los datos. En este trabajo nos centraremos en el tratamiento de imágenes, donde las redes neuronales convolucionales son las utilizadas en los *autoencoders* (CAE).

Los autocodificadores convolucionales son básicamente redes convolucionales con un camino codificador, que progresivamente reduce el tamaño de la imagen substrayendo las características fundamentales en el espacio latente[5], y un camino decodificador que convierte el vector final del camino codificador en la imagen original.

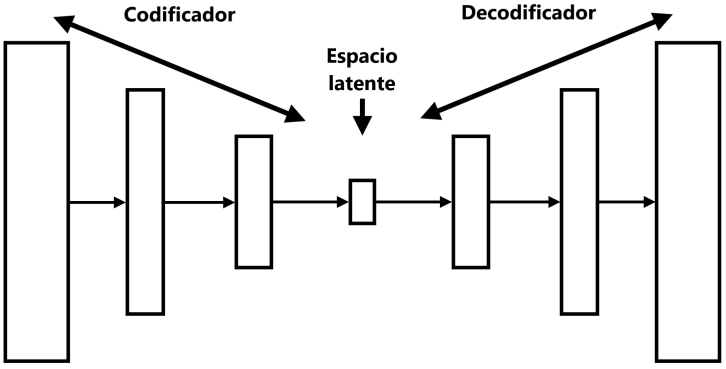


Figura 3.6: Arquitectura Autoencoder

En las redes codificador-decodificador no hay una correspondencia lineal entre el espacio de entrada inicial y el mapeo de salida, habiendo una clara distinción entre el camino codificador y decodificador. Dependiendo del objetivo final deseado, este tipo de estructuras pueden presentar

[5]Espacio multidimensional que contiene una representación abstracta y comprimida de la imagen y es la única información a partir de la cual el decodificador tratar de reconstruir la entrada.

algunas desventajas. En el caso de la segmentación en imágenes, los autoencoders presentan una pérdida de resolución y detalles, que puede ocasionar un impacto negativo en aplicaciones biomédicas como la segmentación celular. La pérdida de resolución de los mapas de características ocurre al pasar de las capas iniciales del codificador a las capas finales, donde, al reducir la dimensión de la imagen, se pierde resolución espacial. Una de las soluciones más potentes es la estructura que presentan las *Convolutional U-Net* (Ronneberger et al., 2015 [1]). En ella, para no perder la información que define los detalles más finos, se aprovechan las características de las capas codificadoras anteriores (con más información espacial) uniéndolas a las características de sus decodificadores equivalentes.

3.7.1. U-Net

La red Unet, al igual que los autoencoders, tiene capas de convolución/downsampling y de deconvolución (*upsampling*) para aumentar la dimensionalidad de la imagen. La diferencia fundamental entre estas redes con respecto a los autoencoders es la existencia de las conexiones (*skip connections*) entre las capas de convolución y deconvolución [27].

En las U-Nets, el mapeo de salida depende directamente del espacio de entrada mediante conexiones de salto. Esto significa que no existe una separación absoluta entre el camino codificador y decodificador, debido a que ya no hay un mapeo de la muestra a un espacio latente bien definido y el calculo posterior de la salida a partir de él, sino que para calcular la salida también se necesita la entrada y todas sus representaciones intermedias.

La ruta de contracción sigue la arquitectura típica de una red convolucional, la aplicación repetida de dos convoluciones 3×3 (en este caso concreto no se utiliza *padding*), cada una seguida de función de activación ReLU y una operación de *max pooling* 2×2 con *stride* 2 para el muestreo descendente. En cada paso de muestreo descendente se duplica el número de canales de características, tal y como se puede ver en la Figura 3.7

Cada paso de la ruta expansiva consiste en la aplicación de *up-convolution* 2×2 sobre el mapa de características que reduce a la mitad el número de canales de características y duplica sus dimensiones, una concatenación con el correspondiente mapa de características recortado de la ruta de contracción equivalente, y dos convoluciones 3×3, cada una seguida de un ReLU. El recorte es necesario debido a la pérdida de píxeles de borde en cada convolución, aunque posteriores versiones de esta red mejoraron la arquitectura y no es necesario este paso. En la

última capa se utiliza una convolución 1×1 para asignar cada vector de características de 64 componentes al número deseado de clases, en este caso concreto, 2 clases. En total, la red tiene 23 capas convolucionales [1].

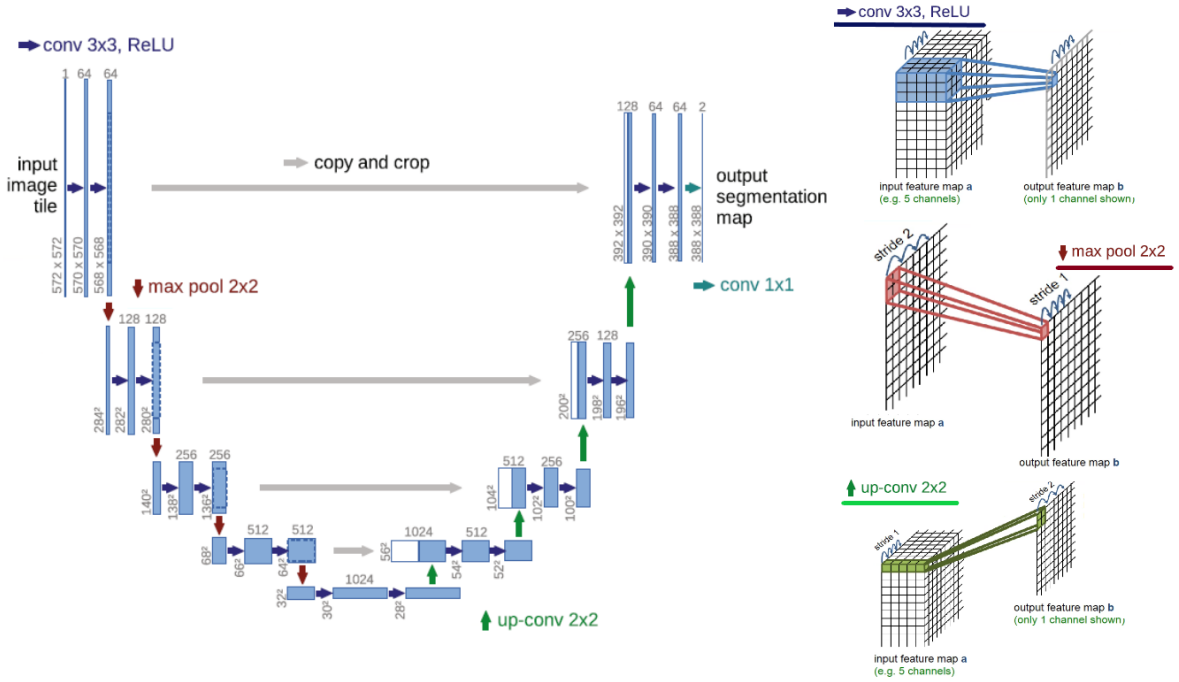


Figura 3.7: Arquitectura U-net. Cada recuadro azul corresponde a un mapa de características multicanal. El número de canales se indica en la parte superior del recuadro. El tamaño de la matriz se indica en el borde inferior izquierdo del recuadro. Los recuadros blancos representan mapas de características copiados de las capas codificadoras equivatentes. Los colores de las flechas indican las distintas operaciones especificadas en la zona derecha de la ilustración. Figura adaptada de [1].

Las redes de aprendizaje profundo que utilizaremos en los capítulos posteriores para procesar el volumen de tomogramas celulares serán distintas variantes de la arquitectura UNet.

Data augmentation

Como hemos mencionado a lo largo de este capítulo, la manera más eficaz de mejorar la capacidad de generalización de un modelo de aprendizaje automático es entrenarlo con más datos. En la práctica, la cantidad de datos de la que se dispone es limitada. Una forma de evitar este problema es modificar los datos de partida aplicando ciertas transformaciones a los datos originales y añadirlos al conjunto de entrenamiento como datos nuevos [14]. Las transformaciones más comunes que se emplean para generar datos adicionales consisten en volteo, rotaciones

y escalado en tamaño de los patches. En especial, en tareas de clasificación, el aumento de datos es esencial para enseñar a la red las propiedades de invariancia y conseguir generalización deseada cuando sólo se dispone de unas pocas muestras de entrenamiento. En el caso de las imágenes y volúmenes tomadas mediante FIB-SEM o TEM, es necesaria la invariancia de desplazamiento y rotación, así como una robustez frente a deformaciones y variaciones del valor de gris. Especialmente las deformaciones elásticas aleatorias de las muestras de entrenamiento parecen mejorar enormemente la capacidad de segmentación de la red con muy pocas imágenes etiquetadas disponibles [1].

Dimensionalidad de la red

La primera UNet convolucional y modelos posteriores toman como entrada a la red cortes 2D del volumen en una sola orientación. Predicen la segmentación para cada corte por separado y luego se combinan las predicciones en un volumen. Este enfoque tiene una precisión limitada porque no considera los otros dos planos en el volumen de la imagen. Sin algún método de posprocesamiento para tener en cuenta las perspectivas desde otras orientaciones, los modelos de este tipo estarán limitados intrínsecamente por lo bien que se pueda detectar las distintas características en una orientación única [27].

Una posible solución para aumentar la coherencia en el volumen sintetizado y mejorar la perspectiva tridimensional del contexto es utilizar una red U-Net 3D. En una capa de convolución 3d, se utiliza un filtro tridimensional para realizar convoluciones, desplazándose en las tres direcciones, mientras que en una CNN 2D solo se realizaban desplazamientos en dos dimensiones. Sin embargo, el uso de volúmenes de entrada y salida de resolución completa limita el número de filtros de las capas de convolución por razones de capacidad computacional.

La Red Neuronal Convolucional 2.5D (CNN 2.5D) proporciona un enfoque de aprendizaje intermedio entre las tareas de aprendizaje de imágenes 3D basadas en *patches* 2D y las redes convolucionales 3D. La CNN 2.5D utiliza un número de *slices* o cortes 2D limitados ortogonales de cada *patch* como entradas para una CNN 2D. Los pesos del kernel convolucional se reparten entre los cortes de entrada en la primera capa convolucional, de manera que el píxel resultante de la convolución posee información de las *slices* de entrada. Esta red puede incorporar mejoras en la capacidad de predicción propias de las capas convolucionales 3D utilizando menos parámetros y permitiendo más muestras generadas con *data augmentation* para una mejor generalización

del modelo. Sin embargo, la falta de información sobre los vóxeles en las direcciones diagonales podría limitar el rendimiento de la CNN 2.5D [28].

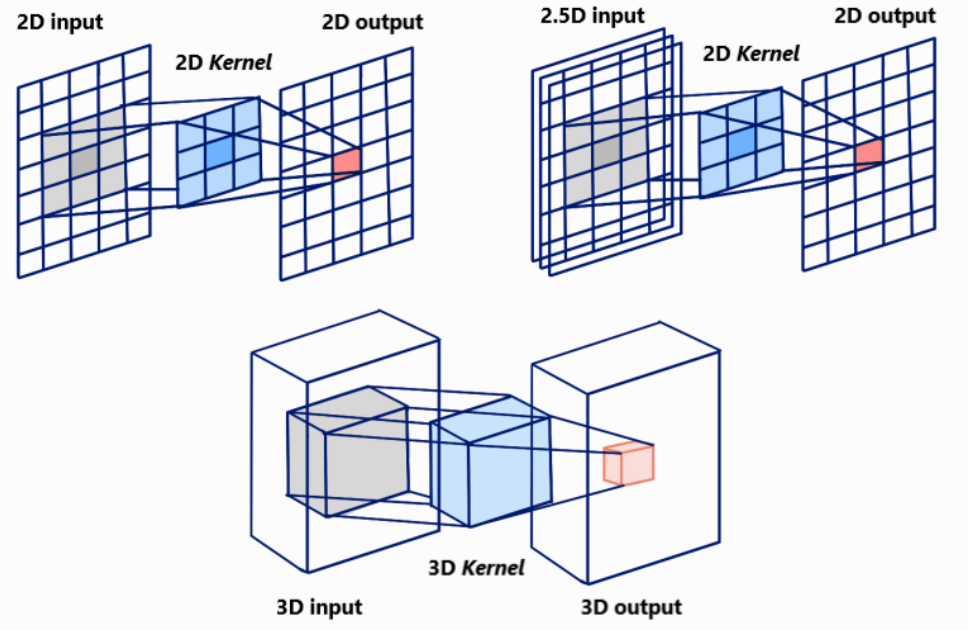


Figura 3.8: Esquemas con las distintas dimensionalidades de la red.

Capítulo 4

Filtrado de ruido con redes neuronales

Tal y como explicamos en capítulos anteriores, los tomogramas de material biológico suelen tener una baja relación señal-ruido, por lo que necesitan ser post-procesados para poder realizar una mejor interpretación [29]. Generalmente, este post-procesado intenta no afectar a la señal, empleando métodos sofisticados de filtrado como los presentados en la Sección 2.3 que consiguen reducir el ruido preservando los detalles estructurales [5].

El filtrado de ruido consiste en tratar una imagen ruidosa $x = s + n$ con el fin de separarla en dos componentes: su señal s y el ruido n que degrada la señal y queremos eliminar. En los últimos años, debido al auge del aprendizaje profundo en la inteligencia artificial se han desarrollado numerosas técnicas de filtrado con redes neuronales convolucionales (CNN). El uso más generalizado de estos sistemas implica el entrenamiento de modelos utilizando pares (x, s) de imágenes de entrada ruidosas x y sus respectivas imágenes de destino limpias s, conocidos como *content-aware image restoration* (CARE)[30]. Los parámetros de la red se ajustan de manera que minimicen una métrica de error (la función de pérdida) entre las predicciones de la red, y, y salida deseada, s. Cuando no se dispone de imágenes "limpias", estos métodos no pueden entrenarse, y, por tanto, resultan inútiles para la tarea de eliminación de ruido. En el contexto de tomografías obtenidas mediante SEM o TEM, la falta de una imagen sin ruido como referencia plantea una dificultad adicional en su filtrado. Para abordar esta limitación, se han desarrollado diversas estrategias que permiten entrenar redes neuronales incluso en ausencia de imágenes limpias. Estudios recientes han demostrado que estos enfoques alternativos pueden

producir resultados prometedores en la eliminación de ruido, lo que amplía las posibilidades de utilización de redes neuronales en este contexto [31].

En este capítulo se abordarán varios métodos de filtrado de ruido, como el enfoque *Noise to Noise* [32] y *Noise to Void* [31]. Se explicarán las diferentes arquitecturas de redes neuronales utilizadas y sus ventajas particulares. A continuación, se presentarán los experimentos y simulaciones llevadas a cabo en un phantom sintético, donde se dispone de la imagen "limpia" como referencia, lo que permitirá una comparación cuantitativa de la calidad del filtrado obtenido. Además, se describirá una estrategia intermedia denominada "Chess N2N", en la cual se generarán dos imágenes ruidosas a partir de un solo volumen. Por último, se aplicarán estos métodos a tomografías reales de células, con el objetivo de evaluar su desempeño en un contexto más complejo. Las redes neuronales utilizadas en este trabajo fueron generadas utilizando el software Dragonfly, Versión 2022.2 para Windows [33], explicado brevemente en el Apéndice D.

4.1. Métodos y arquitecturas de redes neuronales para el filtrado de ruido

Trataremos a la imagen o volumen $x = s + n$ como una extracción de la distribución conjunta[1] y $p(s)$ una distribución arbitraria [2] $p(s, n) = p(s)p(n|s)$, de manera que los píxeles de la señal s_i, no son estadísticamente independientes,

$$p(s_i|s_j) \neq p(s_i) \ . \tag{4.1}$$

Con respecto al ruido n, suponemos una distribución condicional de la forma $p(n|s) = \prod_i p(n_i|s_i)$, es decir, los valores de los píxeles n_i del ruido son condicionalmente independientes a la señal [31]. Además, suponemos que el ruido tiene media cero (en el caso del volumen *phantom* especificado en la Ec. 4.8)

$$\mathbb{E}[n_i] = 0 \ , \text{ y por tanto } \quad \mathbb{E}[x_i] = s_i \ . \tag{4.2}$$

En estrategias como CARE, las redes neuronales convolucionales se han entrenado para

[1] En probabilidad, dados dos eventos aleatorios X y Y, la distribución conjunta de X e Y es la distribución de probabilidad de la intersección de eventos de X e Y (ambos eventos ocurriendo de forma simultánea) [34].

[2] Una distribución arbitraria, cuyos valores y probabilidades son especificados por el usuario, por ejemplo, una distribución Gaussiana con una media y una desviación estándar especificadas.

predecir imágenes sin ruido a partir de *patchs* de imágenes ruidosas, es decir, a partir del campo receptivo de ese píxel $x_{RF(i)}$, dando como salida la predicción $\hat{s}_i$ de un único píxel i localizado generalmente en el centro del *patch* [32]. La mayoría de estos sistemas de regresión requieren el entrenamiento con pares (x_j, s_j) de imágenes de entrada ruidosas x_j y sus respectivas imágenes de destino limpias s_j. A continuación, los parámetros de la red $\boldsymbol{\theta}$ se ajustan para minimizar la función de pérdida L, entre las predicciones de la red $\hat{s}_i$ y la imagen o volumen limpios s_i [31], minimizando el riesgo empírico [3]

$$\underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_i L(f(x_i; \boldsymbol{\theta}) = \hat{s}_i, s_i) \ . \tag{4.3}$$

Los métodos de filtrado que se presentan a continuación no requieren el uso de imágenes limpias y se basan en las asunciones establecidas en las Ec. 4.1 y 4.2. Aunque estos métodos generalmente no alcanzan la misma calidad que los enfoques que utilizan pares de imágenes limpias, son útiles en situaciones donde no se cuenta con un *ground truth* para el entrenamiento.

4.1.1. Noise to noise (N2N)

En lugar de entrenar la red para asignar entradas ruidosas a imágenes limpias, el entrenamiento *Noise to Noise* (N2N) intenta aprender una correspondencia entre pares de versiones degradadas de la misma imagen, es decir $(s+n, s+n')$, que incorporan la misma señal s, pero ruido n y n' diferentes. Se ha demostrado que las redes entrenadas con este método pueden generar predicciones que convergen a las mismos resultados que las redes entrenadas tradicionalmente con acceso a imágenes limpias [32].

Una estrategia habitual para realizar una mejor predicción $\hat{s}$ es encontrar los parámetros θ que den lugar a la menor desviación media $\mathbb{E}$ de las etiquetas s según alguna función de pérdida L

$$\arg\min_{\boldsymbol{\theta}} \mathbb{E}_{(x,s)}\{L(f(x; \boldsymbol{\theta}) = \hat{s}, s)\} \tag{4.4}$$

[3] La minimización del riesgo empírico se refiere al proceso de encontrar los valores óptimos para los parámetros $\boldsymbol{\theta}$ de un modelo de aprendizaje automático, de modo que se minimice la suma de las funciones de pérdida L para todas las muestras de entrenamiento.

realizando un par de transformaciones basadas en la ley de la expectativa total [4]

$$\arg\min_{\boldsymbol{\theta}} \mathbb{E}_x\{\mathbb{E}_{x|s}\{L(f(x;\boldsymbol{\theta}) = \hat{s}, s)\}\} \,, \tag{4.5}$$

donde si $L = L^2(\hat{s}, s) = (\hat{s} - s)^2$.

Haciendo uso de una propiedad inherente a la minimización del error cuadrático medio (L^2), podemos afirmar que la estimación permanece inalterada al sustituir los objetivos s por números aleatorios cuyas medias coincidan con los objetivos. Por ejemplo, si tenemos que encontrar un valor z que resuelva $\arg\min_z \mathbb{E}_y\{L(z, y)\}$, el mínimo se encuentra para $z = \mathbb{E}_y\{y\}$ y se cumple independientemente de la distribución de la que se extraigan los y. Por consiguiente, los parámetros óptimos de la red de la ecuación 4.5 tampoco cambian si las distribuciones de objetivos condicionadas por la entrada $p(s|x)$ se sustituyen por distribuciones arbitrarias que tengan los mismos valores esperados condicionales $\mathbb{E}[x'] = s$ [32]. Esto implica que, en principio, podemos entrenar a la red con objetivos de entrenamiento con ruido de media cero sin alterar su aprendizaje. Basándonos en estas afirmaciones, se obtiene la minimización de la función de pérdida en el contexto específico del enfoque N2N reescribiendo la ecuación 4.3 de la siguiente manera

$$\arg\min_{\boldsymbol{\theta}} \sum_i L(f(x_{RF(i)};\boldsymbol{\theta}) = \hat{s_i}, x_i') \,, \tag{4.6}$$

donde tanto las entradas como los objetivos se extraen ahora de una distribución corrupta diferente, condicionada al objetivo limpio subyacente no observado s_i, de forma que $\mathbb{E}\{x_i'|x_{RF(i)}\} = s_i$. Con datos infinitos, se logran predecir valores de la misma manera que en 4.3. Para datos finitos no podemos afirmar que esto sea cierto debido a la presencia de varianza. La varianza representa la variabilidad de una serie de datos con respecto a su media y afecta la capacidad del modelo para hacer predicciones precisas. La varianza media del ruido en la imagen se divide por el número de muestras de entrenamiento N para obtener una estimación de la varianza promedio por muestra, $\frac{1}{N}\left[\frac{1}{N}\sum_i Var(s_i)\right]$ [32].

La idea fundamental de este método se sustenta en la suposición de que, dado que la señal es idéntica en ambas imágenes pero no contiene información común sobre el ruido que se espera que mapee, se le asigna a la red una tarea imposible. Como resultado, la mejor solución que encuentra la red es quedarse únicamente con la señal.

[4] ley de la expectativa total: $\mathbb{E}(Y) = \mathbb{E}(\mathbb{E}(Y|X))$

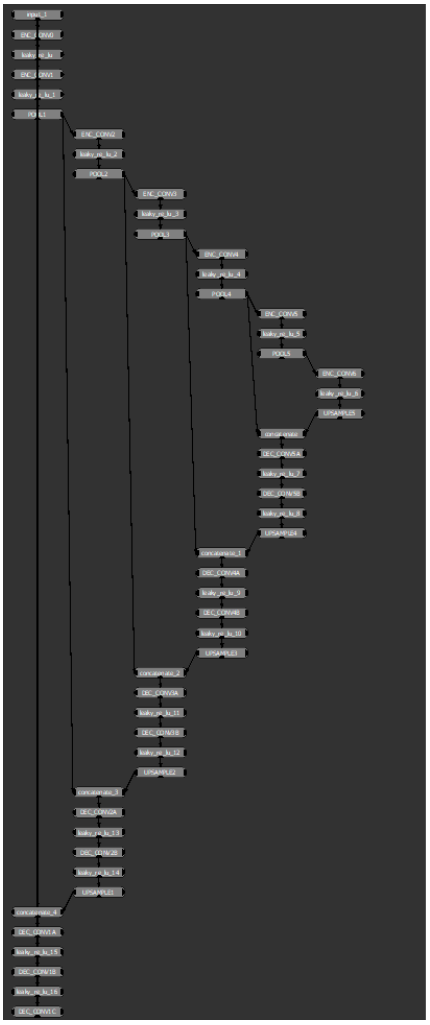
Red empleada

La Figura 4.1 representa la estructura de la red U-net ([1]), la cual se utilizó como modelo de regresión en este estudio. Cabe destacar que esta arquitectura es casi idéntica a la presentada en el artículo original de N2N [32]. La U-net fue implementada utilizando el software DragonFly 2022.2 [33]. En todos los experimentos de eliminación de ruido, tanto en el caso del phantom como en los tomogramas reales, se empleó un solo canal de entrada y un solo canal de salida ($n = m = 1$), ya que se trabajó con imágenes en escala de grises.

La red neuronal consta de cinco bloques de muestreo descendente y cinco bloques de muestreo ascendente con conexiones de salto. En total, la red neuronal cuenta con aproximadamente 950000 parámetros entrenables. Cada bloque descendente contiene una capa convolucional 2D (ENC-CONV), seguida de una activación LeakyReLU y una capa *MaxPooling* (POOL). Los bloques ascendentes constan de una capa *up-convolution* (UPSAMPLE), seguida de una concatenación con el correspondiente mapa de características, y dos capas convoluciones 2D (DEC-CONV), cada una seguida de función de activación LeakyReLU.

La arquitectura de la red se compone de varias capas con diferentes configuraciones. En las capas ENC-CONV y DEC-CONV, se utiliza un tamaño de kernel de $(3, 3)$ con un stride de $(1, 1)$ y una función de activación lineal. La red incluye 17 capas de activación LeakyReLu con $\alpha = 0{,}10$, que se sitúan después de cada capa ENC-CONV y DEC-CONV. Las capas de POOL tienen un tamaño de $(2, 2)$ y un stride de $(2, 2)$. Estas capas se encargan de reducir la resolución espacial de las características de entrada, seleccionando el valor máximo en cada región de $(2, 2)$ píxeles. Por otro lado, las capas de UPSAMPLE tienen un tamaño de $(2, 2)$ y utilizan la interpolación *nearest interpolacion* para replicar cada píxel de entrada en una matriz de salida de 2×2 píxeles.

Se utiliza el *padding* "SAME" cuando el *stride* es de 1, lo cual asegura que las salidas de las capas tengan las mismas dimensiones espaciales que las entradas. Este tipo de *padding* se emplea en las capas de convolución y deconvolución. Sin embargo, en las capas de POOL se utiliza el *padding* "VALID", lo que implica que no se añade ningún *padding* y la salida tendrá dimensiones reducidas debido al *stride* aplicado.



NAME	N_{out}	FUNCTION
INPUT	n	
ENC_CONV0	48	Convolution 3×3
ENC_CONV1	48	Convolution 3×3
POOL1	48	Maxpool 2×2
ENC_CONV2	48	Convolution 3×3
POOL2	48	Maxpool 2×2
ENC_CONV3	48	Convolution 3×3
POOL3	48	Maxpool 2×2
ENC_CONV4	48	Convolution 3×3
POOL4	48	Maxpool 2×2
ENC_CONV5	48	Convolution 3×3
POOL5	48	Maxpool 2×2
ENC_CONV6	48	Convolution 3×3
UPSAMPLE5	48	Upsample 2×2
CONCAT5	96	Concatenate output of POOL4
DEC_CONV5A	96	Convolution 3×3
DEC_CONV5B	96	Convolution 3×3
UPSAMPLE4	96	Upsample 2×2
CONCAT4	144	Concatenate output of POOL3
DEC_CONV4A	96	Convolution 3×3
DEC_CONV4B	96	Convolution 3×3
UPSAMPLE3	96	Upsample 2×2
CONCAT3	144	Concatenate output of POOL2
DEC_CONV3A	96	Convolution 3×3
DEC_CONV3B	96	Convolution 3×3
UPSAMPLE2	96	Upsample 2×2
CONCAT2	144	Concatenate output of POOL1
DEC_CONV2A	96	Convolution 3×3
DEC_CONV2B	96	Convolution 3×3
UPSAMPLE1	96	Upsample 2×2
CONCAT1	$96+n$	Concatenate INPUT
DEC_CONV1A	64	Convolution 3×3
DEC_CONV1B	32	Convolution 3×3
DEV_CONV1C	m	Convolution 3×3, linear act.

Figura 4.1: Tabla con la descripción de las capas de la red obtenida del artículo [32] y red utilizada en el método N2N creada haciendo uso del software [33]

4.1.2. Noise to Void (N2V)

Noise to Void (N2V) es un esquema de entrenamiento que no requiere pares de imágenes ruidosas ni imágenes objetivo limpias. Este planteamiento surge ante la necesidad de poder filtrar haciendo uso de un solo volumen o imagen debido a las dificultades que presenta en ocasiones obtener dos imágenes ruidosas con un nivel de ruido similar. Este método se basa en la suposición de que la señal exhibe cierto grado de predictibilidad al examinar su entorno, lo que implica la existencia de alguna estructura en la señal.

La idea sencilla en la que se basa N2V es utilizar un *patch* como entrada, siendo su píxel central el objetivo de predicción. Si mapeásemos directamente el valor del centro del *patch* de entrada a la salida, la red aprendería la operación identidad. Para evitar esta situación,

debemos utilizar un campo receptivo especial $\hat{x}_{RF(i)}$, suponiendo por ejemplo que este tiene un punto ciego en su centro. Como ya explicamos anteriormente, los métodos de eliminación de ruido suelen basarse en el supuesto de que los valores de los píxeles asociados a la señal s no son estadísticamente independientes. En otras palabras, la observación del contexto de la imagen de un píxel no observado podría permitirnos hacer predicciones sensatas sobre la intensidad del píxel. Por tanto, la predicción de la Unet se ve afectada por todos los píxeles de entrada en una vecindad especifica excepto por el píxel de entrada x_i en su misma ubicación. Denominamos a este tipo de red punto ciego o *blindspot* [31].

Una red con puntos ciegos puede entrenarse utilizando tanto un entrenamiento tradicional donde se disponen de imágenes limpias para comparar su salida como con N2N, utilizando imágenes ruidosas. La red *blindspot* cuenta con menos información para realizar sus predicciones, por lo que cabe esperar que su precisión sea menor en comparación con una red N2N o CARE. Sin embargo, teniendo en cuenta la pequeña fracción de píxeles eliminados de todo el campo receptivo, podemos suponer que sigue funcionando razonablemente bien, manteniendo la ventaja esencial de esta arquitectura, su incapacidad para aprender la función identidad.

$$\arg\min_{\boldsymbol{\theta}} \sum_{i} L(f(\hat{x}_{RF(i)};\boldsymbol{\theta}) = \hat{s}_i, x_{RF(i)}) \tag{4.7}$$

La red N2V creada con DragonFly [33] contará con 4 bloques de muestreo descendente y ascendente. En esta estrategia en concreto, DragonFly no ofrece la misma libertad de manipulación que en la arquitectura N2N. Dada una imagen de entrenamiento ruidosa x_i, extraemos aleatoriamente parches de tamaño 64×64 píxeles, que son mayores que el campo receptivo de nuestras redes. Dentro de cada parche seleccionamos aleatoriamente N píxeles, utilizando un muestreo estratificado para evitar la agrupación, tal y como se puede ver en la Figura 4.2. A continuación, enmascaramos estos píxeles y utilizamos los valores de entrada ruidosos originales como objetivos en su posición. Calculamos los gradientes de todos ellos ignorando el resto de las predicciones, de esta manera, no se utilizan todos los píxeles para calcular la función de pérdida, sino simplemente los seleccionados de manera aleatoria.

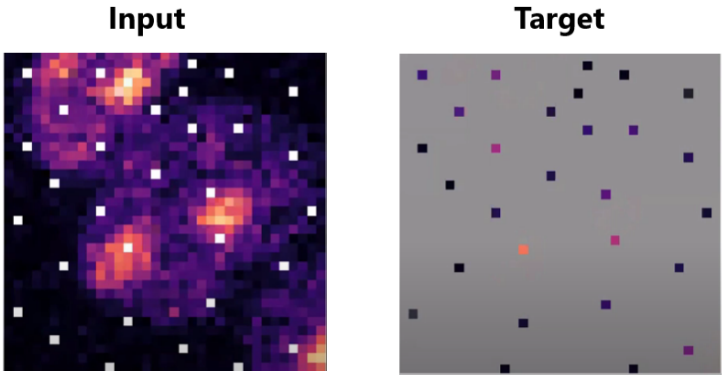


Figura 4.2: Ejemplo de la enmascaración aleatoria de N píxeles en un *patch* y su posterior uso como *target*. Imagen obtenida de Academy@Home webinar `https://www.youtube.com/watch?v=ipp0mxfjhwY&t=1322s`

4.2. Prueba inicial con una imagen sintética: Phantom

Para la evaluación objetiva de los métodos de filtrado de ruido, se empleó un volumen sintético (*phantom*) que simula un entorno celular y que contiene componentes representativos observados en estudios por microscopía electrónica 3D (3DEM), tales como estructuras con membranas, filamentos y complejos macromoleculares (ribosomas, en particular). Este phantom fue diseñado en el contexto de estudios previos del grupo [35]. A partir de los distintos volúmenes sintéticos de 512×512 píxeles y 51 *slices* con distintos niveles de ruido gaussiano, entrenaremos y estudiaremos el desempeño de distintas redes y cómo varían los resultados finales obtenidos en función de los hiperparámetros. Esta imagen sintética proporciona una oportunidad para realizar un estudio más eficiente y explorar diferentes parámetros con mayor flexibilidad. En comparación con las imágenes de tomografía electrónica, esta imagen tiene una menor carga computacional, lo que permite tiempos de entrenamiento más cortos, generalmente menos de 10 minutos para entrenar 10 épocas, y la aplicación de los filtros es casi instantánea. Se han generado 10 versiones del volumen del phantom con 5 niveles de ruido distintos en relación SNR (signal-to-noise ratio) de 0,1, 0,25, 0,5, 1,0 y 2,0. Para poder aplicar la estrategia N2N, se han creado dos versiones con el mismo nivel de ruido distribuido diferentemente. Para crear estas versiones ruidosas se empleó ruido Gaussiano de media 0 y desviación típica acorde con la SNR deseada, siguiendo la definición [36]:

$$\mathrm{SNR} = \frac{\mathrm{Potencia}_{\mathrm{phantom}}}{\mathrm{Potencia}_{\mathrm{noise}}} = \frac{\mathbb{E}[S^2]}{\mathbb{E}[\mathrm{N}^2]} = \frac{\mu_S^2 + \sigma_S^2}{\sigma_N^2}\,, \tag{4.8}$$

donde $\mathbb{E}[\cdot]$ representa la esperanza, S el phantom y N el ruido Gaussiano de media 0. Para generar el ruido se empleó Spider[37], un software clásico del campo de la 3DEM. A continuación se muestra una *slice* del *phantom* con distintos niveles de ruido.

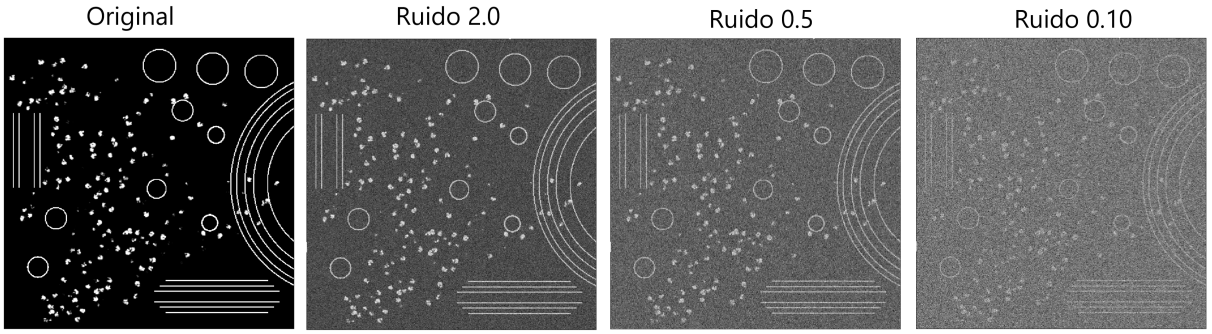


Figura 4.3: Slide 22/51 del Phantom: original, ruido gaussiano 2.0, 0.5 y 0.10

Las estrategias de filtrado *Noise to Noise* y el *Noise to Void* son de especial interés en la aplicación a datos de imágenes biomédicas, donde la adquisición de blancos (*targets*) de entrenamiento, limpios o ruidosos, a menudo no es posible.

El análisis de las curvas de entrenamiento en estas estrategias puede proporcionar información sobre el progreso y la convergencia del modelo durante el proceso de entrenamiento, ayudando a identificar si el modelo está aprendiendo correctamente y si hay mejoras significativas a medida que avanza el entrenamiento. No obstante, en las estrategias N2N y N2V las curvas de entrenamiento tienden a estabilizarse en una zona de plateau donde realizan pequeñas fluctuaciones. Esto es debido a que el modelo no puede predecir el ruido de manera precisa en comparación con la imagen con la que se está comparando. En consecuencia, es posible que el modelo alcance un límite en la reducción del error relacionado con el ruido, indicando que el modelo ha alcanzado su capacidad máxima para mejorar la señal subyacente y no puede reducir aún más el ruido. Esto dificulta la detección de un proceso de sobreajuste mediante el estudio de curvas de entrenamiento. Por lo tanto, las curvas de entrenamiento se utilizarán únicamente para monitorear el progreso del modelo por parte del responsable del entrenamiento y no se incluirán en este capítulo. La métrica que se utilizará para evaluar cada una de las estrategias seguidas en el filtrado de los volúmenes del phantom será el coeficiente de de Correlación Pearson, parámetro utilizado para comprobar y cuantificar la relación entre dos variables. El Coeficiente de Correlación es una potente métrica para medir la información estructural compartida entre los volúmenes *ground truth* y filtrado de forma independiente de la intensidad (más información en el Apéndice E). A la hora de interpretar los valores del coeficiente de correlación de Pearson

se establece que un valor entre 0 y $0,1$ es una correlación inexistente, mientras que valores entre $0,5$ y $1,0$ reflejan una fuerte correlación.

Es importante tener en cuenta que los resultados cuantitativos obtenidos para este volumen sintético no pueden extrapolarse directamente a tomogramas reales. Esto se debe a que ciertos hiperparámetros están fuertemente vinculados al número de *slices* y píxeles del tomograma. Sin embargo, estos resultados nos brindarán una idea de cómo afecta la variación de dichos hiperparámetros durante el entrenamiento.

4.2.1. Entrenamiento y resultados N2N

En el entrenamiento de las redes correspondientes a la Figura 4.1 utilizadas en este estudio, se han tomado en consideración los hiperparámetros basados en investigaciones y publicaciones previas, como los utilizados en [32]. En la Tabla 4.1 se presentan algunos de los parámetros más relevantes que se han mantenido constantes a lo largo de los experimentos

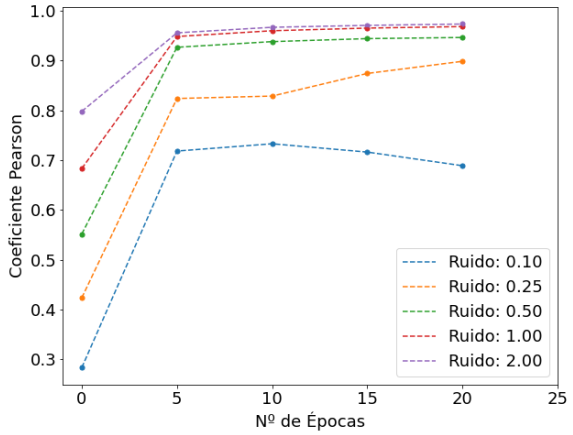
Hiperparámetro	**Elección**
Batch size	32
Patch size	[64, 64, 1]
Stride ratio	1.0
Loss function	MeanSquaredError
Optimization algorithm	Adadelta

Tabla 4.1: Hiperparámetros fijos en los experimentos *Noise to Noise*.

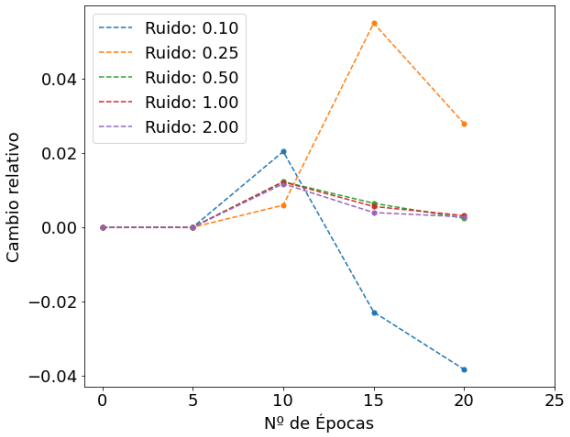
Cada red será entrenada utilizando 2 volúmenes del Phantom con el mismo nivel de ruido pero distribuido de manera diferente. Se utilizará el 80 % de los volúmenes como datos de entrenamiento, mientras que el 20 % restante se utilizará como datos de validación. Los datos de validación no tendrán influencia directa en el ajuste de pesos llevado a cabo por el optimizador, sino que nos servirán de guía para evaluar de manera más objetiva los resultados.

Vamos a investigar cómo el número de épocas afecta el rendimiento de la red neuronal cuando se entrena con el 80 % del tomograma sintético completo, considerando los cinco niveles de ruido. Para esto, evaluaremos el coeficiente de Pearson y el cambio relativo de este coeficiente entre las épocas, lo que nos permitirá cuantificar de manera más precisa la mejora en el entrenamiento. Además, utilizaremos los parámetros comunes en la estrategia N2N, que incluyen el uso de dos

conjuntos de datos (2*DataSet*), una dimensión de kernel 2D y un factor de aumento de datos de 2 (2*data augmentation*), que implica la rotación horizontal y vertical del volumen.



(a) Mejora en la calidad de la imagen cuantificada con el Coeficiente de Pearson en función del número de épocas de entrenamiento para 5 niveles de ruido distintos.



(b) Cambio relativo en el Coef. Pearson en función del número de épocas para 5 niveles de ruido distintos.

Figura 4.4: Resultados obtenidos con la red N2N en el filtrado del phantom tras entrenarla distintas épocas.

Los estudios mostrados son el resultado de aplicar el filtro final de la red al volumen total, utilizando tanto en el porcentaje utilizado en el entrenamiento como el reservado para validación, por lo que una parte de las *slices* con las que se ha calculado el coeficiente de Pearson han sido utilizadas en el propio ajuste de los pesos (no hay una clara diferenciación en el conjunto de entrenamiento, validación y test). Como se puede deducir al observar las Figuras 4.4a y 4.4b, en estos volúmenes, podemos concluir que en 10 épocas de entrenamiento se obtienen los mejores resultados de manera general, a excepción del ruido $\sigma = 0{,}25$, donde entrenar 5 épocas más da lugar a una mejora notable en el coef. de Pearson. No obstante, para poder hacer un estudio lo más generalista posible, el resto de resultados presentados se realizarán con 10 épocas para todos los niveles de ruido.

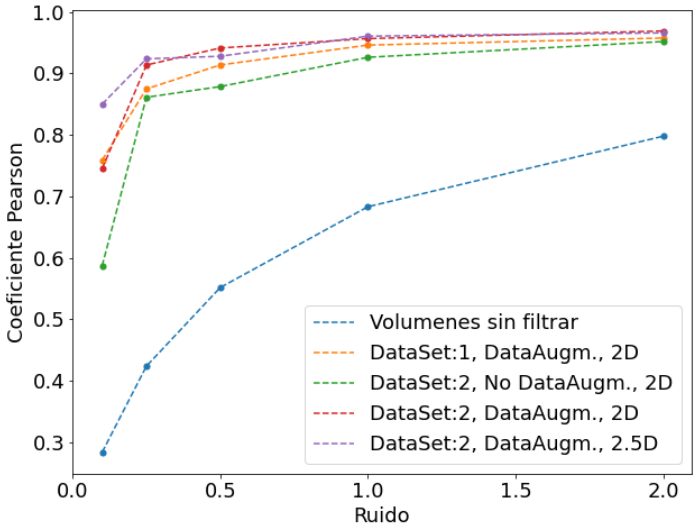
A continuación, mostraremos el estudio realizado variando el resto de hiperparámetros que fijamos anteriormente:

- *Data Augmentation*: En caso de utilizarlo, aumentaremos un factor 2, rotación del volumen horizontal y vertical. Como se explicó en la Sección 3.7.1, el aumento de datos se espera que proporcione a la red una mayor robustez en propiedades de simetría, aparte de aumentar

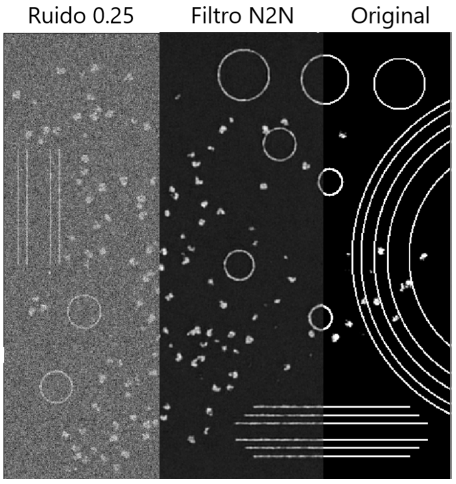
las muestras de datos de entrenamiento en caso de tener un volumen pequeño.

- *Data Set*: Dado que se trata de un entrenamiento N2N, tenemos 2 volúmenes con el mismo nivel de ruido y podemos utilizarlos como entrada y salida indistintamente, pudiendo duplicar los datos de entrenamiento.

- 2.5D: puede entenderse como una combinación de convoluciones 2D y 3D, explicado en la Sección 3.7.1. Se especifica el número de *slices* que se desea, en este caso 3.



(a) Coeficiente de Pearson obtenido en las distintas elecciones de hiperparámetros para cada nivel de ruido.



(b) Secciones de la *slice* 22 del volumen Phanton, donde se puede observar la imagen ruidosa de partida con $\sigma = 0{,}25$, el filtro N2N (con 2.5D, 2 DataSet y Data Augmentation) y el phantom original.

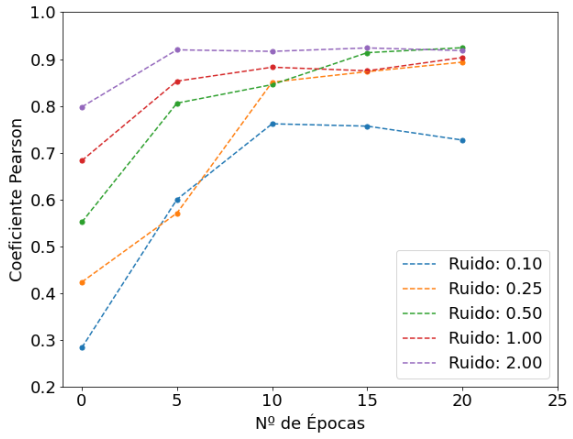
Figura 4.5: Resultados obtenidos con la red N2N en el filtrado del phantom tras varias ciertos hiperparámetros.

Como cabría esperar, añadir *data augmentation* y *data sets* al entrenamiento mejora los resultados obtenidos, ayudando a mejorar la capacidad de generalización y rendimiento del modelo, al proporcionar una mayor diversidad de datos y reducir el riesgo de sobreajuste. La aplicación del *kernel* 2.5D también mejora ligeramente el valor del Coeficiente de Pearson, especialmente en niveles de ruido más altos. Sin embargo, considerando el aumento en el tiempo computacional y la mejora relativa en comparación con otros hiperparámetros, en este volumen en particular no sería una estrategia de interés. Se observa que un mayor contexto de las *slices* vecinas no aporta información tan relevante en el filtrado de ruido como lo haría en otras tareas como la segmentación.

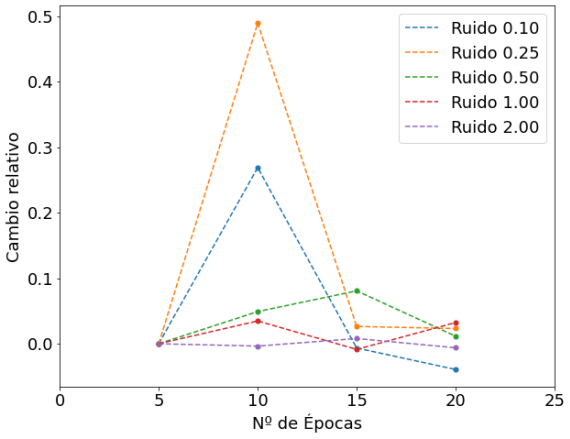
4.2.2. Entrenamiento y resultados N2V

Utilizaremos el *batch size* y *patch size* especificados en la Tabla 4.1, con un píxel ratio de 0,20 %, es decir, el 0,20 % de los píxeles de un *patch* se enmascaran y se toman como *blind-spots*. De manera similar al apartado anterior, mostraremos un estudio del número de épocas y de los hiperparámetros a elegir, teniendo en cuenta que en este caso debemos suponer que solamente tenemos un volumen de entrenamiento.

A diferencia del estudio anterior, los resultados mostrados corresponden a la aplicación del filtro sobre volúmenes que no hemos utilizado en el entrenamiento [5], proporcionando unos resultados ligeramente más objetivos.



(a) Mejora en la calidad de la imagen cuantificada con el Coeficiente de Pearson en función del número de épocas de entrenamiento para 5 niveles de ruido distintos.



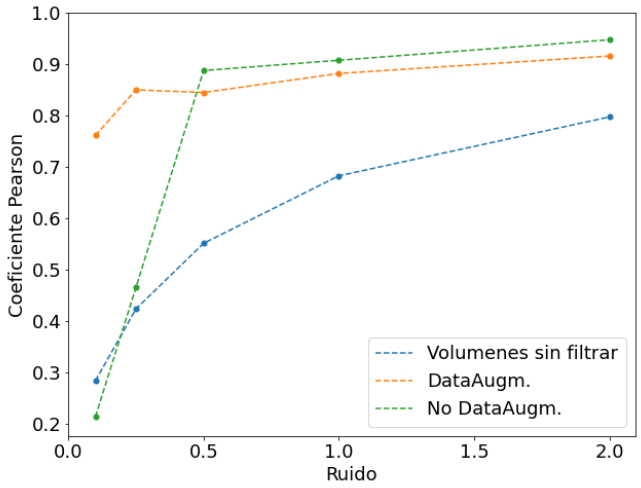
(b) Cambio relativo en el Coef. Pearson en función del número de épocas para 5 niveles de ruido distintos.

Figura 4.6: Resultados obtenidos con la red N2V en el filtrado del phantom tras entrenarla distintas épocas.

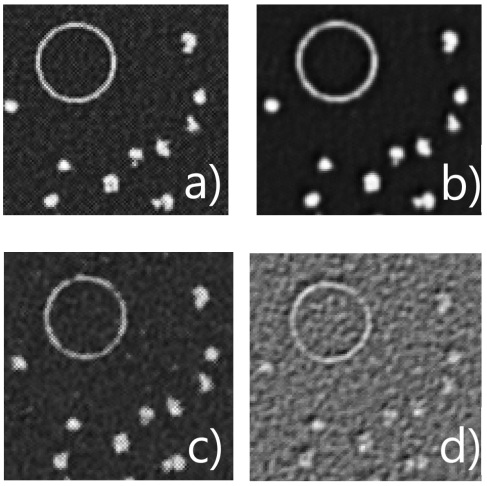
En comparación con la gráfica 4.4a, se puede observar que los valores del coeficiente de Pearson obtenidos en este caso son considerablemente más bajos, lo que indica una correlación menor entre el phantom original y el volumen filtrado. Es interesante destacar que los mejores resultados se obtienen generalmente con 10 épocas de entrenamiento, llegando incluso a empeorar la calidad del filtrado en algunos niveles de ruido.

A continuación, mostraremos el estudio realizado al aumentar de datos de entrenamiento durante 10 épocas con dimensión de kernel 2D.

[5] Debido a que contamos con 2 volúmenes con el mismo nivel de ruido distribuido de manera diferente



(a) Coeficiente de Pearson obtenido en la aplicación de *data augmentation* para cada nivel de ruido.



(b) Zoom 85×85 píxeles del phantom con vesícula (estructura circular) y ribosomas (motas). Las imágenes a) y b) corresponden al volumen $\sigma = 1{,}0$ con *data augm.* y sin *data augm.* respectivamente. c) y d) corresponden al volumen $\sigma = 0{,}25$ con *data augm.* y sin *data augm.* respectivamente.

Figura 4.7: Estudio de la influencia del aumento de datos en el entrenamiento de la red N2V para el filtrado del phantom.

A pesar de lo que se podría esperar en relación con los resultados obtenidos en N2N, Figura 4.5a, en este caso, *data augmentation* para los volúmenes menos ruidosos no resulta beneficioso. Esto es debido a que el entrenamiento con puntos ciegos puede dar lugar a artefactos que denominaremos "tablero de ajedrez" (reportados previamente en artículos como [38]) bastante visibles en la imagen a) y c) de la Figura 5.6b, lo que reduce considerablemente la calidad de las predicciones finales. Estos artefactos parecen aumentar con el número de épocas, y por tanto, con un aumento en los datos de entrenamiento, al realizar un mayor número de ajustes en los pesos.

4.2.3. Estrategia *Chess* N2N

Con el fin de poder obtener la calidad de filtrado propia del N2N a partir de un único volumen como en N2V, planteamos la posibilidad de crear dos volúmenes sintéticos a partir de un único tomograma, intentando minimizar la pérdida de información. Esto podría lograrse separando las *slices* pares e impares y agruparlas para crear dos nuevos tomogramas, provocando una perdida de información entre capas bastante significativa tal y como se puede ver en la siguiente Figura 4.8. Para evitar esta situación, se propone entremezclar por pares las *slices* del tomograma,

creando dos volúmenes.

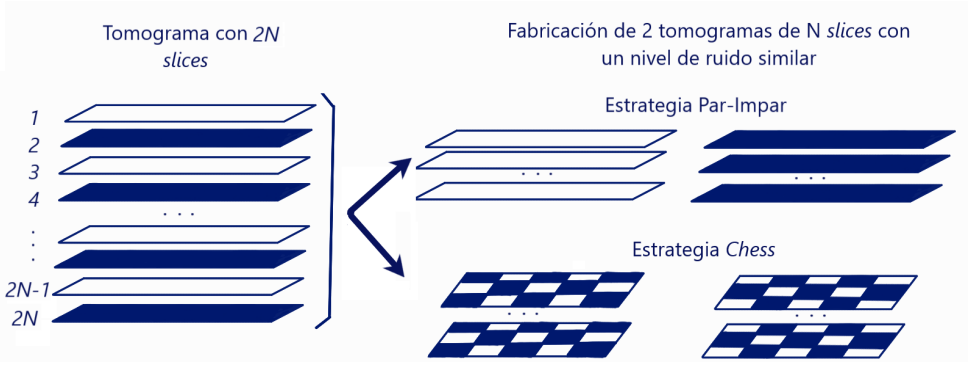


Figura 4.8: Creación de dos tomogramas con nivel de ruido similar para poder aplicar la estrategia N2N a partir de un único volumen. Esta estrategia recibirá el nombre de ajedrez o *chess*.

Una vez obtenidos los 2 tomogramas, se puede aplicar la estrategia N2N utilizando los parámetros que han demostrado un mejor rendimiento en estudios previos.

A continuación, se muestra una comparación de los coeficientes de Pearson obtenidos con los tres métodos estudiados hasta ahora, así como una comparación con un método clásico de filtrado como puede ser un filtro pasa baja de Fourier en 3D, explicado en el Apéndice A.

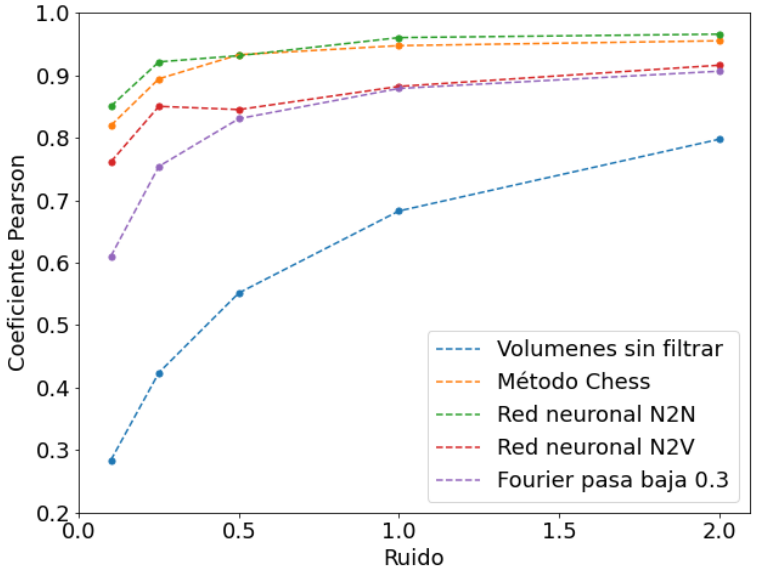


Figura 4.9: Estudio del Coeficiente de Pearson en función del nivel de ruido del volumen para cada una de las estrategias de filtrado explicadas. Se incluye el filtro de Fourier pasa baja con 0.3 para poder comparar con un método clásico (para más información ver Apéndice A).

Los resultados obtenidos de la comparación entre los diferentes métodos de filtrado mues-

tran que el enfoque N2N logra el mejor desempeño en términos de correlación, obteniendo los coeficientes de Pearson más altos. El método Chess también muestra resultados favorables en comparación con el método N2V y el enfoque clásico de Fourier y no presenta unos artefactos tan marcados como los presentados por N2V, tal y como se puede observar en la Figura 4.10.

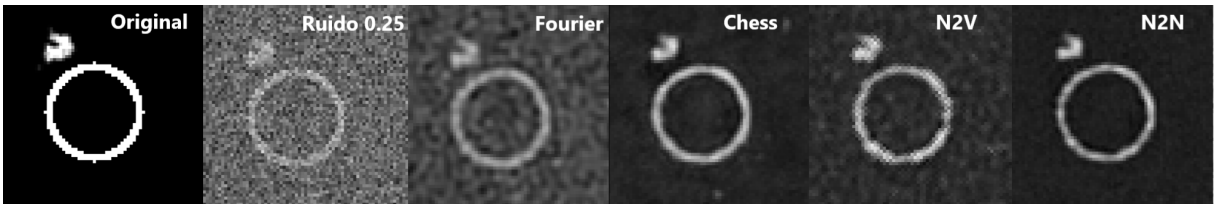


Figura 4.10: Zoom 65×65 píxeles para estudiar una región con una vesícula (estructura circular) y un ribosoma (mota) tras aplicar los distintos filtros.

En conclusión, el enfoque N2N y el método Chess demuestran ser las estrategias más efectivas para el filtrado de tomogramas ruidosos, ofreciendo resultados mejorados en comparación con el método N2V y Fourier.

4.2.4. Estudio de intensidades y morfología

Hasta ahora nos hemos centrado fundamentalmente en el estudio de la morfología de los elementos que aparecen en el tomograma, dejando de lado el efecto de estos filtros sobre la intensidad. El estudio de la intensidad en los tomogramas es de gran relevancia en ciertas aplicaciones, donde la variación de la transmisión de radiación o electrones es un parámetro fundamental. En el estudio de estructuras biológicas, el interés fundamental de los tomogramas reside en la morfología de la imagen y el uso de filtros trata de generar los contrastes correctos entre tejidos independientemente del nivel de intensidad. No obstante, merece la pena mencionar el efecto del filtrado con redes en la intensidad de la señal de cara a la aplicación de los mismos en ciertos estudios físicos.

Para una imagen en escala de grises o blanco y negro, generalmente tenemos valores de píxeles que van de 0 a 255. Como mencionamos en la Sección 3.5.2, las imágenes se han de procesar antes de introducirlas a la red, transformando los datos a la misma escala mediante un proceso de normalización y estandarización que realiza el programa de manera automática. El phantom utilizado en el estudio fue previamente normalizado a una escala de grises [0,1] antes de introducir el ruido de la Ec. 4.8, por lo que la escala de grises de los volúmenes ruidosos creados a partir de él oscilarán entre [-1, 2].

De la misma manera que se normalizan los datos de entrada a la red, los datos con los que se compara a la salida también están normalizados, por lo que la escala de la imagen final filtrada siempre oscilará entre los valores [0,1]. Mostraremos algunos resultados tras estudiar el perfil de intensidad, lo cual nos proporcionará información sobre la conservación o reescalado de la intensidad en los volúmenes filtrados. El estudio del perfil de intensidad se llevará a cabo con el *Software ImageJ* [39], una herramienta analítica utilizada a menudo en el procesamiento digital de imágenes. Para poder distinguir mejor las estructuras, se utilizará un perfil que refleje el promedio de 4 píxeles a lo ancho de la línea estudiada.

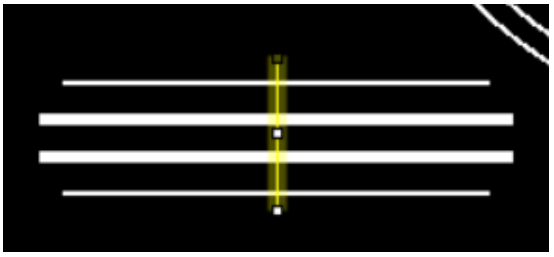


Figura 4.11: Captura de imagen 8/51 de la herramienta de perfil de línea sobre zoom de 230×100 píxeles, con un ancho de línea de 4 píxeles.

En la siguiente imagen se comparará el perfil de intensidad correspondiente a la zona del phantom reflejada en la Figura 4.11 para el volumen original, ruidoso con $\sigma = 0{,}25$ y volúmenes filtrados con las estrategias N2N, N2V y Fourier.

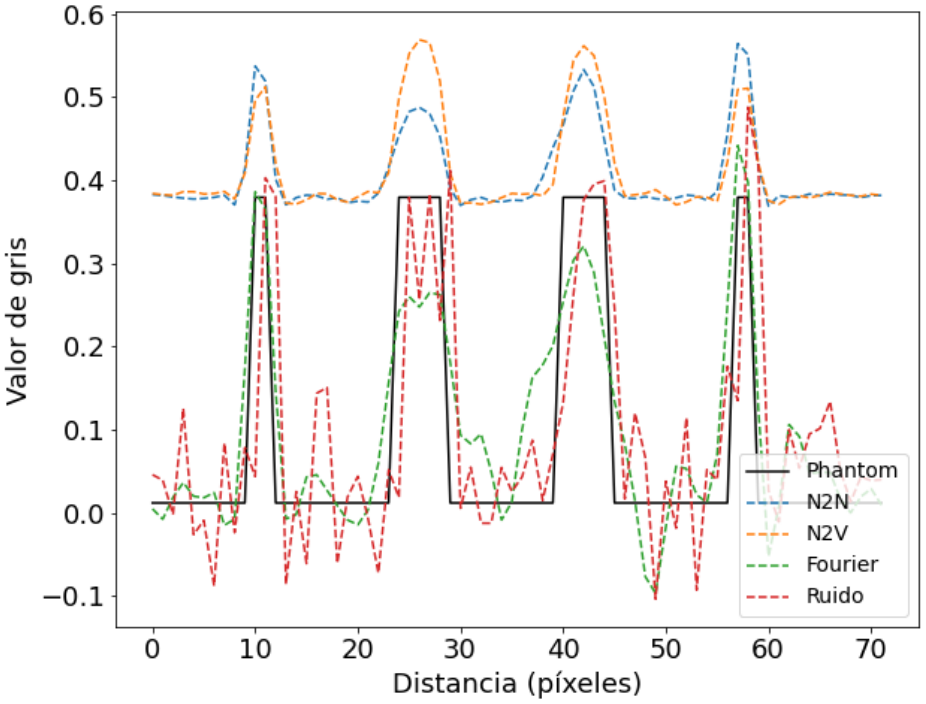


Figura 4.12: Estudio del perfil de intensidad de la estructura correspondiente a la Figura 4.11. Se muestra la variación del nivel de gris sobre el volumen con $\sigma = 0{,}25$ tras aplicarle diferentes filtros.

Se observa la conservación de la intensidad en el filtro clásico de Fourier. Sin embargo, los filtros correspondientes a las redes neuronales sufren un reescalado no lineal. Este reescalado se produce como parte del proceso de entrenamiento y transformación de los datos en la red. Sin embargo, en algunos casos, invertir este reescalado no es posible o no se logra de manera precisa.

En el caso específico de las redes creadas con el software Dragonfly, no se consigue un proceso de inversión del reescalado de manera efectiva. Esto puede generar un problema en estudios donde la intensidad de los datos sea relevante, ya que la información de intensidad original de las imágenes puede perderse o distorsionarse durante el proceso de filtrado y posterior inversión del reescalado.

4.3. Filtrado de tomogramas reales

La estrategia N2N puede ser utilizada en criotomografía electrónica. En esta técnica, durante la adquisición de una imagen TEM en realidad se toman múltiples imágenes (frames) que se alinean mutuamente para compensar el movimiento que sufre la muestra cuando es bombardeada con los electrones. El promedio de estos frames da lugar a la imagen TEM final. La distribución de frames en dos conjuntos independientes permite generar dos imágenes TEM finales con una calidad y ruido similar. En última instancia, esto da lugar al cálculo de 2 volúmenes 3D independientes adecuados para N2N. En cambio, para FIB-SEM, el método N2V o *Chess*N2N resulta más apropiado, debido a que en esta adquisición de datos no contamos con varios *frames*, sino con un barrido de la superficie del volumen.

Se presentaron los resultados del filtrado de tomogramas utilizando los métodos N2N, N2V y *Chess*N2N. Aunque no contamos con métricas de comparación cuantitativa, se realizó una evaluación visual de los tomogramas resultantes.

- Dataset de FIB/SEM, correspondiente a una región de hipocampo de cerebro de rata, con un tamaño de vóxel de 5 nm obtenido de la siguiente dirección [40]. Este dataset se emplea

a menudo como 'benchmark' para evaluar algoritmos de segmentación, en particular de mitocondrias. Los volúmenes de $1024 \times 768 \times 165$ píxeles cuentan con una con segmentación manual de mitocondrias con el etiquetado de cada píxel.Solo disponemos de un dataset, por lo que únicamente podrán filtrarse con la estrategia N2V o *Chess*N2N.

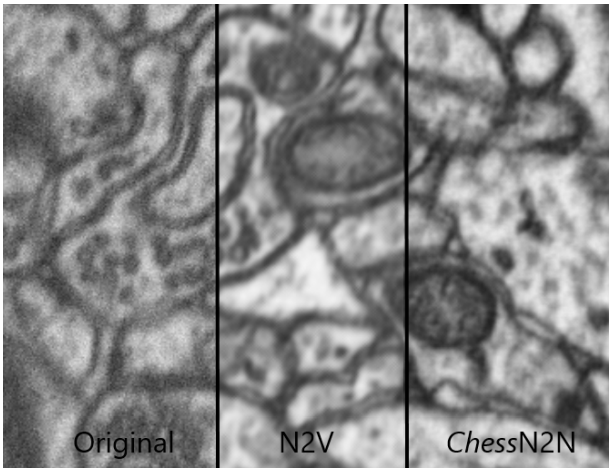


Figura 4.13: Tomografía FIB/SEM filtrada con N2V y*Chess* N2N.

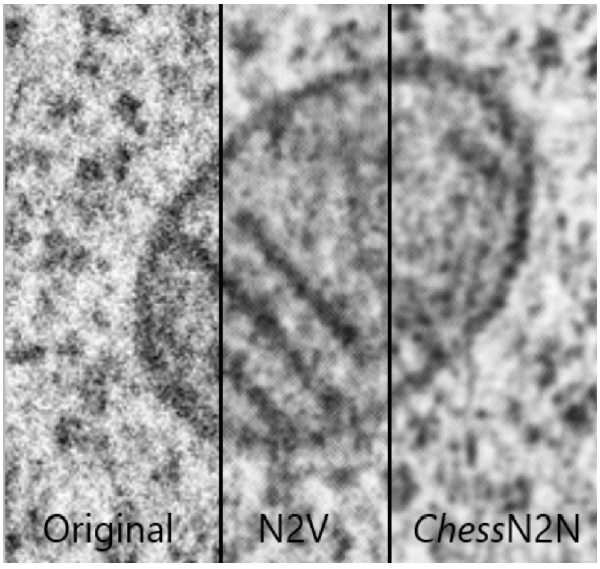


Figura 4.14: Tomografía FIB/SEM filtrada con N2V y*Chess* N2N.

- Dataset empiar10311: correspondiente al FIB/SEM de una célula HeLa, con un tamaño de voxel de 5 nm. Obtenido de la base de datos EMPIAR, con ID: 10311: [41] Solo disponemos de un dataset, por lo que únicamente podrán filtrarse con la estrategia N2V o *Chess*N2N. En la imagen ampliada de una mitocondria, se puede apreciar el resultado final de la aplicación de los filtros."

- Dataset empiar10164, correspondiente a la criotomografía electrónica de virus HIV, con un tamaño de voxel de 5.4 Angstroms. Obtenido de la base de datos EMPIAR, con ID: 10164 [42]

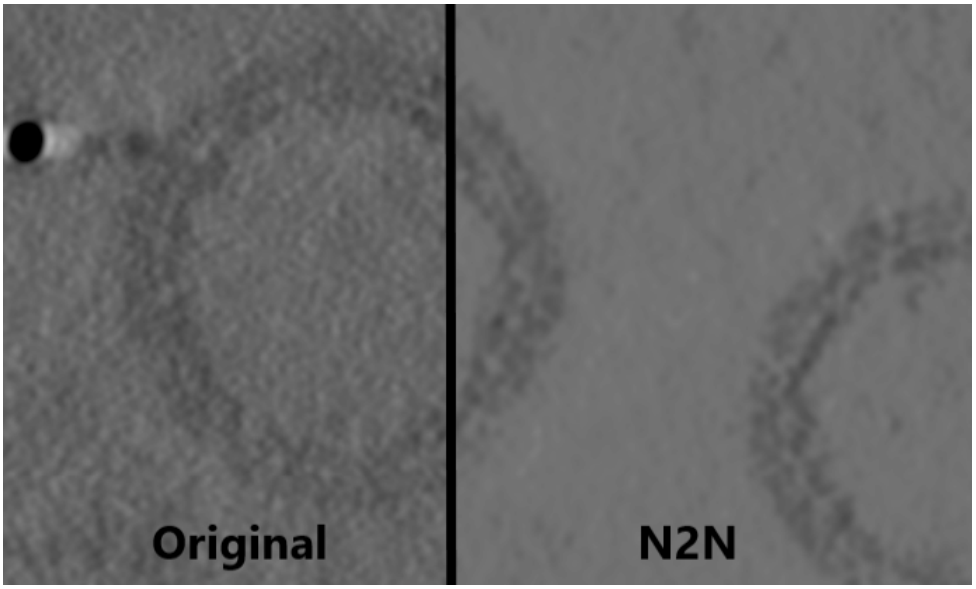


Figura 4.15: Tomografía CrioTEM filtrada con N2N. Se puede observar un marcador fiducial.

El entrenamiento, que constó de alrededor de 10 épocas y se realizó con los hiperparámetros mencionados anteriormente, tardó aproximadamente de 40 minutos a 1 hora. Por otro lado, la aplicación del filtrado a los tomogramas fue bastante rápida, requiriendo tan solo de 5 a 10 minutos.

El método N2N mostró una mayor capacidad de preservar la información estructural y reducir el ruido en los tomogramas. El método N2V también logró mejorar la calidad de los tomogramas, pero se observaron artefactos visuales característicos de este método, claramente visibles en la Figura 4.14. En cuanto al método ChessN2N, se obtuvo un resultado visualmente similar al método N2N en términos de reducción de ruido y preservación de las estructuras.

En comparación con el estudio que se llevó a cabo utilizando un phantom, el tomograma real presenta una mayor complejidad en términos de estructuras y detalles, así como una mayor variedad de intensidades/escala de grises. Esto puede hacer que los resultados obtenidos en el filtrado no sean visualmente tan impresionantes como los anteriores.

A pesar de estas diferencias, es importante tener en cuenta que el objetivo principal de este estudio era evaluar y comparar los diferentes métodos de filtrado en un escenario más realista. Aunque los resultados pueden no ser tan sobresalientes como en el estudio anterior, el análisis cuantitativo y la evaluación visual proporcionan información valiosa sobre el desempeño relativo de los métodos y sus limitaciones en un contexto más desafiante.

Capítulo 5

Segmentación con redes neuronales

La generación de modelos tridimensionales de células completas y sus orgánulos mediante tomografía electrónica ha enfrentado históricamente limitaciones, no solo en términos de la adquisición y análisis de imágenes, sino también debido al laborioso proceso de segmentación manual [43]. La segmentación manual resulta una tarea laboriosa y lenta, propensa a errores y difícil de generalizar. Debido a estas limitaciones, en las últimas décadas se ha realizado una amplia investigación en el uso de redes neuronales para la segmentación. Estas redes tienen la capacidad de automatizar y acelerar el proceso, reduciendo así la dependencia de la segmentación manual.

Existen dos tipos principales de segmentación de imágenes: la segmentación semántica y la segmentación por instancias. En la segmentación semántica, todos los objetos del mismo tipo se marcan con la misma etiqueta de clase, realizando una clasificación por píxeles. En la segmentación por instancias se identifican y separan objetos individuales dentro de una imagen, asignando una etiqueta única a cada objeto segmentado de la imagen [44]. En la Figura 5.1 podemos ver un ejemplo donde se realizan ambas tareas. Por un lado, en la segmentación semántica se han clasificado todas las personas con la misma etiqueta y la mesa con otra, representando cada etiqueta con un único color. En cambio, en la segmentación de instancias se hace una distinción entre personas mediante distintos colores.

Object Detection Semantic Segmentation Instance Segmentation

Figura 5.1: La fotografía central muestra una segmentación semántica aplicada a la imagen izquierda y la derecha una segmentación de instancias. Figura adaptada de [44]

En este capítulo, se presentarán y analizarán algunas de las técnicas más recientes en el campo de la segmentación semántica de imágenes y volúmenes mediante redes neuronales, mostrando los resultados obtenidos para diferentes configuraciones de redes y selección de hiperparámetros.

5.1. Arquitectura de red

En el presente apartado se explica la arquitectura propuesta de la U-net 2D creada con DragonFly [33] para la tarea de segmentación. La Figura 5.2 recoge la explicación de la red desglosada de las capas, mientras que la Tabla 5.1 contiene los parámetros más relevantes que se mantienen constantes en todos los experimentos.

Hiperparámetro	**Elección**
Batch size	32
Stride ratio	1.0
Loss function	CategoricalCrossentropy
Optimization algorithm	Adadelta

Tabla 5.1: Hiperparámetros fijos en los experimentos de segmentación.

La función de coste empleada, *CategoricalCrossentropy*, se utiliza como función de pérdida para modelos de clasificación multiclase en los que hay dos o más etiquetas de salida. A cada etiqueta de salida se le asigna un valor de codificación de categoría en forma de 0 (si no pertenece a esa categoría) y 1 (si pertenece a la categoría).

A diferencia de la Tabla 4.1, no hemos especificado el *Patch size*, ya que se variará en función de la segmentación realizada. En las redes empleadas en segmentación, el tamaño del *patch* cobra especial relevancia, debido a que el tamaño de la subimagen de entrada debe de

ser lo suficientemente grande como para recoger características que permitan asignar la etiqueta correcta, sin comprometer la memoria del dispositivo donde se realice la segmentación.

En comparación con el entrenamiento realizado para la reducción de ruido, el proceso de segmentación requiere un mayor tiempo de entrenamiento debido a la mayor cantidad de parámetros ajustables involucrados. Además, la curva de entrenamiento en la segmentación proporciona información más significativa en comparación con los casos de N2N y N2V. Al ser una tarea de clasificación, es posible alcanzar una función de pérdida igual a cero, lo que permite una detección más confiable en los problemas de entrenamiento. Por tanto, se han utilizado hiperparámetros adicionales específicos para detectar o evitar el sobreajuste.

- Early Stopping: El entrenamiento se detendrá si no hay mejora en la función de pérdida de los datos de validación (*val-loss*) durante 15 épocas consecutivas.
- Model Checkpoint: Este parámetro permite sobrescribir el archivo guardado de la actualización de los pesos basándose en la maximización o minimización de la cantidad supervisada. En nuestro caso, minimizaríamos *val-loss*, de manera que el modelo se guardará después de cada época en la que *val-loss* sea menor. Se debe tener en cuenta que el valor numérico de los pesos en la época final no tiene por qué coincidir con el guardado en la red.
- Reduce LR On Plateau: La tasa de aprendizaje disminuirá si no hay mejora en *val-loss* durante 10 épocas consecutivas. En nuestro caso, comenzaremos con un LR de 1, y se disminuirá a 0,1 si no ocurre la mejora en las épocas establecidas.

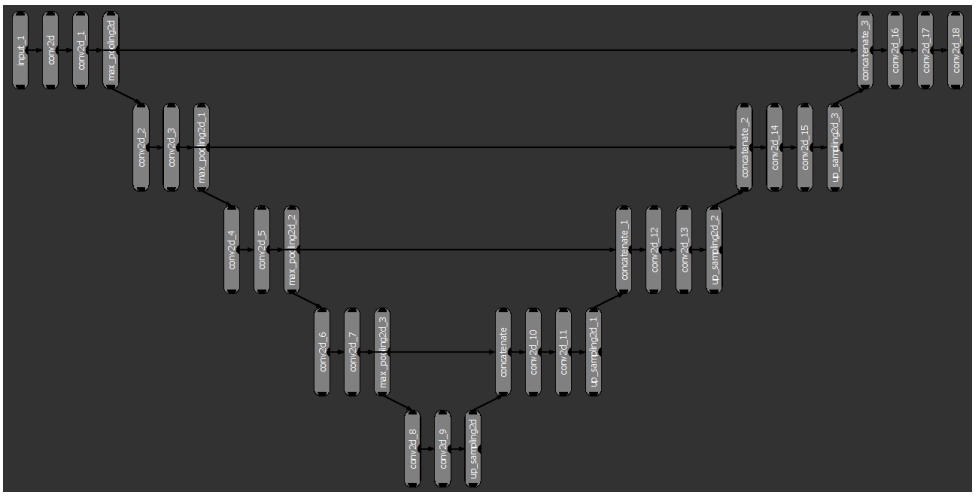


Figura 5.2: Esquema de la arquitectura de la red creada mediante el software DragonFly [33].

Name	***Size***	***Name***	***Size***
input_1	1	concatenate	
conv2d	64 filters	conv2d_10	512 filters
conv2d_1	64 filters	conv2d_11	256 filters
maxpool	2 stride 2×2	up_sampling_1 2d	nearest 2×2
conv2d_2	128 filters	concatenate_1	
conv2d_3	128 filters	conv2d_12	256 filters
maxpool_1	2 stride 2×2	conv2d_13	128 filters
conv2d_4	256 filters	up_sampling_2 2d	nearest 2×2
conv2d_5	256 filters	concatenate_2	
maxpool_2	2 stride 2×2	conv2d_14	128 filters
conv2d_6	512 filters	conv2d_15	64 filters
conv2d_7	512 filters	up_sampling_3 2d	nearest 2×2
maxpool_3	2 stride 2×2	concatenate_3	
conv2d_8	1054 filters	conv2d_16	64 filters
conv2d_9	512 filters	conv2d_17	32 filters
up_sampling 2d	nearest 2×2	conv2d_18	n filters

Tabla 5.2: Tabla con la descripción de las capas de la red.

Cada bloque descendente consta dos capas convolucionales 2D (conv2d) cuya función de activación es ReLU seguidas de una capa maxpooling encargada de codificar la información. Cada bloque ascendente consta de una capa *up-convolution* (UPSAMPLE), seguida de una concatenación con el correspondiente mapa de características procedente de la capa de codificadora, y dos capas convoluciones 2D (conv2d) cuya función de activación es ReLU. Tanto en las las capas de convolución como de deconvolución el tamaño del *kernel* es $(3, 3)$ con *stride* $(1, 1)$ y activación ReLU. Las capas *up sampling2d* utilizan *nearest interpolacion*, replicado de cada píxel de entrada en 2×2 píxeles de salida.

Se usa el mismo criterio para el *padding* que en la arquitectura de la red N2N y N2V. Cuando el *stride* es de 1 se utilizara "SAME" pero en las capas *maxpooling2d* con *stride* 2 se utiliza "VALID". La última capa convolucional de salida tiene tantos filtros como características se desee detectar. En el caso de querer segmentar mitocondrias, el número de filtros es 2, uno

correspondiente al *background* y otro a las mitocondrias.

5.2. Entrenamiento de una Unet 2D para la segmentación de mitocondrias

Los datos utilizados en este capítulo corresponden al volumen de acceso público [40], que fue previamente procesado en el capítulo anterior y se muestra en la Figura 4.13. Esta base de datos incluye la segmentación manual de las mitocondrias en todo el volumen, lo cual será de gran utilidad para la evaluación objetiva de los métodos automáticos de segmentación, pues se usará es información como *ground truth*.

De la misma muestra se obtienen dos tomogramas distintos formado cada uno por 165 *slices* correspondientes a la parte superior de la muestra y a su parte inferior. Utilizaremos el tomograma correspondiente a la parte superior como datos de entrenamiento y validación y la parte inferior para el test, pudiendo realizar una buena diferenciación entre estos tres conjuntos. La Figura 5.3 muestra la segmentación manual del tomograma utilizado para testear las redes entrenadas. El objetivo de la red será obtener una segmentación lo más parecida posible a la aquí mostrada.

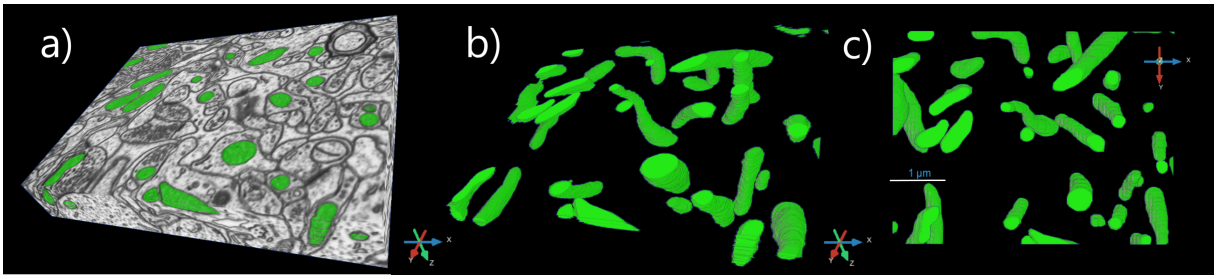


Figura 5.3: a) Volumen de prueba 3D con mitocondrias resaltadas en verde. b) Mitocondrias aisladas mostradas en una imagen 3D con la misma inclinación que en el volumen a). c) Mitocondrias aisladas en vista plano XY

5.2.1. Selección de los datos de entrenamiento y validación

Generalmente, para entrenar las redes en tareas de segmentación se utiliza una parte muy pequeña del volumen total debido a el tedioso proceso de etiquetado de píxeles. Por ello, aunque dispongamos de las etiquetas correspondientes a todo el volumen, seleccionamos 10 *slices* de las 165, reflejando un entrenamiento más "realista". No obstante, debemos tener en cuenta que

la muestra no es homogénea y la elección de unas zonas u otras del volumen utilizadas en el entrenamiento de la red puede cambiar considerablemente su desempeño.

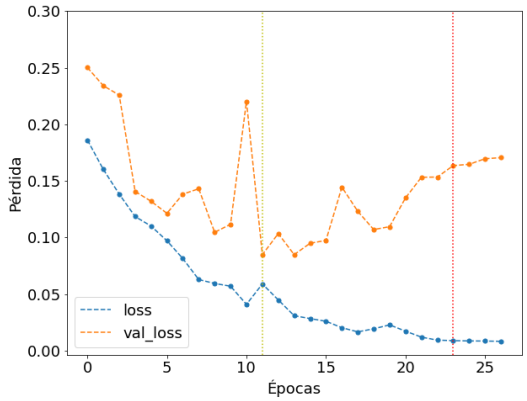
En esta primera segmentación, estamos interesados en detectar únicamente mitocondrias. Considerando que el volumen total del tomograma es $1024 \times 768 \times 165$ píxeles y que las mitocondrias suelen tener un tamaño aproximado de 200×50 píxeles en cada *slice*, el tamaño del *patch* seleccionado en base al tamaño relativo de las mitocondrias será 64×64 píxeles. Utilizar *slices* donde la proporción de este orgánulo este más presente en el entrenamiento será lo óptimo para lograr un mejor desempeño de la red. Para reflejar la relevancia de una elección adecuada de datos de entrenamiento, se compararán los resultados obtenidos al entrenar la red utilizando *slices* donde las mitocondrias constituyen el $\sim 4\,\%$ del número total de píxeles con los resultados obtenidos al duplicar el porcentaje de mitocondrias, $8\,\%$. Este subconjunto final de entrenamiento será el utilizado en el resto de experimentos. Cabe mencionar que los datos de validación en ambos casos serán 4 *slices* independientes del conjunto de entrenamiento con un porcentaje de mitocondrias del $8\,\%$.

Para medir cuantitativamente el desempeño de la red basándonos en la calidad de los resultados, se muestra la curva de entrenamiento para el conjunto de datos de entrenamiento,*loss*, y validación elegidos,*val-loss*. Una vez entrenada la red, se aplica a los datos de test y se calcula la matriz de confusión, que proporcionan información sobre la similitud entre el volumen segmentado y el *ground-truth* del que disponemos. La matriz de confusión es una herramienta que muestra los verdaderos positivos (VP), falsos positivos (FP), falsos negativos (FN) y verdaderos negativos (VN) en forma de matriz $\begin{pmatrix} VP & FP \\ FN & VN \end{pmatrix}$ en tanto %.

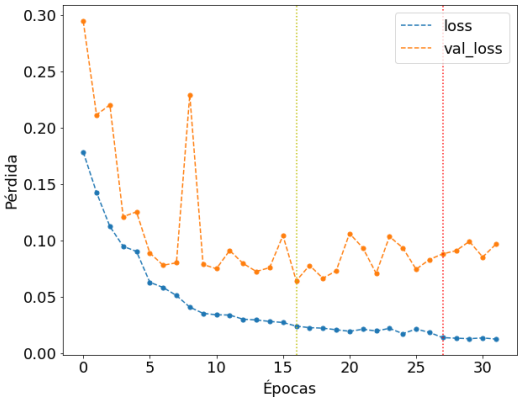
A continuación, se presentan los resultados y las curvas de entrenamiento obtenidas para los datos de prueba en relación a determinados hiperparámetros. La segmentación fue llevada a cabo por la red de la Figura 5.2, la cual fue entrenada utilizando un volumen donde las mitocondrias representan el $4\,\%$.

Hiperparámetro	**Matriz confusión**	**Early Stopping**	**LR reduced**	**Training time**
2 DataAugm. 2D	$\begin{pmatrix} 88{,}80 & 2{,}78 \\ 11{,}20 & 97{,}22 \end{pmatrix}$	Época 27	Época 23	10 min 50 s
4 DataAugm. 2D	$\begin{pmatrix} 92{,}69 & 3{,}00 \\ 7{,}31 & 97{,}00 \end{pmatrix}$	Época 32	Época 28	20 min 39 s
4 DataAugm. 2.5D (3 *slices*)	$\begin{pmatrix} \mathbf{93{,}20} & \mathbf{3{,}15} \\ \mathbf{6{,}80} & \mathbf{96{,}85} \end{pmatrix}$	Época 36	Época 19	24 min 42 s

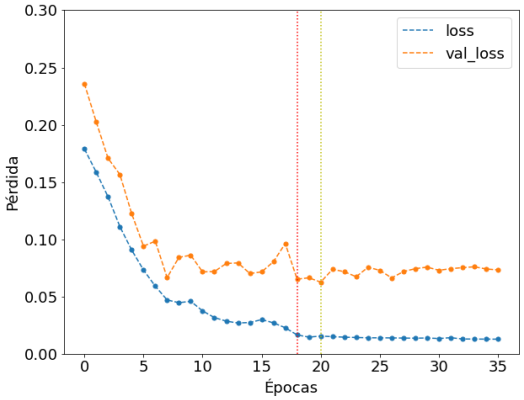
Tabla 5.3: Resultados de la segmentación mitocondrial en el tomograma de test.



(a) Curva de entrenamiento: 2 DataAugm. 2D



(b) Curva de entrenamiento: 4 DataAugm. 2D



(c) Curva de entrenamiento con 4 DataAugm. 2.5D (3 *slices*)

Figura 5.4: Comparación de las curvas de entrenamiento obtenidas con la selección de hiperparámetros correspondientes a la Tabla 5.3. Se indica con una línea discontinua amarilla el punto guardado haciendo uso de *Model Checkpoint* y con una línea discontinua roja el cambio en el *learning ratio*.

Como se puede observar, en todas las curvas de entrenamiento comienza un proceso de *overfitting* transcurridas 25 épocas aproximadamente, razón por la cual a pesar de haber seleccionado 50 épocas de entrenamiento, se aplica el *Early Stopping*. En consonancia con lo esperado teóricamente, se comprueba que un factor mayor de *data augmentation* permite una mayor capacidad de generalización a la red, aumentando considerablemente los verdaderos positivos reflejados en la matriz de confusión de la Tabla 5.3. El uso de una entrada 2.5D ayuda a estabilizar la curva de aprendizaje, debido al aumento de información en el cálculo de los pesos, lo cual hace que estén mas "promediados" y al aumentar el contexto del *patch*, el uso de 2.5D reduce los falsos negativos, dando lugar a un mejor entrenamiento. Por estos motivos, a partir de ahora todos los entrenamientos se realizarán con un factor 4 de *data augmentation* y una dimensionalidad 2.5D.

Al realizar una selección de los datos de entrenamiento donde el porcentaje de mitocondrias asciende al 8 %, se obtiene la siguiente tabla y curva de aprendizaje.

Épocas	**Matriz confusión**	**Time**
25	$\begin{pmatrix} 92{,}95 & 0{,}72 \\ 7{,}05 & 99{,}28 \end{pmatrix}$	13 min
50	$\begin{pmatrix} 92{,}55 & 0{,}59 \\ 7{,}45 & 99{,}41 \end{pmatrix}$	25 min
75	$\begin{pmatrix} 93{,}37 & 0{,}64 \\ 6{,}63 & 99{,}36 \end{pmatrix}$	37 min

Tabla 5.4: Matriz de confusión y tiempo de entrenamiento según el número de épocas con 4 DataAugm. y 2.5D (3*slices*).

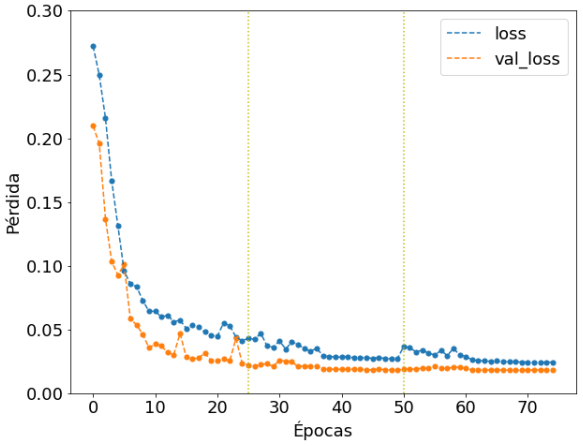
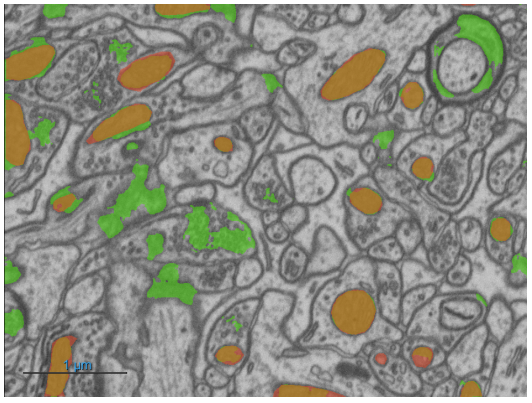
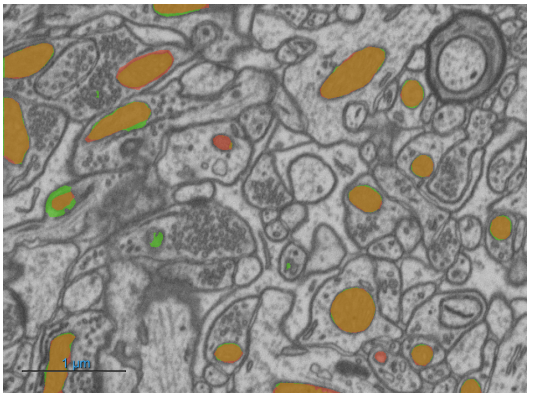


Figura 5.5: Curva de entrenamiento de la U-net 2D. La línea discontinua amarilla separa las 3 etapas de entrenamiento.

La Tabla 5.4 muestra una reducción significativa en los falsos positivos en todas las matrices de confusión en comparación con la Tabla 5.3, lo cual se puede ver de manera cualitativa en la Figura 5.6. Al tener disponible un mayor número de mitocondrias con distintos tamaños y orientadas de maneras diversas, la red es capaz de diferenciar mejor estos orgánulos de otras regiones celulares con características similares.



(a) Segmentación resultante de la red entrenada con los parámetros correspondientes a la curva de entrenamiento de la Figura 5.4b.



(b) Segmentación resultante de la red entrenada con los parámetros correspondientes a la curva de entrenamiento de la Figura 5.5 durante 50 épocas.

Figura 5.6: Comparación de la segmentación realizada por una red entrenada con distintos datos e hiperparámetros. Las áreas coloreadas en rojo representan las etiquetas manuales de las mitocondrias (*ground truth*), mientras que las etiquetas verdes corresponden a la segmentación predicha por la red. La superposición de estas dos etiquetas resulta en un color naranja.

Un aspecto que llama la atención en la curva de entrenamiento de la Figura 5.5, es el menor valor de la función de pérdida en los datos de validación respecto a los datos de entrenamiento [1]. Algunas de las razones por las cuales los datos de validación pueden tener una pérdida menor que los datos de entrenamiento son:

- Los datos de validación seleccionados pueden ser menos complejos que los de entrenamiento, es decir, presentar pocas estructuras que den lugar a un falso positivo o secciones grandes y diferenciadas de mitocondrias que faciliten una correcta clasificación. Por esta razón, en las Figuras 5.4 donde los datos de validación son más complejos que los de entrenamiento, la curva de validación siempre tiene valores de pérdida mayores.
- Las curvas de entrenamiento se obtienen mediante un promedio de los resultados de todos los *batches* de la época. Estos van mejorando desde el principio de la época hasta el final, por lo que el valor de la pérdida de los datos de entrenamiento puede aumentar debido al promediado con los valores iniciales. Sin embargo, las curvas de validación siempre se obtienen con los pesos correspondientes al final de la época, el cual debería ser, en teoría, el mejor modelo de la época. No obstante, el valor del último batch no siempre es el mejor, dando lugar a ciertos picos en las curvas de validación (observables sobre todo en entrenamientos con entradas 2D).

Además de la selección de datos e hiperparámetros mencionados, se exploraron otras opciones de optimización en la red, como:

– Realizar un filtrado con *kernel* 2.5D con un mayor número de *slices*, ya que se proporcionaría más información de la vecindad del píxel a clasificar. Aumentar el número de *slices* aumenta considerablemente el tiempo de entrenamiento. Por ejemplo, una entrada 2.5D de $7 slices$ emplea 57 min en el entrenamiento y no supone una ventaja notoria visualmente ni en su matriz de confusión $\begin{pmatrix} 92{,}58 & 0{,}52 \\ 7{,}42 & 99{,}48 \end{pmatrix}$. Se concluyó que un desarrollo óptimo de la red en comparación con el tiempo de ejecución se consigue aplicando 2.5D ($3 slices$).

– Un mayor tamaño de *patch*. Como las mitocondrias son un orgánulo relativamente grande, se comprobó si aumentar el *patch* a 128 supondría una ventaja, obteniendo la siguiente matriz de confusión $\begin{pmatrix} 92{,}13 & 0{,}69 \\ 7{,}87 & 99{,}31 \end{pmatrix}$ y resultados visuales no muy diferentes a los obtenidos con un *patch* de 64.

[1] De hecho, esto es algo bastante común recogido en *frequently Asked Keras Questions* de Keras [21]

Basándonos en los resultados experimentales, se ha decidido llevar a cabo el resto de entrenamientos con un factor de *data augmentation* 4 y una dimensionalidad 2.5D con 3 *slices*. Además, se ha seleccionado un volumen de entrenamiento con un porcentaje de mitocondrias del 8 %.

En el siguiente apartado se incorporará la detección de las vesículas[2] en la segmentación. Se investigará cómo enseñar a la red a reconocer las vesículas como entidades separadas de las mitocondrias, ya que, en ocasiones, la red neuronal puede tener dificultades para distinguir estos dos orgánulos. Para ello, utilizaremos las tomografías filtradas con la red *Chess*N2N explicada en el anterior capítulo, debido a que, visualmente, facilita la tarea de segmentar a mano, tal y como se puede ver en la Figura 5.7.

Antes de incorporar las etiquetas de las vesículas, comprobaremos si el filtrado afecta a la detección de las mitocondrias. Se calcula la matriz de confusión para un entrenamiento similar al realizado en la Figura 5.5 durante 50 épocas, $\begin{pmatrix} 92{,}56 & 0{,}78 \\ 7{,}44 & 99{,}22 \end{pmatrix}$, comprobando que se obtienen los mismos resultados dentro de las fluctuaciones esperadas por el inicio aleatorio de los pesos en un nuevo entrenamiento.

Este resultado esta en consonancia con lo esperado teóricamente. Filtrar el volumen no debería afectar demasiado a la segmentación, pues la propia naturaleza de la red codifica y descodifica la imagen, recuperando sólo las características más destacadas. Dado que la reconstrucción del ruido rara vez ocurre, los autocodificadores al comprimir las imágenes se comportan como un filtro antirruido [45].

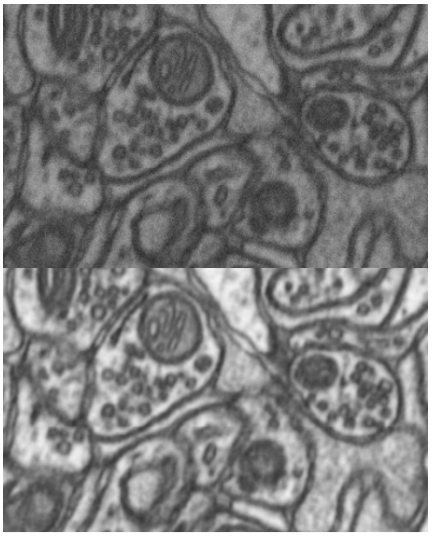


Figura 5.7: Comparación de la visualización de vesículas en el tomograma original (imagen superior) y filtrado con el método (imagen inferior) *Chess* N2N.

5.3. Entrenamiento de una Unet 2D para la segmentación de mitocondrias y vesículas

La Figura 5.6a muestra que, en ocasiones, la red neuronal encuentra dificultades para distinguir un grupo de vesículas de una mitocondria. La proximidad entre las vesículas, sumado al

[2]Estructuras membranosas esféricas o elipsoidales que abundan en las conexiones sinápticas entre neuronas y juegan un papel fundamental en la neurotransmisión.

hecho de que pueden estar rodeadas por membranas, hace que visualmente sean muy similares, lo que acentúa aún más el desafío para la red neuronal en su tarea de diferenciación. Esto es debido a que en el interior de las mitocondrias existen numerosos repliegues (crestas) de su membrana interna que visualmente pueden parecerse a grupos de vesículas. Enseñar a la red a reconocer las vesículas como una entidad separada de las mitocondrias puede ser una solución efectiva para abordar el problema de confusión entre ambas estructuras.

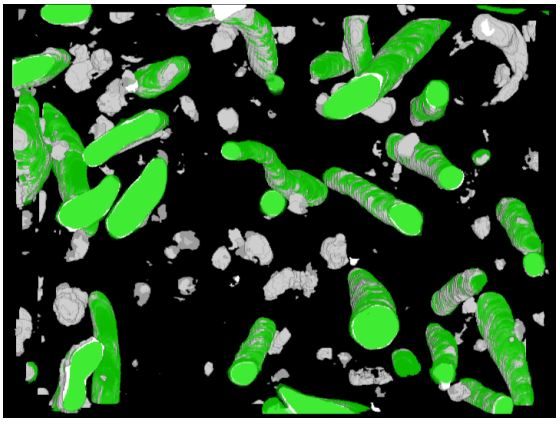
Tras el filtrado de ambos tomogramas, se realizó una segmentación manual de las vesículas sobre el conjunto de entrenamiento. Para la validación, se seleccionó aleatoriamente el 20 % de los datos segmentados manualmente del conjunto de entrenamiento. Aunque la selección de los datos de validación se realizó de forma aleatoria, estos no se utilizaron en la actualización de los pesos durante el entrenamiento de la red. Por lo tanto, este porcentaje de datos se mantuvo oculto durante el proceso de entrenamiento.

La matriz de confusión correspondiente a la segmentación realizada por la red tras ser entrenada durante 50 épocas, con un tiempo total de entrenamiento de 35 minutos y 21 segundos es $\begin{pmatrix} 90{,}73 & 0{,}59 \\ 9{,}27 & 99{,}41 \end{pmatrix}$. Muestra una ligera reducción de los falsos positivos y un aumento de los falsos negativos en comparación a la matriz de confusión obtenida en la segmentación de mitocondrias de los tomogramas filtrados $\begin{pmatrix} 92{,}56 & 0{,}78 \\ 7{,}44 & 99{,}22 \end{pmatrix}$.

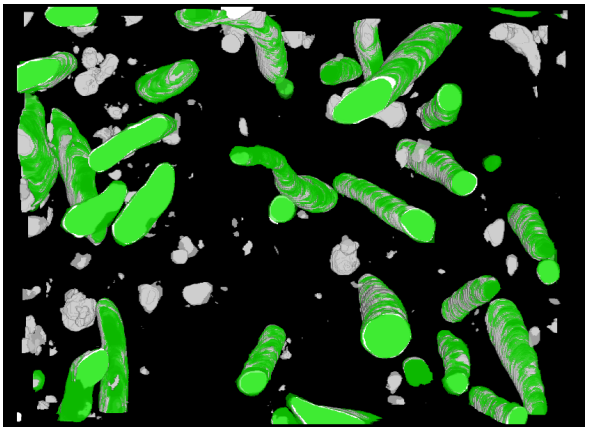
Es posible que algunos detalles visuales importantes no se reflejen directamente en la matriz de confusión. Por ejemplo, a pesar de que la matriz de confusión obtenida por la red entrenada con vesículas no refleja una mejora significativa en la reducción de falsos positivos mitocondriales, al observar la visualización 3D de la segmentación llevada a cabo por ambas redes en la Figura 5.8, si se aprecia una reducción notoria en el número de píxeles erróneamente identificados como mitocondrias coloreados en gris. La matriz de confusión proporciona una evaluación cuantitativa importante de la segmentación, pero puede haber discrepancias entre los resultados de la matriz de confusión y la mejora visual percibida. Por lo tanto, resulta de gran utilidad combinar la evaluación cuantitativa con la evaluación visual subjetiva para obtener una comprensión completa de la calidad de la segmentación.

Además, la evaluación de la segmentación se realiza mediante la comparación con un *ground truth*, resultado de una segmentación manual sujeta a ciertos sesgos y limitaciones inherentes al proceso humano. Estas limitaciones se reflejan en imprecisiones en la delimitación de las mitocondrias o en la clasificación de estructuras que podrían haber sido diferenciadas con anterioridad. Es especialmente notable la falta de homogeneidad en los bordes de las mitocondrias,

que suelen ser segmentados de manera más precisa por las redes neuronales. Como resultado, la matriz de confusión puede estar sesgada por estas limitaciones manuales, lo que puede llevar a clasificar erróneamente como falsos negativos algunas partes de estructuras que en realidad no lo son.



(a) Resultados obtenidos para la red entrenada en la segmentación de mitocondrias.



(b) Resultados obtenidos para la red entrenada en la segmentación de mitocondrias y vesículas.

Figura 5.8: Comparación de la segmentación 3D visualizada en el plano XY llevada a cabo por una red entrenada únicamente en la clasificación de mitocondria (a) y una red donde, además, se ha incorporado la clasificación de vesículas (b). Las estructuras coloreadas en verde se corresponden con una identificación correcta de las mitocondrias, mientras que las estructuras grises se corresponden con falsos positivos.

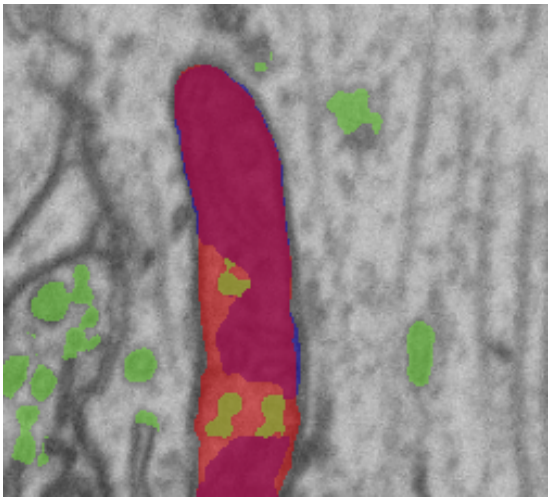


Figura 5.9: Vesículas coloreadas en verde, mitocondrias detectadas por la red entrenada con vesículas coloreada en azul, la cual al superponerse con la mitocondria coloreada en rojo detectada con la red entrenada únicamente con mitocondrias resulta en un color morado.

Un gran porcentaje del incremento de los falsos negativos en esta segmentación son causados por la identificación incorrecta de estructuras internas mitocondriales como vesículas. Esto se puede apreciar visualmente en la Figura 5.9. Es importante evaluar si la reducción en la identificación errónea de vesículas y sacos membranosos como mitocondrias resulta en una mejora significativa en la segmentación, lo suficiente como para compensar el efecto negativo de fragmentar las mitocondrias.

Para evaluar esta cuestión, en el apartado 5.5, abordaremos el refinamiento de la segmentación mitocondrial utilizando herramientas de análisis de imágenes.

5.4. Entrenamiento de una Unet 3D para la segmentación de mitocondrias: Impacto del contexto 3D

Con el fin de evaluar el impacto de un contexto tridimensional, realizamos una comparación del método U-net 2D propuesto en la Figura 5.2 (ya sea con entrada 2D o 2.5D) con la U-net 3D correspondiente a la Figura 5.10.

Para hacer una comparación justa, se ha de mencionar que el número de parámetros entrenables ha descendido considerablemente, pasando a tener 4.000.000 frente a los 21.000.000 de las Unet 2D. El uso de volúmenes de entrada y salida limita el número de filtros de las capas de convolución por razones de capacidad computacional. Por los mismos motivos, el *patch* seleccionado es menor que el utilizado anteriormente, $32 \times 32 \times 32$.

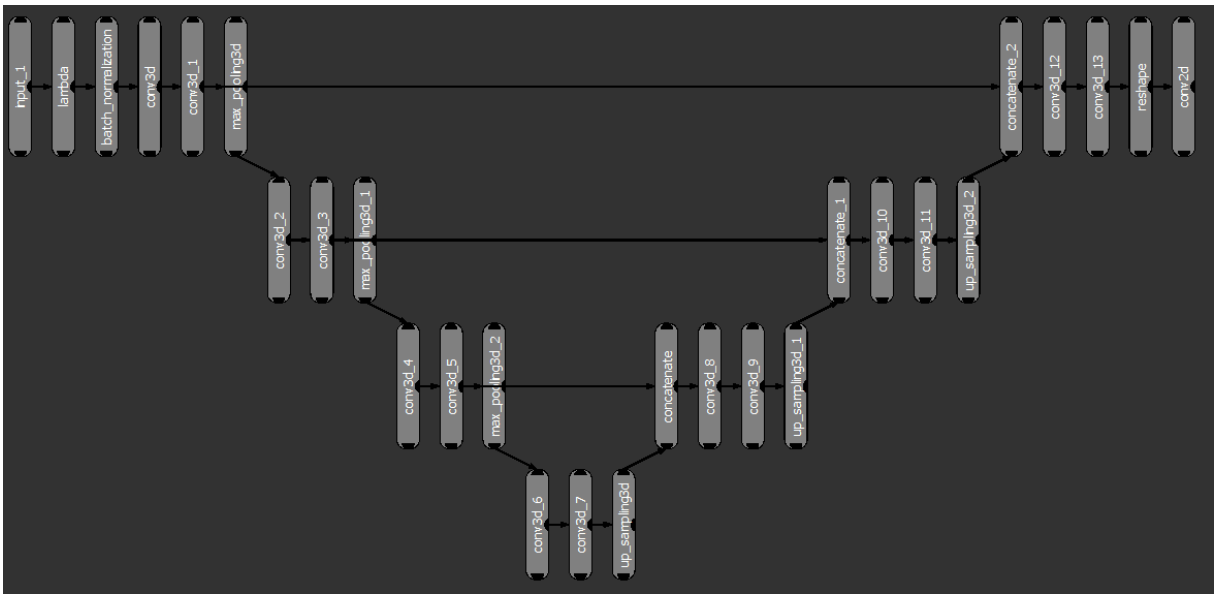


Figura 5.10: Arquitectura de la red neuronal U-net 3D utilizada en la segmentación de mitocondrias y creada a través del software Dragonfly [33].

En particular, el camino descendente o codificador (encoder path) se compone de 4 niveles de procesamiento. El primer nivel consta de una capa de *batch normalization* seguida de dos capas convolucionales 3D (tamaño de kernel de $(3, 3, 3)$) con activación ReLU y finalmente de una capa de convolución maxpooling $(2, 2, 2)$ con *stride* $(2, 2, 2)$. Los 3 niveles restantes están compuestos únicamente por 2 capas de convolución 3D y la capa maxpooling. El máximo número de filtros, 256, se alcanza en la capa de convolución número 6, la cual da paso al camino decodificador. El criterio utilizado para el *padding* es el mismo que se explicó previamente en la U-net 2D.

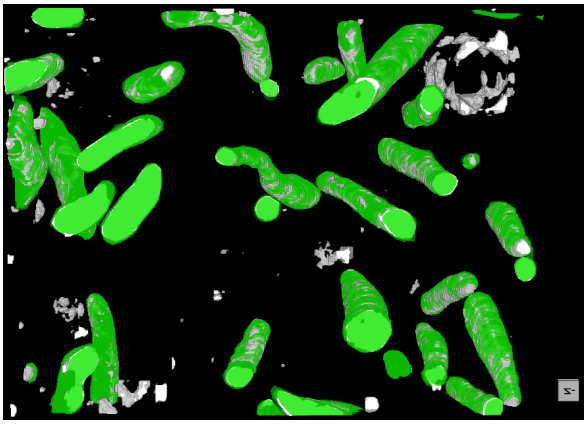
En la siguiente tabla se recoge una comparativa de los resultados obtenidos con la U-net 2D y 3D y sus respectivas estrategias de entrenamiento.

Red	**Épocas**	**Matriz confusión**	**Time**
U-net 2D Mitocondrias	50	$\begin{pmatrix} 92{,}56 & 0{,}78 \\ 7{,}44 & 99{,}22 \end{pmatrix}$	25 min
U-net 2D Mito+Vesículas	50	$\begin{pmatrix} 90{,}73 & 0{,}59 \\ 9{,}27 & 99{,}41 \end{pmatrix}$	35 min 21 s
U-net 3D	20	$\begin{pmatrix} 84{,}88 & 0{,}55 \\ 15{,}12 & 99{,}45 \end{pmatrix}$	5h 53 min 19s

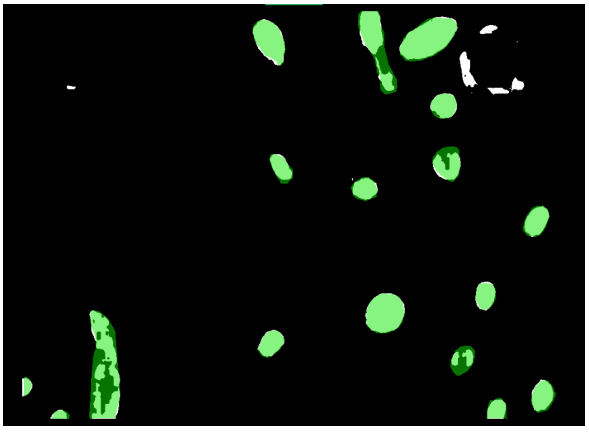
Tabla 5.5: Se presentan los resultados obtenidos para cada método en relación con las hiperparámetros mencionados. Se detalla la matriz de confusión y tiempo de entrenamiento según el número de épocas.

Es importante destacar que, en términos de eficiencia, no se debe considerar únicamente el tiempo de entrenamiento de la red neuronal, ya que el tiempo de aplicación también puede ser significativo. Al aplicar el filtro a todo el tomograma, se realiza una inferencia o predicción en cada punto del volumen, lo que implica realizar cálculos utilizando los pesos y parámetros aprendidos durante el entrenamiento de la red. Sin embargo, durante el entrenamiento de la red neuronal, aunque se realizan múltiples inferencias (épocas) en el conjunto de datos, este conjunto es considerablemente más pequeño en comparación con todo el tomograma. Esto hace que el tiempo requerido para realizar múltiples épocas en un conjunto de datos pequeño sea comparable al tiempo necesario para realizar una única inferencia en todo el tomograma.

Al igual que ocurría en el entrenamiento con vesículas, la información cuantitativa aportada por la matriz de confusión debe ser complementada con la información cualitativa de las imágenes. En la Figura 5.11a se puede observar una reducción notoria en el número de píxeles erróneamente identificados como mitocondrias coloreados en gris, obteniendo una segmentación 3D en principio muy parecida a la deseada de la Figura 5.3.



(a) Reconstrucción 3D de la segmentación realizada con la U-net 3D.



(b) *Slice* 2D de la segmentación realizada con la U-net 3D.

Figura 5.11: Visualización 3D y 2D de la segmentación llevada a cabo por la U-net 3D. Las estructuras coloreadas en verde se corresponden con una identificación correcta de las mitocondrias, mientras que las estructuras grises se corresponden con falsos positivos.

No obstante, al examinar las *slices* en 2D de la Figura 5.11b, se puede observar claramente el error cometido al intentar segmentar una mitocondria completa. En lugar de identificar la mitocondria en su totalidad, la red neuronal identifica solo parches o fragmentos de la estructura, lo que resulta en un aumento considerable en el porcentaje de falsos negativos en comparación con los predichos por la red U-net 2D. Este problema podría ser atribuido, en parte, a la reducción del tamaño de los *patches* a 32 píxeles.

En el siguiente apartado realizaremos una evaluación final de las distintas segmentaciones, las cuales serán refinadas mediante conexión de píxeles vecinos y eliminación de vóxeles inconexos, pudiendo evaluar la utilidad en la segmentación final de la red cuando esta es mínimamente manipulada.

5.5. Resultados de la segmentación final

En este apartado, se abordará la cuantificación de la morfología y el contenido mitocondrial mediante análisis de imagen de las distintas segmentaciones llevadas a cabo por las redes neuronales. En particular, se utilizará el módulo de Análisis de Objetos para Dragonfly, una herramienta que permite medir y cuantificar los componentes conectados en la segmentación obtenida. Este módulo proporciona opciones avanzadas de filtrado basadas en el volumen y el área de superficie de los objetos segmentados, lo que permite clasificarlos según características específicas. Además, se explorarán opciones adicionales que facilitan la división y fusión de

objetos, mejorando la segmentación realizada.

El propósito fundamental es presentar el resultado obtenido mediante un refinamiento rápido y sencillo de la segmentación realizada por la red. No se pretende corregir minuciosamente todos los errores cometidos por la red, ya que esto impediría aprovechar plenamente las ventajas que ofrece una red neuronal, lo que implicaría la necesidad de realizar una cantidad significativa de segmentación manual.

- Relleno de estructuras conectadas: En primer lugar, utilizaremos una herramienta que permite rellenar poros conectados con un diámetro de apertura inferior al umbral seleccionado, en nuestro caso 50 nm. Con esta herramienta intentaremos corregir esrtucturas como la observada en la Figura 5.9.

- Conectividad[3] y vóxel umbral: Realizaremos un etiquetado de componentes conectados, proceso en el que encontraremos todos los componentes conectados de una imagen y marcaremos cada uno de ellos con una etiqueta distintiva. En concreto, utilizaremos la conexión de 26 componentes. La propagación se realiza en todas las direcciones, utilizando las 6 caras, 12 aristas y 8 esquinas adyacentes a la semilla actual. A continuación se eliminan todos los objetos, determinados por el método de 26 conexiones, que tienen un recuento de vóxeles inferior a un umbral seleccionado, en nuestro caso 5000 vóxeles. De esta forma se eliminan aquellos objetos segmentados que sean considerados pequeños o inconexos y no correspondan a mitocondrias de interés.

Los resultados presentados en la Tabla 5.6 permiten realizar una comparativa de la segmentación lograda por las tres redes neuronales después de aplicar el refinamiento "automático" mencionado previamente. Se incluyó el coeficiente DICE [46] para brindar información cuantitativa sobre la similitud entre la segmentación realizada manualmente y la predicha por la red.

El coeficiente de Dice (DICE) es ampliamente utilizado para evaluar la similitud entre dos conjuntos segmentados y se aplica principalmente en problemas de segmentación de imágenes o volúmenes. Con valores en el rango de 0 a 1, proporciona una medida de superposición o solapamiento entre las regiones segmentadas. Se calcula como el doble de la intersección entre las regiones segmentadas dividido por la suma de los tamaños de las regiones

[3]En el procesamiento de imágenes, la conectividad es la forma en que los vóxeles de las imágenes tridimensionales se relacionan con sus vecinos y se etiquetan como componentes separados.

Red	**Nº Mitocondrias**	**Matriz confusión**	**DICE**
Segmentación Manual	34	$\begin{pmatrix} 100 & 0 \\ 0 & 100 \end{pmatrix}$	1
U-net 2D Mitocondrias	48	$\begin{pmatrix} 94{,}08 & 1{,}02 \\ 5{,}92 & 98{,}98 \end{pmatrix}$	0,886
U-net 2D Mito+Vesículas	38	$\begin{pmatrix} 92{,}82 & 0{,}60 \\ 7{,}18 & 99{,}40 \end{pmatrix}$	**0,912**
U-net 3D	37	$\begin{pmatrix} 87{,}91 & 0{,}55 \\ 12{,}09 & 99{,}45 \end{pmatrix}$	0,889

Tabla 5.6: Se presentan los resultados obtenidos para cada modelo, incluyendo la segmentación manual. Se detalla la matriz de confusión, el número de mitocondrias consideradas como componentes conectados y el coeficiente Dice.

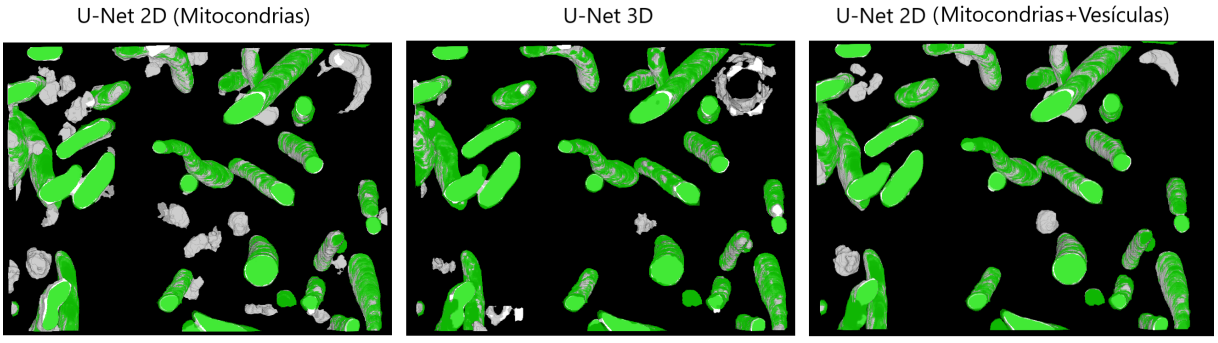


Figura 5.12: Se muestran tres imágenes 3D comparativas de la segmentación de mitocondrias obtenidas utilizando distintas configuraciones de redes neuronales. A la izquierda se presenta la segmentación realizada por la red Unet 2D entrenada exclusivamente para mitocondrias, en el centro la segmentación obtenida con Unet 3D, y a la derecha la segmentación lograda por la Unet 2D entrenada para segmentar tanto mitocondrias como vesículas. Las estructuras coloreadas en verde corresponden a una identificación correcta, mientras que las estructuras grises se corresponden con falsos positivos.

La segmentación realizada por la red neuronal Unet 2D, entrenada para identificar tanto mitocondrias como vesículas, ha demostrado ofrecer una mejor segmentación final de mitocondrias en comparación con otras configuraciones. Esta afirmación se basa en múltiples criterios de evaluación, que incluyen no solo la matriz de confusión, sino también el coeficiente DICE y la mejora visual observada. En la Figura 5.12 se pueden observar algunas diferencias entre la segmentación neuronal y manual, con la presencia de algunas detecciones erróneas por parte de la red neuronal. Sin embargo, en general, la red Unet 2D realiza un buen trabajo en la segmentación, logrando resultados prometedores en la identificación de las estructuras de interés.

Los errores cometidos por la red pueden estar asociados a la presencia de estructuras que presentan una alta similitud visual con las mitocondrias, lo cual dificulta su diferenciación tanto para la red como para un observador humano, tal y como se puede ver en la Figura 5.13. Por

lo tanto, es importante reconocer que la red puede estar desempeñando la tarea de clasificación de manera efectiva en general, a pesar de cometer errores en ciertas clasificaciones debido a la complejidad de las estructuras.

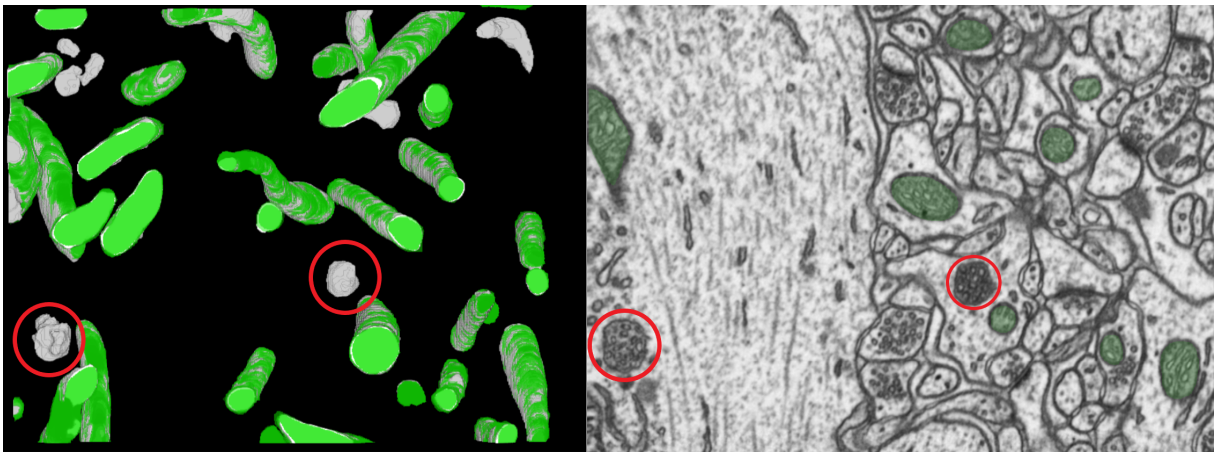


Figura 5.13: Ejemplo de estructuras con alta similitud visual a las mitocondrias en la segmentación final de la Unet 2D.

Una vez se obtiene una segmentación separada por componentes como la que se muestra en la Figura 5.14, se pueden estudiar las estructuras de interés para detectar. Por ejemplo, las alteraciones en la morfología y función de las mitocondrias están asociadas a diversas enfermedades, como trastornos metabólicos, enfermedades neurodegenerativas y cáncer [47] [48]. La segmentación precisa de las mitocondrias permite analizar estos cambios y cuantificar parámetros relevantes para el diagnóstico, pronóstico y seguimiento.

(a) Segmentación *ground truth* tras el análisis de componentes y conectividad.

(b) Segmentación Unet 2D (mitocondrias + vesículas) tras el análisis de componentes y conectividad.

Figura 5.14: Comparación de la segmentación 3D visualizada en el plano XY realizada manualmente (a) y llevada a cabo por una red Unet 2D (b). Ambos volúmenes han sufrido el mismo refinamiento manual.

Por ejemplo, se puede realizar un análisis de *aspect ratio* (o relación de aspecto) para saber la distribución de formas que poseen las mitocondrias y observar anomalías.

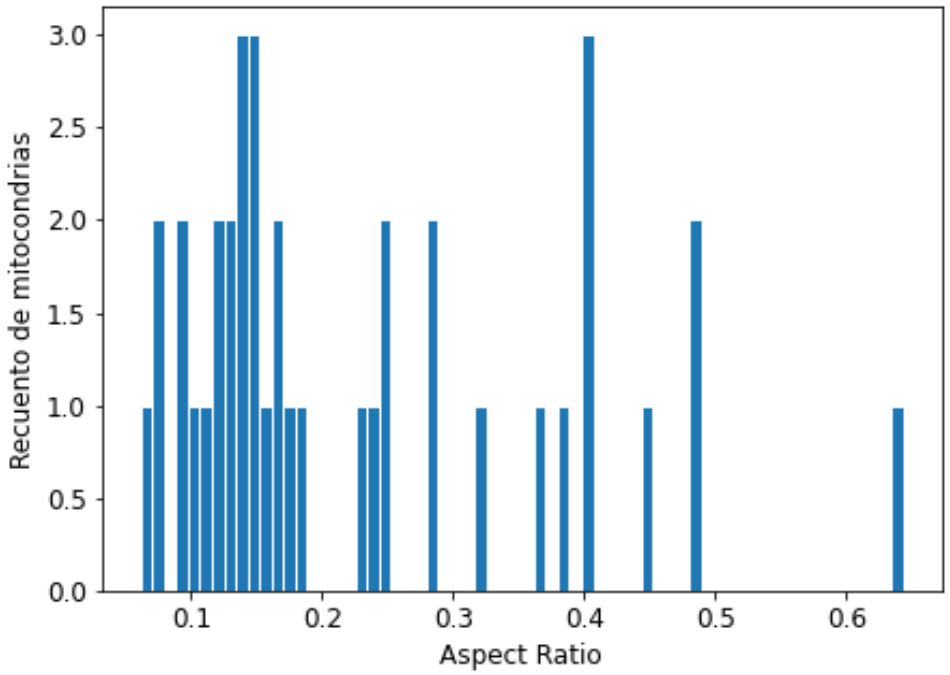


Figura 5.15: Histograma con la distribución de los *aspect ratios* de las mitocondrias correspondientes a la Figura 5.14b.

El aspect ratio proporciona información sobre la proporción entre el largo y el ancho de las mitocondrias segmentadas. Este análisis puede ayudar a detectar posibles variaciones en la forma de las mitocondrias, como elongaciones o esfericidad, que podrían estar relacionadas con cambios en su función o estado fisiológico. En concreto, el valor del aspect ratio aquí mostrado, describe la relación proporcional entre el valor propio más pequeño, correspondiente a la dirección preferente del volumen, y el más grande, correspondiente a la dirección de mayor cambio del volumen, para los vectores propios de inercia. La relación de aspecto se calcula como la relación entre $min(eigenvalue1, eigenvalue2, eigenvalue3)/max(eigenvalue1, eigenvalue2, eigenvalue3)$. De esta manera, un cubo o una esfera perfectos tendrían una relación de aspecto de 1, mientras que una varilla perfecta de un vóxel de ancho o de un punto es 0.

El histograma del aspect ratio de las mitocondrias segmentadas por la Unet 2D correspondiente a la Figura 5.15, revela una distribución en la que predominan valores cercanos a cero, lo que indica que estas estructuras celulares tienden a ser alargadas o incluso muy alargadas.

Los resultados obtenidos en este capítulo han sido fuertemente influenciadas por la cali-

dad del tomograma que se ha utilizado para la segmentación. En tomogramas donde ciertos orgánulos, como el núcleo, ocupen una gran parte del volumen y presenten estructuras internas complejas, puede resultar más difícil detectar con precisión las mitocondrias o las vesículas, tal y como ocurría en el caso de las vesículas etiquetadas erróneamente dentro de las mitocondrias. Estos desafíos adicionales pueden afectar la efectividad de los métodos de segmentación y generar resultados menos precisos en tomogramas más complejos.

Capítulo 6

Conclusiones

El empleo de redes neuronales en investigaciones científicas se presenta como una herramienta prometedora. Si bien es cierto que aún se requiere una supervisión humana debido a la falta de resultados perfectos, la ventaja principal de las redes neuronales radica su capacidad para aliviar a los científicos de tareas laboriosas y tediosas. Al automatizar procesos como la segmentación, las redes permiten a los investigadores centrarse en tareas más creativas y analíticas, donde pueden aplicar su experiencia y conocimiento para interpretar los resultados. Aunque es necesario un seguimiento y ajuste adecuado de las redes neuronales, en este trabajo ha quedado reflejado su capacidad para optimizar y agilizar el trabajo rutinario que supone la segmentación, incluso haciendo uso de redes relativamente sencillas. Por otro lado, su desempeño en el ámbito del filtrado de ruido ha evidenciado ser una destacada alternativa a los métodos convencionales en determinados tomogramas, logrando incluso una mejora en la calidad de las imágenes en comparación con dichos métodos tradicionales, especialmente en la preservación de la morfología de las estructuras. La capacidad de las redes neuronales para aprender características y patrones complejos ha posibilitado obtener resultados superiores en términos de reducción de ruido, brindando una mayor claridad y preservación de los detalles relevantes en las imágenes. Sin embargo, es importante destacar que el uso de redes neuronales en el filtrado de ruido puede dificultar el estudio de la intensidad en las imágenes, ya que esta sufre un reescalado no lineal de la intensidad no invertible, lo que puede afectar la interpretación de los niveles de gris y los valores de píxel.

El estudio llevado a cabo en este trabajo ha supuesto una amplia revisión de la literatura relativa al estado del arte de los campos de la redes neuronales convolucionales y tomogramas

aplicados a la biología celular y molecular. Se ha visto cómo una elección adecuada de datos de entrenamiento e hiperparámetros de la red son aspectos críticos para lograr buenos resultados en el filtrado de ruido y la segmentación mitocondrial. De igual manera, se han explorado diversas arquitecturas de Unets, refinando las configuraciones que ofrecen un rendimiento superior en términos de precisión y velocidad. Además, la eficiencia de las redes y la capacidad de sus estructuras se ven condicionadas por el hardware utilizado, siendo necesario considerar las capacidades computacionales de los ordenadores empleados. Estos aspectos son cruciales para maximizar el potencial de las redes neuronales y garantizar resultados sólidos y reproducibles en el análisis de imágenes tomográficas.

Durante el desarrollo de este trabajo fin de máster, tuve la oportunidad de aprender y ser guiada por expertos en el grupo de investigación del ISPA. Esta experiencia ha sido sumamente enriquecedora, ya que me ha permitido adentrarme en el apasionante mundo de la investigación y adquirir un profundo conocimiento sobre las metodologías y técnicas empleadas en el análisis de imágenes tomográficas. Bajo la tutela de estos profesionales, he podido apreciar la complejidad y los desafíos inherentes a la aplicación de redes neuronales en este campo. Esta experiencia me ha proporcionado una base sólida para continuar mi desarrollo académico y profesional en el ámbito de la investigación científica y el empleo de redes neuronales.

Apéndice A

Transformada de Fourier: Aplicación al filtrado de imágenes

Una imagen puede ser representada a través del dominio convencional espacial o en el dominio de frecuencia. En ambos dominios, podemos discernir aspectos diferentes del ruido y se puede llevar a cabo el filtrado de imágenes. El filtrado en el espacio de Fourier consiste en calcular la transformada de Fourier (FT) de la imagen o tomograma, multiplicar el filtro de Fourier por las componentes de Fourier originales y calcular la FT inversa para obtener la imagen filtrada en el espacio real o dominio espacial. El mayor potencial de los filtros en el espacio de Fourier reside en la eliminación del ruido periódico, es decir, aquel que forma patrones repetitivos a lo largo de la imagen y puede ser identificado y estudiado en el dominio de la frecuencia (no abordada en este trabajo).

En este apéndice se repasarán los conceptos de frecuencia en dos dimensiones, así como la representación de imagenes en el espacio de Fourier y los posibles filtros que se pueden aplicar en dicho espacio. Finalmente, se mostrarán algunos de los resultados obtenidos en el filtrado del Phantom sintético utilizado en el Capítulo 4.

A.1. Concepto de frecuencia en dos dimensiones

Una función sinusoidal de dos dimensiones, se caracteriza por su fase, su frecuencia de oscilación y su dirección de oscilación. Una función sinusoidal en un espacio de dos dimensiones

con la frecuencia normalizada al tamaño de la imagen $M \times N$ se describe de la siguiente manera

$$f(m,n) = \sin 2\pi(\frac{u}{M}m + \frac{v}{N}n) \ , \tag{A.1}$$

donde m y n son las coordenadas espaciales en píxels, U y V son las dos frecuencias (ciclos/píxel) y $1 < u < M$ y $1 < v < N$.

La transformada de Fourier se basa en la idea de que cualquier señal puede ser representada como la suma de un número infinito de señales sinusoidales de diferentes frecuencias. En el caso de las imágenes, la idea es la misma. La imagen resultante es la suma de funciones sinusoidales de dos dimensiones, donde los valores de la escala de grises varían según la función seno, tal y como se puede observar en la Figura A.1.

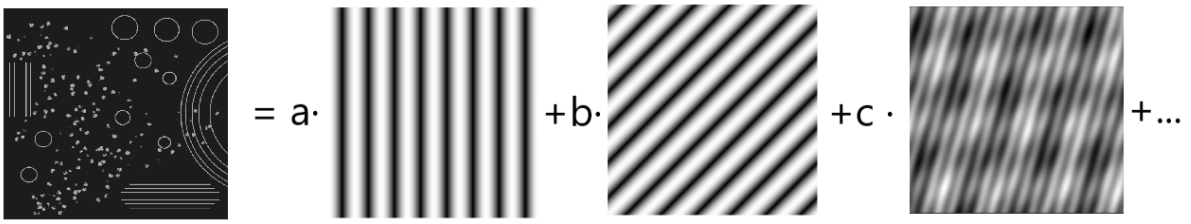


Figura A.1: Descomposición del Phantom en suma de funciones sinusoidales bidimensionales

A.2. Visualización de la Transformada Discreta de Fourier: DFT

La transformada de Fourier discreta en dos dimensiones se puede escribir como

$$F(u,v) = \frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m,n)e^{-i2\pi(\frac{u}{M}m+\frac{v}{N}n)} \ , \tag{A.2}$$

mientras que su transformada de Fourier inversa en forma discreta es

$$f(m,n) = \frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} F(u,v)e^{i2\pi(\frac{u}{M}m+\frac{v}{N}n)} \ . \tag{A.3}$$

Cada punto de la imagen en el dominio de Fourier representa una frecuencia particular contenida en la imagen en el dominio del espacio. Algunos ejemplos sencillos que pueden ayudar a visualizar este concepto se recogen en la Figura A.2.

Las zonas homogéneas en la imagen dan lugar a que los niveles claros de gris en el espectro estén concentrados mayoritariamente en las bajas frecuencias. Zonas con muchos bordes y transiciones bruscas en los níveles de gris de la imagen en el dominio espacial dan lugar a un espectro con componentes de alta frecuencia.

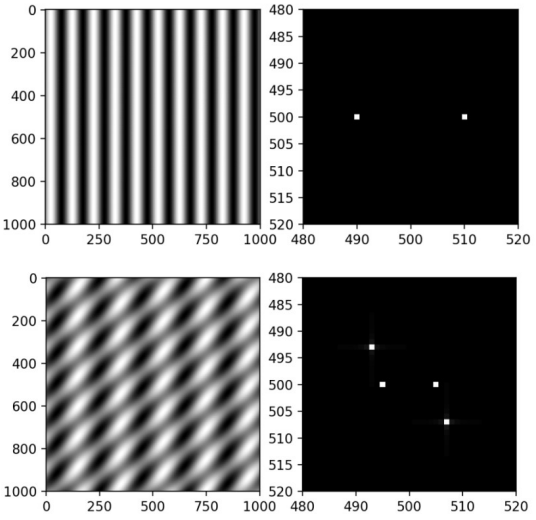


Figura A.2: Transformadas de Fourier de funciones sinusoidales sencillas

A.3. Filtros pasa-baja y pasa-alta

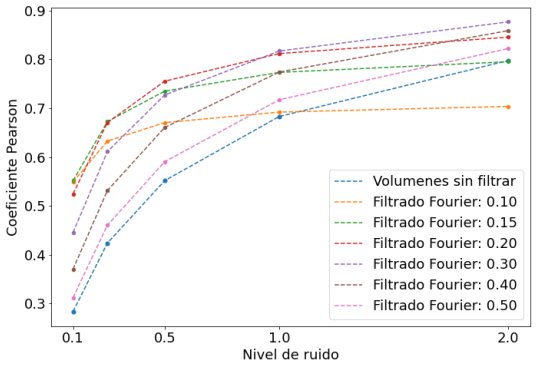
La transformación de una imagen o tomograma en el dominio de Fourier permite la separación directa de las características de baja resolución de los detalles más finos, los bordes y las transiciones nítidas (representados por componentes de media y alta resolución). Dado que el ruido predomina sobre la señal en las frecuencias medias y altas, atenuar o filtrar la amplitud en esas frecuencias reduce la contribución del ruido.

Un filtro que preserva los componentes de baja frecuencia se denomina filtro paso bajo, manteniendo los valores situados en el centro del espectro (alrededor del origen del espacio de Fourier). En cambio, un filtro de paso alto conserva los componentes de alta frecuencia y atenúa o elimina los demás. Los llamados filtros pasa banda conservan una gama determinada de frecuencias medias.

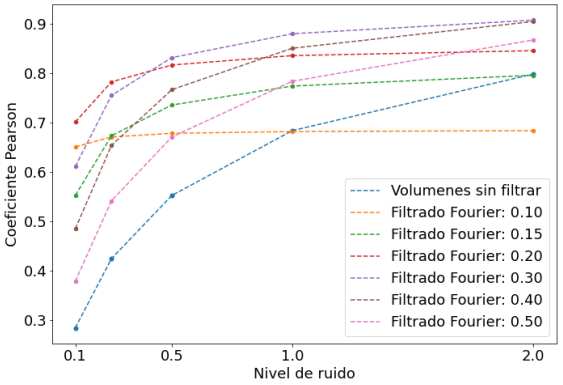
Los filtros de Fourier suelen implementarse mediante funciones suaves, evitando así cortes abruptos que se traducirían en artefactos de *ringing* en el espacio real. El filtrado en el espacio real es un procedimiento que opera directamente sobre los píxeles/voxeles de la imagen o tomograma.

En la Figura A.3 se observa el resultado de aplicar distintos filtros de Fourier al phantom cuantificados con el coeficiente de Pearson con las frecuencias normalizadas a 0,5. Se incluye tanto un filtro realizado sobre las imágenes 2d del tomograma, como un filtro Fourier 3D. No

constituyen filtros abruptos, sino filtro que sigue una función coseno con una transición suave realizada en el intervalo $[-0{,}05, +0{,}05]$ en torno a cada frecuencia de corte. Es decir, si la frecuencia de corte es de 0,3, entonces: todas las frecuencias hasta 0,25 se mantienen intactas, mientras que todas las frecuencias desde la 0,35 se cancelan, consiguiendo una transición suave entre 0,25 y 0,35.



(a) Filtros pasa baja aplicados en el dominio de Fourier 2D.



(b) Filtros pasa baja aplicados en el dominio de Fourier 3D.

Figura A.3: Valores de correlación obtenidos entre las imágenes filtradas en el espacio de Fourier tanto 2D como 3D y el Phantom original. Se puede apreciar un mejor filtrado en el dominio de las frecuencias 3D reflejado en un mayor valor en el Coef. de Pearson. Así mismo, un filtrado óptimo para esta imagen se corresponde con un filtro pasa baja de 0,2 en el caso bidimensional y 0,3 en 3D.

Apéndice B

Regla delta generalizada

La salida de la neurona k de una capa:

$$y_k = f(z_k) \ , \quad \text{con} \quad z_k = \sum_j w_{jk} y_j \ , \tag{B.1}$$

donde los subíndices de ij hacen referencia a la conexión de la neurona j con la neurona i de una capa anterior y f es la función de activación correspondiente.

Cada peso se modifica sumándole una cantidad proporcional a la menos derivada de la función de error, L, donde aplicaremos la regla de la cadena para poder calcularlo

$$\frac{\partial L}{\partial w_{jk}} = \frac{\partial L}{\partial z_k} \frac{\partial z_k}{\partial w_{jk}} \ , \tag{B.2}$$

donde, gracias a la Ec. B.1, sabemos que

$$\frac{\partial z_k}{\partial w_{jk}} = y_j \ . \tag{B.3}$$

Definiendo δ_k como

$$\delta_k = -\frac{\partial L}{\partial z_k} \ , \tag{B.4}$$

cuyo cálculo requiere nuevamente el uso de la regla de la cadena

$$\frac{\partial y_k}{\partial z_k} = f'(z_k) \ , \qquad \delta_k = -\frac{\partial L}{\partial y_k} \frac{\partial y_k}{\partial z_k} = -f'(z_k) \frac{\partial L}{\partial y_k} \ , \tag{B.5}$$

se obtiene Δw_{jk} definido en la Ec. 3.4

$$\Delta w_{jk} = -\eta \frac{\partial L}{\partial w_{jk}} = \eta\ \delta_k\ y_j = \eta f'(z_k)\ y_j\ \frac{\partial L}{\partial y_k} \tag{B.6}$$

necesario para actualizar el peso $w_{ij}(n+1) = w_{ij}(n) + \Delta w_{ij}(n)$.

En el cálculo de $\frac{\partial L}{\partial y_k}$ de distinguen dos casos dependiendo de si es la capa de salida o no:

- Capa de salida: Actúa directamente sobre L, por lo que se aplica la Ec. B.6.
- Si no es la capa de salida: podemos calcular el efecto de la salida de una neurona de la capa oculta y_o en función de su efecto sobre z_f de la neurona de salida

$$\frac{\partial L}{\partial y_o} = \sum_{f=1}^{N_f} \frac{\partial L}{\partial z_f} \frac{\partial z_f}{\partial y_o} = \sum_{f=1}^{N_f} \frac{\partial L}{\partial z_f} \frac{\partial}{\partial y_o} (\sum_{j=1}^{N_o} w_{jf} y_j) = \sum_{f=1}^{N_f} \frac{\partial L}{\partial z_f} w_{of} = -\sum_{f=1}^{N_f} \delta_f w_{of} \tag{B.7}$$

$$\Delta w_{jf} = \eta f'(z_f) y_j \sum_{f=1}^{N_f} \delta_f w_{of} \tag{B.8}$$

En cada iteración, una vez que todas las neuronas disponen del valor del gradiente de la función de pérdida que les corresponde, se actualizan los valores de los parámetros en sentido contrario a la dirección del gradiente.

Apéndice C

Adadelta

El optimizador Adadelta surge como una mejora de los dos principales inconvenientes del optimizador AdaGrad: el continuo decaimiento de las tasas de aprendizaje aprendizaje a lo largo del entrenamiento y la necesidad de ajustar este parámetro manualmente [22].

En el método AdaGrad el denominador acumula los gradientes al cuadrado de cada iteración desde principio del entrenamiento para llevar a cabo la actualización de los pesos

$$\omega_{t+1} = \omega_t - \frac{\eta}{\sqrt{\alpha_t + \epsilon}} g_t \ , \tag{C.1}$$

donde ω son los pesos de las neuronas, ϵ una constante infinitesimal para evitar la división de la tasa de aprendizaje por cero, g es el gradiente y α una nueva constante inicializada a cero e incorporada por este algoritmo que se calcula de la siguiente manera

$$\alpha_t = \alpha_{t-1} \cdot g_t^2 \ . \tag{C.2}$$

Cada término es positivo, por lo que α sigue creciendo a lo largo del entrenamiento, reduciendo la tasa de aprendizaje. Después de muchas iteraciones, esta tasa de aprendizaje será infinitesimalmente pequeña, deteniendo el entrenamiento por completo.

Este método es sensible a las condiciones iniciales de los parámetros y los gradientes correspondientes. Si los gradientes iniciales son grandes, las tasas de aprendizaje serán bajas durante el resto del entrenamiento. Esto puede combatirse aumentando la tasa de aprendizaje global, lo que hace a AdaGrad esté condicionando a la elección de la tasa de aprendizaje.

El método Adadelta consigue superar la sensibilidad a la selección de hiperparámetros, así como para el continuo decaimiento de las tasas de aprendizaje incorporando los siguientes métodos [22]:

$$\text{Se inicializa con} \quad \Delta x = 0 \ , \ \alpha = 0 \tag{C.3}$$

$$\alpha_t = \rho \cdot \alpha_{t-1} + (1 - \rho) g_t^2 \tag{C.4}$$

donde ρ es una constante de decaimiento similar a la utilizada en el método del *momentum*.

$$\Delta \omega_t = -\frac{\sqrt{\Delta x_t + \epsilon}}{\sqrt{\alpha_t + \epsilon}} g_t \tag{C.5}$$

$$\Delta x_t = \rho \cdot \Delta x_{t-1} + (1 - \rho) \Delta \omega_t^2 \tag{C.6}$$

$$\omega_{t+1} = \omega_t + \Delta \omega_t \tag{C.7}$$

En lugar de acumular la suma de gradientes al cuadrado a lo largo de todo el tiempo, restringimos la ventana de gradientes pasados que se acumulan a un tamaño fijo. Con esta ventana de acumulación, el denominador de AdaGrad no puede acumularse hasta el infinito, sino que se convierte en una estimación local utilizando gradientes recientes. Esto asegura que el aprendizaje continúa progresando incluso después de que se hayan realizado muchas iteraciones de actualizaciones [22].

Apéndice D

Software y Hardware

Detallaremos el software y hardware utilizados en el desarrollo de la red neuronal convolucional y todos los experimentos, así como el lenguaje de programación utilizado para tratar los datos.

D.1. Software

Para llevar a cabo el desarrollo de la red neuronal convolucional y todos los experimentos, se ha utilizado el *software* **DragonFly** 2022.2 [33]. Dragonfly es un programa informático, con una interfaz intuitiva y amigable, especializado en el análisis y visualización de datos volumétricos, como imágenes médicas y datos de microscopía. Ofrece una variedad de herramientas y funciones avanzadas para explorar, segmentar y analizar estos datos de manera interactiva. Además, el programa permite la integración de técnicas de aprendizaje automático y redes neuronales para la segmentación y el análisis automatizado de los datos. Utiliza principalmente el lenguaje de programación Python y una variedad de bibliotecas y marcos de trabajo populares para implementar y entrenar redes neuronales. Algunas de las librerías comúnmente utilizadas en Dragonfly son TensorFlow y Keras.

Tensorflow es una biblioteca desarrollada por Google Brain para aplicaciones de aprendizaje automático y redes neuronales profundas. Construido sobre TensorFlow, Keras es una interfaz de alto nivel escrito en Python ampliamente utilizado debido a su facilidad de uso y su enfoque dirigido al usuario, así como su flexibilidad a la hora de realizar experimentos e ideas muy diversos de aprendizaje profundo. **Keras** facilita la construcción y el entrenamiento de redes

neuronales mediante una sintaxis simple y una amplia gama de capas y modelos predefinidos.

Para tratar algunos de los datos se ha utilizado el lenguaje de programación **Python** en su versión 3.8. e **ImageJ** [39], un programa de procesamiento digital de imagen de dominio público programado en Java.

D.2. Hardware

Especificaciones	
Sistema operativo	Windows 10 pro
Ordenador	Estación de trabajo Fujitsu Celsius R940
Procesador	Intel Xeon
Memoria	64 GB
Tarjeta gráfica	Nvidia GeForce GTX 1080 8 GB
SSD	1TB

Apéndice E

Coeficiente de Pearson

La covarianza entre las variables X e Y es una medida que sirve para evaluar la relación lineal existente entre las dos variables

$$\sigma_{XY} = Cov(X,Y) = \mathcal{E}[(X-\mu_X)(Y-\mu_Y)]\,, \tag{E.1}$$

donde μ_X y μ_Y son las esperanzas de X y Y respectivamente. Si (X,Y) es discreta y f_{XY} es su función de probabilidad, la covarianza puede expresarse como

$$\sigma_{XY} = Cov(X,Y) = \sum_{i=1}^{\infty}\sum_{j=1}^{\infty}(x_i-\mu_X)(y_j-\mu_Y)f_{XY}(x_i,y_j) \tag{E.2}$$

Si (X,Y) es continua y f_{XY} su función de densidad de probabilidad:

$$\sigma_{XY} = Cov(X,Y) = \int_{i=1}^{\infty}\int_{j=1}^{\infty}(x-\mu_X)(y-\mu_Y)f_{XY}(x,y)dxdy \tag{E.3}$$

El análisis bivariante es un método estadístico que examina cómo se relacionan dos cosas diferentes cuantitativamente. Implica el análisis de dos variables (a menudo denotadas como X, Y), con el fin de determinar la relación empírica entre ellas. En nuestro caso, estamos especialmente interesados en la independencia con la intensidad.

En general, si X e Y son dos variables aleatorias con una distribución de probabilidad bivariante, su covarianza, en cierto sentido, refleja la dirección y la cantidad de asociación o correspondencia entre las variables, [49, Capítulo 11, *Measures of Association for Bivariate*

Samples]. El coeficiente de correlación de Pearson es una medida de la relación lineal entre X e Y y se definido como

$$p(X,Y) = \frac{cov(X,Y)}{[var(X)var(Y)]^{1/2}} \ . \tag{E.4}$$

La varianza de una variable aleatoria X es el valor esperado de las desviación al cuadrado de la media de X, $\mu = E[X]$. La varianza también puede considerarse como la covarianza de una variable aleatoria consigo misma $var(X) = cov(X,X)$.

En este trabajo, X_i, Y_i, μ_X y μ_Y son los valores de intensidad del píxel i y de intensidad media de las imágenes X_i (sintetizada) e Y_i (ground-truth) respectivamente. Este coeficiente es invariante ante cambios de escala y localización en X e Y, y en estadística clásica este parámetro se suele utilizar como medida relativa de asociación en una distribución bivariante. El valor absoluto del coeficiente de correlación no supera 1, y su signo viene determinado por el signo de la covarianza.

Bibliografía

[1] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," vol. 9351, pp. 234–241. Lecture Notes in Computer Science, 2015.

[2] Y. Demirel, "Chapter 12 - thermodynamics and biological systems," in *Nonequilibrium Thermodynamics*, Y. Demirel, ed., pp. 293–355. Elsevier Science, Amsterdam, 2002. https://www.sciencedirect.com/science/article/pii/B9780444508867500126.

[3] J.-J. Fernandez, "Computational methods for electron tomography," *Micron* **43** no.˜10, (2012) 1010–1030. https://www.sciencedirect.com/science/article/pii/S0968432812001540.

[4] J.-J. Fernandez and A. Martinez-Sanchez, "Computational methods for three-dimensional electron microscopy (3dem)," *Computer Methods and Programs in Biomedicine* **225** (2022) 107039. https://www.sciencedirect.com/science/article/pii/S0169260722004217.

[5] J.-J. Fernandez, "Criotomografía electrónica," *Revista de la Sociedad Española de Bioquímica y Biología Molecular* **201** (2019) 19–24. https://revista.sebbm.es/articulo.php?id=595&url=criotomografia-electronica.

[6] J. Frank, *Electron Tomography*. Springer New York, NY, 2010.

[7] E. Chávez, E. Peña, V. Luque, and C. Prudencio, "La transformada de fourier para explicar el proceso de tomografía computarizada," *Pesquimat* **20** (09, 2017) 77.

[8] L. A. Shepp and B. F. Logan, "The fourier reconstruction of a head section," *IEEE Transactions on Nuclear Science* **21** no.˜3, (1974) 21–43.

[9] N. T. Vo, M. Drakopoulos, R. Atwood, and C. Reinhard, "Reliable method for calculating the center of rotation in parallel-beam tomography," *Optics Express* **22** (07, 2014) 19078–19086.

[10] S. Galindo Uribarri, "Principios matemáticos de la reconstrucción de imágenes tomográficas," *CIENCIA ergo-sum* **10, Nº 3** (2003) 271–281. https://dialnet.unirioja.es/servlet/articulo?codigo=5128966.

[11] J.-J. Fernandez, "Computational methods for materials characterization by electron tomography," *Current Opinion in Solid State and Materials Science* **17** no.˜3, (2013) 93–106. https://www.sciencedirect.com/science/article/pii/S1359028613000156. Electron Tomography.

[12] J. Todd, *Digital image processing*. Optics and Lasers in Engineering, 1988.

[13] W. Burger and M. J. Burge, *Digital Image Processing: An Algorithmic Introduction*. Springer International Publishing, Cham, 2022. https://doi.org/10.1007/978-3-031-05744-1_4.

[14] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[15] J. Torres, *Python Deep Learning: Introducción práctica con Keras y TensorFlow 2*. 2020. https://torres.ai/python-deep-learning/.

[16] F. Chollet, *Deep Learning with Python*. Manning, Nov., 2017.

[17] W. S. McCulloch and W. Pitts, "The perceptron: A probabilistic model for information storage and orfanization in the brain," *Psychological Review* **65** no. 6, (1958) 832–837.

[18] F. Rosenblatt, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics* **5** (1943) 115–133. https://doi.org/10.1007/BF02478259.

[19] M. Schmitt, "On the Complexity of Computing and Learning with Multiplicative Neural Networks," *Neural Computation* **14** no.˜2, (02, 2002) 241–301, https://direct.mit.edu/neco/article-pdf/14/2/241/815069/089976602527411 21.pdf. https://doi.org/10.1162/089976602527411 21.

[20] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature* **323** (1986) 533–536.

[21] F. Chollet *et al.*, "Keras," 2015. https://github.com/fchollet/keras.

[22] M. Zeiler, "Adadelta: An adaptive learning rate method,". https://doi.org/10.48550/arXiv.1212.5701.

[23] P. Radiuk, "Impact of training set batch size on the performance of convolutional neural networks for diverse datasets," *Information Technology and Management Science* **20** (12, 2017) 20–24.

[24] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, G. Gordon, D. Dunson, and M. Dudík, eds., vol. 15 of *Proceedings of Machine Learning Research*, pp. 315–323. PMLR, Fort Lauderdale, FL, USA, 11–13 apr, 2011. https://proceedings.mlr.press/v15/glorot11a.html.

[25] Y. LeCun and Y. Bengio, *Convolutional Networks for Images, Speech, and Time Series*, p. 255–258. MIT Press, Cambridge, MA, USA, 1998.

[26] S. Sakhavi, *APPLICATION OF DEEP LEARNING METHODS IN BRAIN-COMPUTER INTERFACE SYSTEMS*. PhD thesis, 01, 2017.

[27] Y. Xue, F. G. Farhat, O. Boukrina, A. Barrett, J. R. Binder, U. W. Roshan, and W. W. Graves, "A multi-path 2.5 dimensional convolutional neural network system for segmenting stroke lesions in brain mri images," *NeuroImage: Clinical* **25** (2020) 102118. https://www.sciencedirect.com/science/article/pii/S2213158219304656.

[28] S. Liu, D. Zhang, Y. Song, H. Peng, and W. Cai, "Triple-crossing 2.5d convolutional neural network for detecting neuronal arbours in 3d microscopic images," vol. 10541, pp. 185–193. Lecture Notes in Computer Science, Springer Science, 09, 2017.

[29] R. Sanchez-Garcia, J. Gomez-Blanco, A. Cuervo, J. Carazo, C. Sorzano, and J. Vargas, "Deepemhancer: a deep learning solution for cryo-em volume post-processing," *Commun Biol* **24** (07, 2021) .

[30] M. Weigert, U. Schmidt, T. Boothe, A. Müller, *et al.*, "Content-aware image restoration: Pushing the limits of fluorescence microscopy," *Nature Methods* **15** (2018) 1090–1097. https://doi.org/10.1038/s41592-018-0216-7.

[31] A. Krull, T.-O. Buchholz, and F. Jug, "Noise2void - learning denoising from single noisy images," pp. 2124–2132. Conference on Computer Vision and Pattern Recognition (CVPR), 06, 2019.

[32] J. Lehtinen, J. Munkberg, J. Hasselgren, S. Laine, T. Karras, M. Aittala, and T. Aila, "Noise2noise: Learning image restoration without clean data," vol. 7 of *Proceedings of Machine Learning Research*, pp. 4620–4631. International Machine Learning Society, 2018.

[33] Object Research Systems (ORS), "Dragonfly." http://www.theobjects.com/dragonfly.

[34] P. Billingsley, *Probability and Measure*. John Wiley and Sons, second ed., 1986.

[35] A. Martinez-Sanchez, I. Garcia, S. Asano, V. Lucic, and J. J. Fernandez, "Robust membrane detection based on tensor voting for electron tomography," *Journal of Structural Biology* **186** (2014) 49–61.

[36] C. Sherman and J. Butler, *Transducers and Arrays for Underwater Sound*. Springer Science and Business Media, 2007.

[37] T. R. Shaikh, X. Gao, , W. T. Baxter, F. J. Asturias, N. Boisset, A. Leith, and J. Frank, "SPIDER image processing for single-particle reconstruction of biological macromolecules from electron micrographs," *Nature Protocols* **3** (2008) 1941–1974.

[38] E. Höck, T.-O. Buchholz, A. Brachmann, F. Jug, and A. Freytag, *N2V2 - Fixing Noise2Void Checkerboard Artifacts with Modified Sampling Strategies and a Tweaked Network Architecture*, pp.˜503–518. Springer Nature Switzerland, 02, 2023.

[39] C. A. Schneider, W. S. Rasband, and K. W. Eliceiri, "Nih image to imagej: 25 years of image analysis," *Nat Meth* **9** no. 7, (July, 2012) 671–675. http://dx.doi.org/10.1038/nmeth.2089.

[40] A. Lucchi, K. Smith, R. Achanta, G. Knott, and P. Fua, "Supervoxel-based segmentation of mitochondria in em image stacks with learned shape features," *IEEE Transactions on Medical Imaging* **31** no.˜2, (2012) 474–486. https://www.epfl.ch/labs/cvlab/data/data-em/.

[41] J. Hennies, J. Serra Lleti, N. Schieber, R. Templin, A. Steyer, and Y. Schwab, "Amst: Alignment to median smoothed template for focused ion beam scanning electron microscopy image stacks," *Scientific Reports* **10** (02, 2020) . https://doi.org/10.1038/s41598-020-58736-7.

[42] F. K. M. Schur, M. Obr, W. J. H. Hagen, W. Wan, A. J. Jakobi, J. M. Kirkpatrick, C. Sachse, H.-G. Kräusslich, and J. A. G. Briggs, "An atomic model of hiv-1 capsid-sp1 reveals structures regulating assembly and maturation," *Science* **353** no.˜6298, (2016) 506–508, https://www.science.org/doi/pdf/10.1126/science.aaf9620. https://www.science.org/doi/abs/10.1126/science.aaf9620.

[43] M. Guay, Z. Emam, A. Anderson, M. Aronova, I. Pokrovskaya, B. Storrie, and R. Leapman, "Dense cellular segmentation for em using 2d–3d neural network ensembles," *Scientific Reports* **11** (01, 2021) 2561.

[44] A. Arnab, S. Zheng, S. Jayasumana, B. Romera-Paredes, M. Larsson, A. Kirillov, B. Savchynskyy, C. Rother, F. Kahl, and P. Torr, "Conditional random fields meet deep neural networks for semantic segmentation: Combining probabilistic graphical models with deep learning for structured prediction," *IEEE Signal Processing Magazine* **35** (01, 2018) 37–52.

[45] R. Makovetsky, N. Piché, and M. Marsh, "Dragonfly as a platform for easy image-based deep learning applications," *Microscopy and Microanalysis* **24** (08, 2018) 532–533.

[46] L. R. Dice, "Measures of the amount of ecologic association between species," *Ecology* **26** no. 3, (1945) 297–302. http://www.jstor.org/stable/1932409.

[47] D. Wallace, "Mitochondria and cancer," *Nature Reviews Cancer* **12** (09, 2012) 685–698.

[48] D. L. Johannsen and E. Ravussin, "The role of mitochondria in health and disease," *Current Opinion in Pharmacology* **9** no.˜6, (2009) 780–786. https://www.sciencedirect.com/science/article/pii/S1471489209001350. Gastrointestinal/Endocrine and metabolic diseases.

[49] J. Gibbons and S. Chakraborti, *Nonparametric Statistical Inference, Fourth Edition: Revised and Expanded*. Taylor & Francis, 2014. https://books.google.es/books?id=kJbVO2G6VicC.

[50] R. C. Gonzalez, "Deep convolutional neural networks [lecture notes]," *IEEE Signal Processing Magazine* **35** no.˜6, (2018) 79–87.

[51] G. E. Hinton and J. McClelland, "Learning representations by recirculation," in *Neural Information Processing Systems*, D. Anderson, ed., vol. 0. American Institute of Physics, 1987. https://proceedings.neurips.cc/paper_files/paper/1987/file/35f4a8d465e6e1edc05f3d8ab658c551-Paper.pdf.

[52] D. Bank, N. Koenigstein, and R. Giryes, "Autoencoders,". https://doi.org/10.48550/arXiv.2003.05991.

[53] S. Shalev-Shwartz and S. Ben-David, "Understanding machine learning: From theory to algorithms," *Understanding Machine Learning: From Theory to Algorithms* (01, 2013) .