Universidad de la República
Facultad de Ingeniería

PAIM-3D: Pipeline de Análisis de Imágenes de Microscopía Confocal

Memoria de proyecto presentada a la Facultad de Ingeniería de la Universidad de la República por

Emiliano Merlo, María José Millán, Diego Silvera

en cumplimiento parcial de los requerimientos para la obtención del título de Ingeniero Electricista.

Tutor

Federico Lecumberry ..... Universidad de la República
Álvaro Gómez ..... Universidad de la República

Tribunal

Pablo Cancela ..... Universidad de la República
Federico Lecumberry ..... Universidad de la República
Laura Martínez-Palma ..... Universidad de la República

Montevideo
lunes 12 diciembre, 2022

PAIM-3D: Pipeline de Análisis de Imágenes de Microscopía Confocal, Emiliano Merlo, María José Millán, Diego Silvera.

Esta tesis fue preparada en LaTeX usando la clase iietesis (v1.1).
Contiene un total de 178 páginas.
Compilada el lunes 12 diciembre, 2022.
http://iie.fing.edu.uy/

Si lo que tenés que hacer lleva cinco minutos, hacelo.

Emiliano Merlo

Esta página ha sido intencionalmente dejada en blanco.

Agradecimientos

En primer lugar queremos agradecer a nuestros colegas, amigos y familiares. Su paciencia, contención, apoyo, aliento y motivación fueron pilares fundamentales para realizar este proyecto. Queremos agradecer a nuestros tutores, Álvaro y Fefo, por guiarnos a lo largo de este camino y ayudarnos a finalizar una etapa tan importante como lo es este proyecto. Al grupo IMAGINA, especialmente a Erik Winiarski, por facilitarnos las imágenes utilizadas y colaborar con información valiosa para el trabajo realizado. A la Universidad de la República, especialmente a nuestra casa de estudios, la Facultad de Ingeniería y el Instituto de Ingeniería Eléctrica, en cuyos espacios nuestra formación fue llevada a cabo.

Esta página ha sido intencionalmente dejada en blanco.

Resumen

La microscopía es el conjunto de técnicas y métodos destinados a la visualización de objetos que, por ser muy pequeños, están fuera del rango de resolución del ojo humano. En medicina y biología se utiliza la microscopía especialmente para analizar tejidos, células, componentes sanguíneos y microorganismos. En particular, la microscopía fluorescente confocal recoge y detecta la luz emitida por moléculas fluorescentes situadas en un mismo plano del espacio tridimensional, pudiéndose realizar cortes virtuales de las muestras analizadas.

Este proyecto consiste en el estudio de las distintas etapas de procesamiento de imágenes adquiridas mediante microscopía fluorescente confocal. Las etapas analizadas son deconvolución, segmentación y extracción de parámetros morfológicos. Para cada una de estas etapas se estudiaron distintos métodos, con el objetivo de evaluar sus desempeños. Además, se realiza un código de Python que permite a cualquier usuario utilizar los métodos analizados.

Dado que se estudia un enfoque orientado al aprendizaje profundo y esto requiere muchos datos, también fue necesaria la implementación de un generador de imágenes sintéticas.

Finalmente se analizan distintos métodos de clasificación en un conjunto de datos de monocitos en muestras de sangre procesados con el pipeline. Algunas de estas muestras son individuos de control, mientras que otras son de pacientes de Esclerosis Lateral Amiotrófica (ELA).

Esta página ha sido intencionalmente dejada en blanco.

Tabla de contenidos

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 1

Introducción

1.1. Motivación

Probablemente, el desarrollo y la amplia disponibilidad comercial del microscopio confocal sea uno de los avances más significativos de las aplicaciones de la microscopía a las ciencias de la vida [1]. El motivo de su popularidad está dado por la posibilidad de estudiar especímenes de tejidos biológicos en tres dimensiones con la alta sensibilidad y nitidez que brinda la microscopía de fluorescencia.

El modelado en tres dimensiones de distintos organelos es fundamental para extraer características morfológicas de interés médico, dado que muchas veces estas características están directamente ligadas con distintas patologías, enfermedades, anomalías y tratamientos. Sus diferentes aplicaciones abarcan la posibilidad de realizar análisis desde un nivel tejido-estructural hasta el molecular. Los avances científicos que esta técnica aporta, la consolidan como una técnica de vanguardia que permite fortalecer la investigación clínica [2].

Un posible caso de aplicación es el estudio de la mitocondrias, dado que alteraciones en la función, dinámica y/o morfología mitocondrial han sido postuladas como mecanismos patogénicos para diversas enfermedades neurodegenerativas. Algunas de estas enfermedades son el Parkinson, Alzheimer, Huntington y Esclerosis Lateral Amiotrófica (ELA), afectando esta última a las neuronas en el cerebro, el tronco cerebral y la médula espinal que controlan el movimiento de los músculos voluntarios.

Lamentablemente, al día de hoy se desconoce la causa de la ELA en la mayoría de los casos y su diagnóstico viene dado por la sintomatología del paciente, lo que es problemático dado que existen muchas enfermedades cuyos síntomas son similares. Es por esto que desde la Facultad de Medicina surge el interés de estudiar la morfología mitocondrial en los pacientes de dicha enfermedad e investigar posibles diferencias con muestras de individuos saludables.

El procesamiento de imágenes de microscopía comienza con técnicas fundamentales destinadas a reproducir con mayor precisión la información contenida en una muestra microscópica. Esto podría incluir ajuste de brillo y contraste, reducción de ruido y corrección de faltas de uniformidad en la iluminación. Dicho proce-

samiento implica únicamente operaciones aritméticas básicas entre imágenes (es decir, suma, resta, multiplicación y división). La gran mayoría del procesamiento realizado en la imagen del microscopio es de esta naturaleza.

A veces, se emplean técnicas avanzadas con el objetivo de revertir la distorsión del camino óptico del microscopio, eliminando así las distorsiones y el desenfoque causado por la instrumentación. Este proceso se llama deconvolución y será abordado en profundidad en este trabajo.

Dada la necesidad de extraer características morfológicas de las muestras, es fundamental definir exactamente a qué parte de la imagen se le extraerán dichas características. En otras palabras, es necesario definir qué partes de las imágenes son efectivamente relevantes (dichas partes de las imágenes serán llamadas información o señal a lo largo del documento) y qué partes no son de importancia (fondo, *background*). Este proceso es conocido como segmentación y es crucial para la correcta extracción de características.

Este proyecto busca brindar una herramienta de análisis que permita la obtención de información relevante en imágenes de microscopía confocal mediante la deconvolución, segmentación y extracción de distintos parámetros morfológicos de las estructuras biológicas. Las imágenes estudiadas en este trabajo fueron adquiridas en el *Departamento de Histología y Embriología* de la Facultad de Medicina por Erik Winiarski, estudiante de maestría e investigador de IMAGINA [3].

1.2. Objetivos y alcance

El objetivo principal del proyecto es brindar una herramienta que pueda ser utilizada a futuro para el análisis de imágenes de microscopía confocal.

Se plantean dos objetivos específicos, por un lado, idear e implementar un pipeline que permita, a través de imágenes adquiridas por microscopía confocal, la deconvolución, segmentación y medición de características morfológicas del organelo a estudiar. En las etapas de deconvolución y segmentación se investigará el estado del arte y se estudiarán los métodos más utilizados. Se seleccionarán algunos de los métodos estudiados para incluir en el pipeline, pudiendo incluso reutilizar trabajos previos. Queda por fuera del alcance de este proyecto la implementación propia de métodos de deconvolución y segmentación.

Por otro lado, se analizarán las mediciones morfológicas obtenidas, correlacionándolas con la etiqueta (individuo saludable o paciente de ELA) brindada por los referentes médicos. En base al anterior análisis se explorarán distintos métodos de clasificación como posible aplicación del pipeline.

El número de imágenes proporcionadas por la Facultad de Medicina no es suficiente para entrenar algoritmos de aprendizaje automático. Por lo tanto, otro objetivo del proyecto incluye la generación de una base de datos sintética.

A continuación, se muestran los criterios de éxito definidos para el proyecto:

- Establecer contacto periódico con los referentes médicos para adquirir experiencia trabajando en forma interdisciplinaria, logrando comprender el objeto de estudio de la otra disciplina y llevarlo al dominio ingenieril. Entender

cómo funciona el proceso de adquisición de las imágenes.

- Crear o compilar una base de datos con la cual no solo nosotros podamos trabajar, sino que pueda ser utilizada para futuros trabajos.

- Analizar, comparar, discutir e identificar un algoritmo de segmentación que realice la media, votación u otra combinación entre la opinión de distintos profesionales en el área. Se evaluará la segmentación en base a la votación de los referentes médicos, apuntando a una segmentación correcta en por lo menos 70 % de los casos.

- Obtener parámetros morfológicos variados, haciendo mayor el abanico de posibilidades a la hora de la clasificación. Se propone relevar por lo menos 7 características morfológicas.

- Realizar un pipeline enfocado en el análisis de imágenes obtenidas mediante microscopía de fluorescencia confocal y lograr que sea aplicable a otros organelos.

1.3. Antecedentes

Actualmente se está viviendo una etapa de mucho avance en el área del procesamiento de imágenes por computadora, con algoritmos que se podrían denominar clásicos y otros más modernos que utilizan de las nuevas tecnologías. El auge del aprendizaje automático y en particular de las redes neuronales está llevando a nuevos niveles la capacidad de procesar imágenes, con resultados que año a año van mejorando.

La adquisición de señales agrega ruido de distintas características y el caso de la microscopía confocal no es una excepción. A su vez, se incorporan operaciones no lineales como la convolución con una Point Spread Function (PSF) que degradan la imagen, lo cual se detallará en el capítulo 2. Para mitigar estos efectos existe la etapa de deconvolución, en la que se busca contrarrestar los efectos de la PSF.

Existen diversos algoritmos de deconvolución, entre los que destaca el software Huygens Professional, el cual ofrece herramientas de deconvolución y restauración en imágenes de microscopía. Huygens ofrece implementaciones de vanguardia de los algoritmos Classic Maximum Likelihood Estimation [4] (CMLE) y Good's Roughness Maximum Likelihood Estimation [5](GMLE), que funcionan de manera rápida y eficiente en imágenes con ruido. Otros métodos que se utilizan actualmente son: *Tikhonov-Miller* [6] (TM), *Richardson-Lucy* [7] (RL) y *Richardson-Lucy with Total Variation* [8] (RLTV), entre otros.

Para el análisis morfológico de las estructuras biológicas, es necesaria una primera etapa de procesamiento para delimitar la región de interés del cuerpo a estudiar, esto en muchos casos es realizado de forma manual por los grupos de investigación. Esta tarea se vuelve tediosa y subjetiva, por lo que automatizarla contribuiría a un mejor estudio de los parámetros a relevar. Los algoritmos de

segmentación cumplen este fin, tomando una imagen y retornando una máscara binaria con la región de interés.

Para segmentar existen varias técnicas, entre ellas, la umbralización, el *clustering* o incluso las redes neuronales convolucionales. La arquitectura de red convolucional U-Net es uno de los modelos más exitosos para la segmentación de imágenes de microscopía hoy en día. En estudios recientes, los enfoques basados en U-Net han mostrado rendimiento de vanguardia para el desarrollo de sistemas de diagnóstico asistidos por computadora para el diagnóstico temprano y el tratamiento de enfermedades como tumores cerebrales, cáncer de pulmón, Alzheimer, cáncer de mama, entre otros [9].

A continuación se especifican algunas de las herramientas y antecedentes que fueron utilizados para este proyecto.

Para la implementación de la etapa de deconvolución, se utilizará Fiji [10], un paquete de procesamiento de imágenes construido sobre la base de ImageJ [11]. En particular, se utilizará el plugin Deconvolution Lab 2 [12] herramienta de código libre para deconvolución de imágenes 3D de microscopía.

DeepSynth [13] es una herramienta automática para segmentación 3D de volúmenes sintéticos que utiliza modelos de aprendizaje profundo. Los autores generaron una base sintética de núcleos con la cual entrenaron una versión modificada de U-Net, una red neuronal convolucional (CNN) 3D. Comprobaron que este método se encuentra por encima de los métodos tradicionales, logrando además evitar la segmentación manual por parte de un experto. En el presente trabajo se busca automatizar métodos de segmentación sobre imágenes de redes mitocondriales, por lo que se basará en la implementación de U-Net de DeepSynth, tomando también las ideas de generación de una base sintética similar a las imágenes a estudiar.

Zanin *et al* [14] realizaron un análisis de la interacción entre mitocondrias a partir de imágenes 3D. Con el fin de estudiar la existencia de patrones característicos en la interacción mitocondrial en pacientes que padecen la enfermedad de Parkinson, extrajeron características que describen la conectividad de la red mitocondrial. El presente proyecto busca realizar un análisis similar sobre pacientes con ELA, por lo que se basará en el trabajo de Zanin *et al* [14] para la etapa de extracción de características de la conectividad de la red.

1.4. Diagrama de flujo del sistema

En el proyecto se propone construir un pipeline que reciba volúmenes obtenidos mediante microscopía fluorescente confocal, pase por una etapa de deconvolución para mitigar los efectos del PSF, luego por algoritmos de segmentación que permitan delimitar la región de interés y finalmente la medición de las características morfológicas de los cuerpos detectados.

En el diagrama de flujo de la Figura 1.1 pueden visualizarse las distintas etapas del pipeline a abordar. Como se mencionó anteriormente, la etapa de clasificación se concibe como una posible aplicación del pipeline a desarrollar; esto se muestra en la Figura 1.2.

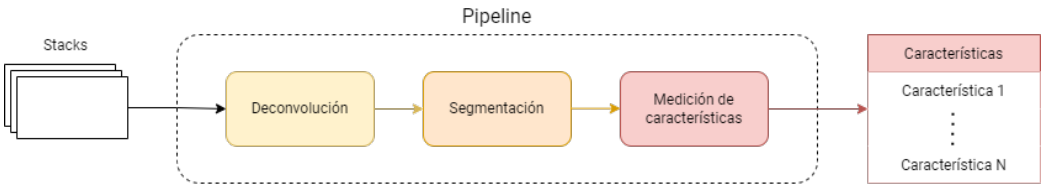


Figura 1.1: Diagrama de flujo del pipeline

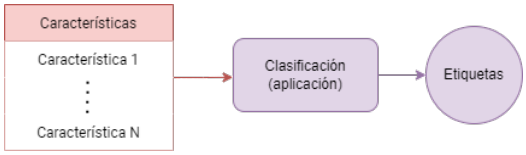


Figura 1.2: Ejemplo de aplicación del resultado del pipeline.

1.5. Organización del documento

Este documento tiene como objetivo explicar las distintas etapas del trabajo realizado. Esto incluye los métodos estudiados, los experimentos realizados en cada etapa del pipeline, la generación de datos sintéticos y la clasificación a partir de los datos reales.

El **Capítulo 2** comienza con una breve introducción a las imágenes de microscopía fluorescente confocal, explicando el funcionamiento de los microscopios para así comprender el fenómeno físico que genera la imagen. Posteriormente, se presenta cómo se modela matemáticamente el degradado de la imagen por los efectos de la *Point Spread Function*, el ruido gaussiano y el ruido de Poisson. Se introduce la base de datos brindada por los referentes médicos y se detallan los parámetros utilizados para la adquisición. En este Capítulo a su vez, se detalla como funciona el generador de datos sintéticos y se muestran algunos ejemplos de lo obtenido. Se propone modelar el ruido a través de redes generativas y se muestran los resultados obtenidos para dos tipos de redes *SpCycleGan* y *Pix2pix*. Finalmente, se presentan las bases de datos a utilizar en las distintas etapas del pipeline.

La etapa de deconvolución es documentada en el **Capítulo 3**, se presenta el marco teórico que fundamenta los distintos métodos probados, utilizándose principalmente métodos probabilísticos. Se detalla la formulación de los distintos métodos utilizados y cómo estiman la señal objetivo. Por último, se muestran los experimentos realizados sobre los distintos métodos y cómo influyen los parámetros.

En el **Capítulo 4** se procede de forma análoga al Capítulo 3, se presenta un marco teórico detallando el funcionamiento de los métodos a probar con especial hincapié en las redes neuronales convolucionales, ya que se probará una *U-Net/ResNet*. Para concluir, se muestran los distintos experimentos realizados con los métodos de segmentación e incorporando transformaciones morfológicas.

El **Capítulo 5** es dedicado a la extracción de parámetros morfológicos, se profundiza sobre los distintos descriptores tanto de las estructuras como del esqueleto de las mismas. A su vez, se comparan los parámetros adquiridos a la salida del pipeline con los obtenidos sobre las imágenes binarias sintéticas originales, pudiendo

comparar que tanto varían estos parámetros entre los stacks originales y luego de pasar por el pipeline.

La clasificación es documentada en el **Capítulo 6**. En este Capítulo se explaya sobre los fundamentos del aprendizaje automático y la posibilidad de las computadoras de aprender de datos sin ser explícitamente programadas. Se realiza un análisis de datos de los distintos parámetros obtenidos en la Capítulo 5 y se entrenan modelos típicos de aprendizaje automático para clasificar entre paciente sano o patológico.

Por último, en el **Capítulo 7** se presentan las conclusiones del proyecto y el trabajo a futuro.

Capítulo 2

Contexto, adquisición y generación de bases de datos

La obtención de información relevante en las imágenes estudiadas requiere conocer los fundamentos de la adquisición. Se debe comprender cómo se forma la imagen y los distintos factores que influyen en la degradación de la misma al ser adquirida. En el presente capítulo se realizará una introducción a la adquisición mediante microscopía confocal de fluorescencia, presentando luego el modelo matemático para este proceso. Se profundizará en la implementación de un generador sintético de volúmenes, el cual tiene como objetivo brindar imágenes similares a las adquiridas con microscopía confocal. Finalmente, se presentarán las bases de datos a utilizarse a lo largo de todo el trabajo.

2.1. Microscopía de fluorescencia confocal

La fluorescencia es uno de los fenómenos físicos más utilizados en la microscopía biológica y analítica, principalmente por su alta sensibilidad y alta especificidad [15]. Con la ayuda de colorantes fluorescentes, la microscopía de fluorescencia se puede utilizar para detectar una variedad de estructuras como por ejemplo mitocondrias [16].

La microscopía de fluorescencia es un tipo especial de microscopía óptica, en la cual se utiliza una fuente de luz de ancho de banda selectivo. El haz incidente debe tener una longitud de onda adecuada para excitar las moléculas fluorescentes o fluoróforos utilizados en la muestra, dichas moléculas luego emiten una luz de una longitud de onda levemente mayor.

En la Figura 2.1 se presenta un esquema del funcionamiento del microscopio de fluorescencia. Para separar los dos caminos de luz se utilizan filtros; un filtro de excitación que deja pasar solamente la longitud de onda del láser, y un filtro de emisión que deja pasar únicamente la longitud de onda de la luz emitida por la muestra. Para direccionar el haz de luz incidente hacia la muestra se utiliza un lente dicroico[1]. La luz emitida es recogida por el lente objetivo, luego pasa por el

[1]Un lente dicroico es un filtro de color, que se utiliza para seleccionar el paso de luz en

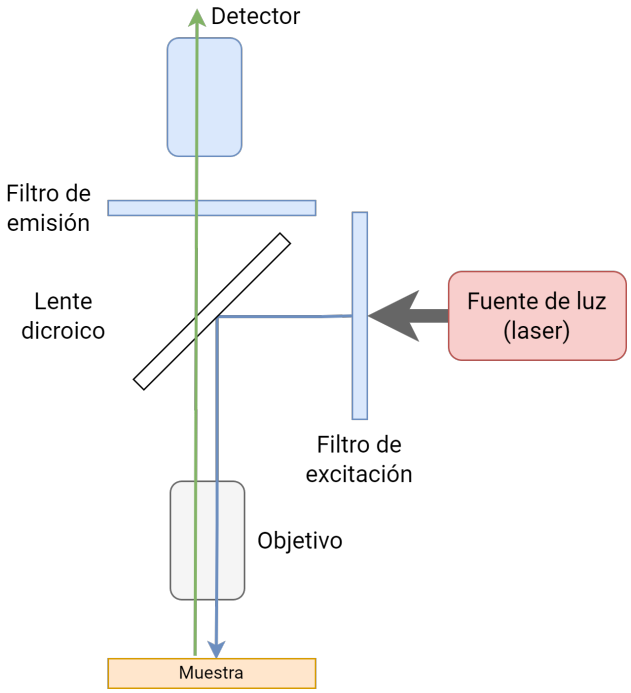


Figura 2.1: Esquema de funcionamiento de un microscopio de fluorescencia. La fuente de luz pasa a través de un filtro de excitación que permite el paso de una determinada longitud de onda, llega a una lente dicroica que permite direccionar el haz incidente hacia la muestra. De esta manera las moléculas fluorescentes utilizadas en la muestra son excitadas y emiten una luz de longitud de onda levemente mayor a la del haz incidente. La luz emitida es recogida por el lente objetivo, pasa por el filtro de emisión que permite el paso de una única longitud de onda y finalmente llega al detector (cámara u ojo del investigador).

filtro y finalmente llega al detector (cámara u ojo del investigador).

La microscopía de fluorescencia confocal es una técnica de adquisición de imágenes que permite una mayor resolución óptica en comparación con la microscopía de epifluorescencia convencional de "campo amplio" (*widefield*). A modo de establecer una comparación y resaltar las ventajas de la microscopía confocal, se mencionarán algunas características de ambos métodos.

En la microscopía de campo amplio, se ilumina toda la muestra al mismo tiempo, por lo que los fluoróforos emiten juntos y los fotones detectados provienen de toda la muestra. En el detector se recoge la luz emitida de múltiples planos focales, enfocados (*in-focus*) y desenfocados (*out-of-focus*). Esto provoca que en una muestra gruesa haya mucho *blur*[2], obteniendo como resultado una buena relación señal a ruido (SNR) pero baja resolución espacial. Esto último dificulta la detección de estructuras pequeñas.

Por otra parte, en la microscopía confocal la luz pasa por un orificio muy pequeño de manera que se bloquea la captación de la mayoría de luz no enfocada

una pequeña gama de colores y reflejan la luz de un color particular.

[2]En la sección 2.2 se desarrolla este concepto.

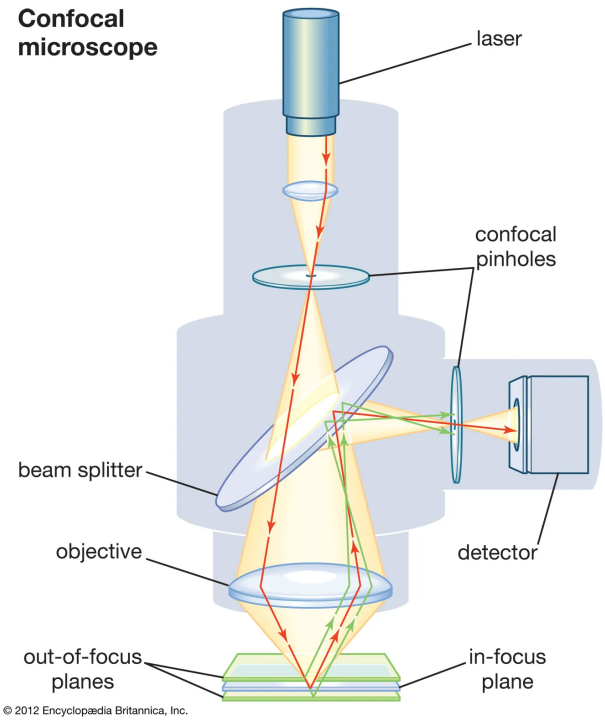


Figura 2.2: Esquema de funcionamiento de un microscopio confocal [17]. La luz emitida por la fuente pasa por un orificio muy pequeño (*confocal pinhole*) de manera que se bloquee la captación de la mayoría de luz no enfocada (*out-of-focus*) y se permite la detección de un único plano focal (*in-focus*). Esto permite obtener imágenes más nítidas y enfocadas. En la Tabla 2.1 se describen las distintas partes del microscopio confocal.

y se permite la detección de un único plano focal. Esto permite obtener imágenes más nítidas y enfocadas, aunque la intensidad de la luz detectada será menor que en el caso anterior. En la Figura 2.2 se muestra cómo está compuesto un microscopio confocal y en la Tabla 2.1 se describe brevemente cada una de sus partes.

2.2. Modelo de formación de la imagen

Al adquirir una imagen de una muestra puntual con un microscopio, la luz emitida por la misma se difracta al pasar por la apertura del objetivo, resultando en un patrón de interferencia tridimensional que se denomina *Point Spread Function* (PSF) [18]. La imagen adquirida estará formada por un centro brillante rodeado de anillos concéntricos claros y oscuros de manera alternada, lo cual recibe el nombre de disco de Airy [18,19]. En la Figura 2.3 se presenta un ejemplo de este fenómeno, junto con el gráfico de superficie de dicho disco. La PSF queda definida por el gráfico de superficie del disco de Airy [19]. En la Figura 2.4 se presenta mediante proyecciones ortogonales, una visualización tridimensional del patrón de Airy.

La PSF puede considerarse como la respuesta al impulso del sistema, puesto

Nombre	Descripción
láser *(laser)*	Fuente de luz que emite con una longitud de onda específica.
orificios confocales *(confocal pinholes)*	Orificios muy pequeños. El microscopio confocal cuenta con dos *pinholes*. El primero de ellos se encuentra entre la fuente de luz y el lente dicroico y permite iluminar una región pequeña de la muestra. El segundo permite recuperar la intensidad proveniente únicamente de dicha región.
lente dicroica *(beam splitter)*	Permite el paso de una longitud de onda específica. Permite que la luz proveniente de la fuente pase a través de sí, pero refleja la luz emitida por la muestra.
objetivo *(objective)*	Lente que recoge la luz emitida por la muestra, mediante una apertura en su parte posterior.
detector *(detector)*	Compuesto por un fotomultiplicador.

Tabla 2.1: Descripción de las distintas partes que componen el microscopio confocal.

que se obtiene como la respuesta del sistema frente a una muestra puntual. En la Figura 2.5 se esquematizan los efectos generados por el microscopio al realizar la adquisición. El primer fenómeno que ocurre es *blur*, esto es una redistribución de la intensidad en la imagen. Este fenómeno se modela como la convolución de la entrada con la PSF. El segundo fenómeno es el ruido, que esencialmente tiene dos fuentes diferentes, se tiene ruido debido a las fluctuaciones estocásticas en los

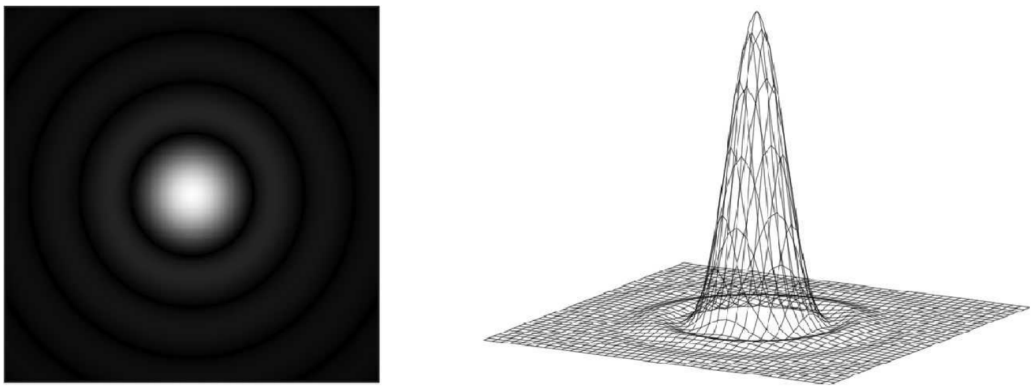


Figura 2.3: Disco de Airy y *Point Spread Function* (PSF). Debido a la difracción, el microscopio produce un disco borroso (A) al adquirir una imagen de una fuente puntual de luz. El gráfico de superficie de este disco define la PSF (B) [19].

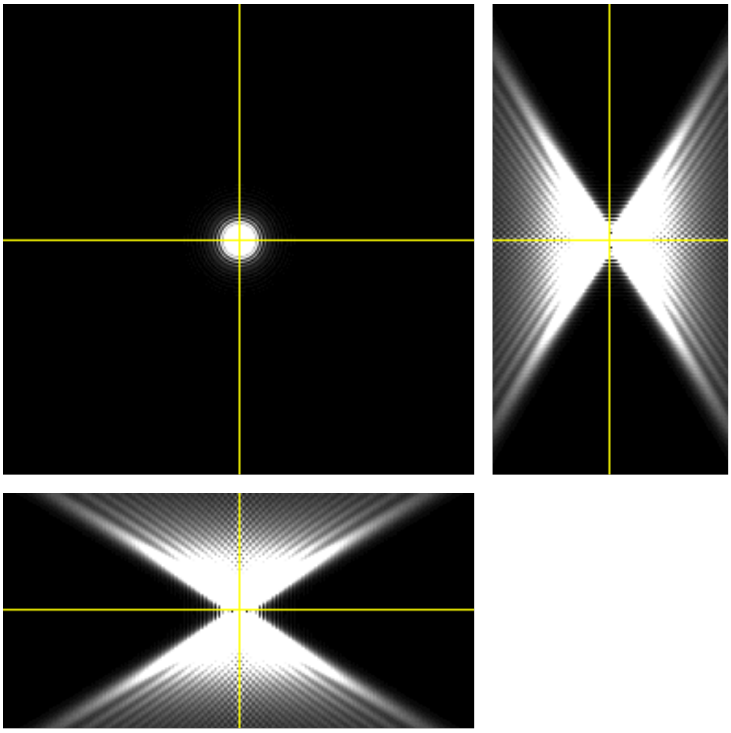
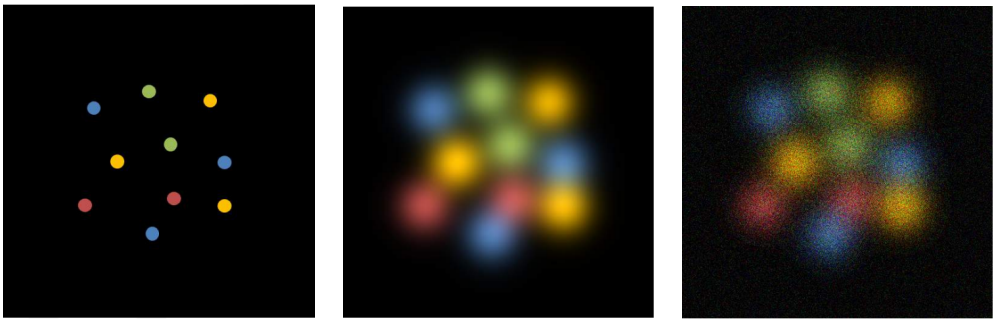


Figura 2.4: Visualización tridimensional del patrón de Airy.

tiempos de llegada de los fotones emitidos por el láser, y además ruido generado por los distintos componentes electrónicos del detector.

La transformación que realiza el microscopio en la adquisición se puede modelar según la ecuación (2.1), donde x e y representan la imagen ideal y la imagen adquirida respectivamente, h es la PSF y n corresponde al ruido, siendo u y v variables espaciales. Este último se modela como la suma entre un ruido aditivo gaussiano, producto de la suma de las distintas fuentes de ruido electrónico del microscopio, y un ruido de Poisson también aditivo que se debe al encolamiento de los fotones al momento de la emisión de luz. En la Figura 2.6 se presenta el



(a) Imagen ideal (b) Convolución con la PSF (c) Adición de ruido

Figura 2.5: Efectos generados en la adquisición por microscopía.

esquema del modelo de adquisición.

$$y(u,v) = x(u,v) * h(u,v) + n(u,v) \tag{2.1}$$

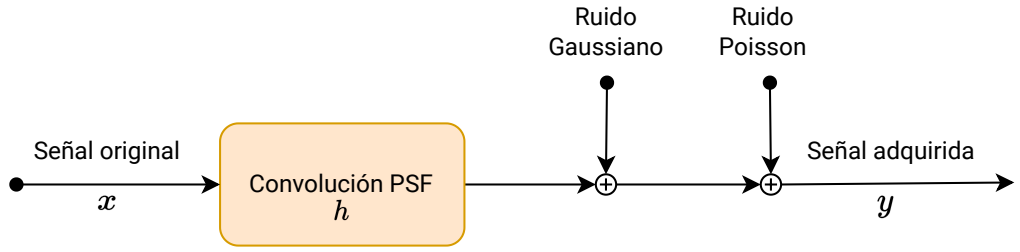


Figura 2.6: Esquema del modelo de formación de imágenes. x es la señal original que interesa estimar, h la PSF del sistema de adquisición e y la salida, que se modela como la ecuación (2.1).

Para lograr obtener información relevante de las imágenes es necesario caracterizar correctamente la PSF y las distintas fuentes de ruido presentes en la adquisición. En la Tabla 2.2 se describen brevemente los parámetros que determinan las características de la PSF.

2.3. Base de datos de IMAGINA

El set de datos proporcionado por el grupo IMAGINA consta de stacks de imágenes obtenidos mediante microscopía confocal. Las imágenes fueron extraídas de muestras de sangre de individuos con ELA y de individuos de control. En la Tabla 2.3 se muestra la cantidad de datos según el individuo.

Nombre	Descripción
Longitud de onda	Longitud de onda de la luz emitida por la muestra fluorescente.
NA	Apertura numérica del lente objetivo. Es la capacidad de la lente para recoger la luz a una distancia fija de la muestra. [20].
Índice de refracción	Determina la medida en que la luz se refractará al pasar a través del líquido de inmersión. La presencia del líquido de inmersión entre la lente del objetivo y la muestra aumenta la resolución. [21].
Tamaño de pixel en xy y paso-z	Son ajustados de forma digital, definen la resolución de la imagen.

Tabla 2.2: Descripción de los parámetros que determinan las características de la PSF.

	Control	Paciente
Mitocondrias	281	210
Núcleos	106	120

Tabla 2.3: Cantidad total de stacks de mitocondrias y núcleos según los individuos (paciente o control).

La PSF del microscopio fue estimada con el plugin "PSF Generator" de Fiji [22], el cual ofrece distintos modelos ópticos 3D para la generación junto con la posibilidad de configurar los parámetros del microscopio. El modelo óptico utilizado fue *Born & Wolf* [23], puesto que este modelo describe la difracción al pasar por una apertura circular cuando la fuente de luz está enfocada.

Los datos proporcionados cuentan con todos los metadatos necesarios para la generación de la PSF; en la Tabla 2.4 se presentan los parámetros utilizados. En la Figura 2.7 se presenta la visualización mediante proyecciones ortogonales del patrón de Airy para la PSF generada con dichos parámetros. Si se grafica la intensidad a través de una línea que atraviese el disco de Airy por el centro (punto de mayor intensidad), se logra observar que la PSF se puede aproximar por una distribución gaussiana entorno al máximo, como se muestra en la Figura 2.8.

2.4. Base de datos sintética

Debido a la cantidad limitada de imágenes proporcionadas por el referente médico, surge la necesidad de generar más datos para poder usar técnicas de redes neuronales y aprendizaje profundo. De esto surge un nuevo desafío, generar estructuras sintéticas con morfología similar a la de las redes mitocondriales y pasar estas imágenes por un proceso de deterioro similar al que sufren las imágenes adquiridas por microscopía fluorescente confocal.

2.4.1. Estructuras Sintéticas

Dan *et al.* [24] determinaron que las redes mitocondriales pueden estar formadas por cinco estructuras distintas. Como se muestra en la Figura 2.9 dichas

Parámetro	Valor
Índice de refracción	1.46
Longitud de Onda	557 nm
NA	1.4
Tamaño de Pixel XY	30 nm
Paso-Z	503 nm

Tabla 2.4: Parámetros de PSF obtenidos de los metadatos de los stacks de IMAGINA.

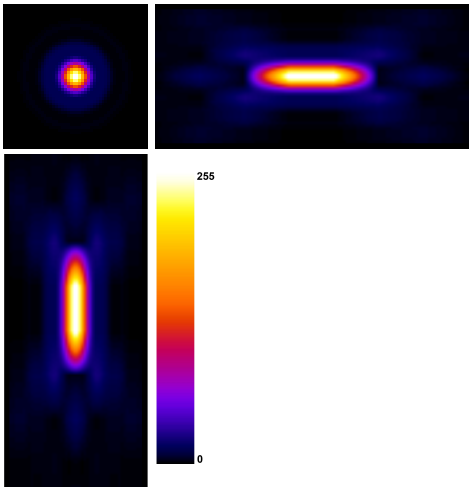


Figura 2.7: Visualización tridimensional del patrón de Airy construido a partir de la metadata de la Tabla 2.2, utilizando el modelo óptico 3D de *Born & Wolf*. En la imagen superior izquierda se presenta el corte xy, en la imagen superior derecha se presenta el corte yz y en la imagen inferior izquierda se presenta el corte xz. Además se presenta el mapa de colores.

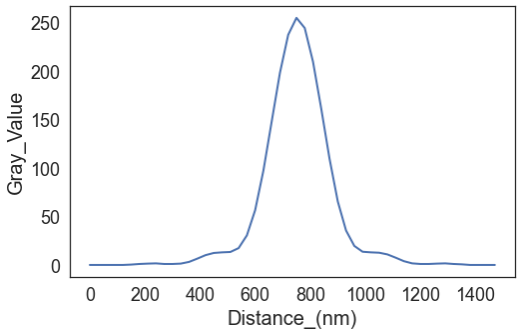


Figura 2.8: Gráfica de la intensidad de la PSF en la dirección axial (corte xy). La PSF fue generada con el PSF Generator con los parámetros de la Tabla 2.4, utilizando el modelo óptico de *Born & Wolf*.

estructuras pueden ser glóbulos, *lumps*, túbulos cortos, túbulos largos o túbulos ramificados. Siguiendo esta línea, se decidió generar estas cinco clases de estructuras.

El generador de volúmenes se desarrolló en Python. La lógica de generación

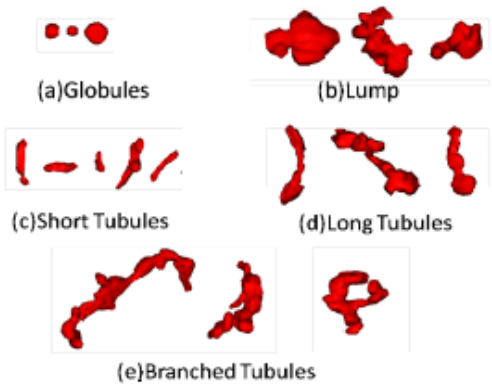


Figura 2.9: Estructuras presentes en las redes mitocondriales. (a) Glóbulos. (b) lumps. (c) Túbulos cortos. (d) Túbulos largos. (e) Túbulos ramificados [24].

Parámetro	Descripción
img_size (N)	tamaño del stack a generar. Los volúmenes serán de tamaño $N \times N \times N$.
min_dist	mínima distancia de separación entre dos estructuras.
n_globules	cantidad de glóbulos.
n_lumps	cantidad de lumps.
n_short	cantidad de túbulos cortos.
n_long	cantidad de túbulos largos.
n_branched	cantidad de túbulos ramificados.
n_ramif	cantidad de ramificaciones por túbulo ramificado.
max_iter	cantidad máxima de iteraciones para intentar agregar una ramificación a un túbulo ramificado.

Tabla 2.5: Parámetros de entrada al script de generación de volúmenes sintéticos.

consiste en cinco funciones, una por cada tipo de estructura. Cada función, al ser llamada, agrega un volumen de su clase siguiendo un mecanismo de control de bordes que evite superposiciones. Estas funciones son llamadas desde un programa principal, en el cual se itera en función de la cantidad de estructuras a agregar. Al fijar el tamaño del volumen a generar (img_size), quedan definidos algunos parámetros internos que definen el tamaño de cada estructura. En la Tabla 2.5 se listan los parámetros de entrada al script principal. Los parámetros internos se detallarán al explicar la generación de cada estructura en particular.

En un trabajo reciente, se generó una base de datos sintética compuesta por núcleos (estructuras esferoidales) [25]. Se utilizó este trabajo como base para generar la función de glóbulos y luego, a partir de esta, se implementaron las restantes. A continuación se detalla cómo fue implementada cada una de las cinco funciones.

En el programa principal se inicializa un volumen S_1 de dimensión $N \times N \times N$ ($N = img\_size$ fijado por el usuario)[3]. Luego se itera con el número de estructuras de cada clase para ir añadiendo de a una al volumen S_1.

Glóbulos

Son pequeñas estructuras nodulares, muy similares a elipsoides. El tamaño de los glóbulos queda definido por sus radios. Una vez fijado el tamaño del volumen por el usuario, quedan determinados los radios mínimo y máximo, r_{min} y r_{max}.

[3]En la sección 4 se explicará por qué fue necesario generar volumenes con esta dimensión.

La función que genera glóbulos recibe como entrada el volumen S_1 y la mínima distancia entre estructuras *min_dist*. Utiliza un volumen auxiliar en el que crea el nuevo glóbulo y finalmente, si no hay problemas de solapamiento, lo añade al volumen S_1. La función devuelve el volumen S_1 actualizado. El procedimiento para la generación se detalla a continuación.

1. Se inicializa un volumen auxiliar S_2 vacío (sin estructuras) de igual dimensión que S_1.
2. Se sortean los radios r_x, r_y, r_z dentro del rango $[r_{min}, r_{max}]$.
3. Se sortean las coordenadas del centro c_x, c_y, c_z para el nuevo glóbulo, asegurando que toda la estructura quepa en S_2.
4. Se genera un elipsoide alrededor del punto (c_x, c_y, c_z) con los radios obtenidos anteriormente.
5. Se rota la estructura según cada eje con ángulos aleatorios.
6. Se realiza un control de solapamiento, utilizando el parámetro *min_dist* fijado por el usuario.
7. Si no hay conflictos en el punto anterior, la estructura se agrega al volumen S_1. Sino, se descarta.

El control de bordes se realiza al seleccionar la ubicación de la estructura a generar. Para evitar que la estructura quede cortada por estar muy cerca de un borde se sortea el centro dentro de un rango "seguro". Las coordenadas (c_x, c_y, c_z) se sortean dentro de un subvolumen de dimensión $(N - 2r_{max}, N - 2r_{max}, N - 2r_{max})$.

Para el control de solapamiento se realiza el mapa de distancias entre el volumen auxiliar S_2 y el volumen de entrada S_1. El mapa de distancias se construye solapando S_2 con S_1 y midiendo las distancias entre cada elemento de S_1 y la estructura originaria de S_2. Esto devuelve un nuevo volumen de la misma dimensión que los volúmenes utilizados, en el cual la intensidad de un punto representa la distancia entre dicho punto y la estructura que se está evaluando agregar. Por lo tanto, se obtiene la distancia mínima del mapa de distancias y si ésta es mayor o igual a la distancia mínima requerida por el usuario (*min_dist*), se permite añadir el nuevo glóbulo en S_1.

Lumps

La lógica es similar a la de los glóbulos, pero en este caso las estructuras tienen una forma más irregular. Se parte nuevamente de un elipsoide pero se irá variando la posición del centro y el tamaño de los radios corte a corte (con cierto criterio para evitar deformaciones exageradas o estructuras que salgan del volumen principal).

La función que genera *lumps* recibe como entrada el volumen S_1 y la mínima distancia entre estructuras *min_dist*. Utiliza un volumen auxiliar en el que crea

el nuevo *lump* y finalmente, si no hay problemas de solapamiento, lo añade al volumen S_1. La función devuelve el volumen S_1 actualizado.

El tamaño de los *lumps*, queda determinado por los valores de $r_{min}, r_{max}, r_{desv}$ y c_{desv}, los cuales quedan definidos al fijar el valor de N. El parámetro r_{desv} determina la máxima variación en el valor del radio al pasar de un corte a otro. De manera similar, c_{desv} es la máxima desviación corte a corte del centro del *lump*. El procedimiento para la generación se detalla a continuación.

1. Se inicializa un volumen auxiliar S_2 vacío (sin estructuras) de igual dimensión que S_1.

2. Se sortean los radios r_x, r_y, r_z dentro del rango $[r_{min}, r_{max}]$

3. Se sortean las coordenadas iniciales del centro c_x, c_y, c_z, asegurando que la estructura entrará en su totalidad en S_2.

4. Se itera en el eje z dentro del rango $[c_z - r_z, c_z + r_z]$, generando una elipse en cada corte siguiendo la siguiente lógica:

   (I) Se genera una elipse con centro en (c_x, c_y) y radios r_x y r_y.

   (II) Se modifican los valores de c_x, c_y, r_x y r_y.

   (III) Se realiza el control de bordes. Si los valores seleccionados en el punto anterior generan conflicto, se vuelve al punto 2. Si no hay conflicto, se pasa al siguiente corte y se vuelve al punto 1.

5. Se rota la estructura según cada eje con ángulos aleatorios

6. Se realiza el control de solapamiento a partir del mapa de distancias entre S_1 y S_2, utilizando el parámetro min_dist fijado por el usuario.

7. Si no hay confictos en el punto anterior, la estructura se agrega al volumen S_1. Sino, se descarta.

En el control de bordes mencionado se verifica que la variación del centro de la estructura no genere que ésta resulte cortada por estar muy cerca de un borde del volumen. En caso de que esto ocurra, se hace una variación del doble de magnitud a la ya realizada pero en la dirección contraria. En cuanto a los radios, se verifica que luego de aplicar cada variación, el valor siga dentro del rango $[r_{min}, r_{max}]$.

Túbulos cortos

Los túbulos son cuerpos con una estructura alargada, asemejándose más a un cilindro que a un elipsoide. Por lo tanto, en lugar de utilizar la ecuación de un elipsoide se utilizó la ecuación de una elipse corte a corte, construyendo un prisma de base elíptica. Al igual que en la generación de *lumps*, se varía la posición del centro y el tamaño de los radios en cada corte para generar deformaciones en el prisma.

La función que genera túbulos cortos recibe como entrada el volumen S_1 y la mínima distancia entre estructuras *min_dist*. Utiliza un volumen auxiliar en el que genera un nuevo túbulo y finalmente, si no hay problemas de solapamiento, lo añade al volumen S_1. La función devuelve el volumen S_1 actualizado.

El tamaño de los túbulos queda determinado por los valores de r_{min}, r_{max}, l_{min}, l_{max}, r_{desv}, c_{desv}, los cuales quedan definidos al fijar el valor de N. Los parámetros l_{min} y l_{max} determinan el rango de longitud válida para los túbulos cortos. El procedimiento para la generación es el mismo que en la función de *lumps*, con la diferencia que en lugar de tener un radio en z se sortea un largo $l_z \in [l_{min}, l_{max}]$ y la iteración en z se realiza en el rango $[c_z - l_z, c_z + l_z]$.

Túbulos largos

Este caso es análogo al de túbulos cortos, con la única diferencia de que tanto las cotas de largo como las de radios tienen un valor mayor.

Túbulos ramificados

Los túbulos ramificados se consideraron como túbulos largos con túbulos cortos superpuestos. Por lo tanto, para la generación de una estructura ramificada se generó un túbulo largo de base y luego varios túbulos cortos, buscando que se superpongan con el largo.

La función que genera túbulos ramificados recibe como entrada el volumen S_1, la mínima distancia entre estructuras *min_dist*, la cantidad de ramificaciones a añadir *n_ramif* y el máximo de iteraciones *max_iter*. Para la construcción de la estructura ramificada utiliza tres volúmenes auxiliares. La función devuelve el volumen S_1 actualizado.

El tamaño de la estructura ramificada queda determinada por las siguientes restricciones a los parámetros:

$$r_{min}^{base} \leq r_{base} \leq r_{max}^{base}$$
$$l_{min}^{base} \leq l_{base} \leq l_{max}^{base}$$
$$r_{min}^{ramif} \leq r_{ramif} \leq r_{max}^{ramif}$$
$$l_{min}^{ramif} \leq l_{ramif} \leq l_{max}^{ramif}$$

A su vez, se define también, r_{desv}^{base}, c_{desv}^{base}, r_{desv}^{ramif}, c_{desv}^{ramif}. Todos estos parámetros quedan definidos al fijar el valor de N. A continuación se detalla el procedimiento para la generación.

1. Se inicializa un volumen auxiliar S_2 vacío, de igual dimensiones que S_1. En este volumen se irá construyendo la estructura ramificada.

2. Se inicializa un segundo volumen auxiliar S_{base} vacío, de igual dimensiones que S_1. Este volumen va a contener únicamente el túbulo base.

3. Se llama a la función que genera túbulos largos, con S_{base} como entrada.

4. Se agrega la estructura base en el volumen S_2.

5. Se itera en el rango $[0, n\_ramif]$ hasta añadir todas las ramificaciones. Para ello se sigue la siguiente lógica:

   (I) Se inicializa el número de intentos en cero.

   (II) Se inicializa un tercer volumen auxiliar S_{ramif} vacío, de igual dimensiones que S_1.

   (III) Se llama a la función que genera túbulos cortos, con S_{ramif} como entrada.

   (IV) Se realiza un control de solapamiento mediante el mapa de distancias entre S_{base} y S_{ramif}.

   Si las estructuras se solapan:

   (v) Se agrega la estructura en S_2.

   (vi) Se incrementa la iteración del número de ramificaciones y se vuelve al punto 1.

   Si las estructuras no se solapan:

   (v) Se incrementa la iteración del número de intentos.
   Si se alcanza *max_iter*:

   (vi) Se vuelve al punto 2.

   Si aún no se alcanza *max_iter*:

   (vi) Se incrementa el contador del número de ramificaciones y se vuelve al punto 1, descartando la posibilidad de agregar esta ramificación.

6. Se rota la estructura obtenida en S_2 según cada eje con ángulos aleatorios.

7. Se realiza el control de solapamiento a partir del mapa de distancias entre S_1 y S_2, utilizando el parámetro *min_dist* fijado por el usuario.

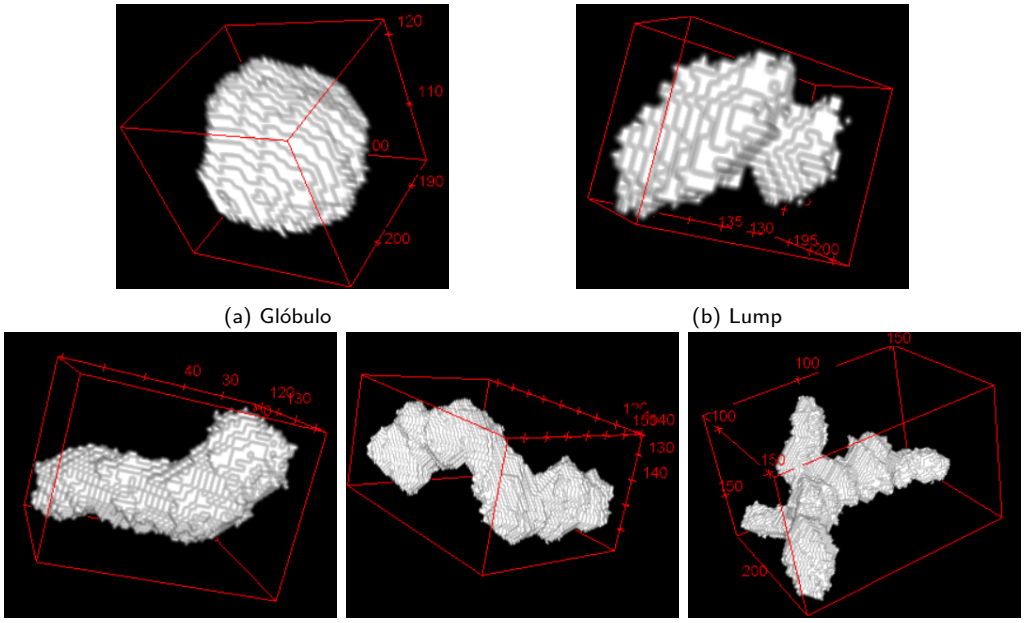
8. Si no hay conflictos en el punto anterior, la estructura se agrega al volumen S_1. Sino, se descarta.

2.4.2. Ejemplos de volúmenes sintéticos

Para la ejecución del script se utilizó el Centro Nacional de Supercomputación, Cluster UY [26]. En la Figura 2.10 se presenta una visualización tridimensional de cada una de las cinco estructuras que pueden obtenerse mediante el generador sintético. Este último fue generado con los parámetros de la Tabla 2.6.

A lo largo del pipeline se trabajará con los volúmenes submuestreados con el fin de representar la apreciación de la confocalidad. Se utilizará el término *stack* para referirse a los volúmenes submuestreados.

Es importante destacar que los volúmenes generados no intentan emular las redes mitocondriales de las imágenes reales. Si bien se generan estructuras detectadas en las redes, no se buscó obtener imágenes con las mismas características.

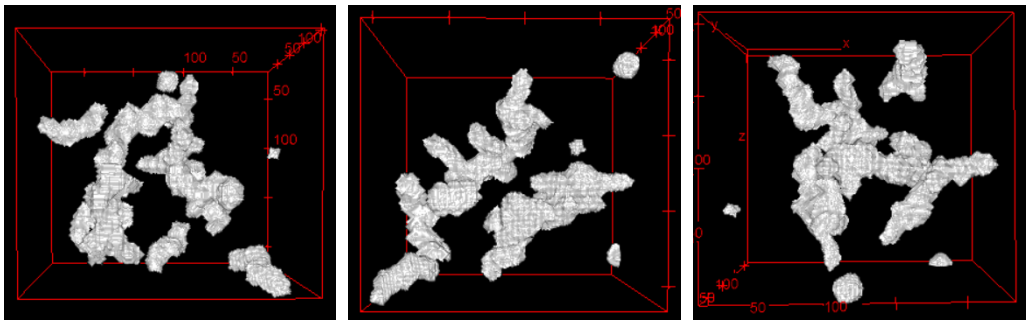


(a) Glóbulo (b) Lump

(c) Túbulo corto (d) Túbulo largo (e) Túbulo ramificado

Figura 2.10: Stacks sintéticos de ejemplo, generados mediante el algoritmo implementado.

La utilización de imágenes reales y sintéticas implica que se debe trabajar en simultáneo con dos dominios, esto tiene sus ventajas y desventajas. Como ventajas principales se tiene capacidad de aumentar el conjunto de stacks y a la vez, se cuenta con un *ground-truth*, mientras que como desventaja se tiene que tomar en cuenta el "cambio de dominio". El cambio de dominio se da cuando se trabajan con dos o más conjuntos de datos, dificultando el desempeño de los algoritmos dado que se intenta obtener buenos resultados para varios conjuntos. En este trabajo se analizará el cambio de dominio y será evaluado qué tanto influye en los resultados.



(a) Corte xy (b) Corte xz (c) Corte yz

Figura 2.11: Stack sintético de ejemplo, generado mediante el algoritmo implementado. Contiene estructuras de los cinco tipos posibles.

Parámetro	Valor
img_size (N)	256
min_dist	5
n_globules	3
n_lumps	4
n_short	3
n_long	3
n_branched	4
n_ramif	5
max_iter	15

Tabla 2.6: Parámetros de entrada al script de generación de volúmenes sintéticos utilizados para la generación del volumen de la Figura. 2.11

2.4.3. Fluorescencia en stacks sintéticos

Al haber generado estructuras sintéticas, se cuenta con un *ground-truth*, por lo que se podrá entrenar y evaluar modelos de segmentación con fidelidad del objetivo al que se pretende llegar.

Para generar el efecto de fluorescencia y el ruido de adquisición se probaron Redes Adversarias Generativas (GAN) [27], en particular dos GAN condicionadas denominadas SpCycleGan [28] y Pix2pix [29]. Se muestra un ejemplo de los resultados que se pueden obtener con Pix2pix en la Figura 2.12. Las GAN son modelos de aprendizaje automático no supervisado que buscan descubrir y aprender los patrones y la estadística de cierta entrada para generar salidas que tengan una distribución estadística similar. Se implementa con dos redes neuronales, una generadora (G) y otra discriminante (D). La red generativa asume el rol de aprender

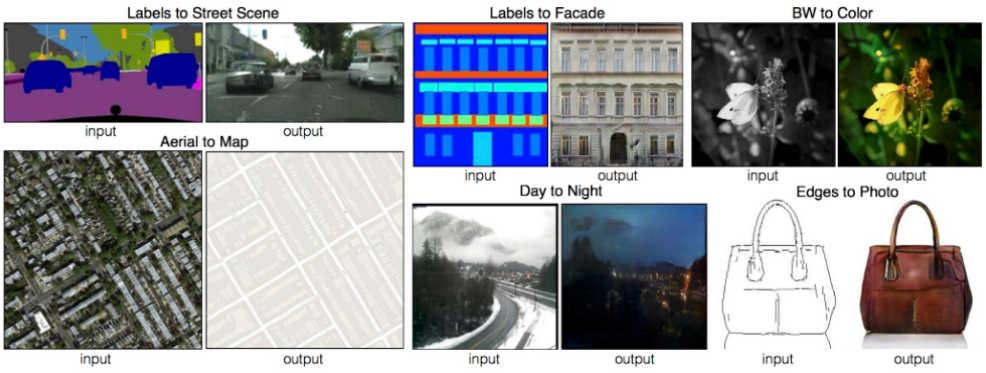


Figura 2.12: Ejemplos obtenidos de la web del proyecto Pix2Pix [30]. Se utiliza la misma arquitectura y el mismo objetivo, solo cambia el conjunto de datos de entrenamiento.

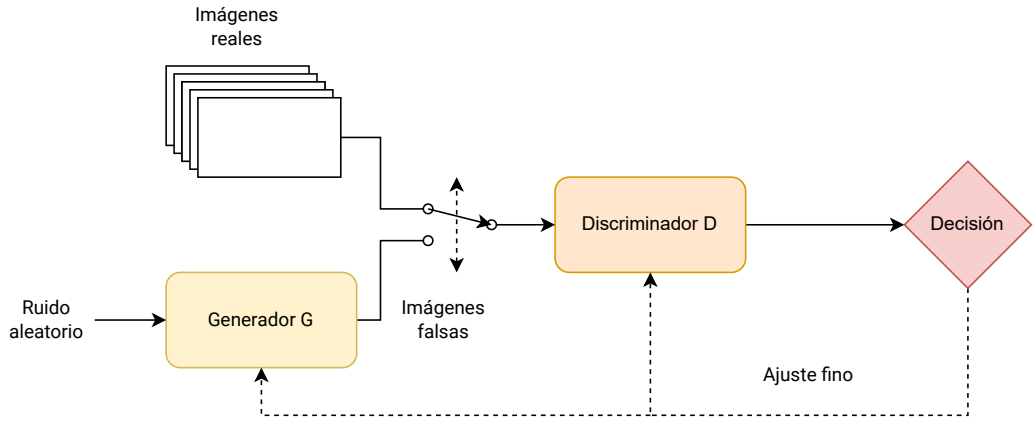


Figura 2.13: Diagrama de funcionamiento de una GAN. Se selecciona aleatoriamente la entrada al discriminador D entre las imágenes reales y la salida del generador G, luego el discriminador define si esa entrada es verdadera o falsa, tomando una decisión. En base a esa decisión se ajustan los pesos y se vuelve a iniciar el proceso. Vale mencionar que a G solo ingresa ruido y en base a los pesos aprendidos se busca generar una imagen con estadística similar a las reales.

la distribución de probabilidad que permite crear imágenes cada vez más similares a la entrada, mientras tanto, la parte discriminante debe recibir una imagen real (del conjunto de entrenamiento) o falsa (del generador) y discernir si es real o falsa. A pesar de ser un modelo no supervisado, se podría considerar como supervisado ya que al incorporar imágenes falsas, se debe asociar una etiqueta a cada tipo de imagen para orientar al discriminador sobre si fue correcta la decisión. El entrenamiento de la red se divide en dos etapas, la primera donde al discriminador se le ingresan datos reales y falsos en la misma medida con su correspondiente etiqueta y la segunda etapa donde el generador genera imágenes falsas con etiqueta real para intentar engañar al discriminador. Los pesos de cada red se ajustan solo en el primer paso para el discrimnante y en el segundo paso para el generador.

En la Figura 2.13 se ve un diagrama del funcionamiento de la GAN, es interesante notar que el generador jamás recibe las imágenes reales sino que aprende en base a la decisión que va tomando el discriminante.

Un trabajo reciente [13,31] encontró que entrenar una GAN condicionada podía generar un conjunto de datos sintéticos para segmentar imágenes de microscopía por fluorescencia. El caso de estudio es muy similar al presentado en este proyecto y por esto se decidió incorporar el software DeepSynth como parte fundamental a probar en el proyecto. DeepSynth se divide en dos partes, una GAN para transformar imágenes de un dominio binario (segmentaciones) a un dominio con la estadística de las imágenes obtenidas por microscopía de fluorescencia y por otro lado, una U-Net modificada [32] para segmentar que será probada en la sección correspondiente 4. Con respecto a la GAN, prueban una CycleGAN simple y una CycleGAN restringida espacialmente (SpCycleGan) diseñadas para trasladar imágenes de un domino A a otro B y viceversa sin contar con imágenes que estén emparejadas (del dominio A y su correspondiente en domino B) [28]. Se muestra un ejemplo de conjuntos de datos emparejados y no emparejados en la Figura 2.14

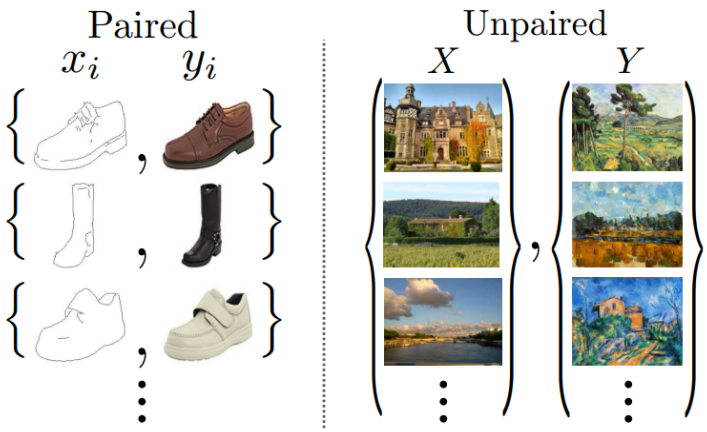


Figura 2.14: A la izquierda imágenes emparejadas. A la derecha imágenes que no se corresponden. Imagen obtenida de [28]

La CycleGAN consiste de dos redes generativas adversarias que asumen roles distintos, una debe aprender a pasar del domino A al dominio B y la otra en sentido inverso, esto podría implicar entrenar por separado dos redes pero los autores incorporaron además un componente a la función de *loss*. Este término denominado *Cycle Consistency Loss* busca relacionar la salida de cada GAN. En la Figura 2.15 se muestra gráficamente el rol de cada GAN y cómo el nuevo término relaciona la imagen original del dominio A, con la transformación de la imagen del dominio B, buscando reducir sus diferencias. Como se mencionó anteriormente, no solo se probará la SpCycleGan sino que también está Pix2pix, que utiliza imágenes emparejadas. Siendo que se cuenta con stacks del grupo IMAGINA y una segmentación avalada por ellos, se considera que los resultados pueden ser mejores que la SpCycleGan.

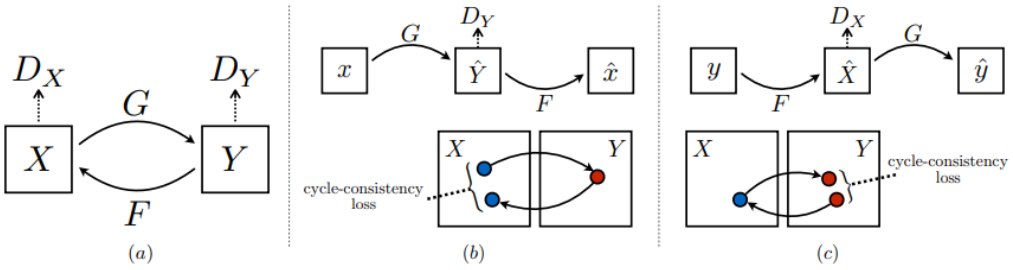


Figura 2.15: (a) Modelo con dos GAN, con funciones que mapean entre dominios $G : X \to Y$ y $F : Y \to X$ y discriminantes D_Y y D_X respectivamente. (b) Se muestra la *loss* propuesta, buscando que la red aprenda que: $x \to G(x) \to F(G(x)) \approx x$ (c) En este caso se satisface que: $y \to F(y) \to G(F(y)) \approx y$. Tomado de [28]

Formulación para la SpCycleGAN

Para comprender el funcionamiento y diferencias de las GAN es crucial entender el objetivo con el cual se entrenan. En la propuesta original de la GAN se utiliza una *loss* denominada Adversarial Loss tal que para una función de mapeo $G : X \rightarrow Y$, con X e Y los dominios entre los que mapear, discriminante D_Y, muestras ${x_i}_{i=1}^{N} \in X$ y ${y_j}_{j=1}^{M} \in Y$ queda:

$$\begin{aligned}\mathcal{L}(G, D_Y, X, Y) = \mathbb{E}_{x \sim p_{data}(x)} \left[log(1 - D_Y(G(x))) \right] \\ + \mathbb{E}_{y \sim p_{data}(y)} \left[log D_Y(y) \right]\end{aligned} \tag{2.2}$$

G busca generar imágenes similares a las del dominio Y, y D_Y es la que busca distinguir entre reales y falsas del dominio Y, esto implica objetivos distintos ya que el entrenamiento de D_Y debe ser para maximizar la probabilidad de que detecte correctamente la etiqueta, mientras que la otra red se entrena para minimizar $log(1 - D_Y(G(x)))$. Se constituye a modo similar otra función de *loss* para la otra GAN: $\mathcal{L}(F, D_X, Y, X)$. En la ecuación 2.3 se detalla la propuesta de *loss* cíclica para condicionar las redes en conjunto.

$$\begin{aligned}\mathcal{L}_{cyc}(G, F) = \mathbb{E}_{y \sim p_{data}(y)} \left[\|G(F(y)) - y\|_1 \right] \\ + \mathbb{E}_{x \sim p_{data}(x)} \left[\|F(G(x)) - x\|_1 \right]\end{aligned} \tag{2.3}$$

La función de *loss* finalmente se compone como:

$$\mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}_{GAN}(G, D_Y, X, Y) + \mathcal{L}_{GAN}(F, D_X, Y, X) + \lambda \mathcal{L}_{cyc}(G, F) \tag{2.4}$$

El objetivo finalmente es minimizar la *loss* presentada en 2.4 donde λ es un factor que mide la importancia relativa de los dos objetivos.

$$G*, F* = arg \min_{G,F} \max_{D_X, D_Y} \mathcal{L}(G, F, D_X, D_Y) \tag{2.5}$$

La CycleGAN no fue la red que dio mejores resultados para el problema concreto imágenes de microscopía, ya que a pesar de aprender una buena traslación entre dominios, se perdía la posición espacial de los núcleos. Por lo tanto se plantearon aumentar la invarianza espacial mediante la incorporación de otro término.

$$\begin{aligned}\mathcal{L}(G, H, F, D_X, D_Y) = \mathcal{L}_{GAN}(G, D_Y, X, Y) + \mathcal{L}_{GAN}(F, D_X, Y, X) \\ + \lambda_1 \mathcal{L}_{cyc}(G, F) + \lambda_2 \mathcal{L}_{spatial}(G, H, X, Y)\end{aligned} \tag{2.6}$$

H es una red con la misma arquitectura que G enfocada en la restricción espacial en la ubicación del núcleo. De esta manera se define $\mathcal{L}_{spatial}$ como:

$$\mathcal{L}_{spatial}(G, H, X, Y) = \mathbb{E} \left[\|H(G(X)) - X\|_2 \right] \tag{2.7}$$

Formulación Pix2Pix

Como fue mostrado previamente, las GAN son modelos generativos que buscan aprender un mapeo desde un vector de ruido z a una imagen salida y, $G : z \rightarrow y$.

En el caso de las redes condicionadas además se utiliza la imagen observada como entrada, $G : \{x, z\} \rightarrow y$. De esta manera la *loss* se puede expresar de la siguiente manera:

$$\begin{aligned}\mathcal{L}(G, D) = \mathbb{E}_{x,z}\left[log(1 - D(x, G(x, z))\right] \\ +\mathbb{E}_{x,y}\left[logD(x, y)\right]\end{aligned} \tag{2.8}$$

Los autores de Pix2pix probaron que si no se considera como entrada z se logran buenos mapeos entre X e Y pero produciendo resultados determinísticos fallando en la tarea de aprender la distribución estadística salvo que sea una delta. Aún así, las redes aprendían a ignorar el ruido y no concluyó como una buena experiencia, esto lo sortearon levemente estableciendo *dropout* en varias capas tanto en el entrenamiento como en la prueba del modelo.

A su vez, para el Pix2pix se propone incorporar una regularización L_1 para forzar la semejanza entre el *ground truth* con la imagen generada, siendo el factor de regularización λ aplicado a $\mathcal{L}_{L1}(G)$

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}\left[\|y - G(x, z)\|_1\right]. \tag{2.9}$$

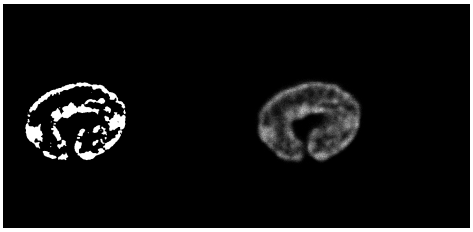
La arquitectura del Pix2Pix consiste en una U-Net para el generador y un discriminante Markoviano (PatchGAN). No se abordará en detalle sobre la PatchGAN, pero el objetivo es analizar la imagen en parches de $N \times N$ y en base al resultado de fake or real de cada parche, se realiza un promedio para asignar una etiqueta.

Entrenamiento y resultados

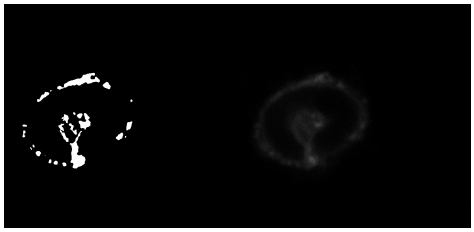
Las dos redes probadas se entrenaron con volúmenes brindados por el grupo IMAGINA, al no disponer de un *ground truth* hecho por ellos, se procedió de modo similar que ellos utilizando Huygens para deconvolucionar los stacks y luego segmentada con una variante de Otsu que tuvo el aval de ellos, esa segmentación fue la asumida como máscara binaria. Se optó por este abordaje por no disponer de una suficiente cantidad de stacks segmentados por parte del equipo. En la Figura 2.16 se muestran cortes de los stacks utilizados de ejemplo. Ambos modelos se entrenan con imágenes en dos dimensiones, no utilizan la información volumétrica por lo que se separó todos los monocitos de cada stack brindados por el grupo e incluso aprovechando los stacks que contenían información del núcleo y no solo de las mitocondrias porque lo que importa es el efecto de fluorescencia.

El conjunto de entrenamiento se constituyó de 4000 imágenes con su correspondiente etiqueta, generados a partir de 477 stacks brindados por el grupo IMAGINA y eliminando las imágenes negras. Para el entrenamiento de ambos modelos se utilizó el Centro Nacional de Supercomputación, Cluster UY [26]. La configuración específica del entrenamiento se hizo de la misma manera en ambos casos con los parámetros propuestos por defecto:

Teniendo en cuenta que el objetivo del proyecto no era la realización de un generador sino que fue una herramienta para la implementación del pipeline, no se hizo una experimentación exhaustiva y los criterios de definición se basaron

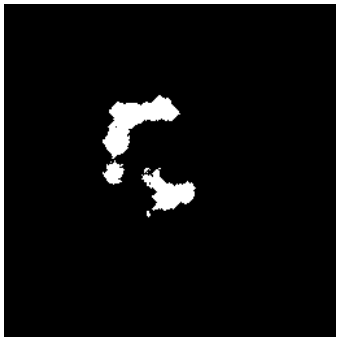


(a) Pareja de imágenes del núcleo de un monocito, a la izquierda segmentación y a la derecha la original.

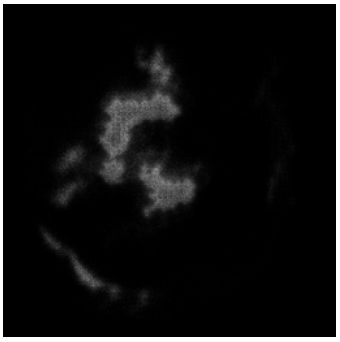


(b) Pareja de imágenes de la mitocondria de un monocito, a la izquierda segmentación y a la derecha la original.

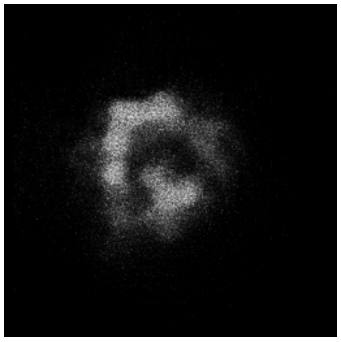
Figura 2.16: Ejemplos de pares de imágenes de IMAGINA utilizados para entrenar el Pix2pix.



(a) Máscara binaria



(b) Inferencia con la SpCycleGan



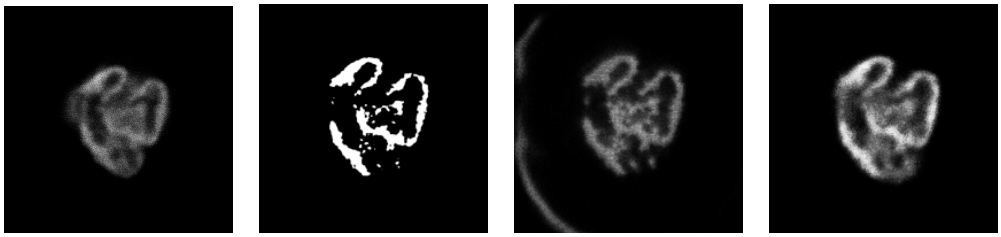
(c) Inferencia con Pix2pix

Figura 2.17: Inferencia a un corte de volumen sintético generado por el Generador presentado en secciones previas.

principalmente en la comparación visual con las imágenes brindads por IMAGINA. En las Figuras 2.17 y 2.18 se pueden ver los resultados de aplicar a cortes seleccionados aleatoriamente los distintos modelos. Del análisis de varias imágenes de stacks diferentes, se consideró mejor utilizar el modelo de Pix2pix, ya que el de SpCycleGan crea estructuras que no deberían estar y en segmentación van a ser muy dañinas para el pipeline. La intensidad de la señal parece ser más acorde en

Parámetro	Valor
Optimizador	ADAM
Tasa de aprendizaje	2×10^{-5}
Momento (ADAM)	0.5
Reescalado	286×286
Recorte	256×256
Inicialización de pesos	Distribución normal

Tabla 2.7: Parámetros por defecto para entrenamiento de la GAN. El reescalado es un paso previo al recorte.



(a) *ground truth* (b) Segmentación OTSU luego de ser deconvolucionada con Huygens (c) Inferencia con SpCycleGan (d) Inferencia con Pix2pix

Figura 2.18: Inferencia a un corte de volumen real brindado por el grupo IMAGINA.

la GAN condicionada espacialmente pero aun así se considera mejor el Pix2pix. En ambos modelos se tuvo problemas con las imágenes totalmente negras ya que ambas generan estructuras, en la Figura 2.19 se muestra sobre la esquina izquierda superior la estructura inventada.

2.5. Selección de base de datos

Para probar y entrenar las distintas etapas del pipeline, junto a su correspondiente clasificación se definieron distintos conjuntos de datos para independizar y evitar el *data snooping*. El *data snooping* sucede cuando se utilizan imágenes del conjunto de entrenamiento para las pruebas o la validación, sesgando los resultados. Esto es adverso al objetivo planteado ya que el modelo en apariencia puede tener buenos resultados pero al probarlo con un conjunto de datos que nunca se vio puede tener un desempeño desastroso. En la Figura 2.20 se muestra gráficamente para qué se utilizó cada conjunto de datos. El manejo de datos en este proyecto fue una cuestión importante y transversal a cada etapa, era necesario una planificación que aprovechara los recursos que se disponían del grupo IMAGINA. La

(a) Imagen binarizada totalmente negra.

(b) Salida de la GAN, genera un artefacto en la esiquina superior izquierda de gran intensidad.

Figura 2.19: Ejemplo de estructura que se genera en la salida del Pix2pix cuando se utiliza como entrada una imagen completamente negra.

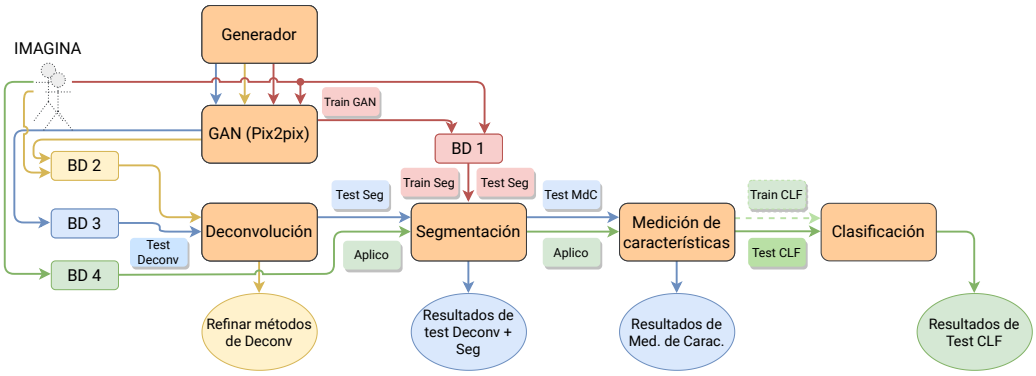


Figura 2.20: Diagrama de flujo y aplicación de los distintos conjuntos de datos que se utilizaron.

importancia de la generación de datos radica no solo en el contar con un *ground truth* que permita realizar pruebas a los algoritmos sino además, nutrir el conjunto de entrenamiento de una CNN como la U-Net.

BD-1

Es el conjunto de datos que se utilizó para entrenamiento y test de la U-Net. En la tabla 2.8 se muestra la cantidad de cada tipo de dato. Los stacks sintéticos ruidosos tienen dimensiones de $128 \times 128 \times 64$, mientras que los stacks reales tienen profundidad de entre 5 y 20 cortes por stack, esto fue una dificultad a sortear para el entrenamiento de la U-Net porque se debe contar con stacks del mismo tamaño y cúbicos. Se optó por dos caminos, apilar stacks reales hasta construir bloques de $128 \times 128 \times 128$ o *paddear* con cortes ruidosos hasta completar el bloque y para los sintéticos apilar dos stacks sintéticos para llegar al mismo tamaño. Hecho esto, el conjunto de datos se compone de: 37 stacks reales en el apilado y 170 sintéticos o 472 stacks reales *paddeados* y 200 stacks sintéticos, de los cuales 30 se utilizan

BD	Reales			Sintéticas		
	Stacks originales	*Ground truth*	Etiqueta	Stacks ruidosos	*Ground truth*	Etiqueta
1	472	SI	NO	400	SI	NO
2	6	SI	NO	5	SI	NO
3	NO	NO	NO	28	SI	NO
4	239	NO	SI	NO	NO	NO

Tabla 2.8: En la tabla se muestra la composición de imágenes de cada conjunto de datos. BD-1 cuenta con 472 stacks originales a los que se le aplico como segmentación OTSU múltiple umbrales para asumir un *ground truth*. A su vez, se cuenta con 200 stack sintéticos tanto su máscara como luego de aplicarles la GAN. BD1 se divide en dos, un conjunto para entrenamiento y otro para probar la segmentación, se apartan 29 stacks reales para pruebas y ninguno sintético. BD2 cuenta con 6 stacks reales con su correspondiente *ground truth* y 5 stacks sintéticos, tanto máscara binaria como luego del Pix2pix. BD3 no tiene volúmenes reales y se compone de 28 stacks sintéticos con su correspondiente *ground truth*. Finalmente BD4 cuenta con 239 stacks reales de monocitos, no tiene ningún stack ruidoso y además, se cuenta con la etiqueta brindada por IMAGINA de "Paciente" o "Control".

para el posterior testeo. El *ground truth* de las imágenes reales se obtuvo aplicando Otsu Multiumbrales con el aval de los referentes médicos que lo consideraron una buena segmentación, esta decisión se estudiará en la parte correspondiente a la U-Net4.

BD-2

Este conjunto de datos es pequeño porque tiene como objetivo probar exahustivamente varios métodos de deconvolución junto a una gran combinación de parámetros. Esto no se podría ejecutar con muchos stacks porque el tiempo de cómputo sería muy grande entre todas las combinaciones de métodos deconvolución, parámetros de deconvolución, segmentación y parámetros de segmentación. Consta de 6 stacks reales con su correspondiente *ground truth* hallado de la misma manera que en BD-1 y 5 stacks sintéticos.

BD-3

La validación del Pipeline depende de este conjunto de datos, por lo cual la idea es que tenga una cantidad de volúmenes que permita un análisis estadístico y así definir los mejores métodos de deconvolución y segmentación para finalmente probar la medición de características. Cuenta con 28 stacks sintéticos, ruido y *ground truth*.

BD-4

Es una base de datos pensada para la etapa de aplicación, es decir, la clasificación. La idea es aplicarle todo el pipeline y así obtener una cierta medición de características para luego entrenar modelos de aprendizaje automático para clasificación. Se cuenta con 239 stacks y su correspondiente etiqueta entre “Paciente” y “Control”.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 3

Deconvolución

La deconvolución es una operación matemática usada en restauración de señales. Se utiliza para restaurar datos que han sido degradados por un proceso físico, el cual puede modelarse mediante una convolución. En la Figura 3.1 puede visualizarse cómo fue modelada la adquisición de las imágenes y su etapa de deconvolución. Se desarrollará un marco teórico basado en algunos papers sobre el tema [33–35].

3.1. Marco Teórico

La señal adquirida en tiempo continuo se expresa con la ecuación (2.1), donde x representa la señal original, h la PSF del microscopio y n la suma de ambos ruidos. Para mostrar cómo se aborda este problema, se realizará un breve análisis partiendo de un escenario ideal donde no se agrega ruido. Para simplificar la notación se presenta el desarrollo con dos dimensiones pero se extiende de igual manera a tres. La ecuación 2.1 se transforma en la siguiente ecuación:

$$y(u, v) = x(u, v) * h(u, v) \tag{3.1}$$

donde (u, v) representan las coordenadas espaciales de la imagen. Al realizar la transformada de Fourier queda:

$$\mathcal{Y}(m, n) = \mathcal{X}(m, n)\mathcal{H}(m, n) \tag{3.2}$$

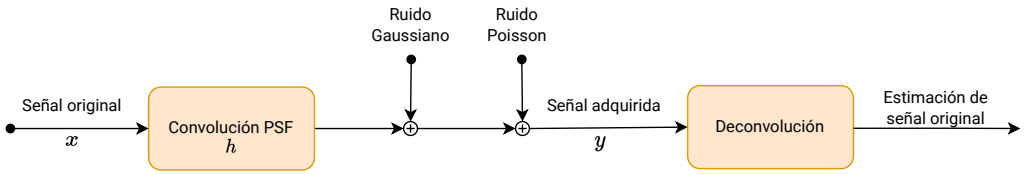


Figura 3.1: Diagrama de bloques del modelado del sistema de adquisición y deconvolución de la señal adquirida.

donde m y n son variables en el dominio de la frecuencia. De esta manera se puede encontrar el filtro $H'(m,n)$ tal que retorne la señal $x(u,v)$ original:

$$x(u,v) = \mathscr{F}^{-1}\left(\frac{\mathcal{Y}(m,n)}{\mathcal{H}(m,n)}\right)(u,v)$$

Si se define el filtro inverso tal que, $\mathcal{H}'(m,n) = 1/\mathcal{H}(m,n)$ se obtiene una expresión espacial para la imagen objetivo:

$$x(u,v) = y(u,v) * h'(u,v)$$

siendo $h'(u,v)$ la transformada de Fourier de $\mathcal{H}'(m,n)$. $\mathcal{H}'(m,n)$ definida como fue presentado tiene el inconveniente de que puede ser cero en el denominador, por lo que se incorpora un parámetro ϵ pequeño para evitar que el filtro inverso amplifique demasiado el ruido y se pierda la señal. En el caso real, este modelo incorpora además el ruido, esto es

$$y(u,v) = x'(u,v) * h(u,v) + n(u,v).$$

La transformada de Fourier en este caso quedaría como sigue:

$$\mathcal{Y}(m,n) = \mathcal{X}'(m,n)\mathcal{H}(m,n) + \mathcal{N}(m,n)$$

Si se procede de forma similar para hallar el filtro inverso se obtendría la siguiente expresión:

$$\begin{aligned}\mathcal{X}'(m,n) &= \frac{\mathcal{Y}(m,n) - \mathcal{N}(m,n)}{H(m,n)} \\ &= \mathcal{X}(m,n) - \frac{\mathcal{N}(m,n)}{\mathcal{H}(m,n)} \\ &= \mathcal{X}(m,n) - \frac{\mathcal{N}(m,n)|\mathcal{H}'(m,n)|^2}{\overline{\mathcal{H}'}(m,n)}\end{aligned}$$

donde $\overline{\mathcal{H}'}(m,n)$ es el conjugado de $\mathcal{H}'(m,n)$ y se utilizó la siguiente propiedad de los números complejos,

$$\mathcal{H}(m,n)\overline{\mathcal{H}}(m,n) = |\mathcal{H}'(m,n)|^2.$$

Se tiene un componente que amplifica el ruido en frecuencia, y siendo que la PSF es un filtro pasabajos, su filtro inverso atenúa las frecuencias bajas y amplifica altas frecuencias, lo que es problemático dado que es donde en la relación señal a ruido suele dominar el ruido. Esto se ejemplifica en la Figura 3.2.

Existen variantes al filtro inverso que sortean las problemáticas aquí planteadas, como por ejemplo el Filtro de Wiener [36] que incorpora un término tal que las zonas con mayor SNR, se utiliza el filtro inverso pero las zonas con menor SNR son atenuadas. Se define el filtro de Wiener según la ecuación (3.3):

$$\mathcal{H}'(m,n) = \frac{1}{\mathcal{H}(m,n) + \frac{1}{SNR}} \tag{3.3}$$

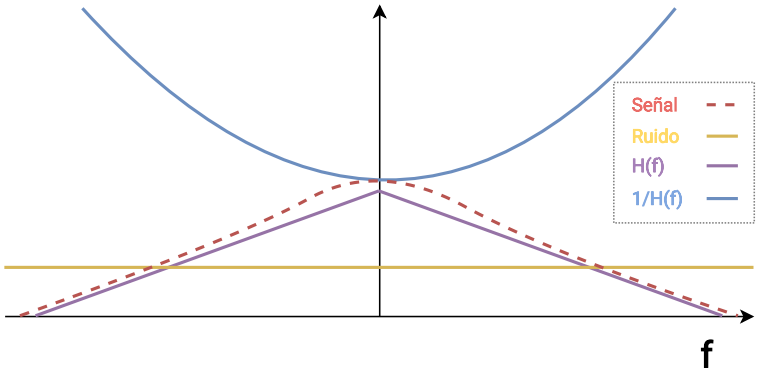


Figura 3.2: Representación gráfica para señales unidimensionales y los efectos del filtro inverso cuando hay ruido presente. Se aprecia que en la zona donde el ruido es mayor a la señal va a ser la zona con mayor amplificación, destruyendo la señal en general. Se asume ruido aditivo blanco por simplicidad.

El abordaje propuesto es a través de métodos computacionales que buscan minimizar una función de costo que relaciona la salida con la entrada, algunos basados en la idea del filtro inverso y otros iterativos. Todos los métodos a probar son métodos probabilísticos ya que incorporan información del ruido. Se realizará un breve esbozo de cómo funcionan los métodos estadísticos. La idea clave surge de estimar una señal original (x), conociendo una señal ruidosa (y), para lo cual se acude al Teorema de Bayes:

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)} \tag{3.4}$$

siendo $P(x|y)$ la probabilidad condicional de obtener x dado que se tiene y. La idea de los métodos consiste en maximizar esta función de probabilidad. Siendo que $P(y)$ es independiente de x, minimizar la ecuación (3.4) implica minimizar solo el numerador. Para abordar esto último se minimizará la función de costo auxiliar $\mathscr{C}(x)$ definida como:

$$\mathscr{C}(x) = \underbrace{-lnP(y|x)}_{L(y,x)} \underbrace{-\gamma lnP(x)}_{\gamma\Omega(x)} \tag{3.5}$$

siendo $L(y, x)$ la función de verosimilitud, $\Omega(x)$ una función de suavizado y γ un parámetro de regularización. La función de verosimilitud depende del tipo de ruido que se desea modelar, por ejemplo: Poisson o Gaussiano. Esta función mide qué tan bien el estimador se ajusta a los datos. Cuando $\gamma = 0$ se tiene el Estimador de Máxima Verosimilitud, si es distinto se está incorporando información a priori y se está hallando el Máximo a Posteriori (MAP) [37].

El anterior análisis fue realizado para tiempo continuo, por lo que se deben realizar algunos comentarios relativos a cómo se trabaja en el dominio discreto. Para trabajar con señales discretas se toman $x \in \mathbb{R}^K$, $n \in \mathbb{R}^K$ e $y \in \mathbb{R}^N$ como vectores con muestras de la señal original, ruido y la señal adquirida respectivamente. K es el producto de las dimensiones de x y N es el producto de las dimensiones

de y. También se trabaja con $\mathbb{H} \in \mathbb{R}^{N \times K}$, definida tal que la discretización de la convolución de la ecuación (3.1) se escriba como la multiplicación de matrices $\mathbb{H}x$. Esto es:

$$S(x * h) = \mathbb{H}x \tag{3.6}$$

donde S es la discretización de la señal.

Para el caso en que $\mathbb{H}$ es *shift invariant* [38], circulante [39] y $K = N$, hipótesis que se cumple, la multiplicación $\mathbb{H}x$ se convierte en una multiplicación elemento a elemento en el dominio de Fourier. Esto permite eficiencia computacional tanto en velocidad como en memoria al trabajar con la transformada rápida de Fourier (FFT por sus siglas en inglés). Por lo tanto, la señal adquirida para tiempo discreto se puede expresar como sigue:

$$y = \mathbb{H}x + n. \tag{3.7}$$

A continuación se presentará una breve descripción de los métodos de deconvolución explorados. Los conceptos presentados aquí se basan fuertemente en el trabajo realizado por Sage *et al.* [12].

3.2. Métodos estudiados

En cada uno de los estimadores utilizados se busca lo mismo, minimizar una función de costo $\mathscr{C}(x)$. Esto es

$$\hat{x} = \arg\min_{x} \left(\mathscr{C}(x)\right) \tag{3.8}$$

donde $\hat{x}$ es la imagen deconvolucionada. Cada método variará la forma que enfrentan al ruido, haciendo distintas hipótesis y tomando distintos enfoques para obtener la mejor solución.

3.2.1. Naive Inverse Filtering (NIF)

El enfoque más simple para la deconvolución consiste en minimizar el módulo de la diferencia entre la señal adquirida y la señal deconvolucionada. En este caso:

$$\mathscr{C}(x) = ||y - \mathbb{H}x||^2. \tag{3.9}$$

Esta diferencia se realiza en el dominio de la frecuencia, donde $\mathcal{X}$, $\mathcal{Y}$ y $\mathcal{H}$ son las FFT de la señal original, la señal adquirida y la PSF, respectivamente.

La solución en frecuencia es:

$$\mathcal{X} = \frac{\mathcal{Y}}{\text{máx}(\mathcal{H}, \epsilon)} \tag{3.10}$$

donde máx denota el máximo elemento a elemento y ϵ es una constante pequeña para evitar divisiones por cero.

Este enfoque se conoce como filtrado inverso "ingenuo" (NIF por sus siglas en ingles), un método cuyo único parámetro es ϵ (en este caso fijado por el *DeconvolutionLab2*, por lo que se lo trata como un método libre de parámetros en la práctica), el cual tiene un costo computacional bajo al trabajar en el dominio de Fourier. El problema al utilizar el NIF es que $\mathcal{H}$ no tiene coeficientes de alta frecuencia, mientras que $\mathcal{Y}$ sí los tiene a causa del ruido. Esto genera amplificación de ruido en altas frecuencias, haciendo al método ineficaz en la mayoría de los casos.

3.2.2. Regularización de Tihkonov

Una solución al problema de amplificación de ruido del NIF es agregar un término de regularización a la función de costo (3.9) que penalice los valores altos de x. Esto es

$$\mathscr{C}(x) = ||y - \mathbb{H}x||^2 + \lambda||x||_2^2, \tag{3.11}$$

siendo λ el parámetro que define el peso del término de regularización. El mínimo de esta expresión se puede hallar explícitamente como sigue:

$$x = \left(\mathbb{H}^T\mathbb{H} + \lambda I\right)^{-1}\mathbb{H}^T y. \tag{3.12}$$

Este método se conoce como el filtro inverso con regularización de Tihkonov (TRIF por sus siglas en ingles). Vale la pena mencionar que este método puede interpretarse como un estimador MAP, en el cual el término de regularización brinda información a priori de x.

3.2.3. Regularized Inverse Filter (RIF)

El RIF mantiene el enfoque del TRIF, pero en este caso el término de regularización penaliza la energía de la derivada de la siguiente forma:

$$\mathscr{C}(x) = ||y - \mathbb{H}x||^2 + \lambda||Lx||_2^2 \tag{3.13}$$

donde L es el operador diferencial Laplaciano. El mínimo de esta expresión es

$$\hat{x} = (\mathbb{H}^T\mathbb{H} + \lambda L^T L)^{-1}\mathbb{H}^T y. \tag{3.14}$$

Nótese que al penalizar valores altos de la derivada se está imponiendo suavidad en la solución.

3.2.4. Algoritmo de Landweber

El algoritmo de Landweber (LW) también minimiza la función de costo (3.9), pero en lugar de realizar la inversa de forma directa, usa descenso por gradiente para alcanzar la solución.

$$x_{k+1} = \mathscr{P}_{(\mathbb{R}^+)^K}\left(x_k + \gamma H^T(y - \mathbb{H}x_k)\right). \tag{3.15}$$

Aquí γ es el paso, $\mathscr{P}_{(\mathbb{R}^+)^K}(x)$ es máx$(x, 0)$ y su función es imponer una restricción de no negatividad en cada iteración, y k representa la iteración. Al ser un algoritmo iterativo, otro parámetro fundamental es el máximo de iteraciones N. Depende directamente de γ y N que el algoritmo converja a una solución.

Dado que este algoritmo tiene la misma función de costo que el NIF, tiene el mismo problema de amplificación de ruido al no contar con un término de regularización. La diferencia es que al ser iterativo se puede compensar entre la convergencia y la amplificación de ruido, de tal manera que N funcione como un pseudo-regularizador.

3.2.5. Algoritmo de Tikhonov-Miller

Es un método iterativo de descenso por gradiente que busca hallar una solución para la función de costo (3.13).

$$x^{k+1} = \mathscr{P}_{(\mathbb{R}^+)^K}\{x^k + \gamma(\mathbb{H}^T y - (\mathbb{H}^T\mathbb{H} + \lambda L^T L)x^k)\}. \tag{3.16}$$

Cabe destacar que hasta el momento este es el único algoritmo iterativo que posee un término de regularización, siendo el método que tiene más parámetros para ajustar.

3.2.6. Algoritmo de Richardson-Lucy

Probablemente el más popular de los métodos presentados. Richardson-Lucy (RL) es un estimador de máxima verosimilitud, pero en este caso se modela el ruido aditivo asociado a la adquisición con una distribución de Poisson. Esta suposición lleva a la siguiente función de costo

$$\mathscr{C}(x) = \mathbf{1}^T\mathbb{H}x - y^T log(\mathbb{H}x) \tag{3.17}$$

donde el logaritmo se aplica componente a componente y $\mathbf{1}$ es un vector unitario de tamaño N. El cálculo iterativo de la estimación esta dado por

$$x_{k+1} = x_k \otimes \mathbb{H}^T\left(\frac{y}{\mathbb{H}x_k}\right). \tag{3.18}$$

siendo $\otimes$ el producto término a término.

3.2.7. Algoritmo de Richardson-Lucy con regularización de variación total

En este algoritmo se agrega un término de regularización que penaliza la norma $\mathscr{L}_1$[1] del gradiente de la señal

$$\mathscr{C}(x) = \mathbf{1}^T\mathbb{H}x - y^T log(\mathbb{H}x) + \lambda||Dx|| \tag{3.19}$$

[1]La norma $\mathscr{L}_1$ también es conocida como la norma de regularización dispersa o regularización de Lasso.

	Parámetros	$\mathscr{C}(x)$	Regularizado	Iterativo
NIF	-	$\|y - \mathbb{H}x\|^2$	✗	✗
TRIF	λ	$\|y - \mathbb{H}x\|^2 + \lambda\|x\|^2$	✓	✗
RIF	λ	$\|y - \mathbb{H}x\|^2 + \lambda\|Lx\|^2$	✓	✗
LW	M_{iter}, γ	$\|y - \mathbb{H}x\|^2$	✗	✓
TM	$\lambda, M_{iter}, \gamma$	$\|y - \mathbb{H}x\|^2 + \lambda\|Lx\|^2$	✓	✓
RL	M_{iter}	$\mathbf{1}^T\mathbb{H}x - y^T log(\mathbb{H}x)$	✗	✓
RLTV	λ, M_{iter}	$\mathbf{1}^T\mathbb{H}x - y^T log(\mathbb{H}x) + \lambda\|Dx\|_1$	✓	✓

Tabla 3.1: Resumen de las características principales de cada estimador.

siendo D la matriz de diferencias para derivadas de primer orden. La solución queda de la forma

$$x_{k+1} = x_k \otimes H^T \left(\frac{y}{Hx_k} \right) \otimes \frac{1}{1 + \lambda g_k}. \tag{3.20}$$

donde g_k es la derivada de la versión regularizada de la norma $\mathscr{L}_1$ de Dx_k.

La Tabla 3.1 resume algunas de las características de los estimadores presentados.

3.3. Experimentos Realizados

Para probar los distintos métodos de deconvolución se utilizó BD2, la cual fue definida para seleccionar algunos algoritmos y parámetros en esta etapa. En la Figura 3.3 se puede visualizar el objetivo de los experimentos que serán descritos a continuación, donde usando datos tanto reales como sintéticos se definirá qué estimadores son de utilidad para el pipeline y con qué conjuntos de parámetros.

Los experimentos se ejecutaron de forma automática utilizando la librería "PyImageJ" [40] de Python, la cual brinda la posibilidad de trabajar con funciones de Fiji. Como se mencionó en la Sección 2.3, la PSF del microscopio fue generada con el plugin de Fiji "PSF Generator", utilizando el modelo óptico de *Born & Wolf*.

Para los experimentos se usa como referencia la deconvolución de las imágenes con *Huygens*, el cual ya tiene incorporado el cálculo de la PSF internamente. Se

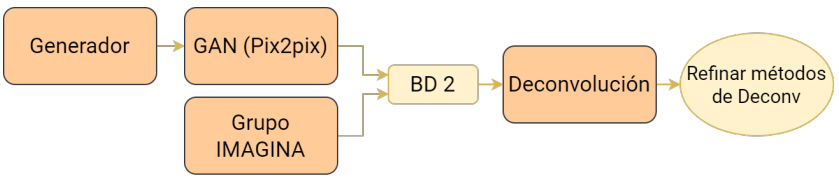


Figura 3.3: Diagrama de flujo en etapa de deconvolución.

Parámetros	Valores
λ	1, 1e-3, 1e-6, 1e-9
M_{iter}	10, 20, 50, 100
γ	1e-1, 5e-1, 1, 2, 10

Tabla 3.2: Parámetros utilizados en todos los métodos para una primera selección

realizaron comparaciones con *Huygens* utilizando su configuración predeterminada o con configuraciones optimizadas dentro de las posibilidades de quienes escriben.

Se ejecutaron todos los algoritmos con los parámetros que se muestran en la Tabla 3.2, probando todas las combinaciones posibles en cada método. Los parámetros fueron refinados empíricamente en función de los resultados observados, teniendo en cuenta el método y el sentido matemático de cada parámetro. Las conclusiones obtenidas en este experimento se explican en detalle en la Sección 3.4.

Para evaluar el desempeño de los distintos métodos, se estudió la SNR y la PSNR (*Peak Signal to Noise Ratio*) de las imágenes, usando como referencia la deconvolución de Huygens. El cálculo de la SNR y la PSNR en tres dimensiones queda determinado por las ecuaciones (3.21) y (3.22) respectivamente.

$$SNR_{dB} = 10\log_{10}\left[\frac{\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}\sum_{k=0}^{O-1}\left[r(i,j,k)\right]^2}{\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}\sum_{k=0}^{O-1}\left[r(i,j,k)-t(i.j,k)\right]^2}\right] \tag{3.21}$$

$$PSNR_{dB} = 10\log_{10}\left[\frac{max\left(r(i,j,k)\right)^2}{\frac{1}{M\times N\times O}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}\sum_{k=0}^{O-1}\left[r(i,j,k)-t(i.j,k)\right]^2}\right] \tag{3.22}$$

La SNR presentada en la ecuación (3.21), puede identificarse como el cociente entre la potencia de la señal y la potencia del ruido:

$$SNR = \frac{P_{señal}}{P_{ruido}}. \tag{3.23}$$

Debido a que las imágenes de microscopía analizadas tienen baja intensidad y ruido considerable, es esperable que la SNR tenga una magnitud pequeña. La diferencia entre la SNR y la PSNR es que en esta última solo se toma el valor máximo de intensidad en el numerador. Al estar observando solo el valor pico de la imagen, el hecho de tener una imagen con poca información ya no es problema.

Teniendo en cuenta que son métricas basadas en la intensidad y se computan elemento a elemento no se utiliza información estadística de las imágenes, por lo que se decidió incorporar dos medidas de similitud que otorguen más información sobre los resultados. Para esto, se exploraron medidas del ambiente de la Teoría de la Información, para agregar análisis y comparaciones estadísticas de los distintos stacks. Se seleccionó la distancia de Kullback-Leibler [41] d_{kl} (entropía relativa) que plantea qué tan distinta es una cierta distribución de probabilidad P de otra

$\overline{SNR}$	$\overline{PSNR}$	$\overline{IM}$	$\overline{d_{kl}}$
-5.088406	35.40091	0.436494	0.73161

Tabla 3.3: Aplicación de las métricas y medidas de similitud entre los stacks reales crudos y la deconvolución de *Huygens*.

Método	**Parámetro**	$\overline{SNR}$	$\overline{PSNR}$	$\overline{IM}$	$\overline{d_{kl}}$
NIF	-	-9.14295	31.34636	0.346758	0.538358
RIF	$\lambda = 0.01$	-5.69057	34.79874	**0.439109**	0.410049
TRIF	$\lambda = 0.01$	-5.74781	34.7415	**0.436895**	0.435519
LW	$M_{iter} = 100$, $\gamma = 1.0$	-5.12532	35.36399	**0.430897**	0.408411
TM	$M_{iter} = 100$, $\gamma = 1.0$, $\lambda =$ 1e-06	-5.12547	35.36384	**0.430897**	0.408448
RL	$M_{iter} = 50$	**-0.80395**	**39.68537**	0.377856	**0.060787**
RLTV	$M_{iter} = 50$, $\lambda = 0.01$	**-0.96148**	**39.52783**	0.37479	**0.061795**

Tabla 3.4: Mejores resultados de deconvolución comparando distintos parámetros de un mismo método. Se presentan el promedio de todos los stacks reales para dos métricas y dos medidas de similitud. En verde se resaltan los resultados que se encuentran por encima del promedio, en el caso de la distancia de Kullback-Leibler significa ser más chico que el promedio. En negrita los mejores 10 resultados. La comparación se realiza entre la combinación de todos los métodos y todos los parámetros.

Q y por otro lado la Información Mutua (MI por sus siglas en inglés) [42], que intuitivamente dice qué tanta información se tiene en una variable aleatoria X si conozco otra variable Y. Se presentan a continuación las expresiones que las definen:

$$d_{kl}(P,Q) = \sum_{i=1}^{M} P(i) log \frac{P(i)}{Q(i)} \tag{3.24}$$

$$\begin{aligned} MI(X,Y) &= H(X) + H(Y) - H(X,Y) \\ &= H(X) - H(X|Y) \\ &= H(Y) - H(Y|X) \end{aligned} \tag{3.25}$$

siendo $H(X) = -\sum_i p(i) ln(p(i))$ la entropía de una variable aleatoria X, P y Q la distribución de probabilidad de X e Y respectivamente e i el nivel de intensidad de gris de la señal. En este caso se tomará P como el histograma de la señal x y Q el histograma de la señal y.

A modo de evaluar si existen mejoras al aplicar los métodos de deconvolución, fue necesario realizar el análisis también sobre los datos crudos. Estos resultados se presentan en la Tabla 3.3. En la Tabla 3.4 se presentan los mejores resultados obtenidos para cada método, definiendo el mejor resultado según su comportamiento con las cuatro métricas. Los resultados completos del experimento se presentan en la Tabla A.1 del Apéndice A.

3.4. Análisis de Resultados

Esta sección tiene como objetivo analizar y discutir sobre los resultados obtenidos en los experimentos realizados. En una primera instancia se discutirán resultados generales de los experimentos realizados. Luego se subdividirán los métodos estudiados en tres agrupaciones: NIF-TRIF-RIF (filtro inverso y sus variantes con distintos términos de regularización), LW-TM (métodos de descenso por gradiente iterativos) y RL-RLTV (Richardson-Lucy con y sin regularización).

3.4.1. NIF, TRIF y RIF

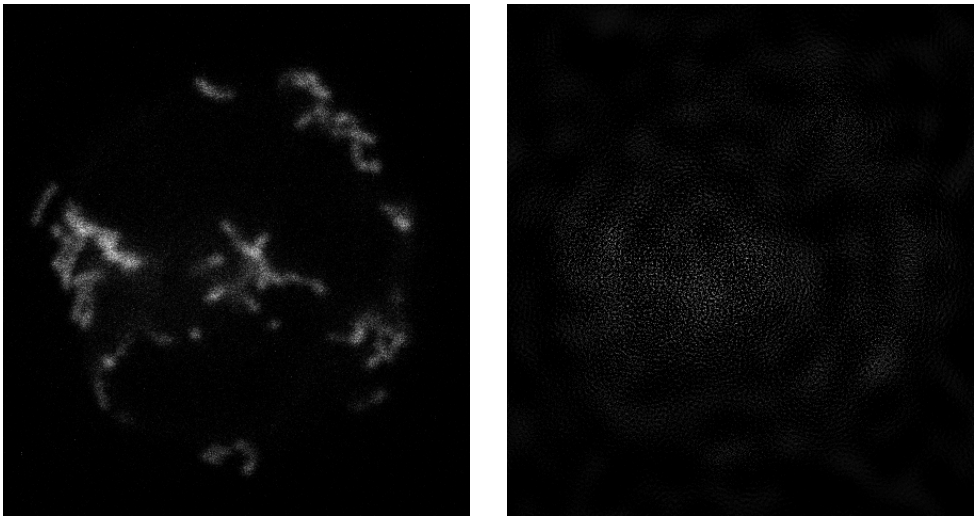
Comenzando con el NIF, de las Tablas 3.3 y 3.4 se desprende que este método es el que tiene peor desempeño. Este resultado se debe a que, como se explicó en la Sección 3.2, el algoritmo amplifica el ruido de la imagen. Esto puede verse claramente en la Figura 3.4, donde es imposible distinguir las mitocondrias. Una forma de evitar esta amplificación de ruido es con un término de regularización.

El TRIF penaliza los resultados con valores de intensidad altos, utilizando $\lambda||x||_2^2$ como término de regularización. Es de suponer que cuando λ tiende a cero la regularización no va a tener el peso suficiente para influir en los resultados. Otra observación es que cuando λ tiende a infinito penaliza demasiado la potencia de la imagen. Esto puede visualizarse mejor en las ecuaciones (3.26) y (3.27).

$$||y - \mathbb{H}x||^2 + \lambda||x||_2^2 \xrightarrow{\lambda \to 0} ||y - \mathbb{H}x||^2 \tag{3.26}$$

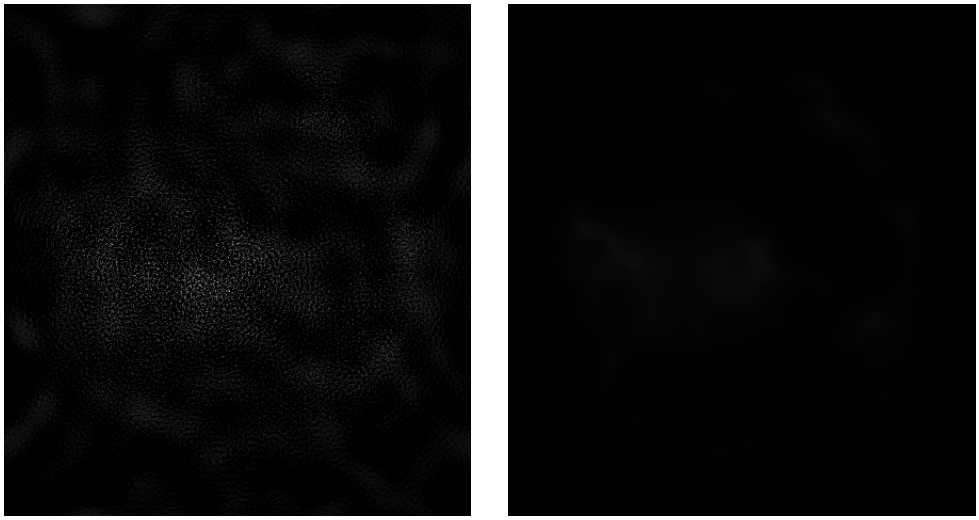
$$||y - \mathbb{H}x||^2 + \lambda||x||_2^2 \xrightarrow{\lambda \to \infty} \lambda||x||_2^2 \tag{3.27}$$

La Figura 3.5 muestra resultados de TRIF con $\lambda = 1 \times 10^{-6}$ y $\lambda = 1000$,



(a) Corte de imagen original (b) Corte de la imagen deconvolucionada

Figura 3.4: Resultados del método NIF en un corte de un stack real perteneciente a BD2.

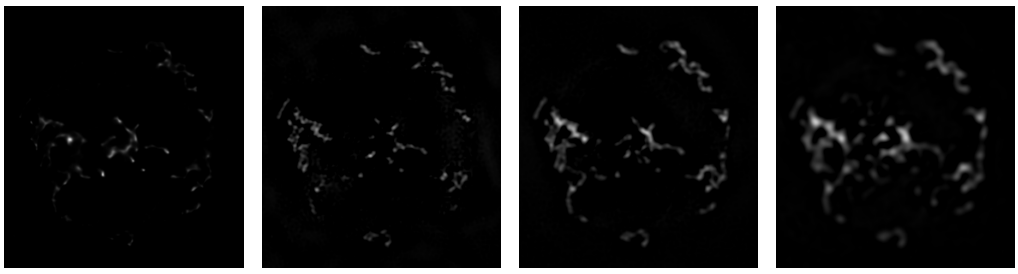


(a) Resultado de TRIF con $\lambda = 1e-6$ (b) Resultado de TRIF con $\lambda = 1000$

Figura 3.5: Resultados del método TRIF en un corte de un stack real perteneciente a BD2.

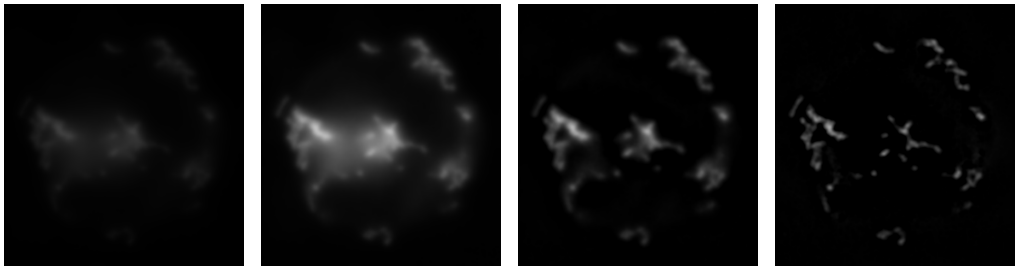
donde se puede ver que 3.5a es prácticamente igual a la imagen obtenida con el NIF, resultado coherente dado que está minimizando aproximadamente la misma función de costo. En la Figura 3.5b se observa que la solución obtenida es casi nula, coincidiendo con minimizar algo muy cercano al módulo de x. Observaciones de este tipo son características de los métodos estudiados. Dado que pueden llegar a ser repetitivas y el objetivo es encontrar los parámetros que mejor funcionen, se desestima el análisis de estos casos.

La regularización genera un compromiso entre qué tan bien se invierte la convolución con la PSF y qué tanto se amplifica el ruido. El parámetro λ se usa para regular sobre este compromiso. La Figura 3.6 refleja justamente esto al variar λ en el RIF. Cuando λ es pequeño (véase Figura 3.6b), los bordes de las mitocondrias están mejor definidos pero la imagen tiene ruido, lo cual es inconveniente para las siguientes etapas del presente trabajo. Un valor de λ alto soluciona el problema del ruido, pero los bordes de las estructuras son suavizados como se puede ver en



(a) Deconvolucion de Huygens (b) Deconvolución de RIF con $\lambda = 1e-6$ (c) Deconvolución de RIF con $\lambda = 0{,}01$ (d) Deconvolución de RIF con $\lambda = 1$

Figura 3.6: Referencia y distintos resultados de RIF al variar λ. Al incrementar el valor de λ la imagen va perdiendo ruido, pero su deconvolución va empeorando.



(a) LW con $M_{iter} = 5$ y $\gamma = 0{,}001$ (b) LW con $M_{iter} = 100$ y $\gamma = 0{,}001$ (c) LW con $M_{iter} = 5$ y $\gamma = 1$ (d) LW con $M_{iter} = 100$ y $\gamma = 1$

Figura 3.7: Resultados de LW variando M_{iter} y γ. Cuando se tienen muy pocas iteraciones o un paso muy pequeño el algoritmo no alcanza la convergencia.

la Figura 3.6d.

Al final del capítulo se mostrarán los que fueron considerados como mejores resultados de TRIF y RIF, junto con otros algoritmos.

3.4.2. LW y TM

Estos estimadores usan descenso por gradiente para aproximarse a la solución, esto implica que los valores de M_{iter} y γ son fundamentales para la convergencia. En la Figura 3.7 se puede ver cómo afecta la elección de parámetros a LW. Debe tenerse en cuenta que LW no tiene un término de regularización, por lo que muchas iteraciones pueden llevarlo a converger a la misma solución del NIF.

Esto refleja nuevamente un compromiso entre qué tan bien se deconvoluciona la imagen y con cuánto ruido se está dispuesto a trabajar, pero en este caso el parámetro con el que se "regula" es M_{iter}. De esta forma, se elige un γ lo suficientemente pequeño para alcanzar una posible convergencia (véase Figura 3.8).

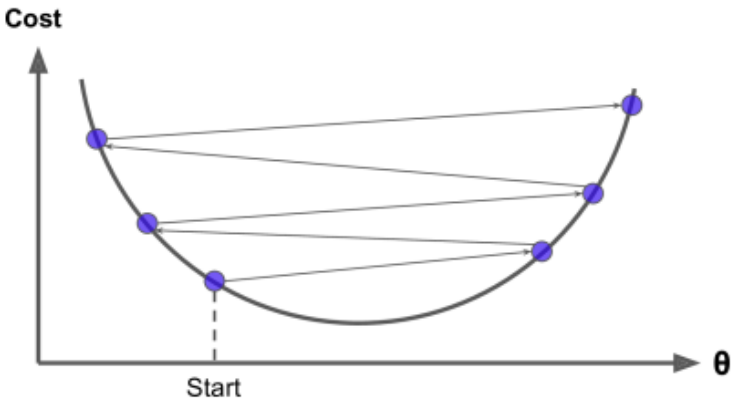


Figura 3.8: Gráfica de función de costo a minimizar donde γ es muy alto, pudiendo llevar al algoritmo a diverger [43]

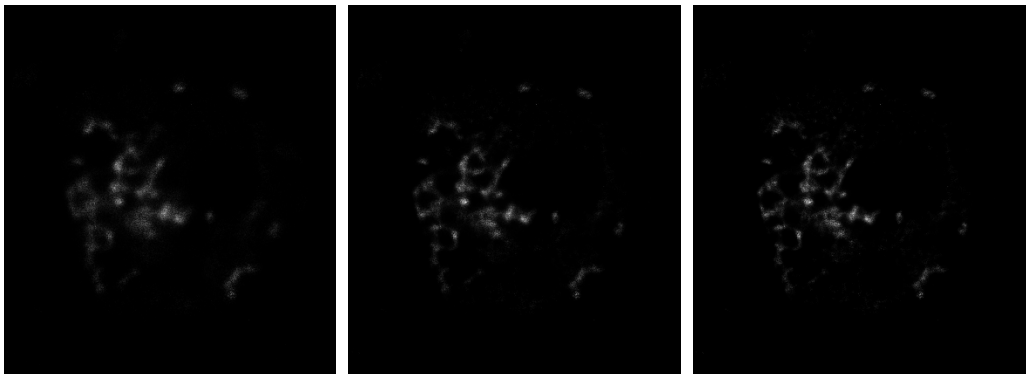
En el caso de TM además de tener M_{iter} y γ se tiene λ nuevamente, dado que busca minimizar la función de costo (3.13). Con el análisis ya realizado de los tres parámetros, se entiende que el comportamiento del método en función de éstos será como sigue:

- Valores muy bajos de M_{iter} y γ hacen que el algoritmo no logre acercarse lo suficiente a la solución.
- Un valor muy bajo de λ genera una solución similar a la de algoritmos no regularizados. En este caso LW y TM generan la misma solución cuando $\lambda \to 0$.
- Existe compromiso entre revertir el efecto de la PSF y manejar el ruido aditivo a la señal, en el caso de TM se puede manejar correctamente tanto con M_{iter} como con λ, siendo este último la mejor opción.

3.4.3. RL y RLTV

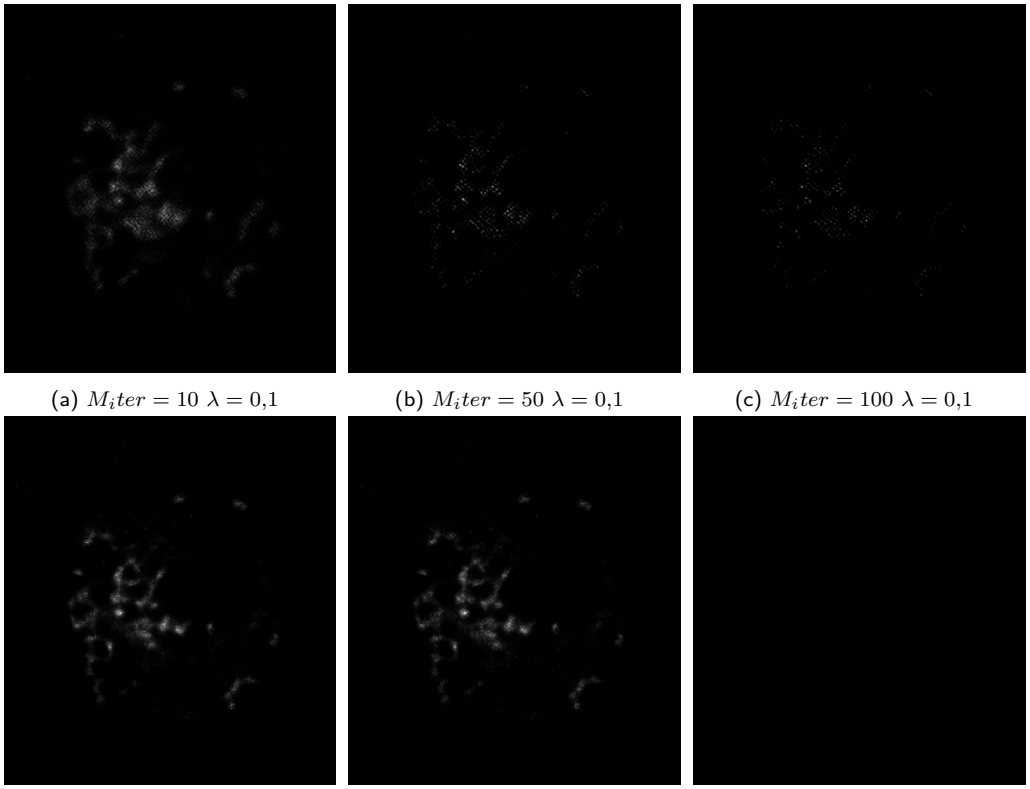
Ambos métodos se analizarán por separado, a pesar de que como ya se ha visto antes, con un parámetro de regularización lo suficientemente chico, RTLV converge a RL. En la Figura 3.9 se aprecian tres imágenes con distinta cantidad de iteraciones. Es claro a partir de ellas que a medida que se aumentan las iteraciones se logra definir aún más la señal, lo cual tiene como contraparte una pérdida de intensidad que puede ser contraproducente en posteriores etapas del pipeline. En los resultados de la tabla A.1 se puede visualizar que todos los métodos de RL, tuvieron un buen desempeño con stacks reales. Esto parece tener sentido con el hecho de que RL es el estimador de máxima verosimilitud [44] (MLE por sus siglas en ingles) para cuando la señal a estimar tiene ruido de Poisson como es este caso.

En el caso del RLTV, se tiene un proceso similar en cuanto a que más iteraciones combate el ruido pero atenúa la señal, esto queda especialmente claro en la



(a) $M_{iter} = 10$ (b) $M_{iter} = 50$ (c) $M_{iter} = 100$

Figura 3.9: Corte de un stack real deconvolucionado con Richardson-Lucy. Se muestran varios casos con distinta cantidad de iteraciones.



(a) $M_iter = 10$ $\lambda = 0{,}1$ (b) $M_iter = 50$ $\lambda = 0{,}1$ (c) $M_iter = 100$ $\lambda = 0{,}1$

(d) $M_iter = 50$ $\lambda = 0{,}001$ (e) $M_iter = 50$ $\lambda = 0{,}01$ (f) $M_iter = 50$ $\lambda = 1$

Figura 3.10: Corte de stack real deconvolucionado con Richardson Lucy y Total Variation. En las imágenes de arriba, de izquierda a derecha se puede apreciar el efecto de aumentar las iteraciones. En la fila de abajo se aumenta la regularización.

Figura 3.10c, donde a pesar de poder apreciar un poco de señal, ya es indistinguible del ruido. En ese caso $SNR_{dB} = 0$, lo que quiere decir que prácticamente no queda señal (el valor del experimento no aparece en la tabla A.1). Mientras tanto el parámetro de regularización se podría considerar como el paso en la convergencia del algoritmo de descenso por gradiente, teniendo los mismos problemas que LW y TM, con un paso suficientemente grande se pierde toda la señal. Y por el otro lado, un λ pequeño genera una solución similar a RL, como puede apreciarse en 3.10d, 3.10e y 3.9b.

3.4.4. Conclusiones

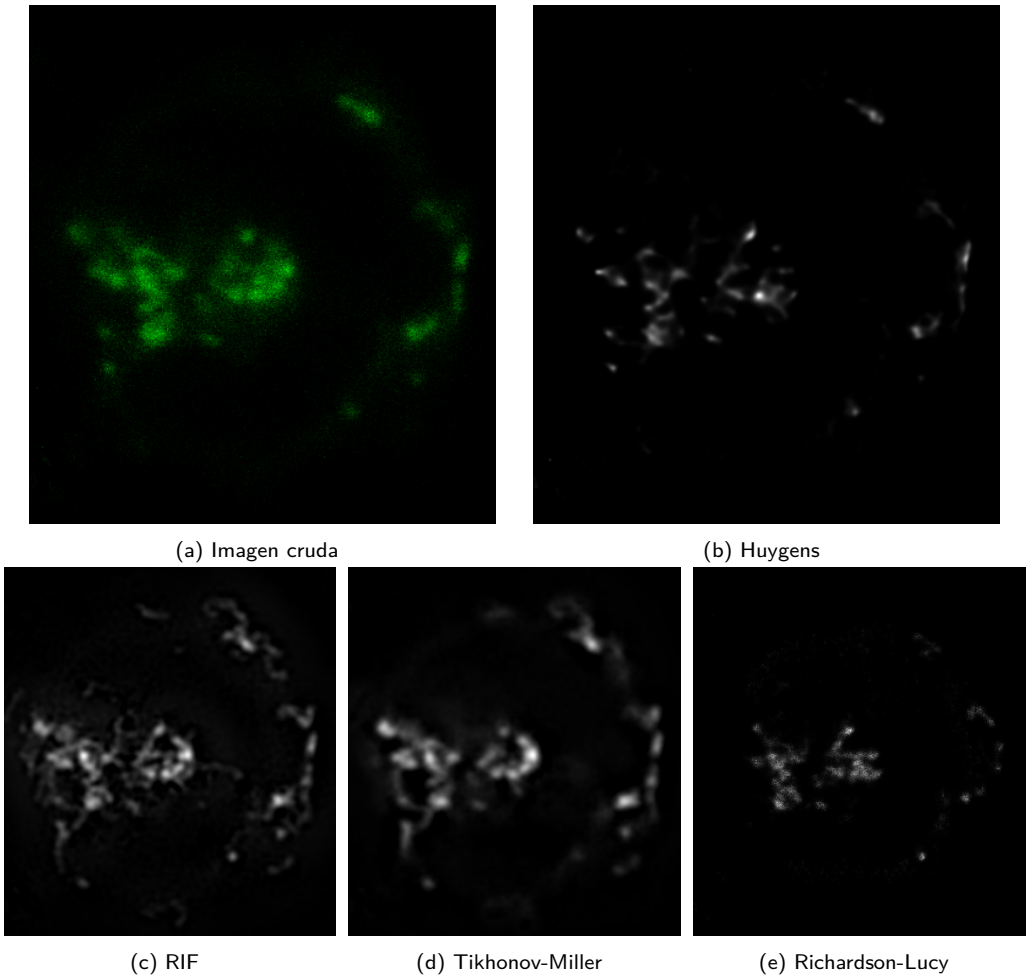
No existe un número mágico o método ideal que otorgue genéricamente los mejores resultados para deconvolucionar PSF, existe un compromiso entre tener una señal más definida y con ruido u otra más degradada pero sin ruido. Se realiza el análisis comparativo de la Tablas 3.3 y 3.4, para así poder definir cuantitativamente el mejor método. Esto se puede visualizar en la Tabla 3.5 en la que se calcula la mejora relativa de cada modelo con respecto al imagen cruda brindada por el grupo IMAGINA.

Método	$\overline{SNR}$	$\overline{PSNR}$	$\overline{IM}$	$\overline{d_{kl}}$
NIF	-80 %	-11 %	-21 %	26 %
RIF	-12 %	-2 %	1 %	44 %
TRIF	-13 %	-2 %	0 %	40 %
LW	-1 %	0 %	-1 %	44 %
TM	-1 %	0 %	-1 %	44 %
RL	**84 %**	**12 %**	**-13 %**	**92 %**
RLTV	81 %	12 %	-14 %	92 %

Tabla 3.5: Mejora relativa de los distintos métodos de deconvolución con respecto al original. La mejora relativa se calcula como: $Mejora_{relativa} = \frac{valor_{dec} - valor_{ref}}{|valor_{ref}|}$. Para el caso de la distancia de Kullback-Leibler se multiplica por -1 ya que cuanto más pequeño el resultado, mejor.

De la tabla 3.5 se concluye que NIF destruye la señal y no es un método que tenga utilidad práctica. Respecto a RIF, TRIF, LW y TM, métodos con la misma función de costo, los mejores resultados redundaron en una mejora relativa de la distancia de Kullback-Leibler, es decir se disminuye la cantidad de información necesaria para obtener el stack objetivo con el stack deconvolucionado. Esta capacidad de aumentar la similitud entre la estadística de las señales, acarrea la perdida de señal en los casos RIF y TRIF, esto permite descartar estos métodos por considerarse sensiblemente peores que a LW y TM. Finalmente los mejores resultados se obtuvieron para Richardson-Lucy sin regularización, vale mencionar que el factor de regularización utilizado es $\lambda = 0{,}01$ que se puede considerar pequeño, por lo que no se está muy lejos del RL sin regularizar. En este caso, se obtiene una gran mejoría en cuanto a la potencia de la señal en relación al ruido, una gran mejoría en cuanto a emparejar la estadística de la imagen original con la deconvolucionada, se fortalecen los puntos de mayor intensidad destacando las zonas más interesantes para segmentar, pero se pierde Información Mutua, posiblemente por una gran degradación de la señal que se lleva consigo al ruido.

En la Figura 3.11 se muestran los resultados para los mejores modelos, donde se confirma el parecido entre 3.11a, 3.11c y 3.11d y por otro lado 3.11e, es la que logra el mejor resultado comparando con 3.11b. Esto permite concluir que Richardson-Lucy es el mejor método para utilizar en el procesamiento de los stacks reales para contrarrestar el efecto de la PSF del microscopio confocal de fluorescencia.



(a) Imagen cruda (b) Huygens

(c) RIF (d) Tikhonov-Miller (e) Richardson-Lucy

Figura 3.11: Corte de stack real al que se le aplicaron varios métodos de deconvolución. La imagen (b) es la deconvolución con Huygens, imagen que se asume como ground truth.

Capítulo 4

Segmentación

La segmentación consiste en separar la imagen en distintos subgrupos, esto es crucial para extraer e interpretar información morfológica en organelos. Es una de las tareas más difíciles en el procesamiento de imágenes, su precisión determina el eventual éxito o fracaso de los procedimientos de análisis computarizados y es por esto que se debe hacer especial énfasis en lograr buenos resultados.

Es particularmente importante la correcta segmentación de imágenes médicas, dado que es una tarea esencial en el diagnóstico clínico y ha sido ampliamente estudiada. Es inmensamente útil para los médicos, ya que no solo especifica que una imagen contiene algo interesante, sino también dónde mirar.

En este capítulo se presentan conceptos relacionados a la segmentación en general, y a los tres enfoques que fueron analizados. Luego se explican en detalle los métodos utilizados y las variaciones implementadas, para luego explicar los experimentos realizados y, finalmente, analizar los resultados y obtener conclusiones de los mismos.

4.1. Marco Teórico

Concretamente, la segmentación consiste en la clasificación de cada pixel (o voxel en el caso 3D) en distintas clases. Uno de los casos más simples de segmentación es la umbralización, donde solo se cuenta con dos clases (*foreground* y *background*, o equivalentemente objetos y fondo) y por lo tanto la imagen resultante es binaria.

En esta sección se explicarán conceptos fundamentales de cada una de las familias de los métodos estudiados, los cuales son una variante de Otsu multicapa a la que se le llamará $Otsu_{2th}$, K-Means y U-Net/Resnet, una variante de la arquitectura de red neuronal U-Net que utiliza bloques residuales.

4.1.1. Métodos de umbralización por histograma

Existen diversos algoritmos que tienen como objetivo hallar el mejor umbral en base a ciertos criterios de optimización. Estos algoritmos parten del histograma de la imagen monocromática y, en función de los valores de dicho histograma,

minimizan distintas métricas como la entropía de Shannon [45] o la distancia de Kullback-Leibler, entre otras.

Algunos de los métodos de umbralización más conocidos son Huang [46], Li [47], Isodata [48], Entropía de Renyi [49], Entropía de Yen [50], Otsu y Otsu Múltiple. Este último fue uno de los métodos utilizados para los experimentos dado que en un trabajo previo [51] fue definido como el algoritmo que mejor se aproximaba al criterio de segmentación del referente médico de IMAGINA.

4.1.2. Métodos de aprendizaje no supervisado

Los métodos de aprendizaje no supervisado son métodos de aprendizaje automático donde los modelos se ajustan a las observaciones. Se distingue del aprendizaje supervisado por el hecho de que no hay un conocimiento a priori.

Dado que la segmentación es un problema de clasificación en el que no necesariamente se conocen las etiquetas de cada voxel, un enfoque de aprendizaje no supervisado parece apropiado. Dentro de los métodos de aprendizaje no supervisado se tiene el *clustering*, el cual tiene como finalidad lograr el agrupamiento de conjuntos de datos no etiquetados, para lograr construir subconjuntos de estos. Dado que es de interés separar entre *foreground* y *background* sin conocer necesariamente las etiquetas de cada voxel, este enfoque parece adecuado para el problema. Es por esto que se estudia *K-Means* [52], un método de *clustering* que tiene como objetivo la partición de un conjunto de datos en K subconjuntos, donde cada voxel pertenece al grupo cuyo valor medio es más cercano. El funcionamiento del algoritmo será explicado en detalle en la Sección 4.2.

4.1.3. Redes Neuronales

Las redes neuronales artificiales son modelos computacionales que procesan información imitando el funcionamiento de las neuronas biológicas. Consisten en un conjunto de unidades, llamadas neuronas artificiales, conectadas entre sí para transmitirse señales. Una neurona es, básicamente, una operación entre dos vectores. Los fundamentos básicos de las redes neuronales pueden encontrarse en el Apéndice B.

En este trabajo se utilizará una variante de la U-Net [53], una arquitectura muy utilizada en el ámbito de las imágenes biomédicas.

4.2. Métodos

Para los métodos a explicar se asume que se está trabajando con volúmenes de tamaño $M \times N \times O$, donde el máximo valor de intensidad es L.

4.2.1. Otsu

Antes de comenzar es importante presentar algunas magnitudes que son necesarias para entender el método de Otsu en tres dimensiones. Comenzando por la

probabilidad de intensidad $p(l)$, esta magnitud se expresa como

$$p(l) = \frac{g(l)}{M \times N \times O} \tag{4.1}$$

y dice cuál es la probabilidad de ocurrencia para una intensidad l. En la Ecuación 4.1 $g(l)$ es la suma de todos los vóxeles de intensidad l.

Suponiendo que se tienen dos agrupaciones, se puede definir la probabilidad de ocurrencia de cada una como

$$P_0(l) = \sum_{i=0}^{l} p_i \tag{4.2}$$

$$P_1(l) = \sum_{i=l+1}^{L} p_i \tag{4.3}$$

donde L es el valor máximo de intensidad, y $P_0(l)$ y $P_1(l)$ son las probabilidades de ocurrencia del *background* y del *foreground* respectivamente. Nótese que estas agrupaciones están definidas por el umbral de intensidad l. Siguiendo con las agrupaciones, se puede hallar la media y desviación estándar de cada una

$$\mu_0(l) = \frac{1}{P_0(l)} \sum_{i=0}^{l} ip(i) \tag{4.4}$$

$$\mu_1(l) = \frac{1}{P_1(l)} \sum_{i=l+1}^{L} ip(i) \tag{4.5}$$

$$\sigma_0^2(l) = \sum_{i=0}^{l} [i - \mu_0(l)]^2 \frac{p_i}{P_0(l)} \tag{4.6}$$

$$\sigma_1^2(l) = \sum_{i=l+1}^{L} [i - \mu_1(l)]^2 \frac{p_i}{P_1(l)} \tag{4.7}$$

Conceptualmente Otsu encuentra el umbral que minimiza la varianza interna de clases ponderada [54] (*weighted within-class variance*). Esta varianza se define como

$$\sigma_w^2(l) = P_0(l)\sigma_0^2(l) + P_1(l)\sigma_1^2(l). \tag{4.8}$$

Además, se sabe que la expresión de la varianza total de la imagen no depende de t y se puede expresar como

$$\sigma = \sigma_w^2(l) + q_1(l)\,[1 - q_1(l)]\,[\mu_1(l) - \mu_2(l)]^2 \tag{4.9}$$

Al ser σ fijo y dado por la imagen, se puede ver que minimizar σ_w^2 equivale a maximizar el segundo término de la suma, el cual representa la varianza entre clases y de aquí en más será llamado σ_B^2.

$$\sigma_B^2(l) = q_1(l)\,[1 - q_1(l)]\,[\mu_1(l) - \mu_2(l)]^2 \tag{4.10}$$

El algoritmo de Otsu se puede resumir de la siguiente manera[1]:

- Inicializar $t = 0$
- Calcular los $p_i, i = 0, 1, ..., L$.
- Calcular las probabilidades de cada clase $P_j(l)$ y sus medias $\mu_j(l) \quad j = 1, 2$.
- Calcular la varianza entre clases $\sigma_B^2(l)$, usando la ecuación (4.10)
- Se obtiene el umbral de Otsu l^*, valor para el cual $\sigma_B^2(l)$ es máximo. Si el máximo no es único, l^* se obtiene promediando los valores correspondientes a los distintos máximos detectados.

4.2.2. Otsu Múltiple

En el caso que se quieren identificar M regiones es posible generalizar el método de Otsu. Esta generalización es conocida como Otsu Múltiple, donde $\sigma_w^2(t)$ se halla como

$$\sigma_w^2(l) = \sum_{i=1}^{M} P_i(l)\sigma_i^2(l). \tag{4.11}$$

Esto implica calcular $P_i(l)$ y $\mu_i(l)$ para cada clase, y hallar una expresión para $\sigma_B^2(l)$ a partir de su definición.

$$\sigma_B^2(l) = \sum_{i=1}^{M} P_i(l) \left[\mu_i(l) - \sum_{j=1}^{M} P_j(l)\mu_j(l) \right]^2 \tag{4.12}$$

Dado que solo interesa distinguir entre señal y fondo, para este trabajo se utiliza Otsu múltiple con dos umbrales, tomando únicamente el más restrictivo. A esta umbralización se le llamará $Otsu_{2th}$ de aquí en más.

4.2.3. K-Means

Es un método iterativo de agrupamiento (clustering), que tiene como objetivo la partición de un conjunto de n observaciones en K grupos $S_1, S_2, ..., S_K$ en el que cada observación pertenece al grupo cuyo valor medio es más cercano. En este caso las observaciones van a ser los vóxeles de las imágenes trabajadas y se evaluará el desempeño del algoritmo con distintos K, siempre tomando el grupo cuya intensidad media es máxima.

La lógica del algoritmo es la siguiente:

1. Inicialización: Se elige la localización de los centroides μ_j de los K grupos.

[1]Nótese que al trabajar con $\sigma_B^2(l)$ no hay necesidad de calcular $\sigma_1^2(l)$ y $\sigma_2^2(l)$.

2. Asignación: Se asigna cada dato x_i al centroide μ_j más cercano, esto es

$$S_j(t) = \{x_n : \ ||x_n - \mu_j(t)|| \leq ||x_n - \mu_l(t)|| \quad l = 1, 2, ..., K\}. \tag{4.13}$$

3. Actualización: Se actualiza la posición del centroide a la media aritmética de las posiciones de los datos asignados al grupo.

$$\mu_j(t+1) = \frac{1}{S_j(t)} \sum_{x_l \in S_j(t)} x_l \tag{4.14}$$

Los pasos 2 y 3 son iterativos, el algoritmo finaliza al alcanzar el límite de iteraciones definido por el usuario o una diferencia entre centroides lo suficientemente pequeña.

$$|\mu_j(t+1) - \mu_j(t)| \leq \epsilon \quad \forall j = 1, 2, ..., K$$

Para adaptar este algoritmo a la umbralización de imágenes, cada intensidad toma el valor de su centroide correspondiente, luego se binariza tomando el mayor valor de los centroides como 1 (objeto) y el resto como 0 (fondo).

4.2.4. U-Net/ResNet

Propuesta por primera vez en 2015, la U-Net es una red neuronal completamente convolucional (es decir, no hay una operación completamente conectada en la red) que clasifica cada voxel de las imágenes de entrada en distintas clases, esto se conoce como segmentación semántica.

La U-Net es además un modelo de tipo *encoder-decoder*. El modelo encoder-decoder consiste en dos caminos, uno constrictivo (codificación) y una expansivo (decodificación). La Figura 4.1 muestra el esquema tradicional de la arquitectura. A continuación se explican las funciones de las capas que componen la arquitectura, las funciones de activación, entre otros.

En el mismo año fue propuesta también la ResNet [55], una arquitectura que brindaba la posibilidad de entrenar redes con muchas capas utilizando bloques residuales. La combinación de estas arquitectura resulta en la U-Net/Resnet, una red neuronal con mayor eficiencia que la U-Net habitual en casi todos los casos gracias a la ayuda que brindan los *Resblocks* para obtener una curva de pérdida suave y a evitar la desaparición del gradiente [56].

Básicamente la U-Net/ResNet consiste en codificar la información de la imagen de entrada a través de distintos filtros convolucionales, cuyos coeficientes son aprendidos por la red. Luego se decodifica la información en el camino expansivo de la red para obtener finalmente una imagen segmentada. Las funciones y los conceptos fundamentales para entender el funcionamiento de esta arquitectura son explicadas en el Apéndice C

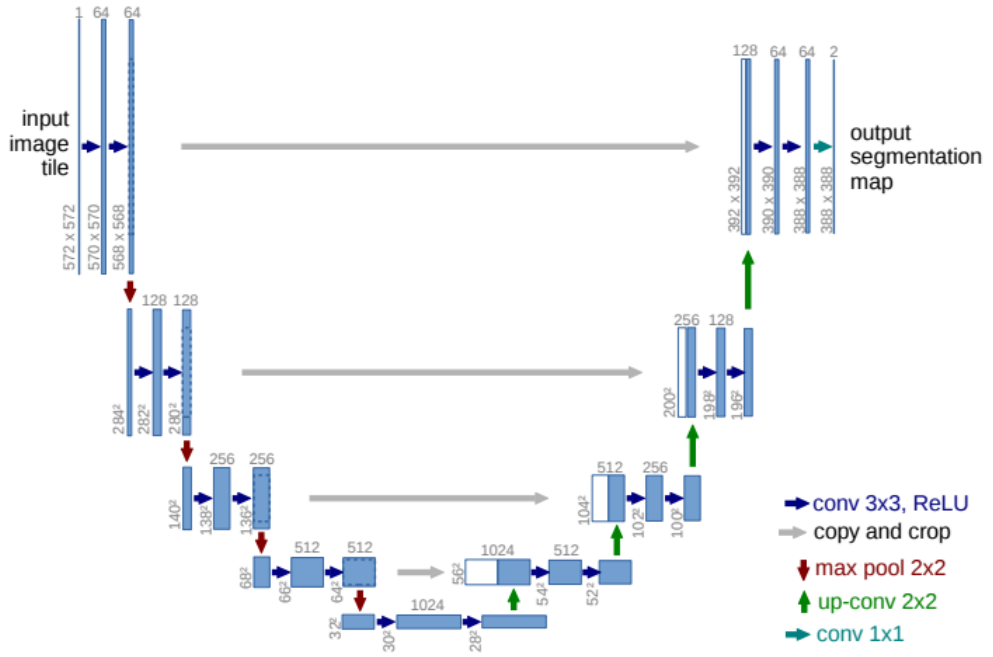


Figura 4.1: Arquitectura de U-Net. Los números sobre cada capa indican la cantidad de feature maps, las flechas grises de "*copy and crop*" representan las *skip connections*, las cuales son fundamentales para captar detalladamente la información a segmentar. [53]

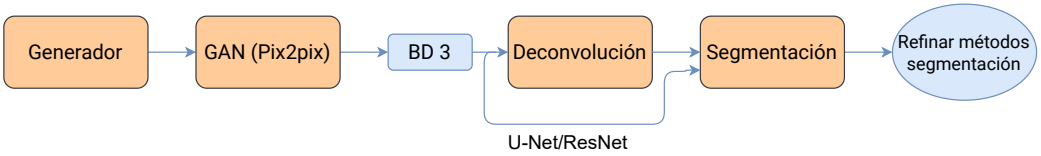


Figura 4.2: Diagrama de flujo en etapa de segmentación

4.3. Experimentos Realizados

Los experimentos de segmentación se realizaron con BD3 como lo muestra la Figura 4.2, donde la deconvolución utilizada es Richardson-Lucy (RL) con 50 iteraciones (N=50) por ser el método que tuvo mejores resultados en el capítulo anterior.

En una primera instancia se analizan K-Means con K variando entre dos y cinco, Otsu múltiple con dos umbrales utilizando el mayor para binarizar ($Otsu_{2th}$), y U-Net/ResNet con distintos entrenamientos donde la época será elegida evaluando el error en un conjunto de validación.

Para la validación de la red se utilizan datos sintéticos, usando $Otsu_{2th}$ como *ground-truth* en estos últimos. En este caso se utiliza un conjunto de datos de validación, los cuales no fueron usados en ninguna de las bases de datos. En cuanto a la evaluación de los modelos se utiliza BD3, base de datos compuesta únicamente por datos sintéticos. Para BD1 (la base de datos utilizada para entrenar la red)

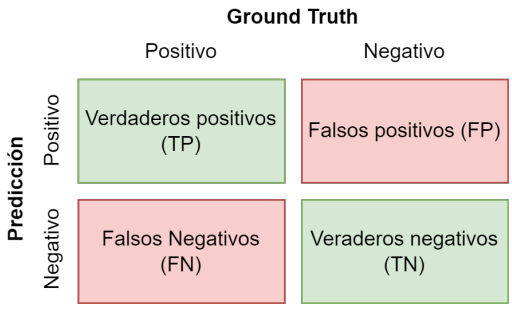


Figura 4.3: Matriz de confusión donde se define en verde los aciertos (verdaderos positivos y verdaderos negativos) y los errores (falsos positivos y falsos negativos).

es fundamental tener imágenes reales y un "*ground-truth*" de las mismas, debido a que no tenerlas implicaría que el cambio de dominio de sintéticas a reales provoque malos resultados en segmentación. Es por esto que para BD1 se utiliza $Otsu_{2th}$. Es importante señalar que esto no implica que la red esté aprendiendo $Otsu_{2th}$, dado que tiene también imágenes sintéticas generadas con la GAN y un verdadero *ground-truth* de las mismas. Además, las imágenes reales fueron deconvolucionadas con Huygens antes de la umbralización de $Otsu_{2th}$, por lo que la red esta aprendiendo una combinación entre revertir el efecto de la GAN, y el conjunto Huygens-$Otsu_{2th}$.

Para BD3 no se utilizan las imágenes reales porque no son un *ground-truth* como tal, y no se quiere tomar una decisión en base a qué tanto se aproximan las segmentaciones a $Otsu_{2th}$, siendo ésta una de las segmentaciones a evaluar.

La red está aprendiendo una combinación de Huygens-$Otsu_{2th}$ y la segmentación de datos sintéticos, cuyo *ground-truth* existe y fue utilizado, por esto, Los resultados de la red se ven influenciados por $Otsu_{2th}$, pero esto no implica ni que $Otsu_{2th}$ sea el mejor método ni que nuestra red tenga resultados equivalentes. En las próximas etapas se estudiarán los resultados de medición de características, donde se podrán comparar los resultados de los parámetros antes y después del pipeline para determinar qué "camino" los afectó menos, y los resultados de clasificación, donde se buscará predecir la etiqueta en base a los parámetros morfológicos obtenidos.

Para evaluar los distintos métodos se debe definir las métricas utilizadas. En la Figura 4.3 se muestra la nomenclatura utilizada para los distintos tipos de acierto y error, que se de detalla en el siguiente listado:

- True Positive (TP): Cantidad de vóxeles con etiqueta positiva tanto en la segmentación como en el *ground-truth*.
- False Positive (FP): Cantidad de vóxeles con etiqueta positiva en la segmentación y etiqueta nula en el *ground-truth*.
- True Negative (TN): Cantidad de vóxeles con etiqueta nula tanto en la segmentación como en el *ground-truth*.

- False Negative (FN): Cantidad de vóxeles con etiqueta nula en la segmentación y etiqueta positiva en el *ground-truth*.

Con estas cantidades se definen las métricas:

$$Dice = \frac{2TP}{2TP + FP + FN} \tag{4.15}$$

$$Recall = \frac{TP}{TP + FN} \tag{4.16}$$

$$Specificity = \frac{TN}{TN + FP} \tag{4.17}$$

$$Precision = \frac{TP}{TP + FP} \tag{4.18}$$

donde *Recall* y *Specificity* representan la tasa de verdaderos positivos (TPR por sus siglas en inglés) y tasa de verdaderos negativos (TNR por sus siglas en inglés) respectivamente. Estas métricas miden qué tantos vóxeles fueron considerados positivos (TPR) o negativos (TNR) respecto a la cantidad total de vóxeles positivos o negativos del *ground-truth*. *Precision* también es conocido como *Positive Predictive Value (PPV)*, e indica cuantos vóxeles fueron asignados correctamente con intensidad 1 respecto a la cantidad total de vóxeles asignados con esta intensidad. En cuanto al *Dice* [57], esta métrica es probablemente la más utilizada en el contexto de imágenes biomédicas [58] y muestra la relación entre la intersección de ambas imágenes (segmentación y *ground-truth*) y la unión de estas. Otra forma de escribir la Ecuación (4.15) es

$$Dice = \frac{2\,|Segmentacion \cap Ground\text{-}truth|}{|Segmentacion| \cup |Ground\text{-}truth|} \tag{4.19}$$

donde se ve claramente la relación.

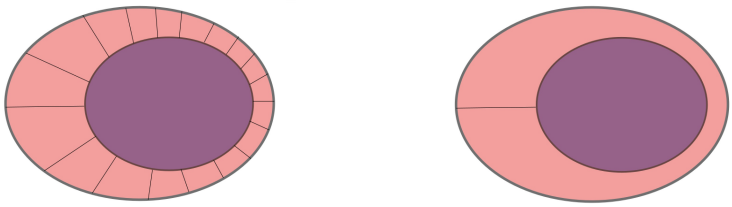
Por último, se calcula también la distancia de *Hausdorff*, la cual mide la distancia entre dos subconjuntos de un espacio métrico. De manera informal, dos conjuntos están cerca según la distancia de Hausdorff si todos los puntos de cualquiera de los conjuntos están cerca de algún punto del otro conjunto. La distancia de Hausdorff es la distancia máxima recorrida dado un voxel con información de una de las imágenes para llegar a un voxel con señal en la otra imagen.

En otras palabras, es la mayor de todas las distancias desde un punto de un conjunto hasta el punto más cercano del otro conjunto. En la Figura 4.4 se muestra en 4.4a varias distancias entre los dos cuerpos a estudiar y en la 4.4b la distancia de Hausdorff, siendo la máxima de estas.

A continuación se procede a explicar cómo se entrenó la U-Net/ResNet y cómo se seleccionó la mejor época para los experimentos de segmentación.

4.3.1. Entrenamiento y validación de la U-Net/ResNet

La U-Net/ResNet se entrenó utilizando tanto stacks sintéticos como reales brindados por el grupo IMAGINA. El entrenamiento de la red necesita stacks cúbicos,



(a) (b)

Figura 4.4: Distancias entre la estructura original (violeta) y la estructura segmentada (rosa), la distancia máxima entre los contornos de ambas estructuras es la distancia de Hausdorff.

es decir de iguales dimensiones en los tres ejes, eso implica una etapa de reescalado, recortado, *paddeo* o apilado. Los stacks sintéticos que se disponía eran de tamaño $256 \times 256 \times 64$ los cuales se recortaron a tamaño $128 \times 128 \times 64$ y posteriormente se apiló con otro stack sintético, obteniendo un stack de dimensiones $128 \times 28 \times 128$, este proceso se puede visualizar en la Figura 4.5a. Para el caso de los stacks reales se tomó dos caminos, apilar a modo similar que en los stacks sintéticos como se muestra en la Figura 4.5b, sustituyendo la etapa de recorte por un reescalado y por otro lado, *paddeando* con cortes con ruido gaussiano. El ruido gaussiano fue estimado en base a la media y la desviación estándar del fondo de los stacks. La Figura 4.5c muestra el procedimiento seguido para éste último caso.

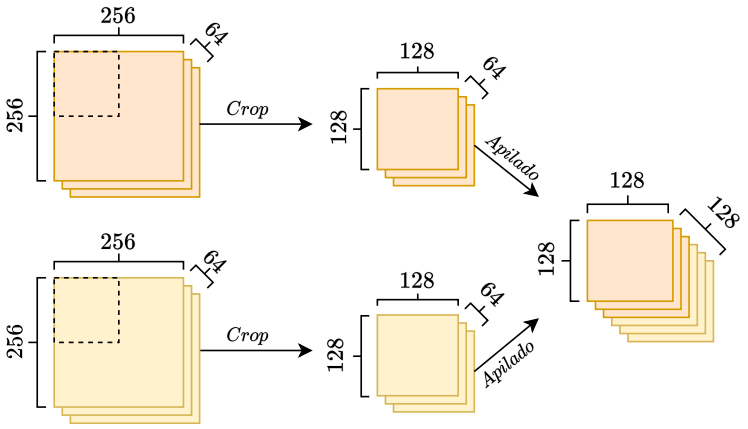
La red se entrenó tres veces con distintos conjuntos de entrenamiento y se incorporó *dropout* para evitar el sobreajuste. El entrenamiento fue de 550 épocas y para validar se utilizan 20 stacks sintéticos evaluando el *dice* promedio por época de todos los stacks utilizados.

Entrenamiento 1: Sintéticos y reales apilados

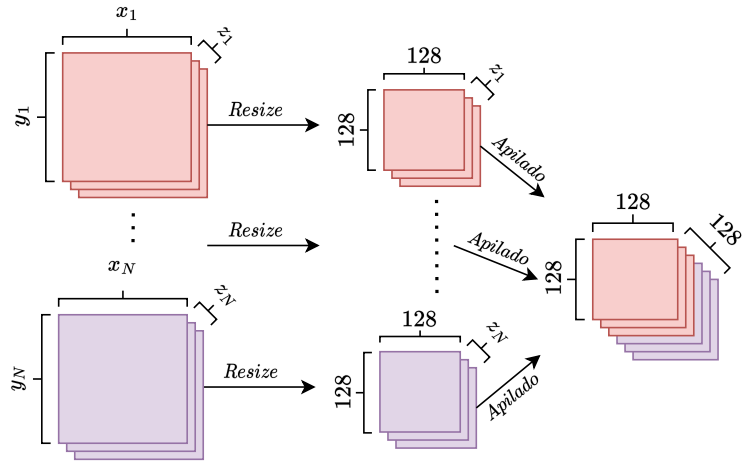
Del preprocesado de los stacks sintéticos se obtuvo 140 stacks sintéticos de tamaño $128 \times 128 \times 128$ y del preprocesado de los reales se produjo 37 stacks con las mismas dimensiones que los sintéticos. En la Figura 4.6 se muestra los resultados obtenidos para este modelo, donde el mejor resultado se obtiene para la época 22 con $dice = 0{,}67$. El modelo se sobreajusta rápidamente y la media móvil permite mostrar la tendencia decreciente.

Entrenamiento 2: Sintéticos apilados y reales con *padding*.

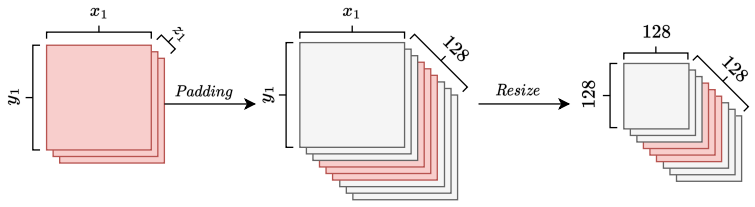
En este caso del preprocesado de los stacks sintéticos se obtuvo 200 stacks sintéticos de tamaño $128 \times 128 \times 128$ y del preprocesado de los reales se produjo 472 stacks con *padding* que tienen las mismas dimensiones que los sintéticos. En la Figura 4.7 se muestran el promedio del *dice* de todos los stacks del conjunto de validación para todas las épocas entrenadas. En este caso el mejor resultado se da en la época 18 con $dice = 0{,}69$. De la gráfica se puede apreciar que la red se sobreajusta rápido, ya que a través de la media móvil se puede apreciar la métrica elegida decrece casi desde las primeras épocas.



(a) El preprocesamiento de los stacks sintéticos (*fakes*) consta de una etapa de recorte (*crop)* y luego de apilado de dos stacks para obtener un stack del tamaño objetivo $128 \times 128 \times 128$. En naranja y amarillo dos stacks sintéticos distintos.



(b) Preprocesamiento de los stacks reales que cuenta con una etapa de reescalado (*resize*) en x e y a $128 \times 128 \times z_n$. Luego se apilan los stacks reales hasta llegar a 128 en profundidad, es decir $z_1 + z_2 + \cdots + z_N = 128$, pudiendo ser parcialmente incorporado los stacks y utilizándose el resto del stack en el siguiente apilado. En rojo y violeta stacks reales distintos.



(c) Preprocesado que realiza *padding* para cubrir lograr la profundidad deseada. El *padding* se realiza estimando la media y la desviación estándar del fondo de la señal y generando ruido gaussiano en los cortes a incorporar en el stack real. Luego hay una etapa de reescalado en x e y. En rojo el stack real y en gris el *padding*.

Figura 4.5: Preprocesamiento de los stacks reales y sintéticos para utilizar en el entrenamiento de la U-Net/ResNet.

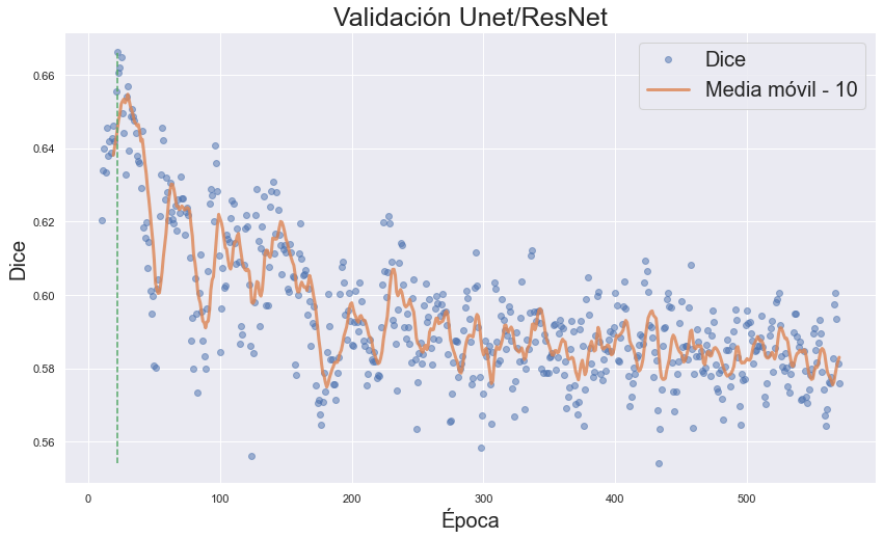


Figura 4.6: Promedio del *Dice* de todos los stacks por época. En naranja se grafica la media móvil con 10 datos, en verde punteado se indica la mejor época.

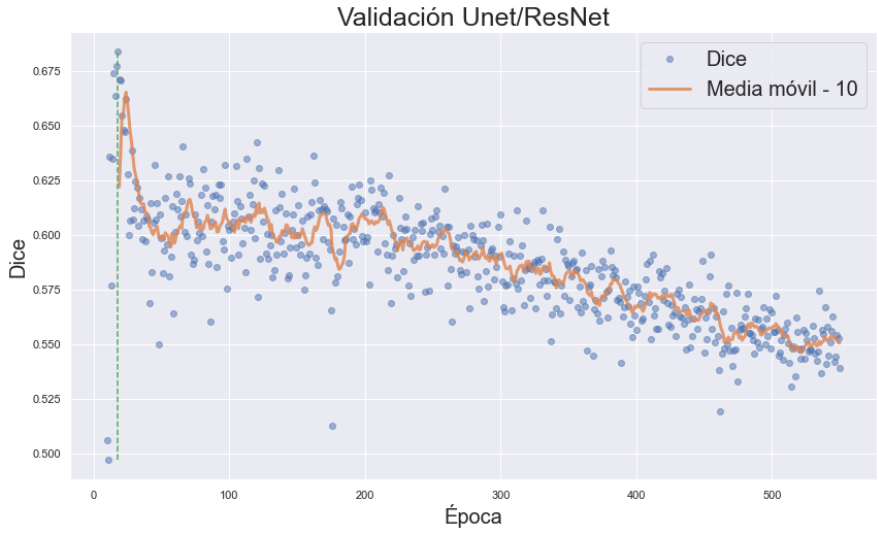


Figura 4.7: Promedio del *Dice* de todos los stacks por época. En naranja se grafica la media móvil con 10 datos, en verde punteado se indica la mejor época. La red no tiene *dropout*.

Entrenamiento 3: Sintéticos apilados, reales con *padding* y red con *dropout*.

El preprocesado en este caso es igual que el caso anterior y con las mismas imágenes, solo cambia la arquitectura de la red, que incorpora *dropout* con probabilidad $p = 0{,}5$. En la Figura 4.7 se muestran el promedio del *dice* de todos los stacks del conjunto de validación. En este caso el mejor resultado se da en la época 331 con $dice = 0{,}63$. Comparando con la Figura 4.7 no se logró resultados tan buenos pero se contrarrestó el sobreajuste ya que el mejor resultado se obtuvo en épocas avanzadas.

Conclusiones del entrenamiento

Teniendo en cuenta que se dispone de un generador de volúmenes las distintas formas de entrenar el modelo, son infinitas y se tomó decisiones en base a la cantidad de datos reales que se disponía y la proporción de datos sintéticos con respecto a los reales. Por otro lado, el entrenamiento de la U-Net/ResNet conlleva grandes tiempos, por lo que no se pudo realizar de forma exahustiva el entrenamiento y se tuvo que seleccionar una cantidad acotada de experimentos. Los tres entrenamientos no logran un resultado que sea claramente decisivo, los modelos sin *dropout* son los que dieron mejor resultado y con pocas épocas de entrenamiento parece lograrse un resultado decente. Mientras que el modelo con *dropout* a pesar de que en magnitud el *dice* sea menor, se obtiene en épocas avanzadas mostrando la utilidad de la regularización para contrarrestar el sobreajuste. En base a las métricas obtenidas el mejor resultado es la arquitectura sin *dropout* y que cuenta con más stacks debido al *padding* y la posibilidad de mantener la cantidad de stacks reales original.

4.3.2. Resultados

La Tabla 4.1 muestra los resultados promedios de las distintas segmentaciones para BD3, utilizando los métodos seleccionados.

En la Figura 4.9 puede verse el proceso de generación de una imagen, su deconvolución y los distintos resultados de las segmentaciones. Nuevamente, se elige mostrar solamente un corte de la imagen por ser más fácil de visualizar, pero es importante tener en cuenta que se ésta trabajando con imágenes 3D.

Puede verse que las segmentaciones detectan señal donde hay ruido, quedando

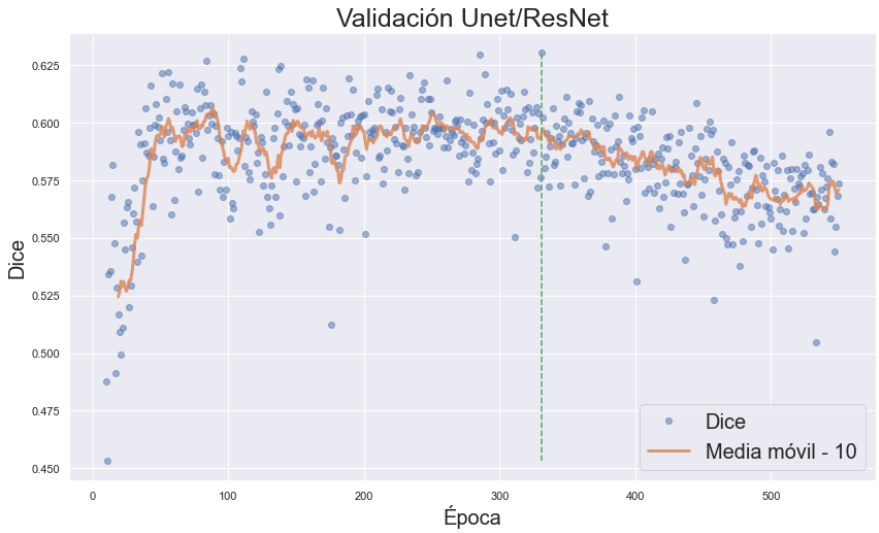


Figura 4.8: Promedio del *Dice* de todos los stacks por época. En naranja se grafica la media móvil con 10 datos, en verde punteado se indica la mejor época. La red tiene *dropout* con $p = 0{,}5$.

	Parámetros	Recall	Specificity	Precision	Dice	Hausdorff
$Otsu_{2th}$	-	0.315	0.9995	0.893	0.462	61.978
K-Means	K=2	0.610	0.997	0.740	0.666	73.246
K-Means	K=3	0.278	0.9995	0.893	0.415	62.904
K-Means	K=4	0.137	0.99989	0.944	0.235	57.499
K-Means	K=5	0.076	0.99995	0.952	0.139	62.880
U-Net/ResNet	T=1, E=22	0.602	0.99797	0.780	0.677	98.313
U-Net/ResNet	T=2, E=18	0.434	0.9987	0.800	0.560	84.927
U-Net/ResNet	T=3, E=331	0.555	0.9983	0.802	0.650	84.239

Tabla 4.1: Resultados de la etapa de segmentación, donde pueden verse los promedios de las distintas métricas explicadas anteriormente para todas las imágenes de BD3. En los parámetros de la U-Net/ResNet la T representa el entrenamiento utilizado y la E representa la época.

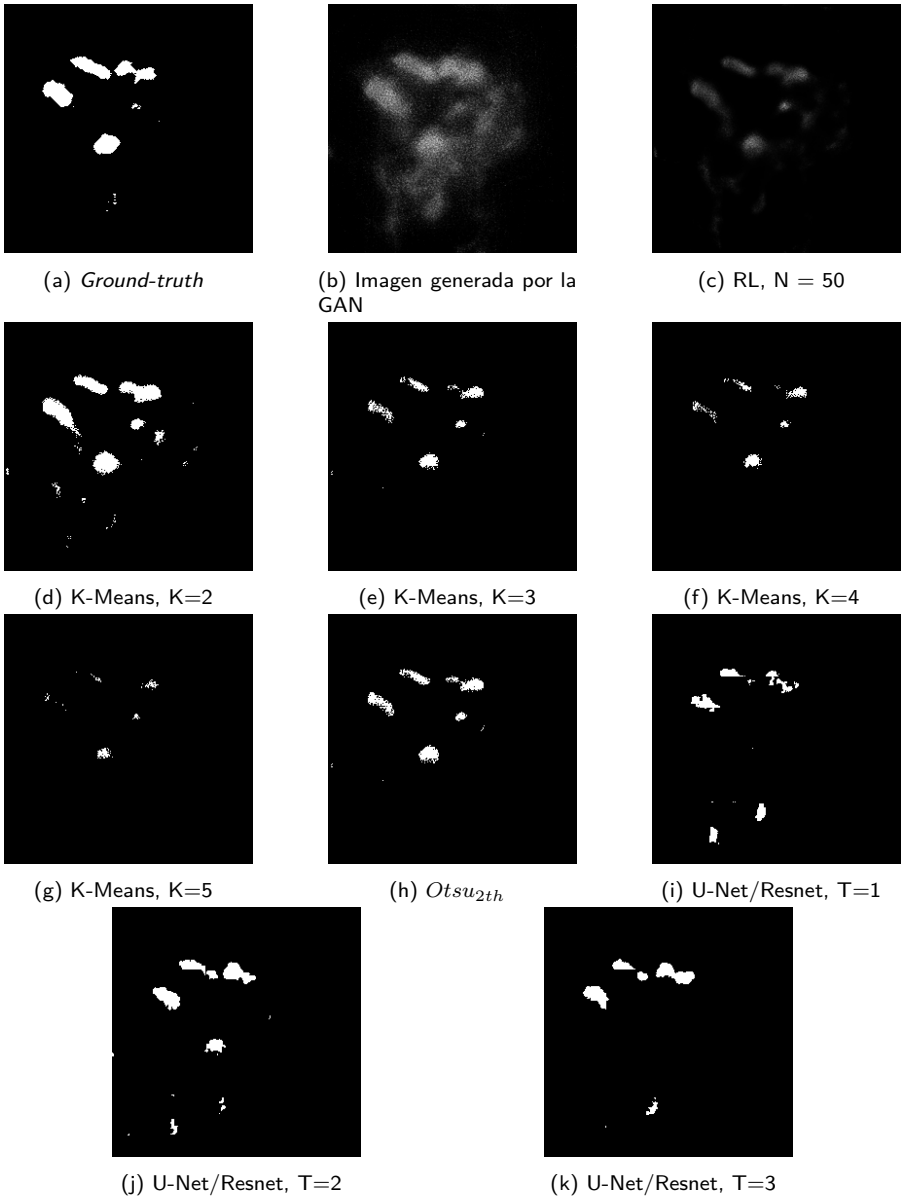
	Parámetros	Recall	Specificity	Precision	Dice	Hausdorff
$Otsu_{2th}$	-	0.244	0.9997	0.922	0.383	80.150
K-Means	K=2	0.577	0.9983	0.809	0.671	32.897
K-Means	K=3	0.211	0.964	0.893	0.335	77.759
K-Means	K=4	0.074	0.99995	0.954	0.139	105.091
K-Means	K=5	0.0288	0.964	0.925	0.055	126.885
U-Net/ResNet	T=1, E=22	0.588	0.9984	0.815	0.680	74.256
U-Net/ResNet	T=2, E=18	0.415	0.99897	0.827	0.547	67.001
U-Net/ResNet	T=3, E=331	0.540	0.9985	0.821	0.646	69.687

Tabla 4.2: Resultados de la etapa de segmentación luego del posprocesado, donde pueden verse los promedios de las distintas métricas explicadas anteriormente para todas las imágenes de BD3.

muchos vóxeles aislados, siendo ésto un problema para la extracción de características morfológicas. Es por esto que se decidió realizar un postprocesado en el que se aplica *opening*, *closing* y se filtran los volúmenes que estén por debajo de cierto umbral, definido en base al trabajo previo realizado con el grupo IMAGINA [51]. A continuación se pueden ver los resultados luego del postprocesado de las imágenes en la Tabla 4.2.

4.4. Análisis de resultados

En las imágenes de la Figura 4.9, puede verse que cuanto más restrictiva es la segmentación, más se alejan los resultados del *ground-truth*. Esto muestra diferencias frente a lo esperado, dado que deja de ser cierto el hecho de que $Otsu_{2th}$, es lo más aproximado a la solución. De hecho, la Figura 4.11 muestra los resultados



(a) *Ground-truth* (b) Imagen generada por la GAN (c) RL, N = 50

(d) K-Means, K=2 (e) K-Means, K=3 (f) K-Means, K=4

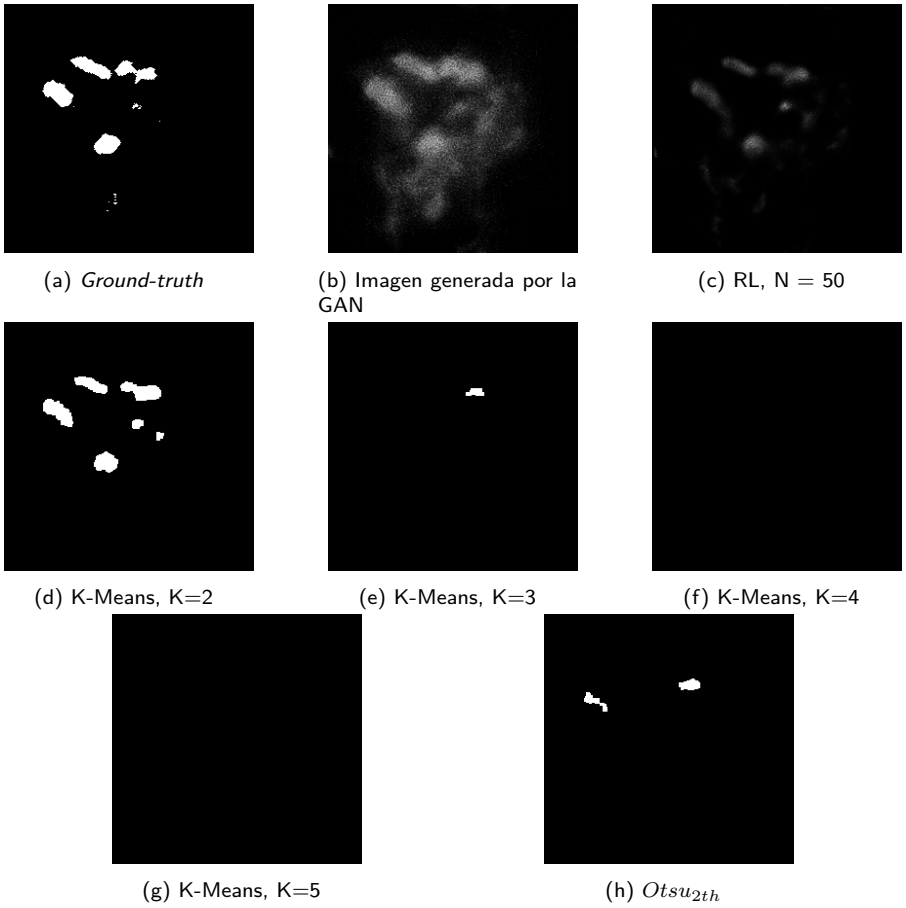
(g) K-Means, K=5 (h) $Otsu_{2th}$ (i) U-Net/Resnet, T=1

(j) U-Net/Resnet, T=2 (k) U-Net/Resnet, T=3

Figura 4.9: Un corte del proceso de generación, deconvolución y segmentación en una imagen sintética. Para $K > 2$ se utiliza la agrupación con intensidades más altas, tomando estas como *foreground* y el resto como *background*.

de Otsu con un solo umbral, siendo notoriamente más similar al *ground-truth*.

La Tabla 4.1 corrobora que el método menos restrictivo es el que brinda mejores resultados. El hecho de que K-Means con $K = 2$ tenga el mejor *Recall* implica que fue el método que acertó más vóxeles con información respecto al *ground-truth*, lo que tiene sentido dado que es el método más permisivo. Sin embarlo, la *Specificity* permite ver que ese resultado no viene dado únicamente por ser permisivo. Esto es porque el hecho de tener buenos resultados de *Specificity* implica acertar vóxeles



(a) *Ground-truth* (b) Imagen generada por la GAN (c) RL, N = 50

(d) K-Means, K=2 (e) K-Means, K=3 (f) K-Means, K=4

(g) K-Means, K=5 (h) $Otsu_{2th}$

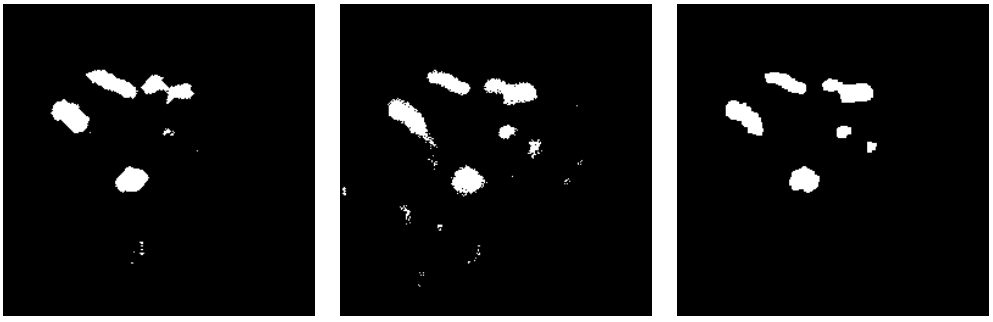
Figura 4.10: Un corte del proceso de generación, deconvolución y segmentación luego del posprocesado en una imagen sintética. Para $K > 2$ se está tomando la agrupación con intensidades más altas, tomando estas como *foreground* y el resto como *background*.

sin señal, o lo que es lo mismo, vóxeles de fondo.

El hecho de que tenga la peor *Precision* tiene mucho que ver con la permisividad del método, dado que tiene sentido que el método más permisivo falle más en cuanto a si los vóxeles a los que se les asigna el valor 1 en la segmentación efectivamente tenían información en el *ground-truth*. Sería raro que los métodos más restrictivos (y por lo tanto que menos valores 1 asignan a las imágenes) fallaran más que si los vóxeles efectivamente tenían información o no.

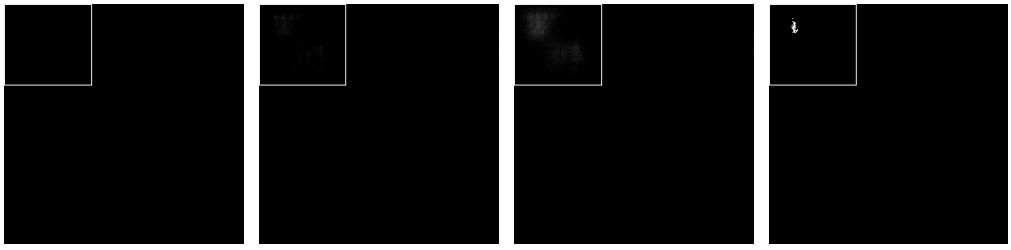
En cuanto a la U-Net/ResNet, se puede ver que los resultados no son tan distintos entre si, obteniendo métricas similares. Se puede observar que cuando la red no tiene *dropout*, las mejores épocas se dan en etapas muy tempranas del entrenamiento. En las Figuras 4.6 y 4.7 puede verse claramente el sobreajuste en épocas tempranas. En la red con *dropout* el sobreajuste se da algunas épocas después, coincidiendo con lo esperado.

Otra observación de las redes entrenadas es que se vuelven más restrictivos a medida que aumentan las épocas, donde las primeras épocas son ruidosas y



(a) GT (b) Otsu (c) Otsu postprocesada

Figura 4.11: Resultados obtenidos con Otsu.



(a) GT (b) GAN (c) RL, N=50 (d) K-Means, K=2

Figura 4.12: Artefacto generado por la GAN en distintas etapas del pipeline.

las últimas filtran demasiada información. Es importante recordar que las redes neuronales fueron entrenadas con imágenes crudas, por lo que también se utilizaron imágenes crudas en la entrada, evitando la etapa de deconvolución. Esto provoca que la red tuviera un desafío mayor en la segmentación que el resto de los métodos. Saltear el proceso de deconvolución es una ventaja ya que el tiempo de cómputo aumenta sensiblemente en el pipeline.

El *Dice Coefficient* muestra que K-Means con K=2 es uno de los mejores métodos, pero la distancia de Hausdorff muestra lo contrario. Esto se debe al artefacto generado por la GAN mencionado en la Sección 2.4.3, el cual puede verse en la Figura 4.12. El problema surge en K-Means porque al ser justamente un método permisivo, permite pasar dicho artefacto. Por la forma en que se calcula la distancia de Hausdorff, es muy sensible a este tipo de estructuras por lo que no es una métrica muy fiable para estos casos.

Como ya se mencionó en la sección anterior, las segmentaciones devolvieron resultados ruidosos, esto está directamente relacionado con el método de deconvolución. Richardson-Lucy es un método que amplifica el ruido, devolviendo una imagen muy bien deconvolucionada pero con ruido superpuesto a la señal (véase Figura 4.13).

Para mitigar esta problemática se decidió hacer un suavizado de las imágenes, el cual consistió en un *opening* para eliminar vóxeles negros cuyos vecinos eran blancos (debido a ruido solapado donde hay información en la imagen deconvolucionada), un *closing* para eliminar vóxeles blancos cuyos vecinos eran negros

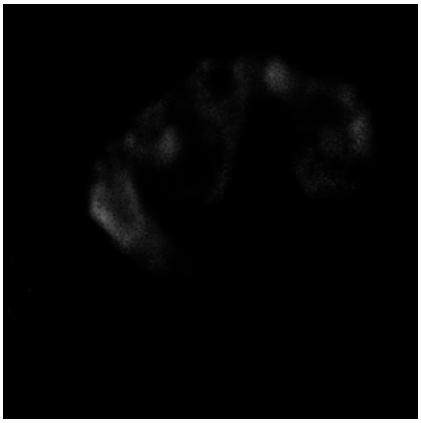


Figura 4.13: Imagen deconvolucionada con zoom a un corte

(debido a ruido en el fondo (*background*) de la imagen deconvolucionada) y se eliminaron estructuras cuyos volúmenes estaban por debajo de un cierto umbral. Esto último logró eliminar correctamente el artefacto generado por la GAN, donde puede verse una clara mejoría en la distancia de Hausdorff en la Tabla 4.2. Además, el *Dice* refleja mejoría en el desempeño de K-Means con K=2. Con otros valores de K esto no sucede debido a que ya de por si eran restrictivos, el *closing* y el filtrado de volúmenes pequeños potenciaron esta restrictividad, obteniendo imágenes con muy poca información. Para la etapa de extracción de parámetros morfológicos se utilizan las imágenes posprocesadas, por lo que el artefacto de la GAN no es un problema.

Algo a destacar es que la amplificación de ruido de RL en la etapa de deconvolución fue abordada en la etapa de segmentación, esto muestra que ambas etapas son complementarias. Es importante tener esto en cuenta, podría pasar que un algoritmo de deconvolución más robusto al ruido, pero con menos efectividad a la hora de revertir el efecto de la PSF y delimitar correctamente los bordes de las estructuras, sea complementado con una segmentación lo suficientemente restrictiva para obtener buenos resultados. Algo de este estilo podría pasar con TRIF y K-Means con un K mayor a 2.

En conclusión, la segmentación es una etapa fundamental del pipeline en la que inevitablemente se deben afrontar problemáticas que quedaron sin solucionar en etapas anteriores (artefacto de la GAN, amplificación de ruido de RL), por ser la última etapa antes de la extracción de características. También se pudo ver que el cambio de dominio es un problema en la segmentación, por lo que no se puede tomar un dominio (los datos sintéticos) como representativo de otro dominio (las imágenes reales). Esto fue demostrado al observar que $Otsu_{2th}$ tiene un peor desempeño que el Otsu simple en los datos sintéticos, cuando en las imágenes reales esta corroborado por referentes médicos que esto no es así. También se concluye que la aplicación de redes neuronales es efectiva para el problema, pero requiere un nivel de atención mayor al resto de los métodos dado que implica comprender la arquitectura utilizada y los parámetros del problema, siendo estos últimos cruciales para el buen desempeño de la red.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 5

Medición y obtención de características morfológicas

El análisis realizado en este capítulo surge de las hipótesis formuladas por los investigadores del grupo IMAGINA respecto a las diferencias presentes entre las redes mitocondriales de personas que padecen ELA y personas sanas. Dichos investigadores plantean que las redes mitocondriales de los primeros tienen mayor división, es decir que tienden a separarse en subredes, y además sostienen que las mitocondrias suelen ser de menor tamaño.

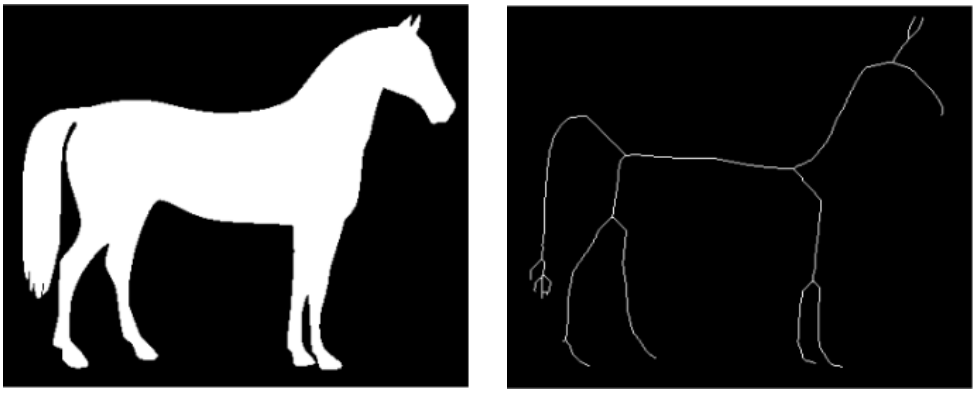
En este trabajo se buscó realizar un análisis que permita obtener tanto las características morfológicas de las mitocondrias, como características de la conectividad de la red. Para esto fue necesario utilizar distintas herramientas según el caso. En el presente capítulo se detallan los enfoques utilizados y los parámetros extraídos.

5.1. Marco teórico

5.1.1. Morfología matemática

La palabra morfología comúnmente denota una rama de la biología que se ocupa de la forma y la estructura de los animales y las plantas. Se usa la misma palabra en el contexto de la morfología matemática como una herramienta para extraer componentes de imágenes que son útiles en su representación y descripción.

En el Apéndice D se introducen algunos conceptos relevantes para la comprensión de las técnicas de morfología matemática aplicadas para la obtención del **esqueleto** de una imagen. El esqueleto es una representación delgada y equidistante del contorno de una estructura. Generalmente enfatiza las propiedades geométricas y topológicas de la morfología como su conectividad, topología, longitud y ancho. La Figura 5.1 muestra un ejemplo de una imagen binaria (Figura 5.1a) y su esqueletización (Figura 5.1b).



(a) Imagen binaria (b) Esqueleto

Figura 5.1: Esqueletización de imagen binaria [59]

5.1.2. Teoría de grafos

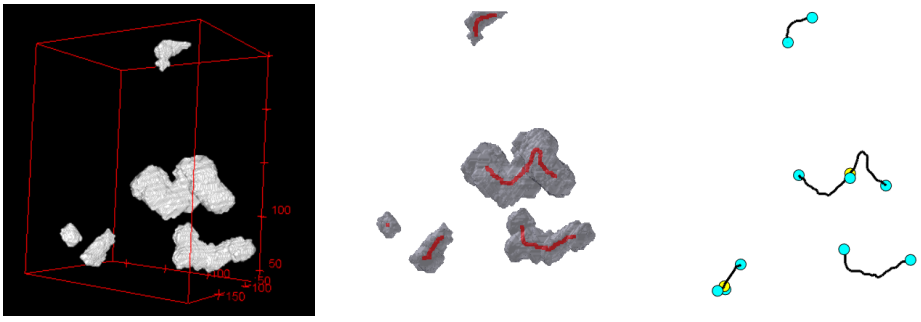
Una vez obtenidos los esqueletos, se utilizan conceptos de **Teoría de grafos** para la extracción de características de red. Un grafo es un objeto matemático conformado por un conjunto de puntos, denominados vértices o nodos, que están relacionados mediante uniones llamadas aristas o *links*. Los nodos se enumerarán y se denotarán por n_i, siendo i el número de nodo. En el Apéndice E se definen varios conceptos relacionados a la Teoría de grafos relevantes para el análisis presentado en el presente capítulo.

5.2. Metodología

Para la extracción de parámetros referidos a la morfología mitocondrial se hizo uso de la librería *SimpleITK* [60]. En primer lugar, se realizó el etiquetado 3D de regiones, entendiendo el etiquetado como el proceso de asignar un número distinto a cada estructura del stack. Es decir, se parte de un stack binario y se obtiene un stack que tiene un nivel de intensidad único por cada estructura. Sobre éstos se utilizó la función *LabelShapeStatisticsImageFilter* que permite obtener parámetros relacionados con la forma a partir de la etiqueta asociada a cierta región de la imagen.

En cuanto a las propiedades de la red mitocondrial, se utilizó como base el trabajo realizado por Zanin *et al* [14]. En dicho trabajo los autores plantean que existe cierta relación entre la enfermedad de Parkinson y la disfunción mitocondrial, pero que esto no se ve reflejado en el análisis de las mitocondrias de manera individual. Por lo tanto, realizaron un análisis enfocado en la red mitocondrial y en cómo interactúan las mitocondrias entre sí. Para ello consideraron la red mitocondrial como un grafo y el análisis se realizó sobre éste.

En primer lugar fue necesario convertir los stacks de imágenes en una matriz de adyacencia que represente un grafo. Para esto, como se sugiere en [14], se utilizaron dos funciones de Matlab [61]. La primera de ellas, *Skeleton3D*, recibe como



(a) Volumen sintético binario. (b) Esqueletización 3D del volumen sintético. (c) Grafo 3D obtenido a partir de la esqueletización 3D.

Figura 5.2: Preprocesamiento de un volumen sintético utilizando las funciones *Skeletonize3D* y *Skel2Graph3D*.

entrada el stack binario y realiza la esqueletización en 3D. La segunda función, *Skel2Graph3D*, toma como entrada el esqueleto 3D generado por *Skeleton3D* y un umbral, y devuelve la matriz de adyacencia del grafo 3D. El parámetro del umbral se utiliza para filtrar posibles artefactos de la esqueletización y su valor se corresponde con el largo mínimo de las ramificaciones. En la Figura 5.2 se presenta un ejemplo del preprocesamiento aplicado a un volumen binario sintético.

Una vez obtenida la matriz de adyacencia se procedió a la extracción de parámetros. La matriz de adyacencia en este caso contiene información del peso de las aristas del grafo, el cual se corresponde con el largo en número de vóxeles de la conexión entre dos nodos.

5.3. Parámetros relevados

Los parámetros relevados se dividen en dos categorías, parámetros morfológicos y parámetros de conectividad, éstos se presentan en las próximas secciones.

5.3.1. Parámetros morfológicos

Se hará referencia a los parámetros morfológicos, entendiendo los mismos como aquellos que permiten realizar un análisis de la forma de cada estructura individual en un volumen. A continuación se detallan los parámetros morfológicos que se utilizarán en este trabajo.

Volumen

Refiere al volumen (Vol) de cada estructura individual presente en el stack. Se considera como estructura individual a aquella identificada con una etiqueta en particular. El volumen se obtiene como el número de vóxeles que poseen dicha etiqueta [62].

Perímetro/Superficie

Este parámetro se corresponde con el perímetro de cada estructura individual si se utilizan imágenes 2D y con la superficie (S) de dicha estructura si se utilizan stacks 3D. Ésta también se calcula en número de vóxeles [62].

Perímetro/Superficie Esferoidal

Si se utilizan imágenes 2D este parámetro retorna el perímetro y para stacks 3D devuelve la superficie. El perímetro esferoidal se corresponde con el perímetro del círculo cuya área es igual al área de la estructura a analizar. La superficie esferoidal (S_e) se corresponde con la superficie de la esfera cuyo volumen es igual al volumen de la estructura a analizar. Su valor se obtiene en número de pixeles [62].

Radio Esferoidal

Similar al parámetro anterior, se define el radio esferoidal (r_e). Es el radio correspondiente a la esfera de volumen igual al de la estructura a analizar [62].

Diámetro de Feret

Si se considera una esfera que encierre la totalidad de la estructura a analizar, el diámetro de Feret (d_F) se define como el diámetro de dicha esfera [62].

Redondez/Esfericidad

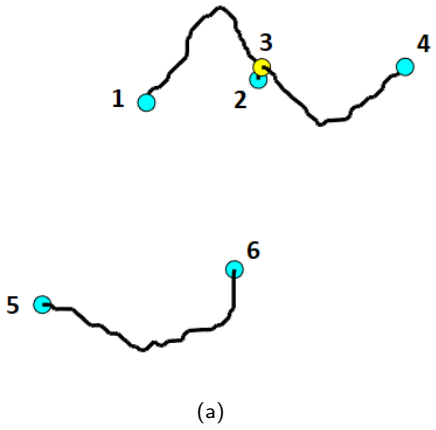
El parámetro de redondez (R) da una idea de qué tan redondeada (2D) o esférica (3D) es la estructura. Es un valor entre 0 y 1, siendo 1 para una esfera [62].

Elongación

Devuelve la elongación (E) de la estructura, computada como el cociente entre el momento principal más grande y el más pequeño. El valor es mayor o igual a 1 [62].

5.3.2. Parámetros de conectividad

Los parámetros de conectividad permiten obtener una descripción de cómo se conforma la red mitocondrial y cómo interactúan las mitocondrias entre sí. La teoría de grafos brinda herramientas útiles para realizar este análisis, por lo que si se considera la red mitocondrial como un grafo, se pueden obtener una gran variedad de características que la describan. Se utilizará el grafo de la Figura 5.3 como referencia para la explicación de los distintos parámetros de interés.



(a)

$$\mathrm{A} = \begin{pmatrix} 0 & 0 & 37 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 & 0 \\ 37 & 4 & 0 & 41 & 0 & 0 \\ 0 & 0 & 41 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 54 \\ 0 & 0 & 0 & 0 & 54 & 0 \end{pmatrix}$$

(b)

Figura 5.3: (a) Grafo 3D obtenido al aplicar *Skeletonize3D* seguido de *Skel2Graph3D* a un volumen sintético binario. Los nodos son de color azul si se conectan con un único nodo y son de color amarillo si se conectan con más de un nodo. El nodo 3 se conecta con los nodos 1, 2 y 4, pero cada uno de estos últimos se conecta únicamente con el nodo 3. Se utiliza un volumen en lugar de un stack para visualizar mejor los resultados. (b) Matriz de adyacencia ponderada por el largo de las aristas que unen dos nodos.

Número de nodos

Se determina el número de nodos totales del grafo, el cual queda dado por el tamaño de la matriz de adyacencia. Ésta será de dimensión $N \times N$, siendo N el número de nodos. En el ejemplo de la Figura 5.3 se tiene $N = 6$.

Densidad de *links*

La densidad de *links* o aristas se corresponde con el número efectivo de conexiones sobre el total de posibles conexiones [14]. Sea N el número de nodos presentes en el grafo, se tiene $a_{ij} = 1$ si existe una arista entre los nodos i y j y $a_{ij} = 0$ en otro caso. Se define la densidad de *links* l_d según la siguiente ecuación

$$l_d = \frac{1}{N(N-1)} \sum_{\substack{i,j \\ i \neq j}} a_{ij}. \tag{5.1}$$

Este parámetro toma un valor entre 0 y 1, siendo nulo cuando no hay conexiones entre nodos ($a_{ij} = 0 \; \forall i, j, i \neq j$) y máximo cuando todos los nodos están conectados entre sí ($a_{ij} = 1 \; \forall i, j, i \neq j$). Para el grafo de la Figura 5.3 el planteo queda como en la Ecuación (5.2), resultando en $l_d = 0{,}267$.

$$l_d = \frac{1}{30} \left(a_{13} + a_{23} + a_{31} + a_{32} + a_{34} + a_{43} + a_{56} + a_{65} \right) \tag{5.2}$$

Grado máximo

Se define como grado máximo al número de conexiones directas del nodo más conectado ponderado por los pesos de cada arista [14]. Siendo r_i el número de co-

nexiones directas del nodo i, contando cada conexión con su peso correspondiente, se tiene que el grado máximo m_d queda definido como sigue

$$m_d = max\left\{\sum_i r_i\right\} \tag{5.3}$$

El grado máximo se obtiene sumando todas las filas (o columnas) de la matriz de adyacencia ponderada y tomando el máximo. En el ejemplo de la Figura 5.3 el nodo más conectado es n_3, el cual se conecta con otros tres nodos n_1, n_2 y n_4. Realizando la suma de sus filas, se obtiene que el valor más alto es el de la tercer fila, resultando en $m_d = 82$.

Eficiencia

Este parámetro refiere a la eficiencia de la información transmitida entre nodos [14]. Para comprender este término, es necesario definir previamente algunos conceptos.

Sea G el grafo que representa una red, G ponderado e incluso no conectado[1]. Se define la eficiencia ϵ_{ij} entre dos nodos n_i y n_j como el inverso del número de *links* entre ellos (véase ecuación (5.4)). El camino más corto d_{ij} entre dos nodos genéricos n_i y n_j se define como la suma más pequeña de las distancias entre todos los posibles caminos en G entre n_i y n_j, donde d_{ij} es medido en cantidad de links. Cuando no existe un camino entre dos nodos, $d_{ij} = +\infty$ y consecuentemente $\epsilon_{ij} = 0$ [63].

La eficiencia E del grafo G se define entonces como el promedio de la eficiencia de cada par de nodos ϵ_{ij} [63]. El cálculo de E queda dado por la ecuación (5.5), de la cual se desprende que el valor de la eficiencia se encuentra en el rango $[0, 1]$. Si todos los nodos estuvieran conectados entre sí, la sumatoria resultaría en $N \times (N-1)$, lo que resulta en $E = 1$.

$$\epsilon_{ij} = \frac{1}{d_{ij}} \; \forall i, j \tag{5.4}$$

$$E = \frac{1}{N(N-1)} \sum_{\substack{i,j \\ i \neq j}} \epsilon_{ij} \tag{5.5}$$

A modo de ejemplo, para fijar el concepto, se explica cómo realizar el cálculo de la eficiencia para el grafo de la Figura 5.3. Partiendo de las ecuaciones (5.4) y (5.5), se deben determinar las distancias d_{ij} entre los pares de nodos. En este caso se observa que:

- La distancia entre los nodos 5 y 6 es 1 (unidos por 1 link) $\Rightarrow d_{56} = d_{65} = 1$. Análogamente, $d_{13} = d_{31} = d_{23} = d_{32} = d_{34} = d_{43} = 1$.

[1] Existen al menos dos nodos en G que no son conectados por ningún camino.

- La distancia entre los nodos 1 y 2 es 2 (unidos por 2 links, el que va del nodo 1 al 3 y el que va del nodo 3 al 2) $\Rightarrow d_{12} = d_{21} = 2$.

  Análogamente, $d_{14} = d_{41} = d_{24} = d_{42} = 2$.

- Los nodos que no están conectados no se consideran para la sumatoria. Esto implica $d_{ij} = +\infty$, por lo que el término ij en la sumatoria es nulo.

De lo anterior, se tiene que el cálculo de E queda como en la ecuación (5.6). Sustituyendo con los valores determinados en los puntos anteriores, se llega a $E = 0{,}367$.

$$E = \frac{1}{N(N-1)} \left[\frac{1}{d_{12}} + \frac{1}{d_{13}} + \frac{1}{d_{14}} + \frac{1}{d_{21}} + \frac{1}{d_{23}} + \frac{1}{d_{24}} + \frac{1}{d_{31}} + \frac{1}{d_{32}} + \frac{1}{d_{34}} + \frac{1}{d_{41}} + \frac{1}{d_{42}} + \frac{1}{d_{43}} + \frac{1}{d_{56}} + \frac{1}{d_{65}} \right] \tag{5.6}$$

Modularidad

La modularidad es una medida de la estructura de la red que cuantifica la división de la misma en subredes o comunidades [14]. Se realizará la partición en comunidades utilizando el algoritmo de Louvain [64], el cual considera la división de forma que se maximice la modularidad. El cálculo de este parámetro se realiza mediante la ecuación (5.7) [64], donde

- A_{ij} representa el peso de la arista entre el nodo i y el nodo j.
- $k_i = \sum_j A_{ij}$ es la suma de los pesos de las aristas conectadas al nodo i.
- c_i es la comunidad a la que pertenece el nodo i.
- $\delta(u, v) = \begin{cases} 1 \text{ si } u = v \\ 0 \text{ si } u \neq v \end{cases}$
- $m = \frac{1}{2} \sum_{ij} A_{ij}$

$$Q = \frac{1}{2m} \sum_{ij} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \delta(c_i, c_j) \tag{5.7}$$

El valor de Q se encuentra en el rango $[0, \frac{\frac{N}{2}-1}{\frac{N}{2}}]$, siendo N el número de nodos[2]. La modularidad aumenta con el número de subredes y es mínima cuando el grafo se conforma por una única comunidad.

Sea A la matriz de adyacencia que contiene los pesos de cada arista (véase Figura 5.3b), k_{ij} se corresponde con la suma de todos los elementos de la fila i de dicha matriz. Para el ejemplo de la Figura 5.3a, se tienen dos comunidades, una contiene los nodos n_1, n_2, n_3 y n_4 y la otra los nodos n_5 y n_6. El valor de m es la

[2]En el Apéndice F se presenta el cálculo para determinar las cotas inferior y superior de Q.

	1	**2**	**3**	**4**	**5**	**6**
1	(1,1)	(1,1)	(1,3)	(1,1)	(1,1)	(1,1)
2	(1,1)	(1,1)	(1,3)	(1,1)	(1,1)	(1,1)
3	(3,1)	(3,1)	(3,3)	(3,1)	(3,1)	(3,1)
4	(1,1)	(1,1)	(1,3)	(1,1)	(1,1)	(1,1)
5	(1,1)	(1,1)	(1,3)	(1,1)	(1,1)	(1,1)
6	(1,1)	(1,1)	(1,3)	(1,1)	(1,1)	(1,1)

Tabla 5.1: Parejas (grado, grado) para cada par de nodos del grafo de la Figura 5.3.

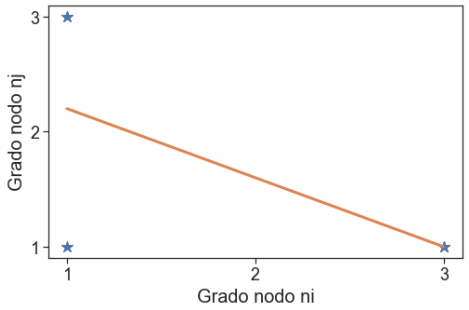


Figura 5.4: Parejas (grado, grado) para cada par de nodos del grafo de la Figura 5.3, junto con la recta que mejor ajusta dichos puntos. La pendiente de esta recta determina el valor de la asortatividad, siendo para este caso $ass = -0{,}6$.

suma de los pesos de todas las aristas, contando las conexiones entre n_i y n_j una única vez. De esta manera, para el grafo del ejemplo se tiene que $Q = 0{,}375$.

$$A = \begin{pmatrix} 0 & 0 & 37 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 & 0 \\ 37 & 4 & 0 & 41 & 0 & 0 \\ 0 & 0 & 41 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 54 \\ 0 & 0 & 0 & 0 & 54 & 0 \end{pmatrix}, \qquad k = \begin{bmatrix} 37 \\ 4 \\ 82 \\ 41 \\ 54 \\ 54 \end{bmatrix}, \qquad m = 136$$

Asortatividad

La asortatividad refiere al coeficiente de correlación de Pearson entre los grados de ambos nodos de un enlace. Los valores positivos indican que los nodos altamente conectados tienden a conectarse con otros nodos de muchas conexiones, mientras que valores negativos indican que los nodos altamente conectados prefieren vincularse con los nodos de la periferia [14, 65].

Para el cálculo de este parámetro, se generan pares grado-grado de los nodos unidos por cada arista del grafo. Tomando el ejemplo de la Figura 5.3, el par grado-grado dado por los nodos 5 y 6 sería $1 - 1$, mientras que el par dado por los nodos 2 y 3 sería $1 - 3$. Los pares grado-grado se cuentan dos veces $((grad\_n_i, grad\_n_j)$ y $(grad\_n_j, grad\_n_i))$.

Tomando las parejas de grados como coordenadas (x, y), se tiene que la asortatividad se corresponde con la pendiente de la recta que mejor ajusta los dichos puntos. En la Tabla 5.1 se presentan todas las parejas del ejemplo y en la Figura 5.4 se muestran dichos puntos en la gráfica, junto con la recta que mejor los ajusta. A partir de la pendiente de dicha recta, se obtiene $ass = -0{,}6$.

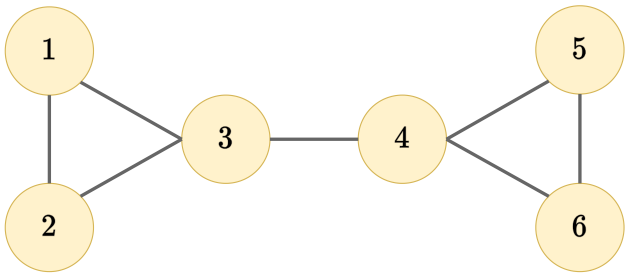


Figura 5.5: Grafo de ejemplo para el cálculo de la transitividad $T = \frac{3 \times \text{num triangulos}}{\text{num ternas conectadas}}$. El coeficiente de transitividad para este grafo es $T = 0{,}6$.

Transitividad

La transitividad hace referencia a la densidad de triángulos, esto es, mide la probabilidad de encontrar ternas de nodos completamente conectadas entre sí [14]. La forma de cuantificar esta información es realizando el cociente entre el "número de triángulos" del grafo y el número de ternas de nodos conectadas del grafo [66].

$$T = \frac{3 \times \textit{Número de Triángulos}}{\textit{Número de ternas conectadas}} \tag{5.8}$$

Una observación importante es que el 3 que aparece multiplicando en la ecuación está debido a que un triángulo ABC, implica otros dos triángulos BCA y CAB.

Véase un ejemplo de esto en la Figura 5.5, donde las ternas 123 y 456 forman triángulos. De esta manera, el número de triángulos es igual a 6 (123, 231, 312, 456, 564 y 645). Por otro lado, el número de ternas conectadas es 10, ya que esta compuesto por las 6 ternas ya nombradas más las ternas 134, 234, 345 y 346. Por lo tanto, se obtiene $T = \frac{6}{10} = 0{,}6$. Nótese que en caso de no existir el enlace entre 3 y 4, se obtendría $T = 1$.

Diámetro

El diámetro de un grafo es la mayor distancia posible entre dos nodos, entendiendo la distancia como el menor número de aristas entre ellos [14]. Este concepto es válido para grafos completos, por lo que no es posible aplicarlo en un ejemplo como el de la Figura 5.3. En el caso de la Figura 5.5, el diámetro queda definido por las distancias de los nodos 1 o 2 a los nodos 5 o 6, resultando en $diam = 3$.

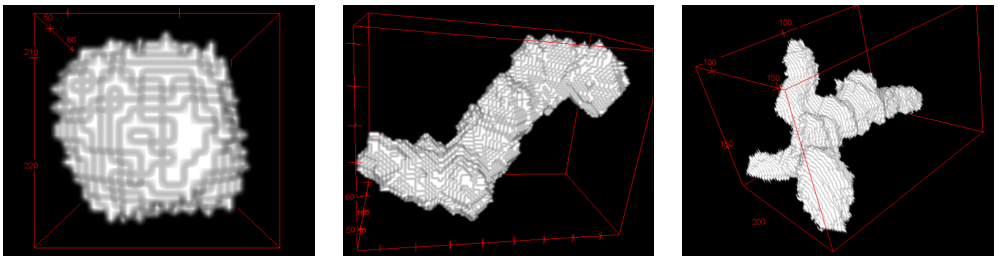
Camino medio más corto (ASPL)

Este parámetro devuelve el promedio de los caminos más cortos entre todas las posibles parejas de nodos [14]. Queda definido por la ecuación (5.9) donde V es el set de nodos de G, $d_{min}(n_i, n_j)$ el camino más corto entre los nodos n_i y n_j y N el número total de nodos [66].

$$ASPL = \frac{1}{N(N-1)} \sum_{n_i, n_j \in V} d_{min}(n_i, n_j) \tag{5.9}$$

Estructura	Parámetros
Glóbulo	$r_x = 6$, $r_y = 6$, $r_z = 6$
Túbulo largo	$r_x^{min} = 2$, $r_x^{max} = 7$, $r_y^{min} = 2$, $r_y^{max} = 7$, $l_z = 75$
Túbulo ramificado	$n_{ramif} = 3$

Tabla 5.2: Parámetros utilizados para la generación de un glóbulo y un túbulo largo, mediante el generador de volúmenes sintéticos. Estos volúmenes se utilizaron para facilitar la comprensión de los parámetros morfológicos y de conectividad extraídos y verificar el correcto funcionamiento del método.



(a) Glóbulo. (b) Túbulo largo. (c) Túbulo ramificado.

Figura 5.6: Volúmenes individuales generados sintéticamente con los parámetros de la Tabla 5.2. (a) Glóbulo utilizado para la comprensión de parámetros morfológicos. (b) Túbulo largo utilizado para la comprensión de parámetros morfológicos y de conectividad. (c) Túbulo con 3 ramificaciones, utilizado para la comprensión de parámetros de conectividad.

Cuanto mayor es el valor de $ASPL$ menos conectado está el grafo, es decir la distancia mínima promedio entre dos nodos es mayor. Notar que si todos los nodos están conectados entre sí, $d_{min}(n_i, n_j) = 1 \ \forall i, j$, por lo que se tiene $ASPL = 1$.

Este cálculo no es válido para el ejemplo de la Figura 5.3 dado que se tienen dos subredes aisladas; no hay un camino de aristas que permita llegar, por ejemplo, del nodo 5 al nodo 3. Por lo tanto $ASPL$ queda indefinido. Por otro lado, en el ejemplo de la Figura 5.5 sí existen caminos entre todos los nodos y se obtiene $ASPL = 1{,}8$.

5.4. Experimentos realizados

En primer lugar se realizó un experimento cuyo fin fue contribuir a la comprensión de las características presentadas en la sección 5.3. Para ello se trabajó con volúmenes binarios generados sintéticamente que contuvieran una única estructura. En particular, se utilizó un glóbulo y un túbulo largo para analizar los parámetros morfológicos y el túbulo y una estructura ramificada para analizar los parámetros de conectividad. Los volúmenes sintéticos fueron generados según la Tabla 5.2 y los mismos se muestran en la Figura 5.6. Los resultados se presentan en la Tabla 5.3.

Para analizar los parámetros de conectividad fue necesario realizar otro ex-

	Glóbulo	Túbulo largo
Volumen	4275	36552
Superficie	1365.5	8725.9
Superficie Esferoidal	1273.8	5326.3
Radio Esferoidal	10.1	20.6
Diámetro de Feret	24.3	87.5
Redondez	0.93	0.61
Elongación	1.1	2.8

Tabla 5.3: Parámetros morfológicos para un glóbulo y un túbulo largo generados con los parámetros de la Tabla 5.2. Se utlizan estos resultados para contribuir a la comprensión de los parámetros morfológicos extraídos.

perimento. La función *Skel2Graph3D* permite fijar la longitud mínima que puede tener una ramificación. Al convertir el esqueleto en un grafo, utiliza este parámetro para definir dónde hay nuevos nodos. Debido a que las estructuras con las que se trabaja tienen formas irregulares, es importante fijar este parámetro correctamente. Para evidenciar la importancia de esto, se presenta en la Figura 5.7 cómo se ve el grafo del túbulo largo y del túbulo ramificado antes y luego de filtrar con *min_path_length = 10*. Este parámetro fue fijado luego de varias pruebas con estructuras aisladas, como el ejemplo presentado.

En las Figuras 5.8 y 5.9 se muestra el volumen original junto con el esqueleto y el grafo 3D para el túbulo largo y el túbulo ramificado, respectivamente. En la Tabla 5.4 se presentan los resultados obtenidos para las características de conectividad calculadas a partir de dichos grafos.

Finalmente, se procedió con el experimento principal de extracción de características para el cual se utilizó BD3. En la Figura 5.10 se presenta el diagrama de flujo correspondiente a esta etapa. Por un lado, se realizó el análisis sobre los stacks generados sintéticamente para obtener un *ground-truth*. Por otro lado, luego de aplicar la generación de ruido de la GAN y procesar estos stacks mediante las etapas anteriores del pipeline - deconvolución y segmentación - se realizó el análisis nuevamente.

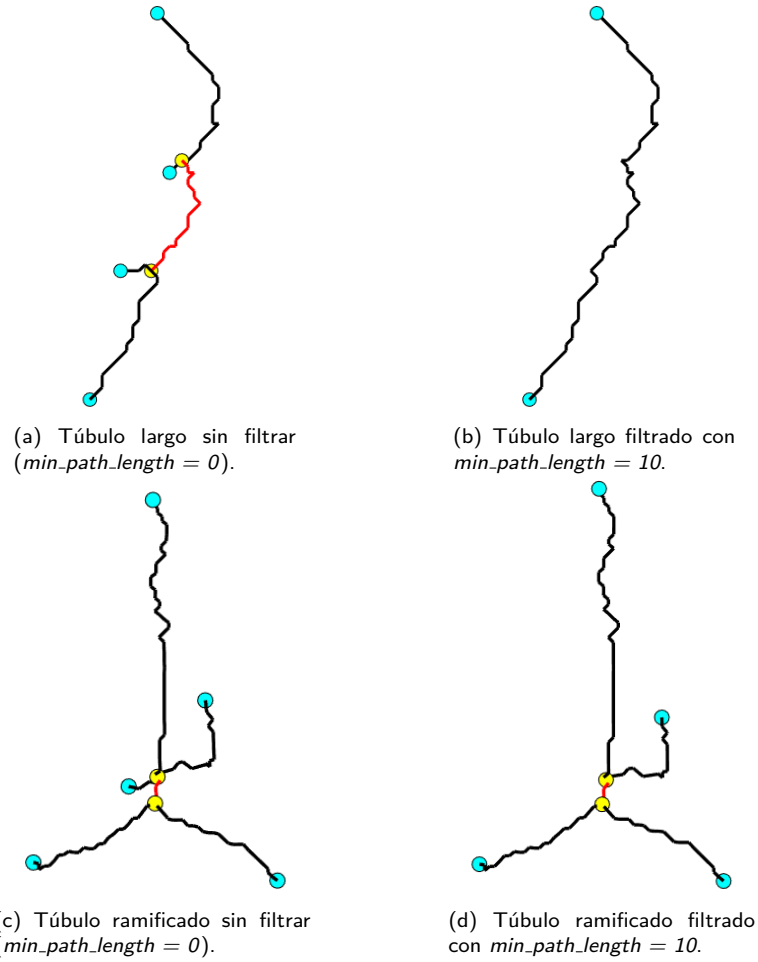
Considerando los resultados obtenidos en los Capítulos 3 y 4, se utilizó como método de deconvolución RL de 50 iteraciones y como métodos de segmentación K-means con $K = 2$ y U-Net. En la Tabla 5.5 se presenta la media y la desviación estándar del error relativo para cada parámetro, utilizando ambos métodos de segmentación. Notar que los parámetros morfológicos brindan información sobre cada estructura individual en un volumen, por lo que para resumir la información de un stack en un único número se decidió utilizar el promedio.

	Túbulo largo	Túbulo ramificado
Num nodes	2	6
Link dens	1	0.33
Max deg	76	96
Efficiency	1	0.62
Modularity	0	0.43
Assortativity	-	-0.67
Transitivity	0	0
Diameter	1	3
ASPL	1	1.93

Tabla 5.4: Parámetros de conectividad para un túbulo largo y un túbulo ramificado generados sintéticamente. Se utlizan estos resultados para contribuir a la comprensión de los parámetros de conectividad extraídos.

	K-means		**U-Net**	
Parámetro	$\mathbf{e_r^{mean}}$	$\mathbf{e_r^{std}}$	$\mathbf{e_r^{mean}}$	$\mathbf{e_r^{std}}$
Volumen (media)	25 %	24 %	52 %	11 %
Superficie (media)	20 %	27 %	41 %	11 %
Superficie esferoidal (media)	24 %	15 %	38 %	10 %
Radio esferoidal (media)	19 %	15 %	20 %	6 %
Diámetro de Feret (media)	32 %	54 %	23 %	9 %
Redondez (media)	27 %	26 %	3 %	3 %
Elongación (media)	23 %	14 %	9 %	8 %
Núm nodos	17 %	14 %	37 %	19 %
Densidad de *links*	20 %	22 %	37 %	36 %
Grado máx	18 %	19 %	23 %	15 %
Eficiencia	17 %	13 %	35 %	17 %
Modularidad	18 %	12 %	4 %	4 %
Asortatividad	18 %	16 %	230 %	430 %

Tabla 5.5: Media y desviación estándar del error relativo entre los parámetros extraídos al procesar BD3 por el pipeline y los parámetros extraídos del *ground-truth*. Resultados obtenidos para la segmentación con K-means con $K = 2$ y con U-Net/ResNet.



(a) Túbulo largo sin filtrar (*min_path_length = 0*).

(b) Túbulo largo filtrado con *min_path_length = 10*.

(c) Túbulo ramificado sin filtrar (*min_path_length = 0*).

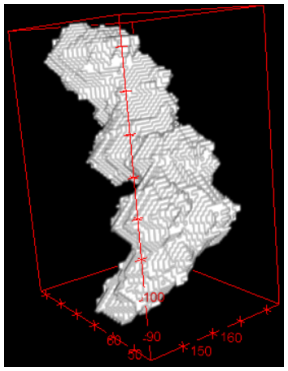
(d) Túbulo ramificado filtrado con *min_path_length = 10*.

Figura 5.7: Comparación entre la generación del grafo 3D de un túbulo largo y un túbulo ramificado sin filtrar y filtrando las ramificaciones de largo menor a 10.

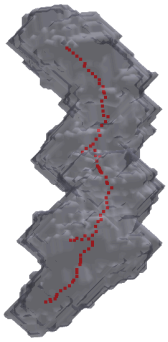
5.5. Análisis de resultados

En la Tabla 5.3 se presentan los resultados obtenidos al extraer parámetros morfológicos sobre el glóbulo y el túbulo largo. El primer aspecto a destacar es que el volumen y la superficie de la estructura globular son más elevados que lo esperado, pues a partir del radio utilizado se puede determinar el volumen $Vol \approx 905$ y la superficie $S \approx 452$. La explicación de esto radica en pequeñas deformaciones causadas en la rotación de las estructuras. Sin embargo, esto no supone un problema para este trabajo pues al aplicarse el mismo suavizado sobre todas las estructuras, se mantienen las características distintivas de cada estructura. Esto se puede corroborar comparando dichos parámetros con el volumen y la superficie del túbulo largo.

La estructura globular permite analizar el radio y superficie esferoidal y el diámetro de Feret. A partir de sus definiciones, es esperable que se verifique $S_e = S$ y $r_e = \frac{d_F}{2} = r$, siendo r el radio del glóbulo. Esto se puede verificar con los



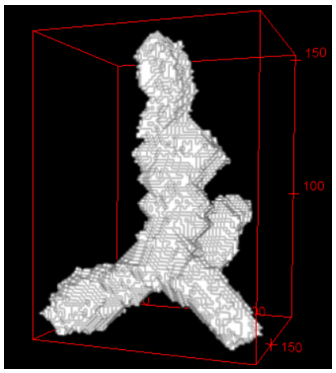
(a) Volumen sintético binario.



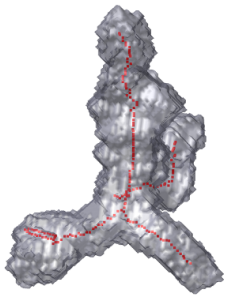
(b) Esqueletización 3D del volumen sintético.

(c) Grafo 3D obtenido a partir de la esqueletización 3D, con *min_path_length* = *10*.

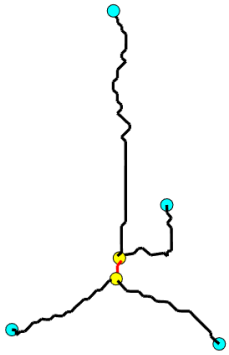
Figura 5.8: Preprocesamiento de un túbulo largo sintético utilizando las funciones *Skeletonize3D* y *Skel2Graph3D* con *min_path_dist* = *10*. A partir del grafo 3D se obtuvieron los parámetros de conectividad de la Tabla 5.4.



(a) Volumen sintético binario.



(b) Esqueletización 3D del volumen sintético.



(c) Grafo 3D obtenido a partir de la esqueletización 3D, con *min_path_length* = *10*.

Figura 5.9: Preprocesamiento de un túbulo ramificado sintético utilizando las funciones *Skeletonize3D* y *Skel2Graph3D* con *min_path_dist* = *10*. A partir del grafo 3D se obtuvieron los parámetros de conectividad de la Tabla 5.4.

resultados de la Tabla 5.3; si bien existen pequeñas diferencias con lo esperado, las mismas se atribuyen al efecto de suavizado ya mencionado dado que éste introduce una pequeña deformación en el glóbulo, provocando que deje de ser, en este caso, una esfera perfecta.

En cuanto a los parámetros redondez y elongación, se verifica que el glóbulo tiene un valor de R cercano al máximo y un valor de E cercano al mínimo. El túbulo en comparación con el glóbulo tiene un valor de redondez menor y elongación mayor.

Para los parámetros de conectividad, se debe resaltar la importancia del filtrado según el tamaño de las ramas. Inicialmente al extraer parámetros se obtuvo que el número de nodos para el túbulo largo y el túbulo ramificado era 6 y 7 respec-

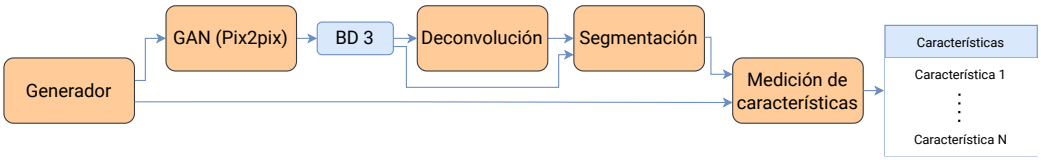


Figura 5.10: Diagrama de flujo para la etapa de extracción de características.

tivamente (véase Figuras 5.7a y 5.7c). Luego del filtrado el túbulo largo contiene dos nodos y una sola arista que los une, lo cual se corresponde con la estructura esperada para un túbulo. Por lo tanto, se observa que este filtrado es esencial para lograr diferenciar un túbulo simple de un túbulo ramificado. Otro aspecto importante a resaltar es que el filtrado no elimina aristas pequeñas que formen parte de la base de la estructura. Esto último se logra verificar en la Figura 5.9c, donde la arista en rojo que une los dos nodos en color amarillo no se elimina aunque su tamaño sea menor que el utilizado para filtrar.

A partir de la definición de la densidad de links se deduce que la misma da una idea de qué tan cerca está el grafo de ser un grafo completo, entendiendo por grafo completo aquél en el que cada vértice está unido a todos los demás. Se tiene entonces que para el túbulo largo la densidad de links debe ser máxima ($l_d = 1$), mientras que para la estructura ramificada se acercará más al mínimo ($l_d = 0$). En la Tabla 5.4 se observa que los valores obtenidos para este parámetro se corresponden con lo esperado.

Si bien la eficiencia también es máxima en el caso de tener un grafo completo, la información que brinda es distinta a la densidad de *links*. Para el cálculo de la eficiencia importa el número de aristas entre dos nodos, mientras que en la densidad de *links* solamente importa si existe una arista que los una directamente o no. Por este motivo, es razonable que el valor de eficiencia obtenido en la Tabla 5.4 para la estructura ramificada sea mayor que la densidad de *links*.

El grado máximo tiene en cuenta el número de conexiones ponderadas por el largo de las aristas, por lo que este parámetro por sí mismo no brindará información sobre la clase a la que pertenece la estructura.

Puesto que el rango de variación del parámetro modularidad depende del número de nodos, se calculó el rango específico para los grafos del túbulo largo y del túbulo ramificado obteniendo $Q_{long} = 0$ y $Q_{branched} \in \left[0, \frac{2}{3}\right]$. Los resultado presentados en la Tabla 5.4 confirman que la modularidad es mínima cuando el grafo se conforma por una única unidad (túbulo largo) y aumenta con el número de comunidades (túbulo ramificado).

El valor de asortatividad brinda información sobre las preferencias de conexión de los nodos altamente conectados. Para determinar este valor es necesario contar con dos puntos distintos en la gráfica de la Figura 5.4, por lo que en el túbulo largo no se puede determinar este parámetro. Observando la Figura 5.7d se observa que en la estructura ramificada del ejemplo cada nodo centro (amarillo) tiene una conexión con otro nodo centro y con dos nodos de la periferia. Por lo tanto, el valor de asortatividad para este ejemplo debe ser negativo, lo cual se comprueba en lo presentado en la Tabla 5.4.

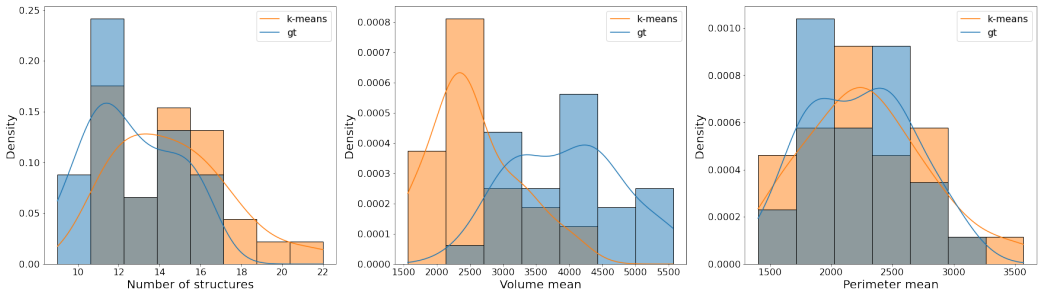


Figura 5.11: Histograma del número de estructuras, volumen promedio y superficie promedio. En azul se muestran los parámetros extraídos a partir del *ground-truth* y en naranja los parámetros extraídos luego de pasar por el pipeline (deconvolución con RL 50 iteraciones y segmentación con *K-means*).

Tanto el túbulo largo como el ramificado tienen transitividad nula. Esto es fácil de ver puesto que en ningún caso se tienen ternas de nodos completamente conectadas entre sí. Observando las Figuras 5.7b y 5.7d también es fácil verificar que el diámetro del túbulo largo es 1, mientras que el de la estructura ramificada es 3.

Observando los resultados de la Tabla 5.4 se logra verificar que un valor más alto de $ASPL$ implica un grafo menos conectado. Este resultado es directo de que, como ya fue mencionado, $ASPL = 1$ para un grafo completo. Este parámetro da una idea de la conectividad del grafo obtenido en las estructuras

Una observación importante sobre los parámetros de conectividad es que para todas las estructuras tubulares simples (túbulos cortos o túbulos largos) estos parámetros son fijos, a excepción del grado máximo que indica el largo del túbulo. Para los túbulos ramificados esto no se cumple pues basta con variar el número de ramificaciones para que los resultados obtenidos difieran.

Debido a que BD3 cuenta con un número elevado de stacks (30), para analizar los resultados se realizaron histogramas comparando la información de cada parámetro extraído luego de la etapa de segmentación con los parámetros extraídos a partir del *ground-truth*.

Análisis de la medición de características para BD3 con segmentación K-means

En primer lugar se analizaron el número de estructuras por stack, el volumen promedio y la superficie promedio. En la Figura 5.11 se presentan los histogramas correspondientes a estos tres parámetros, en los que se logra detectar que luego del procesamiento se tiende a tener un número mayor de estructuras y el volumen tiende a ser menor. Para analizar en mayor profundidad este fenómeno, se estudiaron tres subgrupos G_1, G_2 y G_3, definidos según el tamaño de las estructuras como se muestra en la Tabla 5.6.

En la Figura 5.12 se muestra el número de estructuras para cada subgrupo G_1, G_2 y G_3. En cada stack de BD3 se determinó el volumen de sus estructuras y se obtuvo el número de elementos que pertenecen a cada subgrupo por stack. Esta información es la que se presenta en la Figura 5.12, comparando los resultados

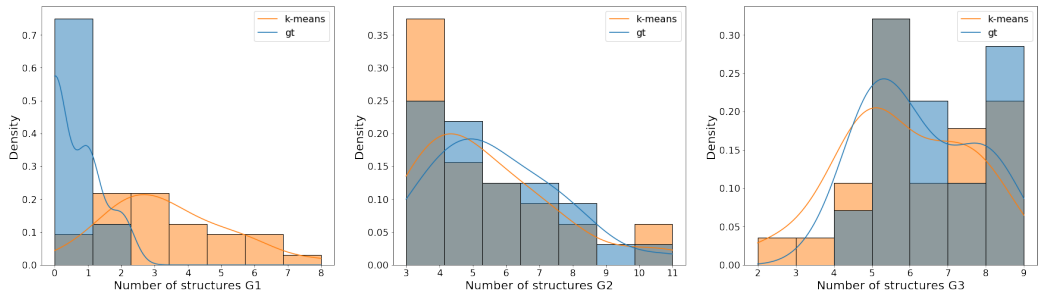


Figura 5.12: Histograma del número de estructuras para cada subgrupo G_1, G_2 y G_3 definidos en la Tabla 5.6. En azul se muestran los parámetros extraídos a partir del *ground-truth* y en naranja los parámetros extraídos luego de pasar por el pipeline (deconvolución con RL 50 iteraciones y segmentación con *K-means*).

post segmentación con el *ground-truth*. Notar que inicialmente en G_1 el número de elementos es prácticamente nulo, mientras que luego de pasar por el pipeline la cantidad de estructuras pequeñas crece. Sin embargo, la distribución en los grupos G_2 y G_3 se mantiene similar. Al pasar por las etapas de deconvolución y segmentación las estructuras se degradan y sufren ciertas deformaciones. En particular, pueden desprenderse pequeñas regiones de una estructura de mayor tamaño, lo cual puede generar la aparición de nuevos elementos más pequeños.

A partir de las definiciones de radio y superficie esferoidal, se tiene que estos parámetros están directamente relacionados con el volumen. Por lo tanto los datos seguirán una distribución similar, lo cual se puede comprobar en la Figura 5.13. El diámetro de Feret, por otra parte, mantiene su distribución muy similar al *ground-truth*. Esto no sorprende dado que, por más que las imágenes segmentadas tienen diferencias respecto a las originales, una variación alta en el diámetro de Feret implicaría un cambio en las estructuras de tal forma que la esfera de radio mínimo que la contiene en su totalidad varíe notoriamente.

En la Figura 5.14 se presentan los histogramas correspondientes a la redondez y elongación. Se observa que la elongación mantiene su distribución al pasar por el pipeline, por motivos similares a los del diámetro de Feret, mientras que la redondez se ve completamente afectada. Esto último se relaciona con la degradación y deformación ya mencionada que producen los procesos de deconvolución y segmentación.

Grupo	Volumen
G_1	$Vol \leq 500$
G_2	$500 < Vol < 2000$
G_3	$Vol \geq 2000$

Tabla 5.6: Subgrupos definidos según el tamaño de las estructuras. Estos grupos se utilizaron para analizar las diferencias detectadas en el volumen y el número de estructuras comparando los valores a la salida del pipeline con el *ground-truth*.

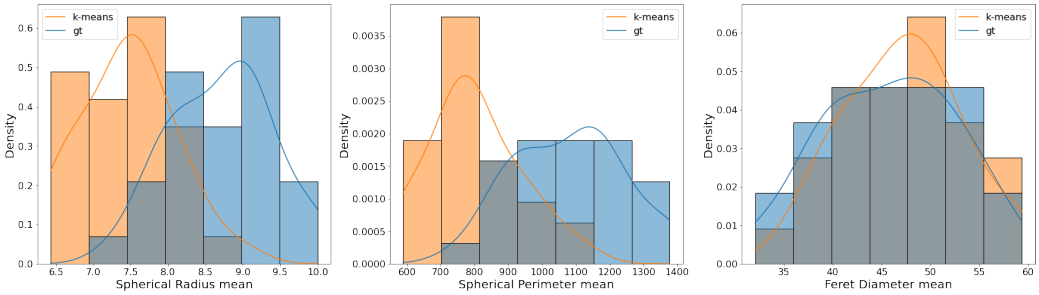


Figura 5.13: Histograma del radio esferoidal promedio, superficie esferoidal promedio y diámetro de Feret promedio. En azul se muestran los parámetros extraídos a partir del *ground-truth* y en naranja los parámetros extraídos luego de pasar por el pipeline (deconvolución con RL 50 iteraciones y segmentación con *K-means*).

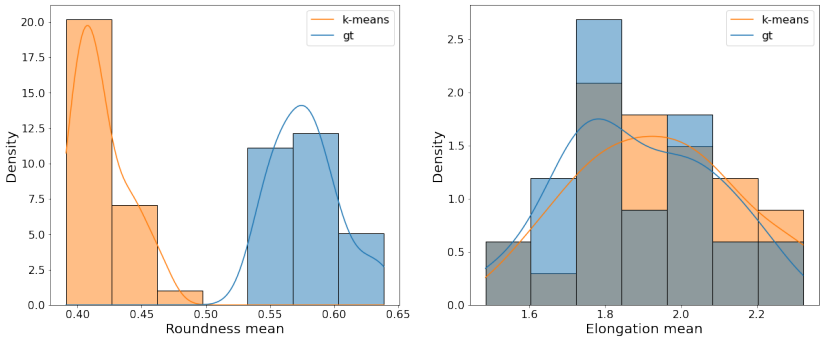


Figura 5.14: Histograma de la redondez y la elongación promedio. En azul se muestran los parámetros extraídos a (partir del *ground-truth* y en naranja los parámetros extraídos luego de pasar por el pipeline segmentación con *K-means*).

En cuanto a los parámetros de conectividad, en la Figura 5.15 se observa que el número de nodos mantiene una distribución similar, mientras que el grado máximo tiene cierta tendencia a aumentar. La densidad de *links* también se mantiene similar entre el *ground-truth* y la salida de la segmentación. Notar que los valores de la densidad de *links* son muy pequeños. Esto se debe a que los volúmenes sintéticos utilizados en BD3 están conformados principalmente por varias estructuras aisladas.

Los valores de eficiencia obtenidos tanto en el *ground-truth* como en la etapa post segmentación son bajos. Se recuerda que la eficiencia puede tomar valores entre 0 y 1, siendo máxima cuando todos los nodos están conectados entre sí. Estos resultados son consistentes con lo observado en la densidad de *links*.

En cuanto a la asortatividad se observa que en el *ground-truth* los valores son principalmente negativos. Esto implica que los nodos altamente conectados tienden a vincularse con nodos de la periferia. Luego de pasar por el pipeline, este valor aumenta, significando que hay nodos que tienden a conectarse con otros nodos altamente conectados. Esto podría vincularse con el aumento en el grado máximo.

La modularidad cuantifica la división en comunidades, siendo mínima ($Q = 0$) cuando el grafo se conforma por una única comunidad. En los resultados obtenidos

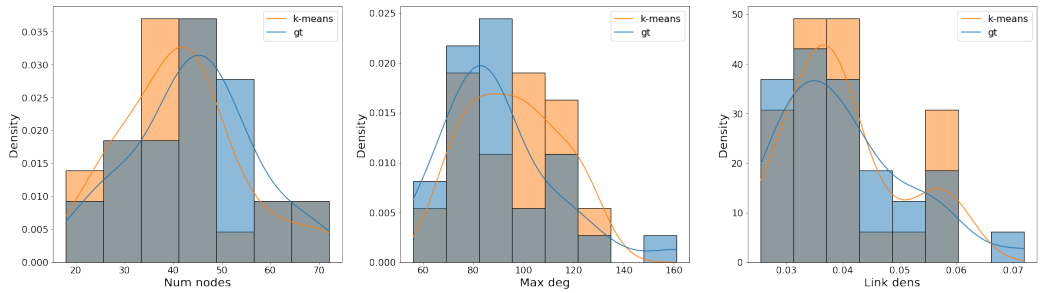


Figura 5.15: Histograma del número de nodos, el grado máximo y la densidad de *links*. En azul se muestran los parámetros extraídos a partir del *ground-truth* y en naranja los parámetros extraídos luego de pasar por el pipeline (deconvolución con RL 50 iteraciones y segmentación con *K-means*).

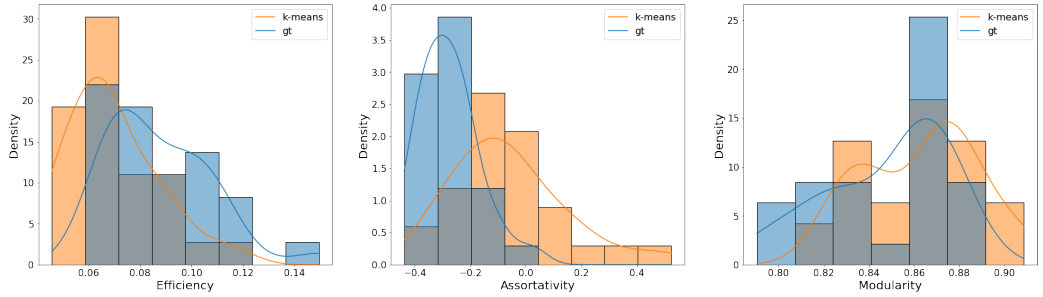


Figura 5.16: Histograma de la eficiencia, la asortatividad y la modularidad. En azul se muestran los parámetros extraídos a partir del *ground-truth* y en naranja los parámetros extraídos luego de pasar por el pipeline (deconvolución con RL 50 iteraciones y segmentación con *K-means*).

se observa que el valor de Q es elevado, tanto para el *ground-truth* como para los resultados post segmentación. Esto es razonable puesto que los stacks sintéticos tienen muchas estructuras aisladas.

Puesto que los stacks utilizados contaban con estructuras aisladas, no se lograron determinar el diámetro y el $ASPL$. La transitividad no se presenta en la Tabla 5.5 ni en las gráficas, puesto que su valor en la gran mayoría de los stacks quedaba indeterminado. Esto se debe a que de las cinco estructuras generadas sintéticamente - glóbulos, *lumps*, túbulos cortos, largos y ramificados - las únicas que deberían generar ternas conectadas son las ramificadas y los *lumps*. Si estas estructuras no están presentes, el valor de T en el *ground-truth* será nulo y el error relativo quedará indeterminado.

A modo de resumen, se tiene que al extraer parámetros sobre los stacks luego del procesamiento realizado en el pipeline, los parámetros morfológicos que podrían brindar información más certera sobre las estructuras originales son el diámetro de Feret y la elongación. De los parámetros de conectividad, se tiene que los que logran mantener su distribución similar al pasar por el pipeline son el número de nodos, la densidad de *links* y la modularidad. Uno de los parámetros que se ve más alterado al pasar por todo el proceso es la redondez.

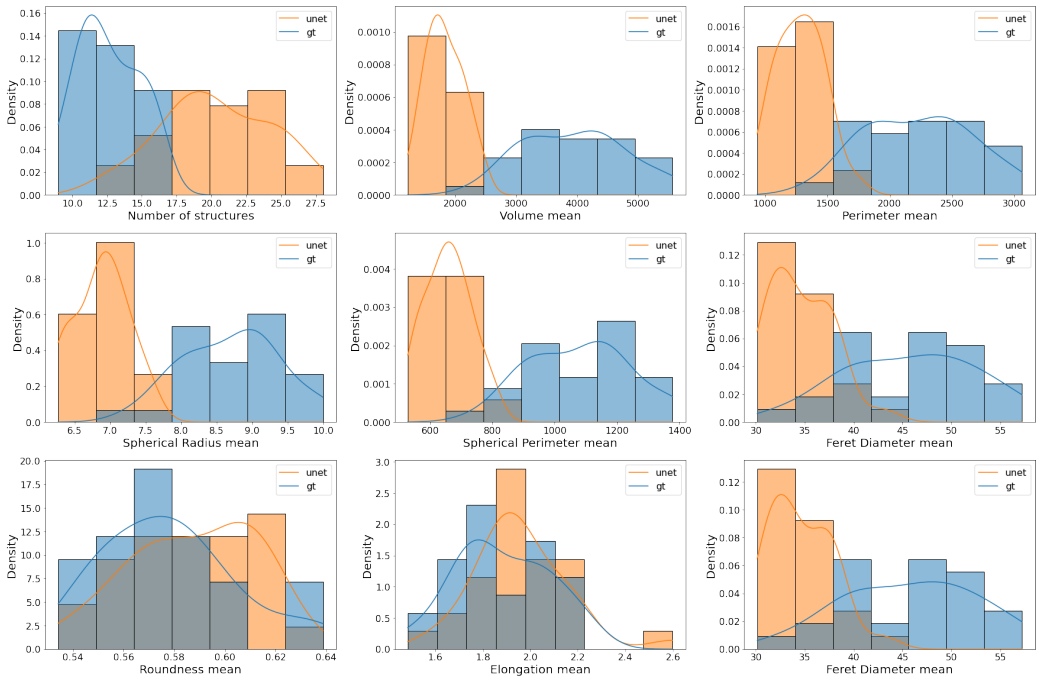


Figura 5.17: Histograma de la cantidad de estructuras y promedios de los volúmenes, de la superficie, del radio esférico, de la superficie esférica, del diámetro de Feret, de la redondez y la elongación. En azul se muestran los parámetros extraídos a partir del *ground-truth* y en naranja los parámetros extraídos luego de pasar por el pipeline (segmentación con U-Net/ResNet).

Análisis de la medición de características para BD3 con segmentación U-Net/ResNet

El análisis a realizar será muy similar al caso donde se utiliza la segmentación con K-means. En la Figura 5.17 se muestran los histograma de todos los parámetros morfológicos extraídos. El primer elemento a destacar es que hay una gran diferencia entre la distribución estadística de la segmentación obtenida comparada con su *ground truth*, teniendo más estructuras en la U-Net/ResNet. Esto se debe a lo poco restrictiva que es la red entrenada donde se logra mantener las regiones de interés más importantes a costa de tener ruido. Otro parámetro que cambia sustancialmente es el promedio de las superficies, teniendo grandes diferencias con el *ground truth*. Se considera que la razón es la misma, al ser una segmentación más permisiva, la región de interés a segmentar será más grande de lo que debería. De forma similar sucede para la superficie esférica y el diámetro de Feret.

La característica redondez tuvo el comportamiento inverso, logrando una distribución similar al *ground truth*, mientras que en el caso de K-means se obtuvieron resultados alejados del *ground truth*. Al ser un parámetro que no depende del tamaño de los cuerpos detectados sino de la forma y teniendo una segmentación permisiva se considera que existe cierta continuidad entre los distintos cortes del stack, en cambio si fuese más restrictiva, es razonable que existan cambios más abruptos lo cual se aleja de una estructura suave como una esfera.

En la Figura 5.18 se muestra la cantidad de estructuras para los distintos

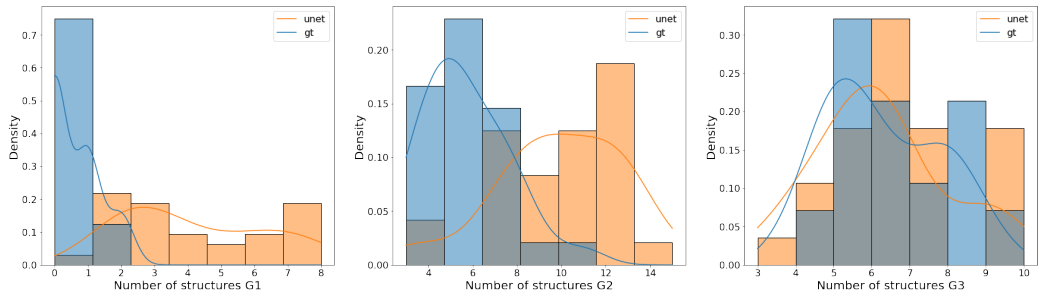


Figura 5.18: Histograma de la cantidad de estructuras de G_1, G_2 y G_3. En azul se muestran los parámetros extraídos a partir del *ground-truth* y en naranja los parámetros extraídos luego de pasar por el pipeline (segmentación con U-Net/ResNet).

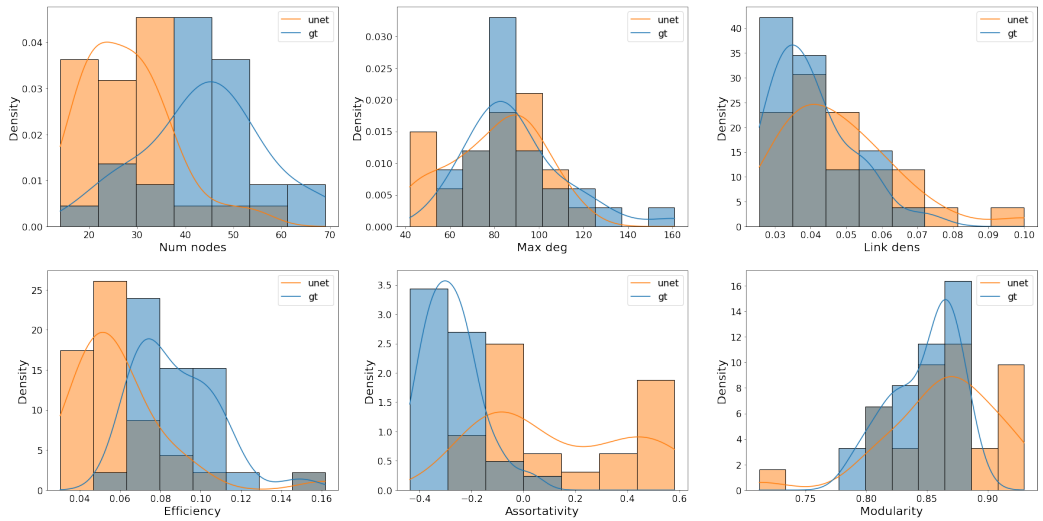


Figura 5.19: Histograma del número de nodos, el grado máximo, la densidad de *links*, eficiencia, asortatividad y la modularidad. En azul se muestran los parámetros extraídos a partir del *ground-truth* y en naranja los parámetros extraídos luego de pasar por el pipeline (segmentación con U-Net/ResNet).

grupos, en este caso, la diferencia con respecto a la otra segmentación, es el segundo grupo. Al segmentar más estructuras y alguna de ellas, más grandes, es razonable que la diferencia entre el *ground truth* y la segmentación sea grande en este caso.

Para los parámetros de conectividad se muestra la Figura 5.19 donde se considera que no existe demasiada variación porque se utiliza principalmente el esqueleto de la segmentación, el cual está más vinculado a la forma que al tamaño del cuerpo. La principal diferencia en este caso es que la cantidad de nodos tiende a disminuir en la segmentación, esto puede deberse a que al segmentar regiones más grandes, las estructuras tubulares tiendan a ser más esferoidales y así tener un esqueleto más compacto.

5.6. Conclusiones

En este Capítulo se mostró que fue posible implementar la medición de características morfológicas como fue realizado en el trabajo anterior [51]. Se pudo computar el error relativo de los distintos parámetros con los stacks sintéticos, teniendo en general un buen resultado siendo menos al 32 %. Por otro lado, se logró incorporar los parámetros de conectividad, fundamentales en las hipótesis esbozadas por el referente médico, brindando resultados aún mejores que los parámetros morfológicos.

La etapa de medición de características cierra el pipeline, disponiéndose de una herramienta automatizada que puede realizar la deconvolución y la segmentación con varios métodos y parámetros para finalmente realizar la medición de características.

Capítulo 6

Clasificación mediante aprendizaje automático

En el presente capítulo se presenta una posible aplicación del pipeline realizado, la clasificación de individuos 'Control' de individuos 'Paciente'. El abordaje propuesto es a través de modelos de aprendizaje automático.

6.1. Marco teórico

Se presentan en el Apéndice G algunos conceptos básicos de aprendizaje automático y análisis de datos. Para el análisis de datos se utilizará la Correlación, la Matriz de Correlación, el Principal Component Analysis(PCA), a su vez se estudiará los efectos que genera la calidad y cantidad de los datos.

Los modelos a probar se presentan en el Apéndice H junto a qué función de *loss* minimizan y los hiperparámetros a probar.

6.2. Conjunto de entrenamiento

Todos los modelos a probar serán de aprendizaje supervisado ya que se posee la etiqueta que se desea predecir para un cierto conjunto de entrenamiento. En esta parte se obtendrán los parámetros descriptos en la Sección 5 para la BD4 (detalle de BD4 en la Tabla 6.1. Se probaron dos preprocesados, por un lado, deconvolucionado con Richardson-Lucy con 50 iteraciones y segmentación K-Means con dos *clusters* y por otro segmentación con U-Net/ResNet entrenado con los stacks apilados, sin *dropout* y época 22. En ambos casos se filtran las estructuras con volumen menor a 200 píxeles por considerarlas información no relevante o ruido.

6.3. Análisis de datos

Para la aplicación se dispone de 238 medidas para los stacks brindados por el grupo IMAGINA. Se cuenta con un conjunto de 119 características que claramente

Etiqueta	**Cantidad**	**Total**
Paciente	85	238
Control	153	

Tabla 6.1: Detalle de la base de datos 4. Cuenta con stacks brindados por el grupo IMAGINA de pacientes patológicos y control.

para la cantidad de datos que se dispone puede derivar en sobreajuste, por lo que se realizará un análisis de las características para obtener las más importantes. La dimensión de las características es superior a las mediciones que se presentaron en el Capitulo 5 debido a que los stacks del grupo IMAGINA no tienen un solo cuerpo por stack, por lo que se debe tomar decisiones sobre cómo se calcula las características cuando existe más de un cuerpo dentro del stack. En el caso que se presenta en esta sección se incorporó para el análisis estadístico la separación en subgrupos de los distintos cuerpos, obteniendo tres agrupamientos ($G_1, G_2\, y\, G_3$), las estructuras se filtraron por grupos en base a su volumen habiéndose elegido como umbrales: $TH^1_{VOL} = 500$ y $TH^2_{VOL} = 2000$. Dentro de cada subgrupo se tomó para cada volumen: media, desviación estándar, mediana, mínimo y máximo de cada parámetro.

6.3.1. Método 1: segmentación con K-Means

Correlación y PCA

Para comenzar con la selección de características se realizó un estudio de la correlación entre las distintas características y la etiqueta asignada, primariamente se mostrará la matriz de correlación utilizando el coeficiente de Pearson para mostrar de forma general, que existe una diferencia apreciable entre los mejores y los peores parámetros. De la matriz 6.1 se puede apreciar que existe una evidente diferencia entre los mejores parámetros y los peores, por lo que la etiqueta está vinculada con los parámetros medidos. Por otro lado, en los recuadros violetas se pueden apreciar agrupamientos de características muy correlacionadas entre sí, por lo que a pesar que se disponga de las mejores 20 características, no todas aportan información nueva.

Como ya fue esbozado, se utilizará la correlación de Pearson, Kendall y Spearman para no obviar que puedan existir correlaciones no lineales. Se muestran los mejores 10 parámetros en la tabla 6.2. En este caso se realizó un filtrado de algunos parámetros que tenían una correlación elevada entre sí según Pearson, como fue mostrado en la matriz de correlación. De este análisis se muestra que independientemente del tipo de correlación los parámetros que influyen mayoritariamente son los que pertenecen a G_3 y los parámetros que provienen del análisis de la red mitocondrial. Para refinar la selección se realiza una nueva matriz de correlación con los métodos de Spearman y Kendall, éstas se pueden visualizar en la Figura 6.2

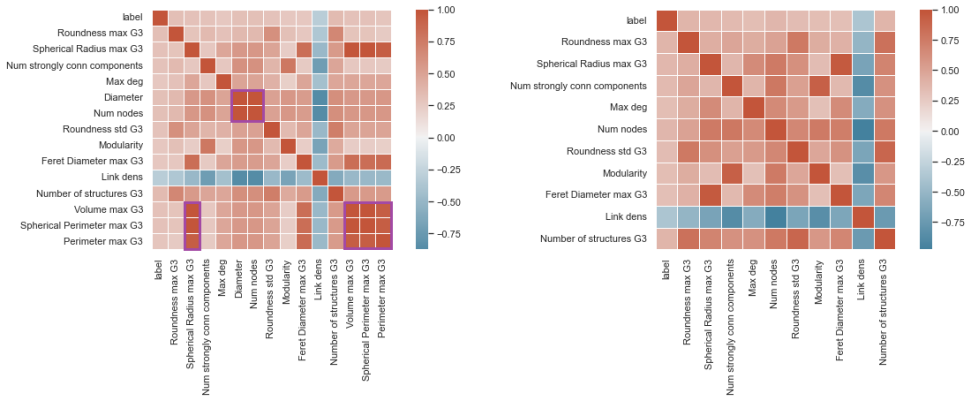
En base a este análisis se establece como un experimento el entrenamiento de los modelos utilizando: el máximo de la esfericidad de G_3, el radio esférico máximo

Figura 6.1: Matriz de correlación con las mejores 20 y peores 20 características según la correlación de Pearson. La última fila muestra como se relacionan los parámetros con respecto a la etiqueta. Los recuadros violetas muestran agrupamiento de parámetros que están muy correlacionados entre sí. En el recuadro amarillo se muestran los peores 20 parámetros.

de G_3, la cantidad de componentes fuertemente conectados, la cantidad de nodos, la desviación estándar de la esfericidad en G_3, la modularidad, el diámetro de Feret máximo en G_3, la densidad de *links*, el grado máximo y finalmente, el número de

Parámetro	Pearson	Parámetro	Kendall	Parámetro	Spearman
Roundness max G3	0.419002	Number of structures G3	0.344845	Roundness max G3	0.387936
Spherical Radius max G3	0.383039	Roundness max G3	0.321668	Number of structures G3	0.387862
Num strongly conn components	0.349466	Link dens	-0.316559	Link dens	-0.386730
Max deg	0.338170	Diameter	0.313247	Diameter	0.381091
Diameter	0.332167	Num nodes	0.313142	Num nodes	0.380964
Num nodes	0.331113	Spherical Radius max G3	0.304643	Volume max G3	0.367391
Roundness std G3	0.330668	Volume max G3	0.304643	Spherical Perimeter max G3	0.367391
Modularity	0.327983	Spherical Perimeter max G3	0.304643	Spherical Radius max G3	0.367391
Feret Diameter max G3	0.325376	Roundness std G3	0.302400	Num strongly conn components	0.358962
Link dens	-0.312063	Num strongly conn components	0.299631	Perimeter max G3	0.348637

Tabla 6.2: Mejores 10 párametros según distintas correlaciones. De los parámetros que tienen máximo, mínimo, desviación estándar, mediana o media, se seleccionó al mejor por considerarlos correlacionados entre sí y que otros parámetros pueden aportar más información nueva.



(a) Método de correlación: **Kendall**.

(b) Método de correlación: **Spearman**.

Figura 6.2: Matrices de correlación con distintos métodos. Se utilizan menos parámetros porque se realizó un refinamiento seleccionado solo un parámetro cuando mantenía gran correlación con otro. En violeta se recuadran los parámetros que mantienen una correlación alta.

estructuras de G_3. En la Figura 6.3 se pueden ver los histogramas de los mejores parámetros obtenidos donde se puede apreciar que las distribuciones estadísticas son distintas para los datos dependiendo de la etiqueta.

El otro abordaje propuesto para el análisis de datos es la utilización del PCA. Para esto, se utilizó la clase *PCA* de *Scikit-Learn* y se tomó $n\_components = 10$. En la Figura 6.4a se puede visualizar la evolución de la varianza acumulada a medida que se aumenta la cantidad de componentes principales. El primer componente explica el 82,3 % de la varianza y el segundo componente 16,1 % siendo claramente los portadores de la mayoría de la información, con estos dos se supera el umbral de 95 % e incorporando un componente más se supera el 99 %. En la Figura 6.4b se muestra un diagrama de dispersión de los dos primeros componentes principa-

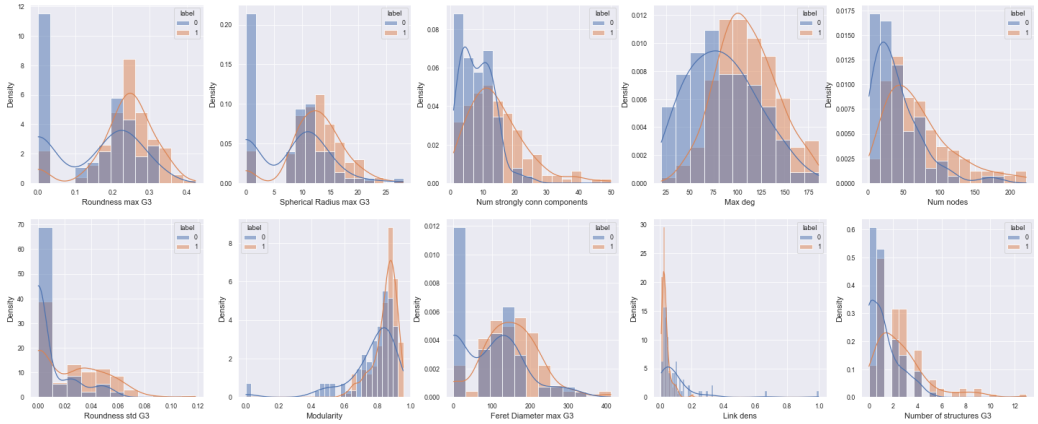
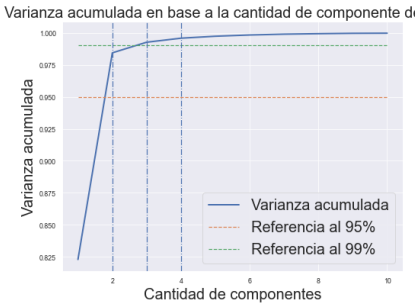
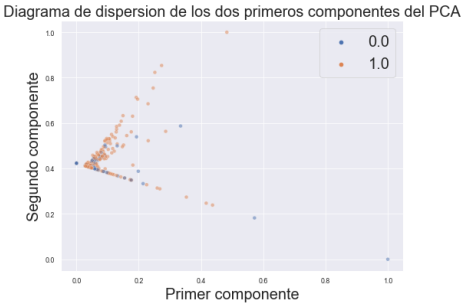


Figura 6.3: Histogramas junto a una estimación de la distribución de probabilidad, en azul la distribución correspondiente a la etiqueta 0 y en naranja la distribución correspondiente a la etiqueta 1. Todos los histogramas se normalizan para contrarrestar el efecto del desbalance de datos



(a) Varianza acumulada en función de los componentes principales obtenidos con PCA. Se grafican rectas de referencia para mostrar cuando se supera el 95 % y el 99 % de la varianza acumulada.



(b) Diagrama de dispersión de los dos primeros componentes principales detectados con el PCA. En azul los puntos que se corresponde con la etiqueta 0 y en naranja los que se corresponden con la etiqueta 1. Las magnitudes están normalizadas.

Figura 6.4: Análisis de los componentes principales derivado del PCA.

Parámetro	Pearson	Parámetro	Kendall	Parámetro	Spearman
Volume max G3	0.287083	Number of structures G3	0.261702	Number of structures G3	0.283169
Spherical Perimeter max G3	0.286784	Volume max G3	0.247664	Volume max G3	0.281111
Roundness max G3	0.279922	Spherical Perimeter max G3	0.247664	Spherical Perimeter max G3	0.281111
Spherical Radius max G3	0.278154	Spherical Radius max G3	0.247664	Spherical Radius max G3	0.281111
Roundness mean G3	0.330668	Roundness max G3	0.246860	Roundness max G3	0.280219
Info content	0.265619	Feret Diameter max G3	0.221176	Spherical Perimeter mean G3	0.264918
Number of structures G2	0.264479	Elongation max G3	0.216637	Perimeter max G3	0.256753
Perimeter max G3	0.264085	Info content	0.190448	Feret Diameter max G3	0.251058
Num nodes	0.260815	Number of structures G2	0.185226	Elongation max G3	0.245912
Diameter	0.260815	Diameter	0.181540	Info content	0.232723

Tabla 6.3: Mejores 10 párametros según distintas correlaciones para el caso con segmentación U-Net/ResNet. De los parámetros que tienen máximo, mínimo, desviación estándar, mediana o media, se seleccionó al mejor por considerarlos correlacionados entre sí y que otros parámetros pueden aportar más información nueva.

les del PCA, en éste se puede evidenciar que es posible agrupar los parámetros por etiqueta, fortaleciendo la idea de que es posible aprender con los modelos de aprendizaje automático.

6.3.2. Método 2: segmentación con U-Net/Resnet

Correlación y PCA

El análisis a realizar en este caso es muy similar al presentado para el caso de K-Means. En la Figura 6.5 se muestra la matriz de correlación para los 20 mejores y 20 peores parámetros según la correlación de Pearson. En ésta se puede visualizar nuevamente que los parámetros más correlacionados son los que se corresponden al tercer grupo. Pero a su vez, se evidencia claramente que hay mayor cantidad de parámetros correlacionados entre sí. En la Tabla 6.3 se muestras los mejores resultados para los distintos tipos de correlación, se seleccionó solo una medida estadística (media, mediana, máximo, mínimo y desviación estándar) por parámetro, de aquí lo principal que se puede evidenciar es que la magnitud en todas las correlaciones es inferior que el caso K-Means.

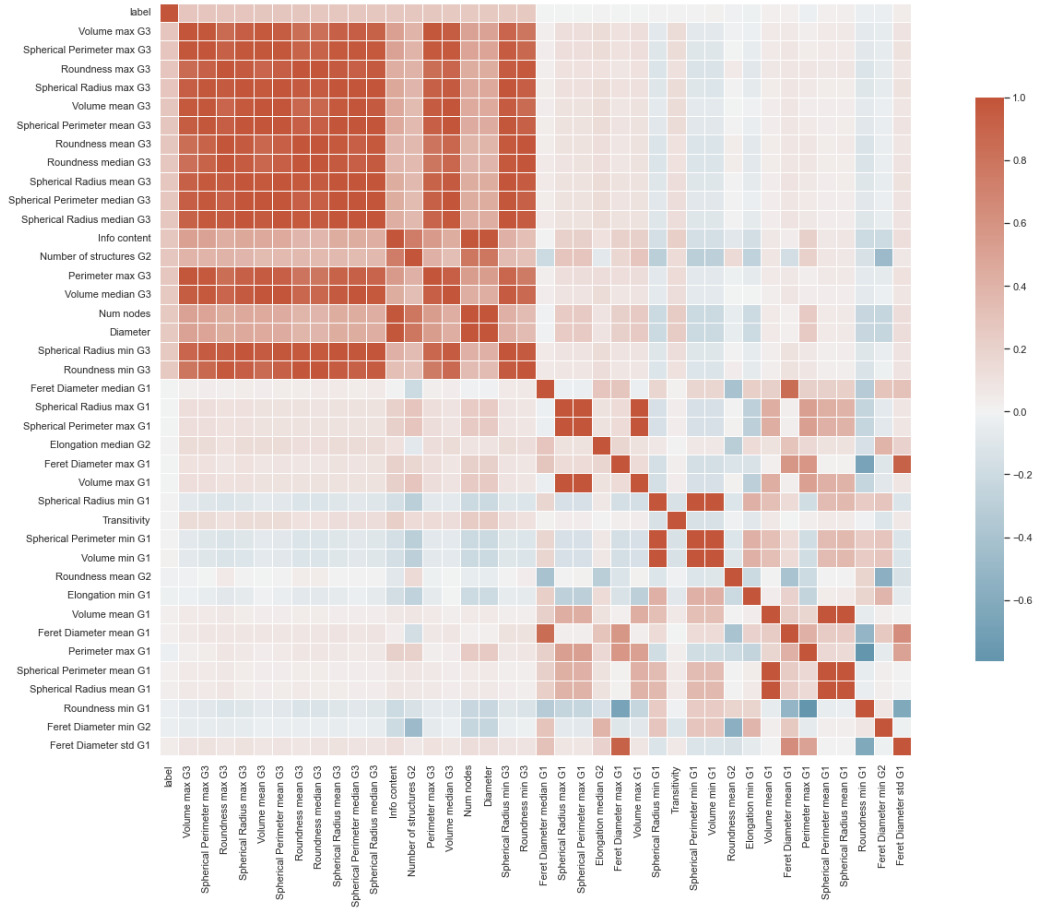
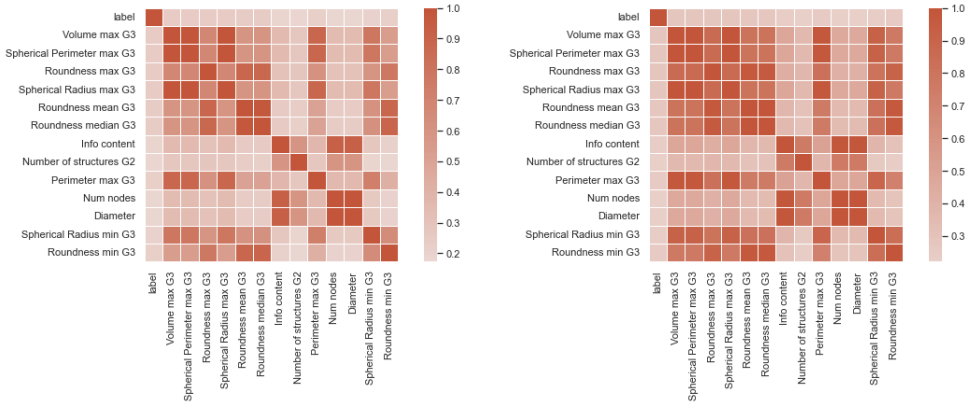


Figura 6.5: Matriz de correlación con las mejores 20 y peores 20 características según la correlación de Pearson. La última fila muestra como se relacionan los parámetros con respecto a la etiqueta.

En las Figuras 6.6a y 6.6b se puede ver el proceso de selección de características, habiéndose seguido el mismo procedimiento que con el método 1.

Finalmente, se realiza el análisis de los componentes principales. En la Figura 6.7a se puede visualizar la varianza acumulada donde a diferencia del método 1, se necesitan mayor cantidad de componentes para explicar más del 99 % de la varianza acumulada, en este caso se supera luego del quinto componente principal. Por otro lado en la Figura 6.7b se muestra el diagrama de dispersión donde se puede apreciar que existirían algunas regiones (cuando las dos componentes son grandes) que parece corresponderse con la etiqueta naranja, pero son pocos datos y eso indica dificultad para el aprendizaje.

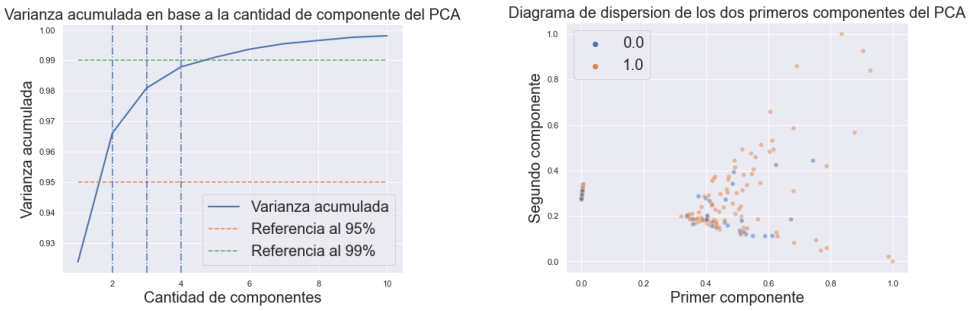
Como conclusión para este análisis, se tiene que existe una correlación pequeña con la etiqueta pero los resultados fueron peores que en el método 1. En base a esto, es esperable que el método 1 permita realiza una mejor clasificación.



(a) Método de correlación: **Kendall**.

(b) Método de correlación: **Spearman**.

Figura 6.6: Matrices de correlación con distintos métodos. Se utilizan menos parámetros porque se realizó un refinamiento seleccionado solo un parámetro cuando mantenía gran correlación con otro. En violeta se recuadran los parámetros que mantienen una correlación alta.



(a) Varianza acumulada en función de los componentes principales obtenidos con PCA. Se grafican rectas de referencia para mostrar cuando se supera el 95 % y el 99 % de la varianza acumulada.

(b) Diagrama de dispersión de los dos primeros componentes principales detectados con el PCA. En azul los puntos que se corresponde con la etiqueta 0 y en naranja los que se corresponden con la etiqueta 1. Las magnitudes están normalizadas.

Figura 6.7: Análisis de los componentes principales derivado del PCA.

6.4. Entrenamiento de modelos

Se seleccionó tres modelos, Regresión Logística (LR), *Support Vector Machine* (SVM) y *Random Forest* (RF) como método de ensamble. Los modelos se implementaron con la librería de Python *Scikit-Learn* y utilizando en particular la clase *Pipeline*. Se probó concatenar *Standard Scaler*(Normalizar), *PCA* con dos componentes principales y el modelo (LR, SVM o RF). Los experimentos que se realizaron son: parámetros por defecto de cada modelo, utilizar todos los datos (sin selección de características), utilizar *PCA* de dos componentes o solo los mejores parámetros obtenidos de la sección anterior. En el caso de *Random Forest* se incorporo una última prueba con los mejores parámetros sin utilizar el *StandardScaler*.

El primer experimento consiste en entrenar los modelos con todos las características y PCA pero con parámetros por defecto. El objetivo de este experimento

LR	SVM	RF
$C = 1$	$C = 1$	$Estimadores = 100$
$Norma\ regularización = L_2$	$Kernel = RBF$	$Profundidad\ máxima = None$

Tabla 6.4: Parámetros por defecto utilizados en el entrenamiento de los modelos. Estos no son los únicos híıperparámetros de cada modelo.

	LR		**SVM**		**RF**	
	Train	*Val*	*Train*	*Val*	*Train*	*Val*
Accuracy	0.74	0.74	0.75	0.72	1.00	0.69
Precision	0.73	0.74	0.74	0.72	1.00	0.74
Recall	0.92	0.93	0.94	0.93	1.00	0.80
Balanced Accuracy	0.66	0.66	0.67	0.63	1.00	0.64

Tabla 6.5: Resultados del promedio de los folds de la validación cruzada con el conjunto de entrenamiento. Los datos presentados en la Tabla se corresponden con el método 1.

era comprobar si era posible aprender en base a estos datos. Se presenta en la tabla 6.4 los parámetros por defecto utilizados.

6.4.1. Método 1 - Segmentación con K-Means

De los resultados obtenidos en la Tabla 6.5 se constata la importancia de utilizar varias métricas comprendiendo lo que sucede por detrás de ellas, si el resultado se midiese solo con la *Accuracy* se podría inferir incorrectamente ya que se cuenta con un conjunto de datos desbalanceado, con aproximadamente 66 % de una etiqueta y 34 % de la otra etiqueta, por lo que una *Accuracy* mayor a 0.66 sería lo deseable. Como se puede ver, en todos los métodos se obtuvo, sin ajuste de hipérparámetros, un clasificador que permite inferir la etiqueta en base a las características. El rendimiento del clasificador es bajo, pero para un primer abordaje es interesante que efectivamente se pueda clasificar.

Debido a que se tiene pocos datos, se probó entrenando con varios conjuntos cambiando el *random seed*, obteniendo resultados similares.

Finalmente, se entrenaron los modelos con todo el conjunto de entrenamiento y se evaluó con el conjunto de test apartado inicialmente (20 %) del conjunto de entrenamiento, este conjunto es estratificado para mantener la proporción de las clases.

Los resultados se muestran en la Tabla 6.6, donde se puede visualizar que

	LR		**SVM**		**RF**	
	Train	Test	*Train*	Test	*Train*	Test
Accuracy	0.74	0.73	0.75	0.77	0.99	0.63
Balanced Accuracy	-	0.68	-	0.72	-	0.60

Tabla 6.6: Resultados con el conjunto de test de los distintos métodos probados. Hiperparámetros por defecto. Los datos presentados en la Tabla se corresponden con el método 1.

LR	SVM	RF
$C \in [0{,}01, 1]$, $step = 0{,}01$	$Kernel \in [Lineal, RBF]$	$N\_estimadores \in [10, 250]$, $step = 10$
-	$C \in [0{,}01, 1]$, $step = 0{,}01$	$max\_depth \in [1, 2, 3, 4, 5]$

Tabla 6.7: Rango de valores para los hiperparámetros utilizados en el Grid Search. Los datos presentados en la Tabla se corresponden con el método 1.

		LR		**SVM**		**RF**	
		Train	*Test*	*Train*	*Test*	*Train*	*Test*
Todos las características	*Accuracy*	0.72	0.73	0.77	0.64	0.73	0.68
	Balanced Accuracy	-	0.70	-	0.60	-	0.67
Mejores características	*Accuracy*	0.74	0.75	0.73	0.77	0.73	0.73
	Balanced Accuracy	-	0.71	-	0.73	-	0.68
Todas las características PCA (2 componentes)	*Accuracy*	0.74	0.73	0.75	0.77	0.75	0.73
	Balanced Accuracy	-	0.68	-	0.72	-	0.67

Tabla 6.8: Resultados de entrenamiento y con el conjunto de test de los distintos métodos probados y los distintos experimentos. Los mejores parámetros se pueden ver en la Tabla 6.9. Los datos presentados en la Tabla se corresponden con el método 1.

tanto regresión logística como SVM tienen un buen desempeño y efectivamente clasifican. El Random Forest está sobreajustado, por lo que regularizarlo podría permitir una mejor generalización.

Para seleccionar los hiperparámetros y así contrarrestar el sobreajuste del *Random Forest* o mejorar el desempeño de *SVM* y *LR* se realizó un *Grid Search* con los parámetros presentados en la Tabla 6.7.

Como ya fue mencionado se realizaron tres experimentos, del *Grid Search* de cada experimento se desprenden los resultados obtenidos en la Tabla 6.8. La mejor *Regresión Logística* se obtuvo para el caso donde se utilizan solo las mejores 10 características, aunque con una diferencia muy pequeña en relación a utilizarlas todas. El *SVM* con todas las características fue el peor estimador, claramente sobreajustado si vemos la diferencia entre el *accuracy* de *Train* y de *Test*, pero cuando utiliza menos características se tienen los estimadores con mejores desempeño. El *Random Forest* tiene resultados muy similares en los tres casos, donde se evidencia que se pudo contrarrestar el sobreajuste y permitir una mejor generalización. El último experimento realizado fue no estandarizar los datos para el *Random Forest*, debido a que no es necesario realizar este tipo de preprocesado cuando se utilizan árboles de decisión. En la Tabla 6.10 se indica que sin utilizar el *StandardScaler* se obtienen mejores resultados para el *Random Forest* obteniendo 71 % de *balanced accuracy* cercano a los mejores resultados obtenidos.

6.4.2. Método 2 - Segmentación con U-Net/ResNet

En la Tabla 6.11 se muestran los resultados para el primer experimento, es decir, sin ajustar los parámetros por defecto de los modelos. En esta Tabla, se puede apreciar que los modelos no logran aprender a clasificar en base a los datos brindados ya que la *accuracy balanceada* se encuentra alrededor del 50 %, no muy distinto a una moneda.

	LR	**SVM**	**RF**
Todos las características	$C = 0{,}05$	$C = 0{,}28$ kernel = lineal	max_depth=3 estimators=20
Mejores características	$C = 0{,}04$	$C = 0{,}68$ kernel = RBF	max_depth=2 estimators=40
Todas las características PCA (2 componentes)	$C = 0{,}02$	$C = 0{,}26$ kernel = RBF	max_depth=4 estimators=30

Tabla 6.9: Hiperparámetros obtenidos mediante *Grid Search* para todas las combinaciones de modelos y experimentos probados. Los datos presentados en la Tabla se corresponden con el método 1.

StandardScaler	***Params***	***Train***	***Val***	***Test***
TRUE	max_depth=2 estimators=40	0.73	0.73	0.68
FALSE	max_depth=2 estimators=210	0.74	0.75	0.71

Tabla 6.10: Resultados obtenidos cuando se utiliza *StandardScaler* y cuando no se preprocesan los datos. Para el entrenamiento se seleccionó las mejores características. Los datos presentados en la Tabla se corresponden con el método 1.

	LR		**SVM**		**RF**	
	Train	*Val*	*Train*	*Val*	*Train*	*Val*
Accuracy	0.66	0.55	0.66	0.63	1.00	0.63
Precision	0.69	0.64	0.66	0.65	1.00	0.69
Recall	0.87	0.96	1.00	1.00	1.00	0.80
Balanced Accuracy	0.57	0.49	0.54	0.48	1.00	0.57

Tabla 6.11: Resultados del promedio de los folds de la validación cruzada con el conjunto de entrenamiento. Los datos presentados en la Tabla se corresponden con el método 2.

	LR		**SVM**		**RF**	
	Train	Test	*Train*	Test	*Train*	Test
Accuracy	0.65	0.65	0.67	0.63	1.00	0.60
Balanced Accuracy	-	0.55	-	0.48	-	0.56

Tabla 6.12: Resultados con el conjunto de test de los distintos métodos probados. Todos los hiperparámetros por defecto. Los datos presentados en la Tabla se corresponden con el método 2.

		LR		SVM		RF	
		Train	*Test*	*Train*	*Test*	*Train*	*Test*
Todos las características	*Accuracy*	0.62	0.67	0.65	0.65	0.68	0.67
	Balanced Accuracy	-	0.57	-	0.50	-	0.58
Mejores características	*Accuracy*	0.65	0.65	0.65	0.65	0.71	0.69
	Balanced Accuracy	-	0.50	-	0.50	-	0.63
Todas las características PCA (2 componentes)	*Accuracy*	0.63	0.69	0.65	0.65	0.67	0.69
	Balanced Accuracy	-	0.61	-	0.50	-	0.60

Tabla 6.13: Resultados de entrenamiento y con el conjunto de test de los distintos métodos probados y los distintos experimentos. Los mejores parámetros se pueden ver en la Tabla 6.14. Los datos presentados en la Tabla se corresponden con el método 2.

	LR	SVM	RF
Todos las características	$C = 0{,}01$	$C = 0{,}01$ kernel = RBF	max_depth=4 estimators=220
Mejores características	$C = 0{,}01$	$C = 0{,}35$ kernel = RBF	max_depth=3 estimators=50
Todas las características PCA (2 componentes)	$C = 0{,}02$	$C = 0{,}01$ kernel = Lineal	max_depth=4 estimators=200

Tabla 6.14: Hiperparámetros obtenidos mediante *Grid Search* para todas las combinaciones de modelos y experimentos probados. Los datos presentados en la Tabla se corresponden con el método 2.

En la Tabla 6.12 se indican los resultados obtenidos entrenando con todo el conjunto y validando con los que fueron apartados oportunamente. Aquí se confirma que efectivamente los modelos no están aprendiendo y los resultados obtenidos se deben principalmente al azar. Vale recordar que se está utilizando PCA con dos componentes principales y en este caso, no se explicaba tanta varianza acumulada como en el caso K-Means por lo que utilizar más componentes principales podría ayudar.

Los hiperparámetros a utilizar en el refinamiento de los modelos se detalló en la Tabla 6.7 de la sección anterior.

Para los parámetros que brindan los mejores resultados (Tabla 6.14) se muestran los resultados obtenidos en la Tabla 6.13. De esta Tabla se puede concluir que no se logró resultados aceptables ya que los resultados oscilan alrededor del 65 % que es la proporción entre las dos clases, es decir, un clasificador que siempre asigne la misma etiqueta, podría obtener el mismo resultado.

6.5. Conclusiones

En este Capítulo se mostró una posible aplicación del Pipeline desarrollado, la utilización de la medición de características obtenidas para clasificar en dos subgrupos. Del análisis preliminar de los datos se evidenció la correlación con la etiqueta a predecir, así como armar un conjunto de características para el entrenamiento. El análisis de componentes principales fortaleció la idea de que se podía aprender de las medidas obtenidas y sirvió como un experimento distinto a la selección

realizada mediante la correlación.

El tiempo de entrenamiento de los distintos modelos no fue evaluado porque en general la velocidad fue alta y ningún *Grid Search* duró más de dos minutos, en un caso donde se tengan más datos, esto debería ser parte del análisis ya que el experimento de entrenar utilizando las 118 características puede volverse inviable.

Se establecieron tres experimentos, utilizando todas características, las diez mejores o todas las características pero incorporando un *PCA* de dos componentes principales. Los resultados para el caso de segmentar con K-Means y deconvolucionar con RL son satisfactorios mostrando la posibilidad de aprender a clasificar en base a las medidas obtenidas por el Pipeline, los tres métodos permitieron superar el umbral de 70 % de *balanced accuracy* que era el objetivo que el equipo se había propuesto. Lamentablemente la U-Net/Resnet no brindó buenos resultados, lo cual podría era de esperar en base a la poca correlación obtenida entre los datos y la etiqueta a predecir. Esto no quiere decir que no se pueda utilizar la U-Net/ResNet, sino que se deberían probar distintas alternativas para mejorar su desempeño, por ejemplo, entrenar de otras maneras, alterar la arquitectura, utilizar imágenes deconvolucionadas o elegir el *ground truth* de otra manera.

Capítulo 7

Conclusiones

7.1. Análisis General del Desarrollo del Proyecto

El proyecto contó con distintas etapas, las cuales propusieron distintos desafíos a enfrentar. En un principio fue necesaria una instancia de investigación del estado del arte, logrando adquirir conocimientos generales y específicos sobre la microscopía de fluorescencia confocal. Esto permitió dar contexto al problema a abordar, comprendiendo los pasos a seguir para tratar el mismo.

La primera dificultad a enfrentar fue la insuficiencia de datos. Esto implicó una etapa de generación de datos sintéticos para probar las distintas etapas del pipeline y entrenar la red neuronal convolucional. Se logró generar estructuras sintéticas similares a las identificadas en las redes mitocondriales, así como la implementación de una GAN que busca transformar los volúmenes binarios a otro dominio que se asemeja al efecto de la microscopía de fluorescencia. Si bien el generador de datos sintéticos podría utilizarse en otras aplicaciones, es necesario realizar una optimización en su implementación puesto que los tiempos de generación son muy elevados.

Se logró probar la deconvolución con distintos métodos y parámetros, evaluando su desempeño según algunas métricas seleccionadas. En base a los resultados obtenidos por la U-Net/ResNet se confirmó la importancia de esta etapa para segmentar y se considera clave para trabajos que involucren microscopía de fluorescencia confocal. La deconvolución permite contrarrestar el efecto del *blur* y mejorar la definición de las estructuras, mejorando la precisión al momento de segmentar.

La segmentación fue de las etapas más complejas ya que la implementación de una red convolucional como la U-Net/ResNet, conllevo no solo desafíos de programación sino de cómo entrenarla en un escenario donde se disponía de pocos datos y dos dominios distintos. Este abordaje no arrojó buenos resultados, lo cual no quiere decir que le modelo no sirva, sino que se debe entrenar de otra manera, preprocesar distinto, etc. Finalmente, se logró tener buenos resultados de segmentación con métodos clásicos y de *clustering*.

Como fuera mencionado en el capítulo correspondiente, la medición de carac-

terísticas fue satisfactoria, ya que se logró implementar los parámetros morfológicos pero además, implementar nuevos parámetros derivados de la conectivdad de la red mitocondrial. Estos nuevos parámetros son de importancia en las hipótesis del grupo IMAGINA y teniendo en cuenta que tienen poco error con respecto al *ground truth* (menor al 22 %) se consideran parámetros de gran confianza. Los parámetros morfológicos también fueron de gran confianza pero tuvieron un error relativo sensiblemente mayor.

La etapa de clasificación como fuera mencionado anteriormente, era una posible aplicación del pipeline. En esta etapa nuevamente se disponía de pocos datos y con un largo proceso de selección de parámetros que podía concluir en una mala correlación de las medidas obtenidas con la etiqueta a predecir. Se logró entrenar clasificadores obteniendo un resultado decente teniendo en cuenta los elementos anteriormente mencionados. En esta etapa se vió que la medición de características cuando se utiliza la U-Net/ResNet tiene demasiado ruido y no se pudo entrenar los modelos.

Finalmente, se logró construir un pipeline de punta a punta, que implementa las etapas anteriormente probadas, utilizando librerías o software de terceros, todo automatizado mediante Python.

7.2. Comparación entre Objetivos Iniciales y Resultados

A modo de determinar si el proyecto fue exitoso, se plantearon al comienzo del mismo algunos criterios de éxito. A continuación se listan los mismos, junto con un breve análisis evaluando si fueron cumplidos.

- *Establecer contacto periódico con los referentes médicos para adquirir experiencia trabajando en forma interdisciplinaria, logrando comprender el objeto de estudio de la otra disciplina y llevarlo al dominio ingenieril. Entender cómo funciona el proceso de adquisición de las imágenes.*

  Se logró el abordaje de un problema del ámbito médico a la ingeniería manteniendo contacto con los investigadores de IMAGINA, quienes no solo colaboraron proporcionando datos, sino además información y sugerencias para el trabajo. En cuanto a los fundamentos de la adquisición de imágenes, estos fueron estudiados, entendidos y explicados en la Sección 2.1.

- *Crear o compilar una base de datos con la cual no solo nosotros podamos trabajar, sino que pueda ser utilizada para futuros trabajos.*

  La generación de estructuras obtuvo un desempeño satisfactorio, pudiendo crear 5 subgrupos de estructuras con parámetros que pueden ser variados al llamar a las funciones. El mayor inconveniente de este generador es el tiempo de ejecución, pero esto no fue considerado un problema debido a que no era prioridad. Con la GAN, se obtuvieron resultados suficientemente satisfactorios, aunque produce un artefacto en los cortes negros que es perjudicial en las etapas del pipeline. Una evaluación más exhaustiva de la GAN, sus parámetros y arquitectura, podría brindar mejores resultados en

la generación, lo cual no fue abordado en profundidad por no ser parte de la formulación del proyecto original.

- *Analizar, comparar, discutir e identificar un algoritmo de segmentación que realice la media, votación u otra combinación entre la opinión de distintos profesionales en el área. Se evaluará la segmentación en base a la votación de los referentes médicos, apuntando a una segmentación correcta en por lo menos 70 % de los casos.*

  En el transcurso del proyecto se eligió otra forma de evaluar la segmentación, utilizándose métricas de similitud y estadística para analizar los resultados. El intercambio con los referentes médicos hubiera enriquecido los resultados pero a su vez, tener métricas objetivas permiten reducir la subjetividad de los resultados.

- *Obtener parámetros morfológicos variados, haciendo mayor el abanico de posibilidades a la hora de la clasificación. Se propone relevar por lo menos 7 características morfológicas.*

  En cuanto a la extracción de características, el objetivo planteado fue ampliamente superado, obteniendo 7 parámetros morfológicos y además 9 parámetros que describen la conectividad de la red mitocondrial. También se exploraron distintas maneras de resumir los parámetros morfológicos para un stack, considerando la media, la mediana, la desviación estándar, máximo y mínimo.

- *Realizar un pipeline enfocado en el análisis de imágenes obtenidas mediante microscopía de fluorescencia confocal y lograr que sea aplicable a otros organelos.*

  Se logró la implementación de un pipeline que permite procesar stacks de microscopía confocal, aplicando deconvolución, segmentación y extracción de características sobre las estructuras presentes en las mismas. El pipeline permite realizar algunas variaciones en los métodos de deconvolución y segmentación, de forma que se ajusten de mejor manera a las imágenes a analizar.

7.3. Comentarios Generales y Trabajo a Futuro

Al tratarse con un pipeline de punta a punta, se abarcaron muchos temas en los que se puede hacer un estudio de mayor profundidad. A continuación se muestra un punteo de posibles trabajos a futuro relacionados.

- **Mayor cantidad de datos:** La toma de datos es fundamental, aumentar la cantidad y mejorar la calidad de estos es crucial para las etapas posteriores.

- **Interfaz gráfica:** Implementar una interfaz gráfica para el generador y el pipeline que permita seleccionar los parámetros visualmente. Debido al uso

general que se le da a ImageJ en el ámbito de investigación médica, podría ser de interés diseñar un *plugin* del mismo.

- **Prioridad en clasificación:** Los modelos de clasificación probados no fue una selección muy exhaustiva debido a que se tomó solo como una aplicación del trabajo realizado. Se pueden incorporar más métodos comúnmente utilizados, como por ejemplo redes neuronales.
- **GAN 3D:** Dados los resultados obtenidos en la generación de datos sintéticos, es interesante implementar una GAN que trabaje en tres dimensiones, evaluar distintas arquitecturas y parámetros que deriven en stacks con mayor similitud estadística al objetivo.
- **Variación de U-Net:** Debido a la gran popularidad de esta arquitectura, existen diversas alternativas y variaciones de la misma que obtienen resultados interesantes. Podría estudiarse más a fondo su funcionalidad y proponer alternativas.
- **Función distancia:** Incorporar la distancia entre la red mitocondrial y el núcleo de cada monocito es un parámetro de gran relevancia segun el grupo IMAGINA y por lo tanto, sería interesante relevarlo y probar cómo afecta la clasificación.
- **Clasificación de estructuras:** Una alternativa de clasificación podría ser evaluar que tan bien funciona el pipeline apuntando a una clasificación de los distintos organelos que se generaron.

Apéndice A

Tabla de resultados de deconvolución

Método	**Parámetro**	**SNR**	**PSNR**	**MI**	**KL**
LW	$M_{iter} = 100$, $\gamma = 0.001$	-12.3722	28.11709	0.264886	8.21525
LW	$M_{iter} = 100$, $\gamma = 1.0$	-5.12532	35.36399	**0.430897**	0.408411
LW	$M_{iter} = 20$, $\gamma = 0.001$	-11.6379	28.85138	0.308891	7.94978
LW	$M_{iter} = 20$, $\gamma = 1.0$	-5.90905	34.58026	0.428012	0.624669
LW	$M_{iter} = 50$, $\gamma = 0.001$	-12.4284	28.06088	0.257436	8.230718
LW	$M_{iter} = 50$, $\gamma = 1.0$	-5.22256	35.26675	0.429511	0.471349
NIF	-	-9.14295	31.34636	0.346758	0.538358
RIF	$\lambda = 0.001$	-7.07974	33.40957	0.430772	0.414474
RIF	$\lambda = 0.01$	-5.69057	34.79874	**0.439109**	0.410049
RIF	$\lambda = 0.1$	-5.74058	34.74873	0.426397	0.534604
RIF	$\lambda = 1$	-7.17412	33.31519	**0.434358**	0.848803
RIF	$\lambda = 100$	-10.7697	29.71958	**0.432425**	1.826652
RIF	$\lambda =$ 1e-06	-7.52335	32.96596	0.420635	0.420429
RIF	$\lambda =$ 1e-09	-7.52581	32.9635	0.420635	0.420549
RIF	$\lambda = 50$	-10.3509	30.13842	**0.438788**	1.60475
RL	$M_{iter} = 10$	-1.58477	38.90454	0.410938	0.229972
RL	$M_{iter} = 100$	**-0.87111**	**39.6182**	0.344595	**0.0312**
RL	$M_{iter} = 20$	**-1.07805**	**39.41126**	0.414198	**0.137064**
RL	$M_{iter} = 50$	**-0.80395**	**39.68537**	0.377856	**0.060787**
RLTV	$M_{iter} = 10$, $\lambda = 0.001$	-1.58404	38.90528	0.410417	0.23306
RLTV	$M_{iter} = 10$, $\lambda = 0.1$	**-1.4502**	**39.03912**	0.412876	**0.142734**
RLTV	$M_{iter} = 100$, $\lambda = 0.1$	**-0.12184**	**40.36747**	0.153803	**0.104867**
RLTV	$M_{iter} = 20$, $\lambda = 0.001$	**-1.10905**	**39.38026**	0.415505	**0.141704**
RLTV	$M_{iter} = 20$, $\lambda = 0.1$	**-0.66944**	**39.81987**	0.362748	**0.047072**
RLTV	$M_{iter} = 50$, $\lambda = 0.001$	**-0.86063**	**39.62869**	0.377934	**0.063892**
RLTV	$M_{iter} = 50$, $\lambda = 0.01$	**-0.96148**	**39.52783**	0.37479	**0.061795**
RLTV	$M_{iter} = 50$, $\lambda = 0.1$	**-0.24225**	**40.24707**	0.243056	**0.047129**
TM	$M_{iter} = 100$, $\gamma = 0.001$, $\lambda = 0.001$	-12.3722	28.11714	0.26481	8.215273
TM	$M_{iter} = 100$, $\gamma = 0.001$, $\lambda = 0.01$	-12.3712	28.11811	0.264886	8.215468
TM	$M_{iter} = 100$, $\gamma = 0.001$, $\lambda =$ 1e-06	-12.3722	28.11709	0.264886	8.21525
TM	$M_{iter} = 100$, $\gamma = 0.001$, $\lambda =$ 1e-09	-12.3722	28.11709	0.264886	8.21525
TM	$M_{iter} = 100$, $\gamma = 0.001$, $\lambda = 2$	-12.2671	28.22223	0.262797	8.240621
TM	$M_{iter} = 100$, $\gamma = 1.0$, $\lambda = 0.001$	-5.28091	35.2084	**0.43192**	0.426891
TM	$M_{iter} = 100$, $\gamma = 1.0$, $\lambda =$ 1e-06	-5.12547	35.36384	**0.430897**	0.408448
TM	$M_{iter} = 100$, $\gamma = 1.0$, $\lambda =$ 1e-09	-5.12532	35.36399	**0.430897**	0.408411
TM	$M_{iter} = 20$, $\gamma = 0.001$, $\lambda = 0.001$	-11.638	28.85136	0.308891	7.949781
TM	$M_{iter} = 20$, $\gamma = 0.001$, $\lambda = 0.01$	-11.6379	28.85142	0.310227	7.949785
TM	$M_{iter} = 20$, $\gamma = 0.001$, $\lambda =$ 1e-06	-11.6379	28.85138	0.308891	7.94978
TM	$M_{iter} = 20$, $\gamma = 0.001$, $\lambda =$ 1e-09	-11.6379	28.85138	0.308891	7.94978
TM	$M_{iter} = 20$, $\gamma = 1.0$, $\lambda = 0.001$	-5.94506	34.54425	0.430522	0.6303
TM	$M_{iter} = 20$, $\gamma = 1.0$, $\lambda = 0.01$	-9.5035	30.98581	0.308586	0.658084
TM	$M_{iter} = 20$, $\gamma = 1.0$, $\lambda =$ 1e-06	-5.90906	34.58025	0.428012	0.624676
TM	$M_{iter} = 20$, $\gamma = 1.0$, $\lambda =$ 1e-09	-5.90905	34.58026	0.428012	0.624668
TRIF	$\lambda = 0.001$	-8.32163	32.16768	0.393958	0.468521
TRIF	$\lambda = 0.01$	-5.74781	34.7415	**0.436895**	0.435519
TRIF	$\lambda = 0.1$	-7.23102	33.25829	**0.440351**	0.98246
TRIF	$\lambda = 1$	-11.3927	29.09658	0.38681	4.856613
TRIF	$\lambda = 10$	-12.3977	28.09163	0.269942	8.142254
TRIF	$\lambda = 100$	-9.48086	31.00846	0.372573	7.365782
TRIF	$\lambda =$ 1e-06	-9.21926	31.27005	0.33938	0.54712
TRIF	$\lambda =$ 1e-09	-9.32171	31.1676	0.342309	0.561478
TRIF	$\lambda = 50$	-11.5242	28.96515	0.318094	7.923301

Tabla A.1: Tabla que muestra el promedios de los resultados para los stacks reales aplicando distintas combinaciones de métodos y parámetros. Se utiliza M_{iter} como M_{iter} iter para facilitar la lectura de la tabla. En verde están resaltados los resultados que son mejor que el promedio de la columna, en el caso de Kullback-Leibler significa estar por debajo del promedio. En negrita están los mejores 10 resultados.

Apéndice B

Conceptos básicos de Redes Neuronales

Una neurona es, básicamente, una operación entre dos vectores. Una operación muy utilizada en distintas arquitecturas de redes es el producto interno, este producto es realizado entre x, vector de entrada a la neurona, y w, vector de pesos de la neurona, luego de esto se aplica una función de activación, la cual se definirá más adelante, para obtener finalmente la salida de la neurona. La Figura B.1a muestra el esquema básico de funcionamiento de una neurona, donde la sumatoria representa la suma del producto interno entre x y w, f es la función de activación y, finalmente, y es la salida de la neurona. Matemáticamente esto es

$$y = f\left(\sum_{i=1}^{n} x_i w_i\right). \tag{B.1}$$

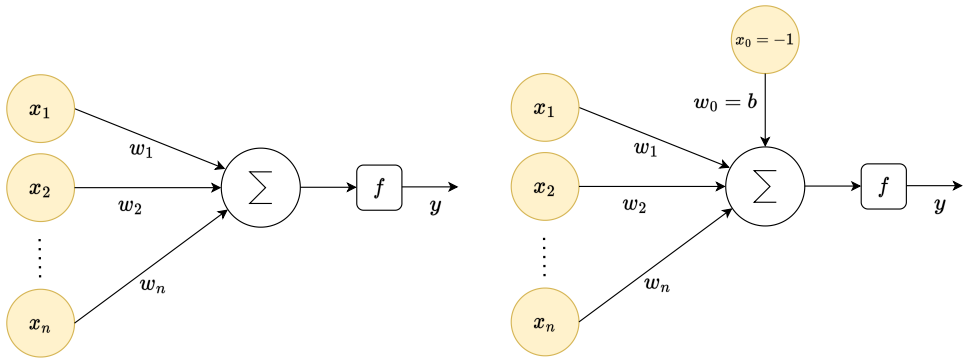
La red neuronal de la Figura B.1a es el ejemplo más sencillo de redes neuronales dado que solo cuenta con una neurona, y se la conoce como **perceptrón simple**. Existe la posibilidad de sumarle un sesgo al perceptrón, de tal forma que siga manteniendo el producto interno entre el vector de entradas y los pesos. Para esto se define $x_0 = -1$ y $w_0 = b$, siendo b el sesgo incorporado. A esto se le llama *bias trick* (truco del sesgo) y puede verse implementado en la Figura B.1b. La Ecuación B.2 cambia únicamente el índice de la sumatoria.

$$y = f\left(\sum_{i=0}^{n} x_i w_i\right) \tag{B.2}$$

Funciones de activación

Una función de activación es una función no lineal utilizada para entrenar redes, ésta define la salida de una neurona dado un conjunto de entradas. La función se usa en cada neurona de la red para ayudar a la red a aprender los patrones complejos en los datos, lo que le permite hacer predicciones. Además, el propósito de las funciones de activación es introducir no linealidades en la red neuronal [67].

Dos de las problemáticas que enfrentan las redes neuronales y están relacionadas a las funciones de activación son: *vanishing gradient* y "muerte de neuronas".



(a) Esquema de funcionamiento de una neurona artificial.

(b) Esquema de funcionamiento de una neurona artificial con el *bias trick*.

Figura B.1: Esquema de funcionamiento de una neurona donde x_i son los datos de entrada, w_i los pesos del modelo, $\sum$ la operación que implementa la neurona y f, la función de activación.

El *vanishing gradient* o desvanecimiento de gradiente ocurre cuando los valores del gradiente se acercan a cero a medida que la propagación hacia atrás (backpropagation) se adentra más en la red, lo que hace que los pesos se saturen y no se actualicen correctamente. Como resultado, la función de pérdidas deja de disminuir y la red no se entrena adecuadamente. Las neuronas cuyos pesos no se actualizan correctamente se denominan neuronas saturadas. [67]

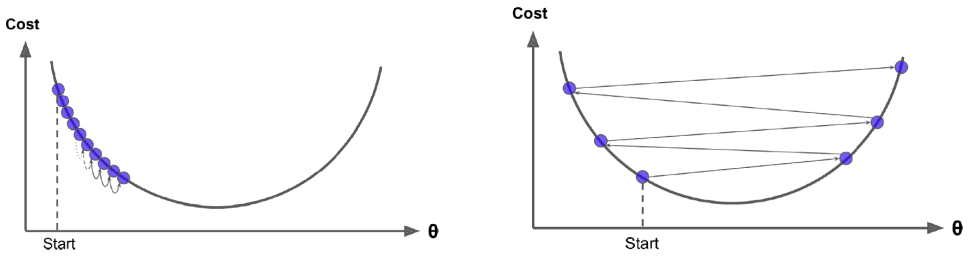
La muerte de neuronas se da cuando el valor de salida de las funciones es cercano a cero, obligando a las neuronas correspondientes a estar inactivas, por lo que no contribuyen a la salida final. Además, los pesos pueden actualizarse de tal manera que la suma ponderada de una gran parte de la red se fuerza a cero. Este escenario puede forzar la desactivación de una gran parte de la entrada, lo que resulta en un problema irrecuperable durante el rendimiento de la red. Por lo tanto, estas neuronas que han sido desactivadas a la fuerza se conocen como "neuronas muertas", y el problema se conoce como el "problema de las neuronas muertas". [67]

Entrenamiento de las redes neuronales

A la salida y se le llama **predicción**. La forma en la que se entrena una red neuronal es ajustando los pesos de cada neurona cuando se realiza una predicción incorrecta. Este proceso se repite muchas veces y la red sigue mejorando sus predicciones hasta haber alcanzado uno o varios criterios de parada. Para determinar que tan buena es una predicción se utiliza una **función de pérdida**. Las funciones de pérdida miden la calidad de una predicción utilizando esta predicción y la etiqueta original, devolviendo un valor alto cuando la predicción no es buena.

Para actualizar los pesos de tal forma que garanticen una reducción en la función de pérdida se utiliza el **descenso por gradiente**. Sea $\mathcal{L}(y_k, y_{gt})$ la función de pérdida entre la predicción en la k-ésima iteración y_k y la etiqueta original y_{gt}[1],

[1]Se le llama y_{gt} por *ground-truth*, término recurrente a lo largo del documento



(a) Tasa de aprendizaje muy chica. (b) Tasa de aprendizaje muy grande.

Figura B.2: Gráficas de función de costo respecto a vector de pesos. Puede verse que la tasa de aprendizaje determina si se logra alcanzar la convergencia.

se define al gradiente de $\mathcal{L}$ como

$$\nabla_w \mathcal{L}(y_k, y_{gt}) = \frac{\partial \mathcal{L}}{\partial w}(y_k, y_{gt}) = \left(\frac{\partial \mathcal{L}}{\partial w_0}, \frac{\partial \mathcal{L}}{\partial w_1}, ..., \frac{\partial \mathcal{L}}{\partial w_n} \right). \tag{B.3}$$

El descenso por gradiente es un método iterativo que consiste en aproximarse a un mínimo local de la función moviéndose en la dirección opuesta al gradiente. Esto garantiza que el valor de la función efectivamente disminuye. Matemáticamente puede verse como

$$w_{k+1} = w_k - \delta \nabla_w \mathcal{L}(y_k, y_{gt}) \tag{B.4}$$

donde k es el paso de iteración, y δ es la **tasa de aprendizaje**. Dependiendo de que tan grande o chico sea la tasa de aprendizaje pueden ocurrir los escenarios de la Figura B.2, donde puede verse que una tasa de aprendizaje muy grande puede llegar a provocar que los pesos se alejen de su valor óptimo en cada iteración, y una tasa de aprendizaje muy chica puede implicar no alcanzar la convergencia o alcanzarla muy lentamente.

Existen distintas variaciones del descenso por gradiente, dependiendo de la frecuencia con la que se actualicen los pesos de la red. Para el caso en que los pesos son actualizados por cada dato (en este caso imagen) analizado, se dice que se esta utilizando el **descenso por gradiente estocástico**. Esta implementación permite observar de forma inmediata el rendimiento de la red y en algunos casos puede llegar a buenos resultados más rápido, pero implica un costo computacional muy grande cuando se trabaja con muchos datos.

Otra variante es el ***batch gradient descent*** o **descenso por gradiente de lotes**, el cual actualiza los pesos de la red después de analizar todos los datos de entrenamiento. Esto da como resultado menos costo computacional y una convergencia más estable en la mayoría de los casos, pero el hecho de analizar todo el conjunto de entrenamiento antes de cada actualización puede llevar a un tiempo de cómputo innecesariamente alto para alcanzar la convergencia cuando se tienen muchos datos. Un ciclo a través de todo el conjunto de datos de entrenamiento se denomina **época** de entrenamiento. Por lo tanto, a menudo se dice que el descenso de gradiente de *batch* realiza actualizaciones del modelo al final de cada época de entrenamiento [68].

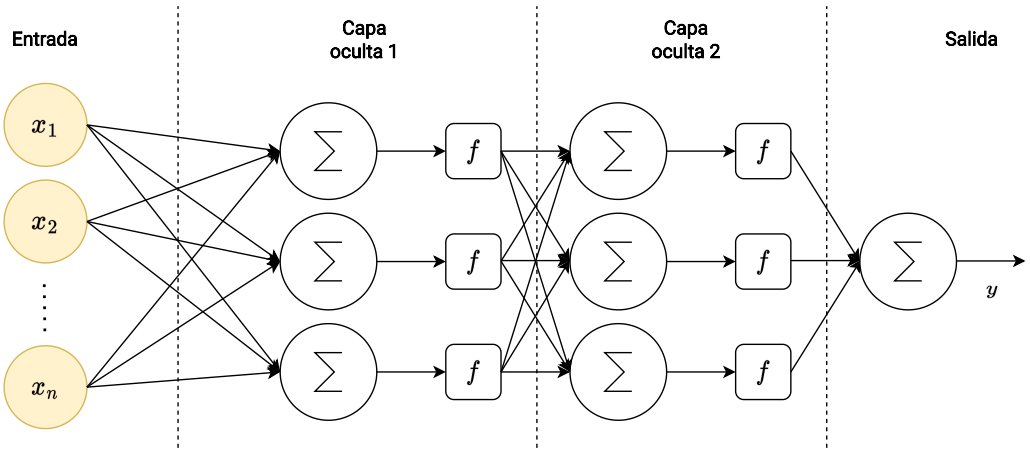


Figura B.3: Arquitectura de perceptrón multicapa. En este caso se tienen dos capas ocultas, cada una con tres neuronas.

Finalmente, el ***minibatch gradient descent*** o **descenso por gradiente de minilotes** es un método que divide al conjunto de entrenamiento en subconjuntos, cada uno de estos llamado ***minibatch***, y actualiza los pesos de la red luego de analizar los datos de cada *minibatch*. Dado que el tamaño del *minibatch* es una parámetro de entrenamiento, este puede ser elegido de tal forma que se tenga un equilibrio entre la robustez del descenso de gradiente estocástico y la eficiencia del descenso de gradiente de *batch*.

Todos los conceptos mencionados pueden extenderse al **perceptrón multicapa**, donde se tienen múltiples capas de neuronas. En la Figura B.3 puede verse un perceptrón multicapa con 2 capas intermedias, normalmente llamadas capas ocultas, cada una con 3 neuronas. Como fue mencionado anteriormente, cada neurona de la red cuenta con una función de activación.

Para los casos en los que se tienen múltiples capas se utiliza ***backpropagation*** [69] o **propagación hacia atrás**. Este método consiste en hallar los gradientes a partir de la regla de la cadena. Luego de realizar una pasada "*forward*" (de las entradas a la salida), se calculan los gradientes locales (evaluados en un punto) de cada neurona realizando una pasada "*backward*" (de la salida a las entradas) de la red. Al hacer gradientes locales utilizando la regla de la cadena, este paso logra ser rápido y preciso [43]. Un ejemplo de *backpropagation* puede verse en la lectura de la Universidad de Stanford [70], donde se explica paso a paso como funciona la propagación hacia atrás.

Red neuronal convolucional

Existen otros tipos de neuronas además del perceptrón, que realizan otro tipo de operaciones. Por ejemplo, si se tiene una red cuyas neuronas no operan haciendo el producto entre dos vectores, sino que se hace una convolución, entonces se tiene una **red neuronal convolucional** (CNN por sus siglas en inglés). De hecho, debido a la correlación que existe entre píxeles o vóxeles en una imagen, las redes

neuronales convolucionales son muy usadas en el procesamiento de imágenes.

Al igual que el perceptrón multicapa, las redes neuronales convolucionales consisten en una capa de entrada, capas ocultas y una capa de salida. Cada capa cuenta con neuronas que operan con una convolución y una función de activación, normalmente la ReLU. La convolución es realizada entre la entrada de la neurona y un **núcleo** o ***kernel*** cuyos valores son aprendidos por la red de la misma forma que los pesos en el perceptrón. Luego de la función de activación se obtiene lo que se conoce como ***feature maps***, salidas de cada neurona. Estos *feature maps* no son considerados imágenes porque, como su nombre lo dice, luego de la convolución pasan a ser un mapeo de una o más características que aprende la red. En cada capa se puede especificar la cantidad de salidas que se quiere, o lo que es lo mismo, la cantidad de *kernels* que se utilizarán. Esto permite ir aumentando la cantidad de entradas a cada capa de la red, teniendo más *feature maps* para procesar.

En la convolución con imágenes 2D y 3D, el *kernel* se recorre toda la imagen generando un *feature map* de salida. Para entender este recorrido y la dimensión de salida del *feature map* es necesario definir ***stride*** y ***padding***.

El *padding* consiste en agregar píxeles o vóxeles en los bordes de la imagen, es utilizado para obtener un *feature map* de salida con la misma dimensión que la entrada. Existen diversas técnicas, aunque en este caso simplemente se uso *zero-padding*, el cual consiste en agregar filas y columnas con ceros, como puede verse en la Figura B.4.

El *stride* o "salto" indica de cuantos píxeles o vóxeles es el desplazamiento del *kernel* al recorrer la imagen en la convolución. Podría haber un stride para desplazamiento vertical distinto al stride de desplazamiento horizontal, pero no es el caso en este trabajo. Un emeplo de *stride* puede verse en la Figura B.5, donde puede verse que este vale uno en ambas direcciones.

La dimensión de salida (O) de una neuronal convolucional queda definida por la dimensión de la entrada (I), la dimensión del *kernel* (F) de la neurona, el *stride* (S) y el *padding* (P), siguiendo la siguiente ecuación B.5.

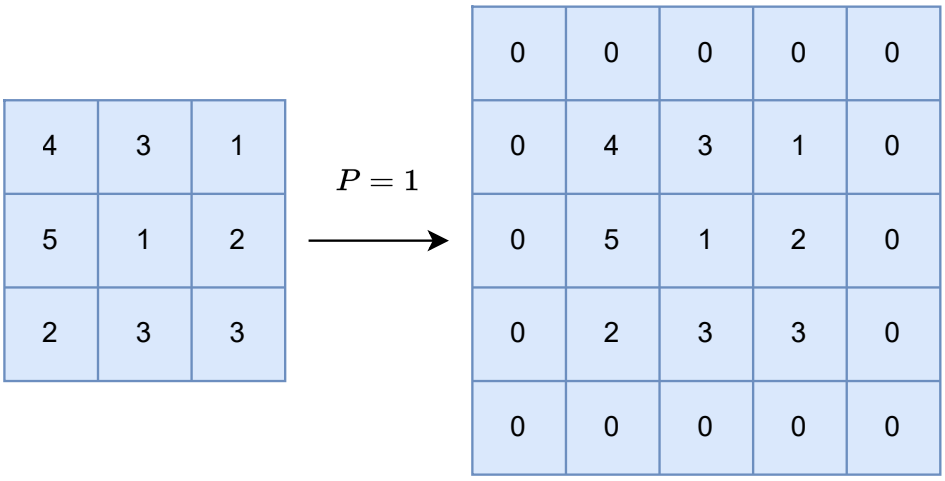


Figura B.4: Imagen original e imagen luego de hacer *zero-padding*.

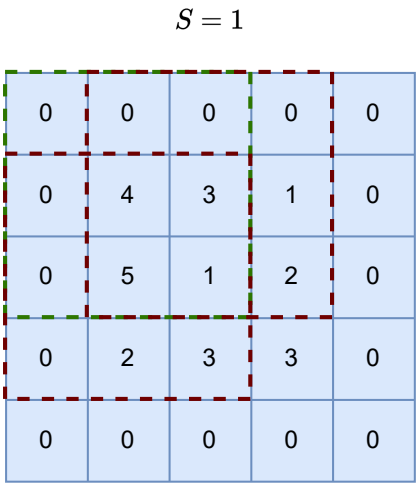


Figura B.5: Imagen *paddeada* donde la linea punteada verde representa el *kernel* en la posición inicial, y las lineas punteadas rojas muestran el desplazamiento vertical y horizontal, siendo de un pixel en ambos casos.

$$O = \frac{I - F + 2P}{S} + 1 \tag{B.5}$$

Al estar realizando una convolución, existe interacción espacial entre los píxeles o vóxeles vecinos. Esto da la posibilidad de reducir la dimensionalidad de la salida, dado que cada píxel o vóxel de salida tiene información de sus vecinos. Es por esto que las redes convolucionales tienen una etapa de ***pooling*** en la que se reduce la dimensionalidad de la salida. El método utilizado en este trabajo se llama ***maxpooling*** y será explicado en el Apéndice C.

Apéndice C

Funciones y conceptos importantes de U-Net/ResNet

ReLU, LReLU y PReLU

A diferencia de otras funciones de activación como la sigmoide o el arcotangente, ReLU no es exponencial, por lo que tiene un menor costo computacional. Esto la hace mejor candidata para redes neuronales, alcanzando mejor rendimiento [67].

$$ReLU(x) = max(x, 0). \tag{C.1}$$

Un problema que sufre esta función es que las entradas negativas generan una salida nula, lo que implica la posible muerte de neuronas. Para solucionar la muerte de neuronas es que se propone la *Leaky ReLU*(LReLU).

$$LReLU(x) = \begin{cases} 0{,}01x & x \leq 0 \\ x & x > 0. \end{cases} \tag{C.2}$$

Esta función, al no truncar los valores negativos se mitiga el efecto de la muerte de neurona pero al ser tan pequeño el gradiente para valores negativos, es factible el problema del vanishing gradient [67].

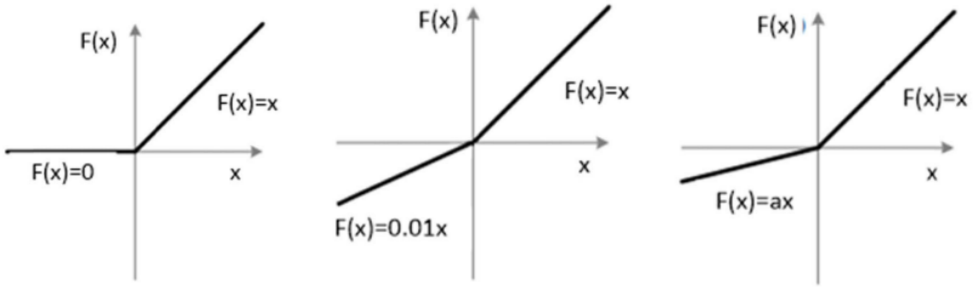
Otra variación es la *Parametric ReLU* [71], la cual recibe como entrada la pendiente de la recta que deja pasar valores negativos. Esto es

$$PReLU(x) = \begin{cases} ax & x \leq 0 \\ x & x > 0. \end{cases} \tag{C.3}$$

Nótese que para $a = 0$ y $a = 0{,}01$ PReLU es equivalente a ReLU y LReLU, respectivamente. Para este trabajo se utiliza la PReLU con $a = 0{,}2$. En la Figura C.1 pueden verse las tres funciones de activación.

Maxpooling

Formalmente, la función de *pooling* reduce progresivamente el tamaño espacial de los feature maps para reducir la cantidad de parámetros y costo computacional.



(a) ReLU (b) LReLU (c) PReLU

Figura C.1: Gráficas de ReLU, LReLU y PReLU [72]

El *maxpooling* básicamente toma el máximo valor de intensidad en una ventana de cierto tamaño, como lo indica la Figura C.2. Nótese que las dimensiones de la imagen se reducen a la mitad con una ventana 2×2, para imágenes 3D la ventana es de tamaño $2 \times 2 \times 2$, incorporando profundidad. Para este caso se quiere evitar repetir píxeles o vóxeles, por lo que el tamaño de la ventana de *maxpooling* es igual al *stride*. En este caso se tiene $S = 2$.

Sobremuestreo o "Up-Sampling"

Comúnmente llamado "upsampling" en inglés, el sobremuestreo de la imagen es fundamental dado que permite la decodificación de la información. Puede verse como que la codificación brinda información de la imagen a través de distintos *feature maps*, y la decodificación revela dónde está dicha información.

Existen muchas técnicas de sobremuestreo tales como vecino más cercano, interpolación bilineal, interpolación de ceros, unpooling o convolución transpuesta [1]. Esta última consiste en un filtro convolucional cuyos pesos son aprendidos por la red, con un tamaño y stride tales que la salida del filtro tiene mayor dimensión que la entrada. Esto puede visualizarse en la Figura C.3, donde una entrada de dos elementos devuelve una salida de cinco. Generalizando, la dimensión de salida

[1] También llamada deconvolución en algunas fuentes, no debe confundirse esta operación con la etapa de deconvolución del Capítulo 3

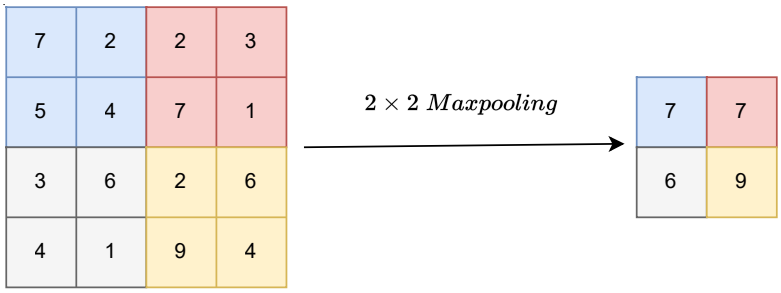


Figura C.2: Maxpooling con ventana de 2×2.

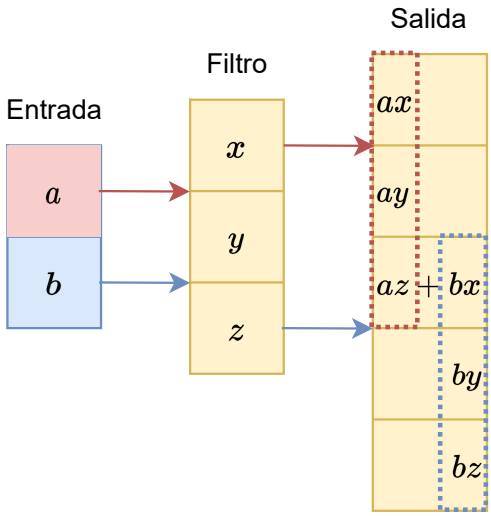


Figura C.3: Ejemplo de una convolución transpuesta en un arreglo de dimensión 1, donde la dimensión de salida esta definida por la dimensión de la entrada (2), la dimensión del filtro (3) y el stride (2)

cumple la Ecuación (C.4), donde S es el stride e I, O y F son las dimensiones de la entrada, la salida y el filtro convolucional, respectivamente.

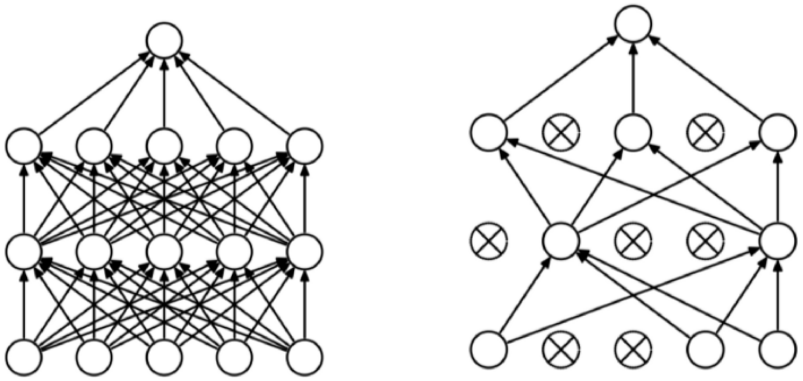
$$O = (I - 1)S + F \tag{C.4}$$

Skip-connections

Representadas con flechas grises en la Figura 4.1, las *skip-connections* o conexiones de salto son las que permiten capturar al detalle la información codificada. Como su nombre lo sugiere, consisten en pasar por alto algunas neuronas, evitando así el *vanishing gradient* en redes profundas.

En redes profundas, al aplicar *backpropagation* es muy común observar *vanishing gradient* debido a que, en general, las derivadas parciales dan valores menores a 1 en valor absoluto. Las *skip-connectios* proveen un camino alternativo para el gradiente, evitando que se desvanezca y así logrando actualizar los pesos correctamente.

En la U-Net, para cada nivel de la red se concatenan los *feature maps* de la etapa de expansión con los *feature maps* del mismo nivel de la etapa de contracción, y todos son convolucionados y sobremuestreados al nivel anterior. De esta manera no solo se evita el *vanishing gradient*, sino que además no hay pérdida de información en la decodificación gracias al hecho de estar usando los *feature maps* de la etapa de contracción.



(a) Red neuronal estándar

(b) Red neuronal después de aplicar dropout

Figura C.4: Ejemplo de red neuronal antes y después de aplicar dropout [74]

Dropout

Durante el entrenamiento, pone a cero aleatoriamente algunos de los pesos con probabilidad p utilizando muestras de una distribución de Bernoulli(p). Cada elemento se pondrá a cero de forma independiente. Se ha demostrado que estas técnica es eficaz para regularizar las redes neuronales [73]. Además, las salidas están escaladas por un factor de $\frac{1}{1-p}$ durante el entrenamiento. En la Figura C.4 puede verse un ejemplo de un perceptrón multicapa sin *dropout* (Figura C.4a) y con *dropout* (Figura C.4b).

Batch Normalization

Consiste en normalizar los *mini-batches* con los que trabaja la red, restándoles la media y dividiéndolos por la desviación estándar del conjunto. Esto estabiliza y agiliza la red, reduciendo el costo computacional y reduciendo el número de épocas necesarias en el entrenamiento [75].

U-Net

Como ya se dijo, la U-Net cuenta con dos vías, contracción y expansión. La contracción cuenta con capas convolucionales y de *pooling*, las cuales permiten crear una variedad de *feature maps* con un tamaño razonable para disminuir la cantidad de parámetros de la red. Entre estas dos capas esta la función de activación de cada neurona, la cual es la PReLU con $a = 0{,}2$ para tener la eficiencia de la ReLU en entrenamiento mientras se evita la muerte de neuronas.

Muchos *feature maps* con poca dimensión se entienden como información codificada que será decodificada en la vía expansiva, es por esto que el modelo es considerado un *encoder-decoder*.

En la expansión se produce el efecto contrario, se disminuyen la cantidad de *feature maps* y sube la dimensión de los mismos a medida que se avanza en la red,

obteniendo finalmente una única salida de la misma dimensión de la entrada. Para tener buenos resultados a la salida no es suficiente con codificar la información en *feature maps* pequeños, debido a que al disminuir la dimensión de estos se pierden detalles. Es por esto que las *skip connectios* forman parte de la arquitectura, permitiendo que no se pierda ningún tipo de información y así, obtener buenos resultados de segmentación.

Finalmente, se tiene una red convolucional que codifica la imagen de entrada en distintos *feature maps* de un tamaño relativamente pequeño para reducir la cantidad de parámetros, y decodifica esta información para lograr clasificar correctamente cada píxel o voxel de la imagen.

Este proceso se repite las veces que se consideren necesarias (tres en este trabajo) hasta llegar al "cuello de botella", etapa en la que la que se tiene la mayor cantidad de *feature maps* con el menor tamaño utilizado. En este nivel nuevamente se realizan dos convoluciones con una ventana de 3x3x3 y se empieza la decodificación. Para decodificar se hacen *up-sampling* de los *feature maps*, *skip-connections* con las *feature maps* del camino de contracción y se vuelven a hacer dos convoluciones con PReLU. Por último, se realiza una convolución final para devolver imágenes del mismo tamaño de las imágenes de entrada. Esta arquitectura y sus variaciones son ampliamente utilizadas en el ámbito biomédico [9].

ResNet

Antes de que esta arquitectura fuera propuesta en 2015 ganando el *ImageNet Large Scale Visual Recognition Challenge* [76], era difícil el entrenamiento de redes neuronales profundas de muchas capas. Un ejemplo de esto puede verse en la Figura C.5, donde se visualiza el error tanto de entrenamiento y testeo en función de las épocas para dos redes convolucionales con distinta cantidad de capas. El hecho de que el error en el conjunto de test sea mayor para la red con más capas puede indicar sobreajuste, pero al ver el error en el entrenamiento se puede apreciar que también es mayor en la red con mayor profundidad, descartando la posibilidad del sobre ajuste. Esto implica que las capas adicionales sufren un problema de aprendizaje tan severo que no logran ni siquiera aprender a no hacer nada, perjudicando el desempeño de la red anterior.

Lo que propone la red residual, también llamada ResNet [55], es aprender un mapeo de la función residual $\mathcal{F}$(x) = $\mathcal{H}$(x) - x, donde $\mathcal{H}(x)$ es la función objetivo. Para esto utiliza *skip connections* en el entrenamiento que, como fue mencionado anteriormente, implica que los pesos que "saltan" de una capa a otra se vean menos degradados en la propagación hacia atrás.

En la Figura C.6 puede verse un diagrama de flujo con la lógica de las redes residuales. A esto se le conoce como *Resblock* o bloque residual.

Las redes residuales no solo permiten agregar más capas a la red, sino que por su arquitectura son capaces de reducir considerablemente el tiempo de entrenamiento en algunos casos [43].

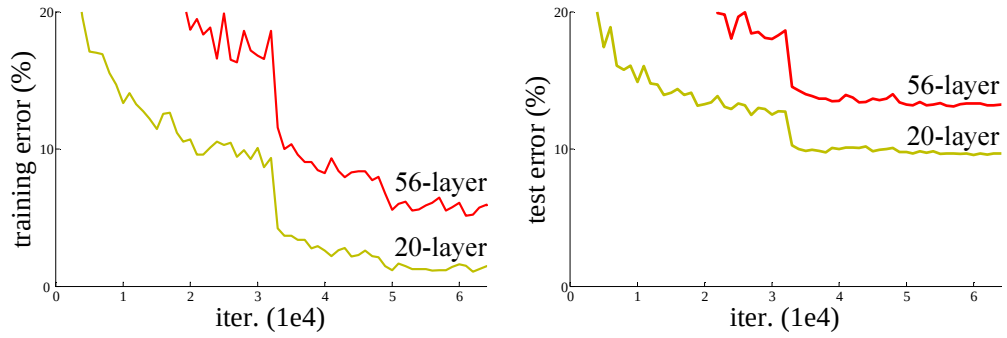


Figura C.5: Gráficas de la función de pérdidas en cada época para una red de 20 capas y otra de 56, ambas con la misma arquitectura [55]

U-Net/ResNet

La arquitectura utilizada para la segmentación es una combinación de la *U-Net* y la *ResNet*, siendo llamada *U-Net/ResNet*. Esta arquitectura cuenta con *skips-connections* largas, dadas por la arquitectura *U-Net*, y cortas, dadas por la *ResNet*. Esto no solo evita la pérdida de información, sino que además permite capturar mejores detalles de la imagen original.

Como se puede ver en la Figura C.7, cada nivel de la etapa de contracción y expansión de la red es un bloque residual, dado que tienen el mismo formato. Más específicamente son bloques residuales convolucionales, porque dentro de cada *skip connection* se tiene una convolución y *batch normalization*, esto es para no tener problemas de dimensiones y/o cantidades de *feature maps* (en este caso solo aumenta la cantidad de *feature maps*, dado que el *padding* y *stride* fueron elegidos de tal forma que la salida tenga las mismas dimensiones que la entrada). La Figura C.6 muestra cómo fueron implementados los bloques residuales convolucionales utilizados en esta arquitectura.

Finalmente, es importante aclarar que la red de este trabajo no es exactamente

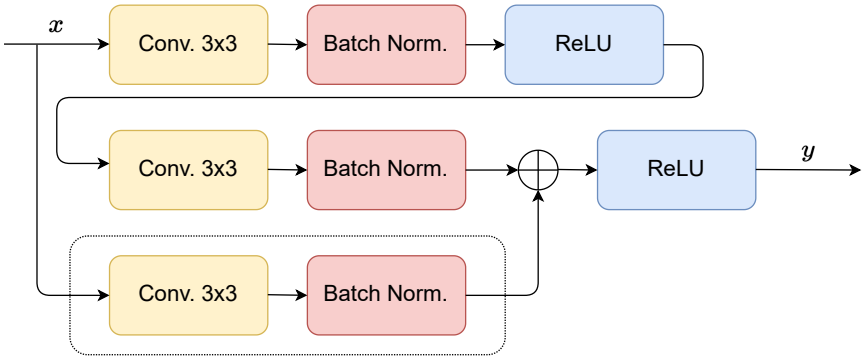


Figura C.6: Arquitectura de bloque residual convolucional. Nótese que la convolución en la *skip connection* (recuadrada con línea punteada) está únicamente para tener la misma cantidad de *feature maps* en la suma.

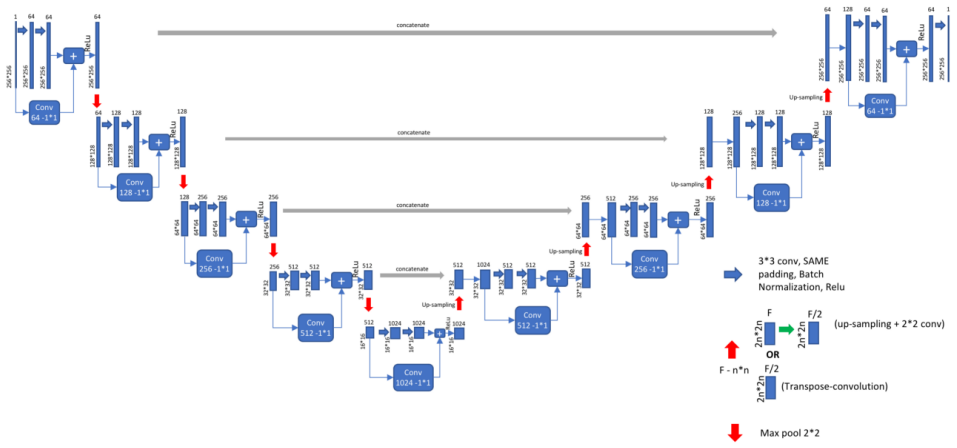


Figura C.7: Arquitectura U-Net/Resnet, donde cada nivel de contracción y expansión es un bloque residual. La convolución realizada en la *skip connection* está únicamente para que ambas entradas a la suma tengan las mismas dimensiones [56].

igual a la de la Figura C.7, dado que sus bloques residuales convolucionales cuentan con tres etapas de activación, las cuales se dan:

- Después de la primera convolución y *batch normalization*.
- Después de la segunda convolución y *batch normalization*.
- Después de la suma.

En la red utilizada se respetan los bloques de la Figura C.6, con activación:

- Después de la primera convolución y *batch normalization*.
- Después de la suma.

Esta página ha sido intencionalmente dejada en blanco.

Apéndice D

Conceptos básicos de morfología matemática

Para explicar ciertas operaciones morfológicas es fundamental definir los **elementos estructurales** o ***structuring elements (SEs)***. Éstos son subimágenes o ventanas utilizadas para estudiar propiedades en imágenes. Varios ejemplos de SEs pueden verse en la primera fila de la Figura D.1, mientras que la segunda fila convierte estos SEs a una matriz. Nótese que estas operaciones son aplicadas en imágenes binarias.

La mayoría de las operaciones morfológicas están basadas en dos operaciones básicas, la **erosión** y la **dilatación**.

La erosión de una imagen A por un SEs B da como resultado el conjunto de puntos (x, y) tales que $B_{(x,y)} \subseteq A$. Formalmente se define como

$$A \ominus B = \{(x, y) | B_{(x,y)} \subseteq A\}. \tag{D.1}$$

Por otro lado, la dilatación de una imagen A por una SEs B da como resultado el conjunto de puntos (x, y) donde la intersección entre A y $B_{(x,y)}$ no sea nula. Esto es

$$A \oplus B = \{(x, y) | A \cap B_{(x,y)} \neq 0\}. \tag{D.2}$$

La Figura D.2 muestra ejemplos de erosión y dilatación de una imagen. Es importante observar que por sus definiciones, la erosión es útil para eliminar píxeles de información (*foreground*) aislados en una imagen, mientras que la dilatación puede utilizarse para eliminar píxeles del fondo (*background*) que estén aislados.

De aplicar estos operadores con cierto orden se definen las operaciones ***opening*** y ***closing***. El *opening* consiste en aplicar primero erosión y luego dilatación, mientras que el *closing* es el caso inverso. Esto es

$$A \circ B = (A \ominus B) \oplus B, \tag{D.3}$$

$$A \bullet B = (A \oplus B) \ominus B \tag{D.4}$$

donde la ecuación D.3 es la operación que define el *opening* y la ecuación D.4 es define el *closing*.

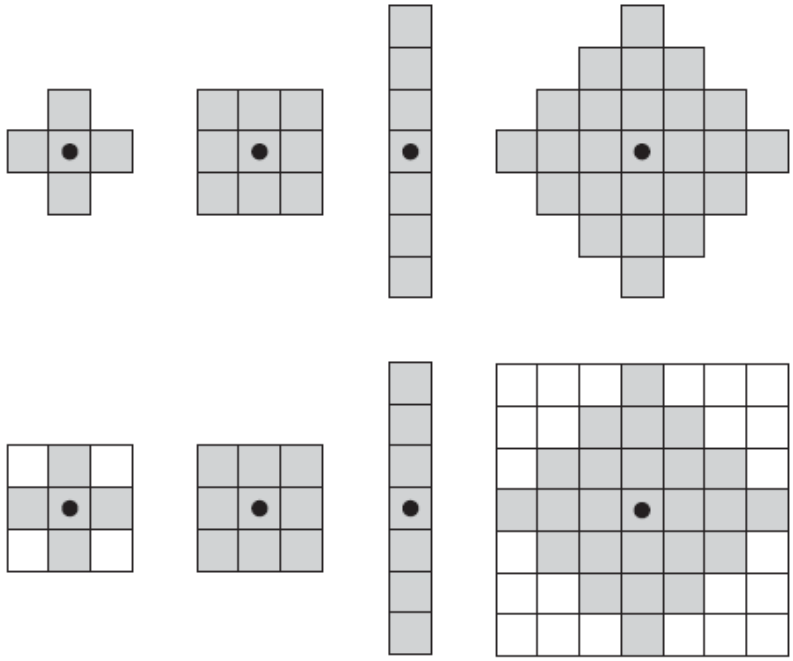


Figura D.1: Ejemplos de elementos estructurales (primera fila) y elementos estructurales pasados a matrices (segunda fila). El punto representa el centro de los SEs [54].

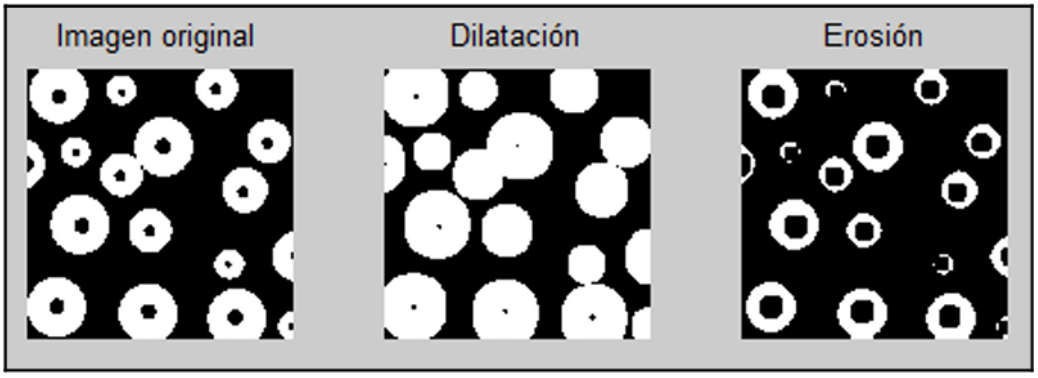


Figura D.2: Ejemplo de erosión y dilatación de una imagen, tomada de [77].

Una característica que puede ser expresada en términos de erosiones y *openings* es el **esqueleto** de una imagen [54]. En 2D, el esqueleto puede entenderse como el lugar geométrico de los centros de las circunferencias que son tangentes al contorno de la estructura en al menos dos puntos. La Figura D.3 muestra los esqueletos de distintas estructuras, y algunas circunferencias tangentes a los contornos centradas en distintos puntos de los esqueletos.

Otro concepto importante y complementario al de esqueleto es el de *pruning*. Éste consiste en remover ramificaciones parásitas que no brindan información re-

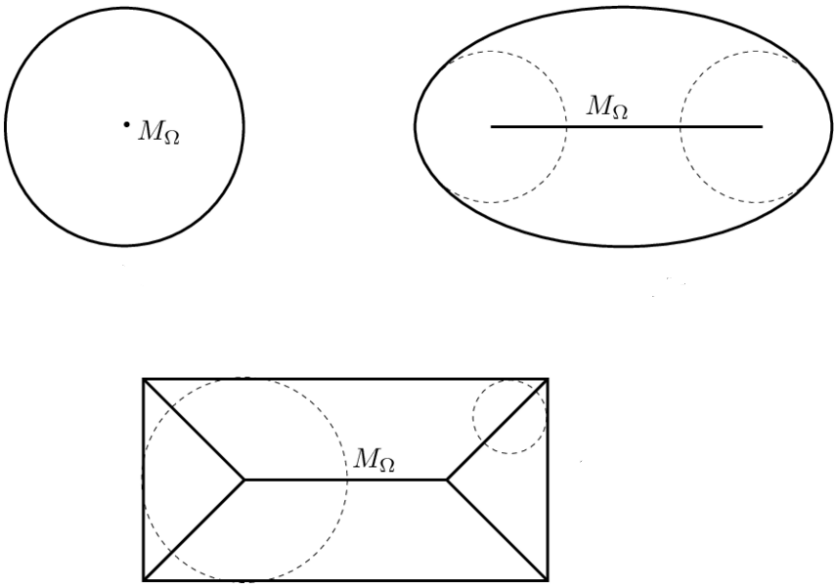


Figura D.3: Distintas figuras geométricas con sus esqueletos (M_Ω). Las circunferencias muestran la propiedad geométrica que cumplen los puntos del esqueleto.

levante y deben ser eliminadas para una mejor caracterización. Para definir el *pruning* matemáticamente es necesario definir algunas operaciones. La primera es la transformación *Hit-or-Miss*, definida como

$$A \circledast B = (A \ominus B_1) \cap (A^C \ominus B_2) \tag{D.5}$$

donde A^C representa el complemento de A y $B = (B_1, B_2)$ siendo B_1 el conjunto de puntos del SEs asociados al *foreground* y B_2 los puntos asociados al *background*.

Otra operación necesaria para definir el *pruning* es *thinning*, definida como

$$A \otimes B = A - (A \circledast B). \tag{D.6}$$

La explicación de *Hit-or-Miss* y *thinning* [54] no se detallará en este trabajo, debido a que estos se presentan únicamente para poder dar una definición matemática del *pruning*.

Finalmente, el *pruning* se logra realizando los siguientes pasos:

1. Aplicar *thinning* n veces para eliminar cualquier ramificación de n o menos píxeles.
$$X_1 = A \otimes B \tag{D.7}$$
2. $$X_2 = \bigcup_{i=1}^{8} (X_1 \circledast B_i) \tag{D.8}$$
donde
$$B_1 = \begin{pmatrix} \times & 0 & 0 \\ 1 & 1 & 0 \\ \times & 0 & 0 \end{pmatrix} \; B_2 = \begin{pmatrix} \times & 1 & \times \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \; B_3 = \begin{pmatrix} 0 & 0 & \times \\ 0 & 1 & 1 \\ 0 & 0 & \times \end{pmatrix} \; B_4 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ \times & 1 & \times \end{pmatrix}$$

$$B_5 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \; B_6 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \; B_7 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \; B_8 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

Nótese que $\times$ es notación para "no importa", pudiendo haber un 1 o un 0 indistintamente en esa posición.

3. Realizar la dilatación usando una matriz de 3×3 (H) que consta de todos los 1's y solo inserta 1 donde la imagen original (A) también tenía 1.

$$X_3 = (X_2 \oplus H) \cap A \tag{D.9}$$

4. Tomar el resultado del paso 1 y realizar la unión con el paso 3 para lograr los resultados finales.

$$X_4 = X_1 \cup X_3 \tag{D.10}$$

Apéndice E

Conceptos relevantes sobre la Teoría de grafos

Un grafo es un objeto matemático conformado por un conjunto de puntos, denominados vértices o nodos, que están relacionados mediante uniones llamadas aristas o *links*. Un grafo se designa por $G = (V, E)$, siendo V el conjunto de nodos y $E \subset V \times V$ es un conjunto que contiene las parejas de nodos conectados. Los nodos se enumerarán y se denotarán por n_i, siendo i el número de nodo.

Definición 1 *Sea V un conjunto finito, no vacío y sea $E \subset V \times V$. El par (V, E) es un* grafo dirigido*, donde V es el conjunto de vértices o nodos y E es su conjunto de aristas. Dicho grafo se denota por $G = (V, E)$ [78].*

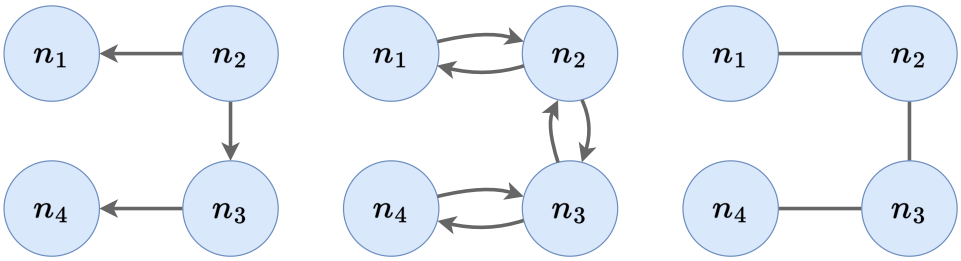
Definición 2 *Se dice que un grafo es* no dirigido *si el conjunto de aristas E es simétrico, esto es, $(n_i, n_j) \in E \Leftrightarrow (n_j, n_i) \in E$.*

Las aristas pueden ser direccionadas, cuando el par está ordenado, o no direccionadas, cuando son definidas por un par no ordenado de nodos. En las Figuras E.1a y E.1b se presenta un ejemplo de grafo dirigido y no dirigido respectivamente, de acuerdo con las definiciones 1 y 2. En un grafo no dirigido, la notación del conjunto E se puede simplificar tomando $\{n_i, n_j\}$ como el conjunto que contiene ambas combinaciones posibles: $\{(n_i, n_j), (n_j, n_i)\}$ y la visualización de dicho grafo puede simplificarse como en la Figura E.1c.

Un grafo puede tener lazos o bucles, esto es, una arista que relaciona un nodo con sí mismo. Además, puede tener más de una arista que une la misma pareja de nodos. En la Figura E.2 se presenta un ejemplo de un grafo de 4 nodos con multiaristas y un lazo.

Definición 3 *Se denomina* grafo simple *a aquél que no posee bucles o aristas múltiples.*

Definición 4 *Se denomina* grafo vacío *a aquél donde E es el conjunto vacío, $E = \emptyset$. No existen aristas en el grafo, ningún nodo está conectado con otros.*



(a) Grafo dirigido G_A (b) Grafo no dirigido G_B (c) Grafo no dirigido G_B simplificado

Figura E.1: Representación de un grafo dirigido y un grafo no dirigido, con 4 nodos. (a) Grafo $G_A = (V_A, E_A)$ con $V_A = \{n_1, n_2, n_3, n_4\}$ y $E_A = \{(n_2, n_1), (n_2, n_3), (n_3, n_4)\}$. G_A es un grafo dirigido. (b) Grafo $G_B = (V_B, E_B)$ con $V_B = \{n_1, n_2, n_3, n_4\}$ y $E_B = \{(n_1, n_2), (n_2, n_1), (n_2, n_3), (n_3, n_2), (n_3, n_4), (n_4, n_3)\}$. El grafo G_B es no dirigido. (c) Visualización simplificada del grafo no dirigido G_B. En un grafo no dirigido, el conjunto E se puede simplificar como $E_B = \{\{n_1, n_2\}, \{n_2, n_3\}, \{n_3, n_4\}\}$.

Definición 5 *Sea el conjunto V con N elementos $V = \{n_1, n_2, \ldots, n_N\}$ y sea $E = \{(n_i, n_j) : \forall i, j \in [1, N], i \neq j\}$. Es decir, cada nodo está relacionado con todos los demás, salvo consigo mismo. El grafo $G = (V, E)$ se denomina* grafo completo *de N vértices.*

Definición 6 *Sea V con N elementos, $N \geq 3$. Se denomina* grafo ciclo *de N vértices a un grafo $C_N = (V, E)$ tal que sus nodos están conectados en forma cíclica.*

En las Figuras E.3a y E.3b se presenta un ejemplo de un grafo completo y un grafo cíclico respectivamente, de 6 vértices, siguiendo las definiciones 5 y 6. En la Figura E.3c se presenta un grafo cíclico de 3 nodos. Notar que en este caso el grafo cíclico es también un grafo completo.

De aquí en adelante al referirse a un grafo se asumirá un grafo simple, no dirigido. Se dice que dos vértices x_0, x_1 están conectados si existe un camino de x_0 a x_1, entendiendo camino según la definición 7.

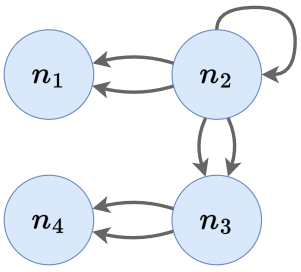
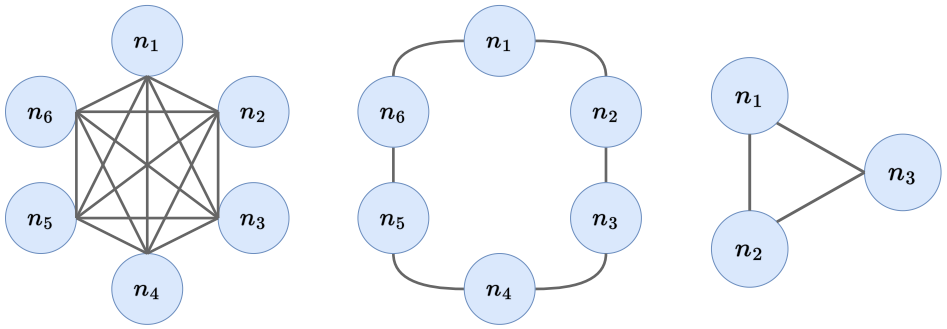


Figura E.2: Grafo $G = (V, E)$ con 4 nodos. El grafo presenta aristas múltiples, existe más de una arista con la misma dirección que une dos nodos. Además, presenta un lazo o bucle en el nodo n_2.



(a) Grafo completo de 6 vértices. (b) Grafo cíclico de 6 vértices. (c) Grafo cíclico y completo de 3 vértices.

Figura E.3: (a) Representación de un grafo completo de 6 vértices según la definición 5. (b) Representación de un grafo cíclico de 6 vértices, según la definición 6. (c) Representación de un grafo cíclico de 3 vértices. Notar que este grafo es además un grafo completo.

Definición 7 *Sean $x_0, x_1 \in V$. Se define un* camino de longitud r *de x_0 a x_1 como una sucesión de vértices $(n_1, n_2, \ldots, n_r)$ tal que $n_1 = x_0$ y $n_r = x_1$ y $(n_i, n_{i+1}) \in E$ (se tiene una arista de G para pasar de n_i a n_{i+1}). La longitud r es la cantidad de aristas que se recorren al pasar de x_0 a x_1.*

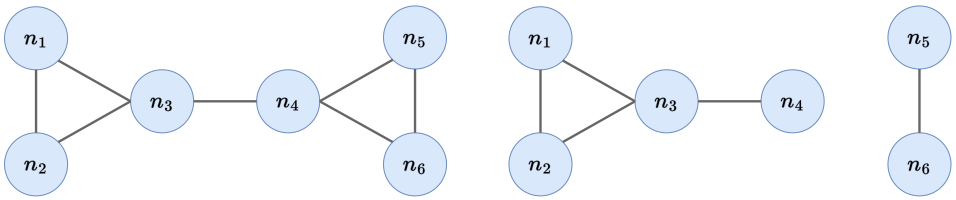
En la Figura E.4 se muestra un grafo conexo y uno no conexo, según la definición 8. En la Figura E.5 se muestra un grafo conexo con dos caminos resaltados en amarillo para ejemplificar cómo se mide la longitud del camino.

Definición 8 *Un grafo G es* conexo *si para todo par de vértices $(n_i, n_j) \in V$ existe un camino de n_i a n_j.*

Definición 9 *Un* subgrafo G' *de un grafo $G = (V, E)$ es un par $G' = (V', E')$ que verifica que el conjunto de vértices $V' \subset V$ y el conjunto de aristas $E' \subset E$. Además, para que G' sea un grafo se debe verificar $E' \subset V' \times V'$.*

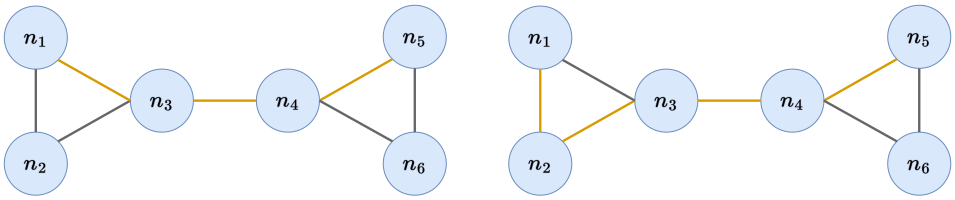
Definición 10 *Sea $G = (V, E)$, el* grado de un vértice $x \in V$, *es la cantidad de aristas incidentes al vértice x. Como no se tienen multiaristas, se puede determinar como:*

$$gr(x) = \#\{y \in V : \{x, y\} \in E\}$$



(a) Grafo conexo. (b) Grafo no conexo.

Figura E.4: (a) Grafo *conexo*; para todo par de vértices $(n_i, n_j) \in V$ existe un camino de n_i a n_j. (b) Grafo *no conexo*; no existe un camino entre los vértices n_5 o n_6 y el resto de los vértices.



(a) Camino de longitud 3.

(b) Camino de longitud 4.

Figura E.5: Grafo G conexo, de 6 nodos. En amarillo se indican posibles caminos entre los nodos n_1 y n_5. La longitud del camino es la cantidad de aristas que se recorren al pasar de n_1 a n_5.

Otro concepto importante en Teoría de Grafos es el de comunidad. Una comunidad puede entenderse como un subconjunto de nodos que están densamente conectados entre sí y débilmente conectados con los demás nodos del grafo. Este concepto es un tanto ambiguo puesto que la caracterización de una conexión como densa o débil no es directa. Existen varios métodos de detección de comunidades, que difieren en el criterio que utilizan para definir si las conexiones son débiles o no. En particular, se trabajará con el algoritmo de Louvain, el cual será presentado en la sección 5.3.2.

Apéndice F

Cotas de modularidad

Aquí se presentan los cálculos realizados para determinar las cotas superior e inferior del parámetro modularidad.

$$Q = \frac{1}{2m} \sum_{ij} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \delta(c_i, c_j) \tag{F.1}$$

Para el desarrollo de la ecuación se obviará el factor δ y se evaluará más adelante su influencia.

$$Q = \frac{1}{2m} \sum_{ij} A_{ij} - \frac{1}{2m} \sum_{ij} \frac{k_i k_j}{2m} \tag{F.2}$$

$$= \frac{1}{2\frac{1}{2} \sum_{ij} A_{ij}} \sum_{ij} A_{ij} - \frac{1}{(2m)^2} \sum_{ij} k_i k_j \tag{F.3}$$

$$= 1 - \frac{1}{(2m)^2} \sum_{ij} k_i k_j \tag{F.4}$$

Recordando la definición de k_i, este valor se corresponde con la suma de los pesos de todas las aristas conectadas al nodo i. Puesto que en nuestro caso los pesos se asocian a un valor de distancia, van a ser siempre valores positivos. Por lo tanto, en (F.4) se observa que el término que queda restando será siempre mayor a cero. En consecuencia, se obtiene que $Q \leq 1$.

Analizando el término $\frac{1}{(2m)^2} \sum_{ij} k_i k_j$ se logra restringir aún más la cota superior. La modularidad es máxima cuando cada comunidad está formada por el mínimo posible de nodos y todos los pesos son iguales. Es decir, se considerará el caso que las subredes están formadas por dos nodos que se conectan únicamente entre sí y la arista que los une tiene peso p.

Se tiene entonces que $k_i = p \; \forall i$ y $m = \frac{1}{2} N p$, siendo N el número total de nodos en el grafo. Sustituyendo en (F.4), se llega a la cota superior de Q (F.8). Aquí entra en consideración el término δ, al determinar cuáles términos aportan a la sumatoria en (F.5). En este caso, por cada comunidad se tienen dos nodos. Esto es, cuatro

términos por comunidad $(i_0i_0, i_0j_0, j_0i_0, j_0j_0)$. El número de comunidades es $\frac{N}{2}$. Por lo tanto, se tiene que $\sum_{ij} p^2 = \frac{N}{2}4p^2$.

$$Q = 1 - \frac{1}{(2\frac{1}{2}Np)^2}\sum_{ij} p^2 \tag{F.5}$$

$$= 1 - \frac{1}{N^2p^2}2Np^2 \tag{F.6}$$

$$= 1 - \frac{2}{N} \tag{F.7}$$

$$\Rightarrow \boxed{Q = \frac{\frac{N}{2} - 1}{\frac{N}{2}}} \tag{F.8}$$

De manera similar, se determina la cota inferior. El caso en el que la modularidad es mínima es cuando se tiene una única comunidad, en la que todos los nodos están conectados entre sí y todos los pesos son iguales. Esto implica $k_i = (N-1)p$ $\forall i$ y $m = \frac{1}{2}N(N-1)p$. Sustituyendo en (F.4) se llega a la cota inferior de Q (F.11). Nuevamente entra en consideración el término δ para determinar cuáles términos de la sumatoria en (F.9) contribuyen al cálculo de Q. En este caso se tiene una única comunidad con N nodos. Por cada nodo hay N términos. Por lo tanto, $\sum_{ij}[(N-1)p]^2 = N^2[(N-1)p]^2$.

$$Q = 1 - \frac{1}{(2\frac{1}{2}N(N-1)p)^2}\sum_{ij}[(N-1)p]^2 \tag{F.9}$$

$$= 1 - \frac{1}{N^2(N-1)^2p^2}N^2(N-1)^2p^2 \tag{F.10}$$

$$\Rightarrow \boxed{Q = 0} \tag{F.11}$$

Apéndice G

Conceptos básicos de Aprendizaje Automático

Correlación y Matriz de Correlación

La correlación es una medida que permite conocer la cantidad de información que se puede obtener de un cierto dato, utilizando otro dato. Habitualmente cuando hay referencias a la correlación, se utiliza el Pearson Correlation Coefficient (PCC), que es la correlación lineal entre dos conjuntos de datos. Esta correlación se puede expresar a través de la siguiente ecuación:

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y} \tag{G.1}$$

siendo X e Y variables aleatorias, $cov(X,Y)$ la covarianza de X e Y y finalmente σ_X y σ_Y las desviaciones estándar de X e Y respectivamente. En la Figura G.1 se muestra distintos resultados de la correlación de Pearson y posibles escenarios que generan esa correlación.

La correlación de Pearson no es la única, es habitual utilizar las correlaciones de Spearman y Kendall. Todas estas correlaciones tienen rango de valores de -1 a 1, con el mismo significado que Pearson, la diferencia es permiten buscar relaciones no lineales entre los datos, por ejemplo, Spearman encuentra qué tan bien se puede estimar la relación de dos variables aleatorias con una función monótona.

La matriz de correlación de n variables aleatorias $X_1, \ldots, X_n$ es la matriz con dimensiones $n \times n$ tal que la entrada en (i,j) es $corr(X_i, X_j)$ y los elementos de la diagonal son idénticamente uno. Debido a que $corr(X_i, X_j) = corr(X_j, X_i)$ es una matriz simétrica. En la Figura G.2 se muestra un ejemplo con una selección de parámetros del problema planteado.

Generalización

El problema fundamental que se busca abordar con el aprendizaje automático es la posibilidad de que las computadoras aprendan de datos sin ser explícitamente programadas. Esto quiere decir, lograr que el modelo se comporte de una manera

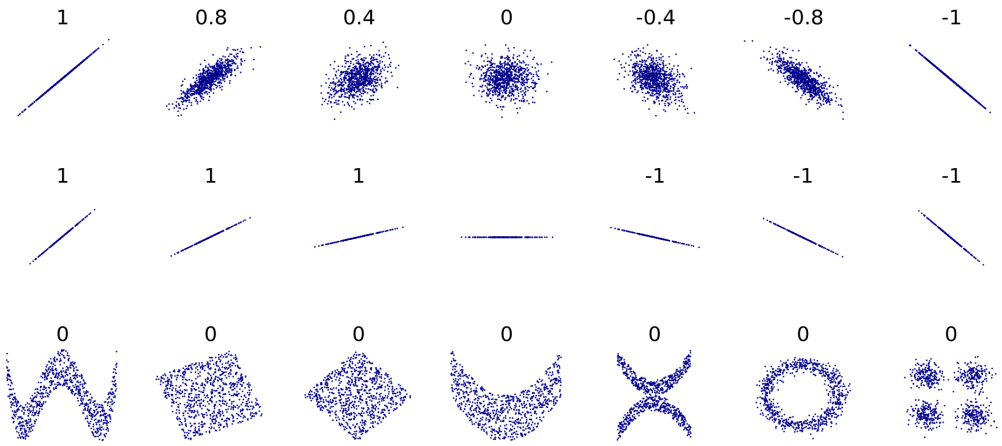


Figura G.1: La correlación de Pearson toma valores entre -1 y 1, donde los casos que es 0 indica una correlación nula. Los casos en los que son 1 o -1 indica correlación máxima, los datos tienen una dependencia lineal entre sí. En las imágenes de la fila inferior se visualiza claramente una cierta correlación entre los datos pero el coeficiente de Pearson es 0 porque no es una correlación lineal. Tomado de [43]

similar al conjunto de entrenamiento para datos nunca antes visto, este concepto se le conoce como Generalizar. En el aprendizaje automático el primer debate que se enfrenta es si es posible aprender de los datos, es decir la capacidad para generalizar. Para esto se hará referencia a algunas pruebas y resultados del libro de Yaser S. Abu-Mostafa, et al [79]. En el libro se plantea que desde un abordaje determinístico no es posible aprender, pero abordándolo desde la probabilidad es posible. En la Ecuación (G.2) se presenta la Ecuación de Hoeffding donde g es una hipótesis que pertenece al conjunto de todas las hipótesis posibles $\mathcal{H}$, $E_{in}(g)$ el error dentro de la muestra, $E_{out}(g)$ el error fuera de la muestra, N, la cantidad de datos en la muestra y ϵ un valor de tolerancia,

$$\mathbb{P}[|E_{in}(g) - E_{out}(g)| > \epsilon] \leq 2e^{-2e^2 N}, \forall \epsilon > 0. \tag{G.2}$$

A medida que crece la cantidad de muestras N, se reduce exponencialmente la probabilidad de que el error fuera de la muestra diverja del error dentro de la muestra con tolerancia ϵ. En el caso anterior se asumió una sola hipótesis en el conjunto $\mathcal{H}$ pero el caso que puede ser de interés es cuando este conjunto tiene una cantidad M finita de hipótesis. $\mathcal{H} = \{h_1, h_2, ...h_M\}$. En este caso no es que se desee que para todo $h_m \in \mathcal{H}$ la desigualdad de Hoeffding sea chica sino que para la última hipótesis (llamémosle g), la desigualdad de Hoeffding sea chica. Para acotar la probabilidad deseada se utilizan dos propiedades probabilísticas:

$$Si\, \mathcal{B}_1 \Rightarrow \mathcal{B}_2,\, entonces\, \mathbb{P}[\mathcal{B}_1] \leq \mathbb{P}[\mathcal{B}_2], \tag{G.3}$$

$$\mathbb{P}[\mathcal{B}_1 \textbf{ or } \mathcal{B}_2 \textbf{ or } \ldots \textbf{ or } \mathcal{B}_M] \leq \mathbb{P}[\mathcal{B}_1] + \mathbb{P}[\mathcal{B}_2] + \cdots + \mathbb{P}[\mathcal{B}_M], \tag{G.4}$$

siendo $\mathcal{B}_i$ un evento y donde $\mathcal{B}_1 \Rightarrow \mathcal{B}_2$ significa que $\mathcal{B}_1$ implica $\mathcal{B}_2$. Con las propiedades (G.3), (G.4) y asumiendo que $\mathbb{P}[\mathcal{B}_M]$ está acotado por la desigualdad de

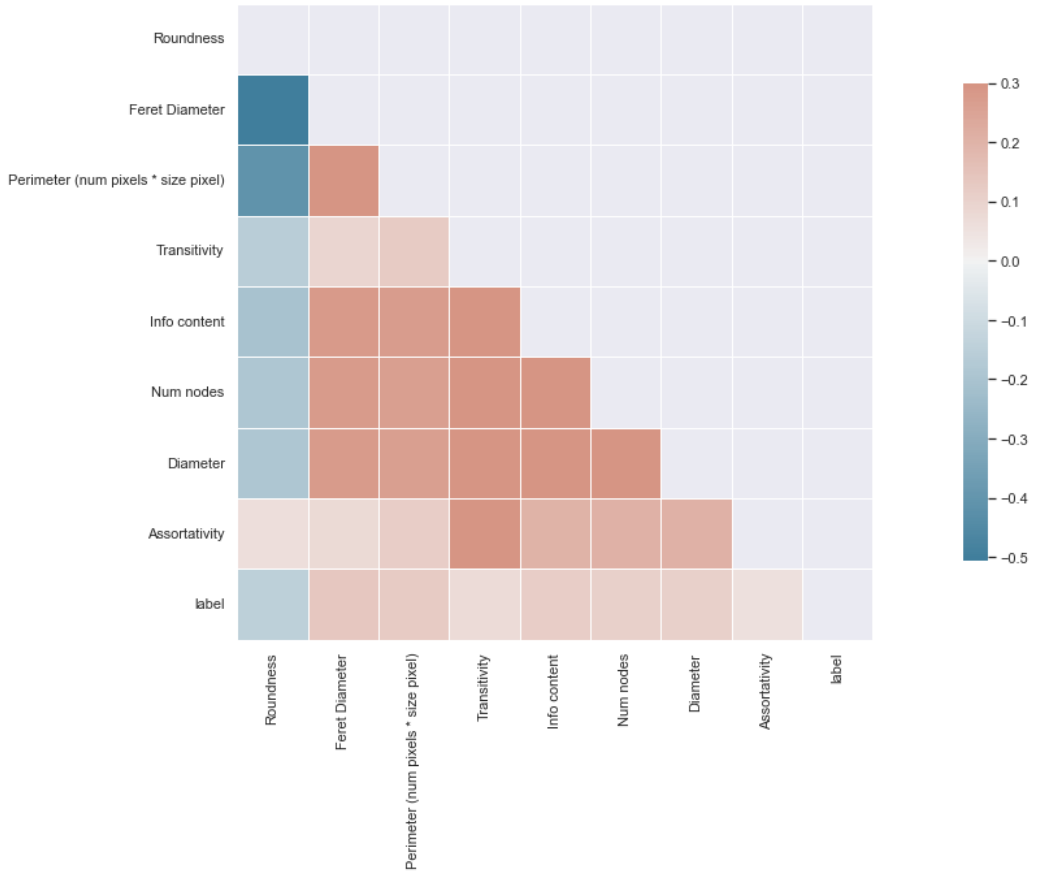


Figura G.2: Matriz de correlación para algunos parámetros de prueba en el problema de interés.

Hoeffding para la hipótesis M, de esta manera se obtiene como nueva cota la sumatoria de todas las otras cotas obtenidas por Hoeffding, la nueva cota para la hipótesis g se muestra en la ecuación G.5.

$$\mathbb{P}[|E_{in}(g) - E_{out}(g)| > \epsilon] \leq 2Me^{-2e^2N} \tag{G.5}$$

Esta nueva cota es lo que se buscaba pero tiene un nuevo problema y es que en general los conjuntos de hipótesis son grandes, e incluso pudiendo ser infinitos, por lo que no sería una cota útil en la práctica. El problema ahora se traduce en, ¿qué tanto se pueden acercar $E_{out}(g)$ y $E_{in}(g)$? y ¿qué tan chico se puede hacer $E_{in}(g)$? La desigualdad de Hoeffding solo subsana la primera pregunta, la segunda se puede saber una vez que se entrena el modelo y se obtiene un error de entrenamiento para el conjunto de datos. Hechos estos comentarios, se puede avanzar en la definición de que el error de generalización es la diferencia entre E_{in} y E_{out}, Hoeffding brinda una forma de caracterizar ese error con una cota probabilística ((G.5)). Se reescribe la Ecuación (G.5) tal que; $\mathbb{P}[|E_{in}(g) - E_{out}(g)|] \leq \epsilon] \leq 1 - 2Me^{-2N\epsilon^2}$, que implica que con probabilidad de por lo menos $1 - 2Me^{-2N}$, $E_{out}(g) \leq E_{in}(g) + \epsilon$. Definiendo $\delta = 2Me^{-2N\epsilon^2}$, se obtiene que $\epsilon = \sqrt{\frac{1}{2N} ln \frac{2M}{\delta}}$, esto permite escribir la cota de

generalización como la ecuación (G.6)

$$E_{out}(g) \leq E_{in}(g) + \sqrt{\frac{1}{2N} ln \frac{2M}{\delta}}. \tag{G.6}$$

Aquí δ es una tolerancia, por lo que cuanto más pequeña, más dispares se admiten los errores. M es la cardinalidad del conjunto de hipótesis y N la cantidad de muestras, de esto se desprende que para que la cota tenga utilidad práctica se debería buscar acotar la cardinalidad de $\mathcal{H}$ ya que incluso los modelos más sencillos de aprendizaje automático pueden tener infinitas hipótesis como el perceptrón. Para avanzar se enuncia a continuación el Teorema de la Cota de Generalización de Vapnik-Chervonenkis,

$$E_{out}(g) \leq E_{in}(g) + \sqrt{\frac{8}{N} ln \frac{4 m_{\mathcal{H}}(2N)}{\delta}}, \tag{G.7}$$

para todo δ y la cota se cumple con probabilidad $\geq 1 - \delta$. $m_{\mathcal{H}}$ es la función de crecimiento, que indica la cantidad máxima de dicotomías que pueden generarse por $\mathcal{H}$ en N muestras dadas. Yaser S. Abu-Mostafa, et al [79] demuestran a su vez que si $m_{\mathcal{H}}(k) < 2^k$ para algún k, entonces:

$$m_{\mathcal{H}}(N) \leq \sum_{i=0}^{k-1} \binom{N}{i} \tag{G.8}$$

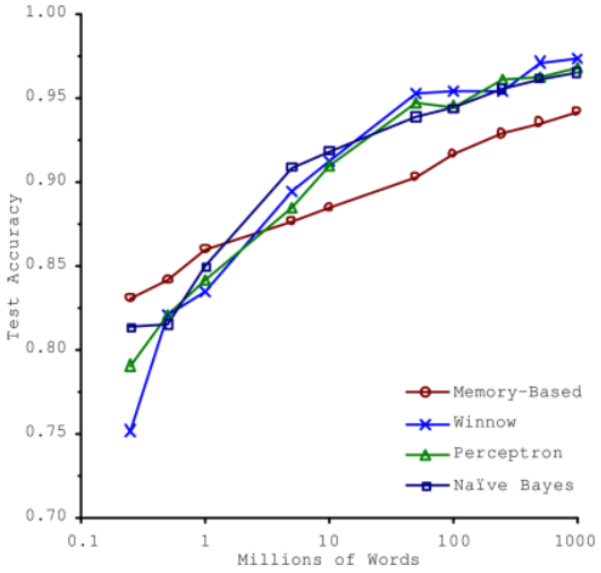
para todo N. Donde la parte derecha de la ecuación es un polinomio en N de grado $k-1$. De esta manera se obtiene una cota polinomial para la función de crecimiento. Volviendo a la ecuación (G.7) se tiene que hay una dependencia polinomial según N pero aplicada dentro de un logaritmo, por lo que cuando N crece, $1/N$ se reduce más rápido que el logaritmo. Este resultado alentador permite inferir que si se dispone de una suficiente cantidad de datos se puede lograr una generalización que permita aprender a la máquina. En la Figura G.3 se muestra los resultados obtenidos por M. Banko y E. Brill, donde muestran que con una suficiente cantidad de datos, independientemente del modelo, se logran buenos resultados [80].

Calidad de datos y características

Poseer una gran cantidad de datos es necesario para lograr que una máquina aprenda pero no es suficiente, se debe disponer además, de datos con calidad, representativos de los nuevos casos que se desea generalizar y con características relevantes al problema.

En la base de datos puede haber ruido, errores, *outliers*, datos faltantes, entre otros, estos problemas habilita una parte sustancialmente importante en el proceso de entrenamiento de modelos de aprendizaje automático, la limpieza de los datos, donde se puede descartar el dato, completar los datos faltantes o incluso descartar la característica.

Las características de los datos también son fundamentales, si se posee un mapa de características que no tiene correlación con la etiqueta que se desea aprender,



.

Figura G.3: Se muestra la *Accuracy* en el conjunto de prueba en relación a la cantidad de datos utilizados para el entrenamiento. Esta imagen fue obtenida de [43] donde citan a un trabajo que en 2001 demostró que para distintos modelos de aprendizaje automático, incluyendo algunos simples, si se cuenta con suficiente cantidad de datos se obtiene un comportamiento similar [80]

difícilmente se encuentre un modelo que infiera correctamente la etiqueta viendo solo esos datos. Por lo que tener una etapa de selección de las características más importantes o incluso creando nuevas, es fundamental en el proceso.

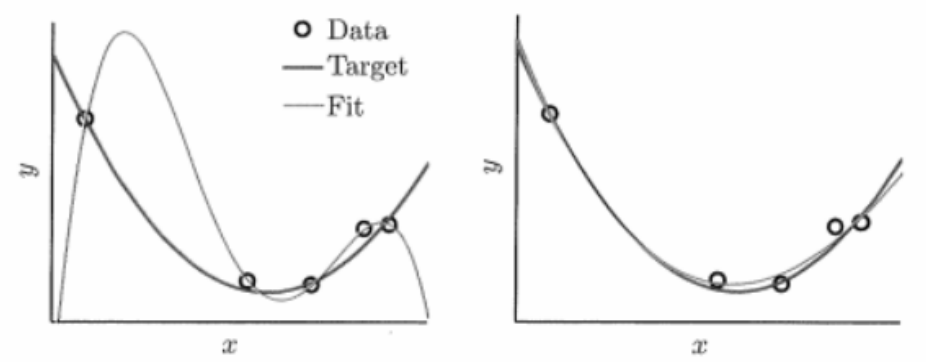
Ajuste a los datos

Los problemas presentados anteriormente sumados a una mala elección de los modelos puede llevar a una mala generalización. Cuando sucede esto último, el problema puede estar; sobreajustando o subajustando.

El sobreajuste sucede cuando el modelo es demasiado complejo relativo a la cantidad de datos de entrenamiento y su ruido. En la Figura G.4 se muestra gráficamente el efecto de sobreajustar con un modelo de gran complejidad. Para solucionar este problema, pueden existir varios abordajes, aumentar la cantidad de datos y por ende mejorar la posibilidad que el conjunto de entrenamiento sea representativo de las muestras por fuera, reducir el ruido o eliminar los valores atípicos o simplificar el modelo utilizado, ya sea eligiendo alguno con menos parámetros, reduciendo la cantidad de características utilizadas o restringiendo el modelo.

Cuando se imponen restricciones al modelo, se le conoce como regularización, en la Figura G.4 a la derecha, se muestra qué sucede cuando se incorpora un término de regularización al modelo seleccionado anteriormente, donde se nota un mejor ajuste a la función objetivo.

Para el problema del subajuste sucede lo opuesto al sobreajuste, el modelo utilizado no posee la suficiente complejidad como para representar la función objetivo,



.

Figura G.4: A la izquierda se tiene una función objetivo (*target*) y una función (*fit)* que se ajusta a los datos, en este caso el modelo se está sobreajustando a los datos, obteniendo poco error dentro de la muestra pero generalizando mal por fuera de ésta. Es claro el ejemplo que con un modelo de menos orden se podría estimar mejor. A la derecha se utiliza el mismo modelo pero incorporando regularización para limitar la complejidad del modelo. Imágenes obtenidas de [79]

por lo que se debe seleccionar un modelo con más parámetros, utilizar características con mejor correlacionadas respecto a lo que se desea aprender o reducir las restricciones del modelo.

Habiendo realizado estos comentarios sobre el entrenamiento, vale mencionar que la única forma de saber qué tan bien se está generalizando, es a través de las pruebas con un conjunto de nuevos datos.

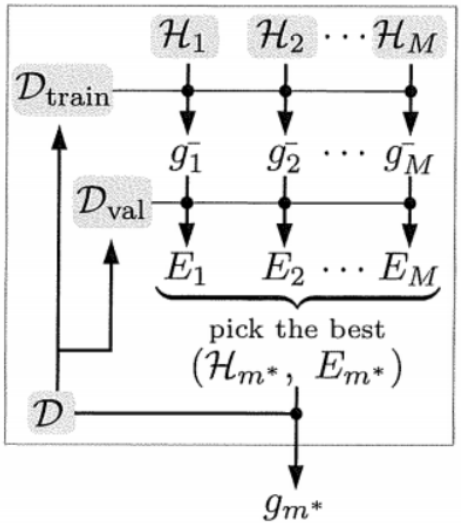
Para definir los hiperparámetros de los modelos se utiliza un tercer conjunto de datos nuevos que se denomina validación, este set de datos es distinto al de test porque se utiliza para incorporar información al modelo, es decir, para definir qué hiperparámetros brindan la mejor generalización. Para la selección de hiperparámetros se entrenan varios modelos con distintos hiperparámetros y se validan con un conjunto de validación, el desempeño del modelo se mide finalmente con el conjunto de pruebas. En la Figura G.5 se muestra el procedimiento para elegir la mejor hipótesis utilizando el conjunto de validación.

Métricas y Error

Hasta ahora se ha planteado varios elementos generales de comparación, se plantean errores pero no se ha definido cómo se computa el error. La selección de métricas de desempeño y error son fundamentales en los modelos de aprendizaje automático para explicar que está sucediendo en el entrenamiento y en los resultados. En la figura 4.3 se muestra la matriz de confusión que sirve para mostrar los distintos tipo de error, necesario para la explicación de cómo funciona cada métrica.

La primera métrica a definir es la *accuracy*,

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}, \tag{G.9}$$



.

Figura G.5: Se muestra en la figura, el proceso de selección de modelo utilizando un conjunto de entrenamiento y otro de validación. $\mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_M$, son conjuntos de hipótesis posibles para distintos modelos, g_1^-, g_2^-, g_M^-, las hipótesis seleccionadas para cada conjunto respectivamente y E_1, E_2, E_M los errores obtenidos para el conjunto de validación. $\mathcal{D}_{train}$ y $\mathcal{D}_{val}$ son el conjunto de entrenamiento y validación respectivamente. g_m^* es la mejor hipótesis teniendo en cuenta el error de validación. Imagen obtenida de [79]

esta métrica es la más intuitiva pero es muy sensible a problemas como que los datos estén desbalanceados. Por ejemplo, si se tiene un problema de dos etiquetas donde el 90 % del conjunto a probar tiene etiqueta 0 y el otro 10 % tiene etiqueta 1, una función que siempre asigne la etiqueta 0 redundará en una *accuracy* de 90 %. Otras métricas comúnmente utilizadas son la *precision* y el *recall* definidas en el Capitulo 4. Retomando el ejemplo presentado para la *accuracy* si $TP = 1$, $TN = 90$, $FP = 1$, $FN = 8$, el cálculo de la *recall* queda: $recall = \frac{1}{1+8} = 0{,}11$ y la precisión: $precision = \frac{1}{1+1} = 0{,}5$. Demostrando esto que dependiendo de la métrica que se utilice, se obtienen distintos resultados y es importante poder interpretarlos correctamente. La definición de umbrales para evaluar los algoritmos implica que para una cierta métrica, ese umbral puede ser mejor y para otra métrica peor, por lo que siempre se debe tener en cuenta el problema de estudio, qué tipo de error es más o menos admisible . Presentadas estas dos métricas se define a continuación la *accuracy balanceada* como:

$$balanced\_accuracy = \frac{1}{N} \sum_{i=1}^{N} recall_i, \tag{G.10}$$

donde N es la cantidad de clases. Ésta métrica es invariante al desbalance de clases.

Esta página ha sido intencionalmente dejada en blanco.

Apéndice H

Técnica clásicas de Aprendizaje Automático

Reducción de la dimensionalidad - PCA

Como se ha mencionado anteriormente, la cantidad de características puede llevar a una mala generalización si no se cuenta con la suficiente cantidad de datos. El análisis de componentes principales permite detectar las dimensiones que brindan mayor información y así reducir de la cantidad de dimensiones sin perder las más importantes, claramente la pérdida de dimensiones puede implicar pérdida de información pero en los casos donde las características no aportan demasiado, puede realizarse sin perjuicio. El objetivo del PCA es detectar el hiperplano de dimensión n más chica que la dimensión original, que contiene la mayor cantidad de información y así preservar la mayor cantidad de varianza. En la Figura H.1 se muestra un caso en dos dimensiones, donde el objetivo es reducir a una, esa reducción muestra que dependiendo de la dirección que se elija, se mantiene más o menos información.

Se omite el procedimiento de cálculo del PCA, solo para orientar al lector, se utiliza *Singular Value Decomposition(SVD)* que es una técnica de factorización de matrices [81].

Regresión Logística

El primer modelo que se presentará es la Regresión Logística pero previo a formalizarlo, se presenta la regresión lineal para mostrar la idea fundamental que rige en la regresión. La regresión lineal se expresa simplemente multiplicando las características por los pesos más un término de sesgo (*bias*). Se presenta a continuación el modelo de regresión lineal,

$$\hat{y} = b + \theta_1 x_1 + \cdots + \theta_n x_n, \tag{H.1}$$

donde $\hat{y}$ es el valor predicho, n la cantidad de características, x_i la característica i, θ_i el peso correspondiente a x_i y b el sesgo. En notación vectorial se puede escribir como:

$$\hat{y} = \theta^T \mathbf{x}. \tag{H.2}$$

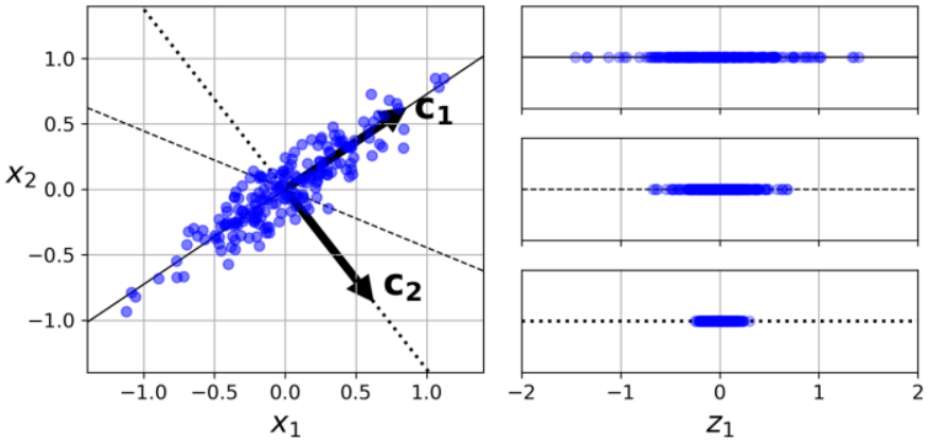


Figura H.1: A la izquierda se muestra la distribución para dos características x_1 y x_2, a la derecha distintas proyecciones unidimensionales. Imagen obtenida de [43]

Habiendo introducido la regresión lineal se está en condiciones de expresar la regresión logística en función de esta. Se expresa la regresión logística en la ecuación (H.3),

$$\hat{p} = \sigma(\mathbf{x}^T\theta), \tag{H.3}$$

siendo $\sigma(t) = \frac{1}{1+e^{-t}}$. Este resultado brinda una probabilidad, por lo que se debe definir un umbral para asignar una etiqueta.

Para el entrenamiento del modelo se minimiza la función de costo (H.4).

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}log(\hat{p}^{(i)}) + (1-y^{(i)})log(1-\hat{p}^{(i)})\right], \tag{H.4}$$

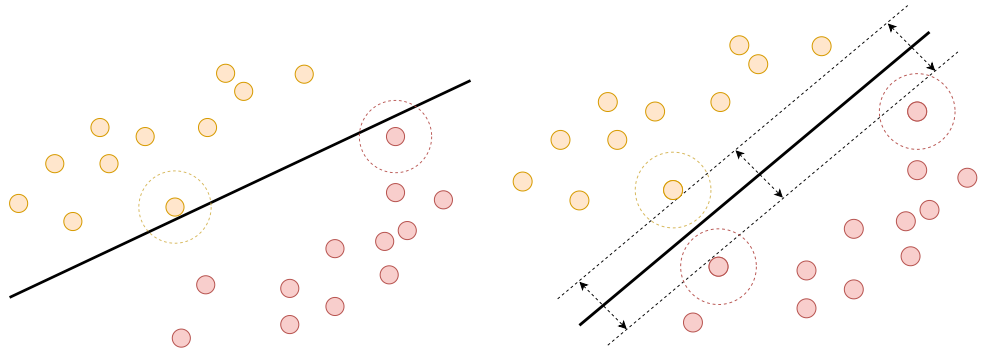
ésta función de costo no tiene una expresión cerrada que la minimice pero es una función de costo convexa, por lo que se puede utilizar el descenso por gradiente [82].

Support Vector Machine

Support Vector Machines (SVM) son modelos de aprendizaje automático capaces de aprender clasificación a modo similar que un perceptrón, pero incorporando un margen que permite elegir el modelo que mejor se comportará con nuevas muestras. Se muestra un ejemplo en la Figura H.2b, donde se grafican circunferencias alrededor de los datos más cercanos al hiperplano para mostrar que el margen permite aumentar la capacidad de generalización, si el hiperplano se elige aleatoriamente puede suceder que un dato muy cercano a uno bien clasificado se le asigne la etiqueta incorrecta, como se muestra con las circunferencias en la Figura H.2a. Sea h un hiperplano, éste divide los dos subconjuntos de datos, si se cumple que:

$$y_n(\mathbf{w}^T\mathbf{x}_n + b) > 0, \tag{H.5}$$

para $n = 1, \ldots, N$. Siendo w los pesos aprendidos, $x_1, \ldots, x_N$ los datos, b el *bias* e y_n la etiqueta que corresponde para el dato x_n. Para hallar el margen del hiperplano, se necesita calcular la distancia entre el hiperplano y el dato más cercano.



(a) Separación de los subconjuntos con un hiperplano seleccionado aleatoriamente entre las posibles soluciones.

(b) Separación de los subconjuntos con el hiperplano óptimo que separa ambos conjuntos. En línea punteada negra el margen que se obtiene. El margen es rígido ya que no se aceptan datos dentro de ese margen.

Figura H.2: En color naranja y rojo los datos correspondiente a clases distintas, en negro el hiperplano que separa ambos conjuntos.

Se utilizará la Figura H.3 para ejemplficar cómo se calcula el margen y se tomará la deducción de [79]. El objetivo es calcular la distancia del hiperplano h (amarillo) al dato $\mathbf{x}$, donde $d(\mathbf{x}, h)$ es la distancia perpendicular de h a $\mathbf{x}$. Sea cualquier punto $\mathbf{x}'$ que pertenezca al hiperplano, es decir $\mathbf{x}^T\mathbf{x}' + b = 0$ y sea $\mathbf{u}$ un vector normal a h, entonces

$$d(\mathbf{x}, h) = |\mathbf{u}^T(\mathbf{x} - \mathbf{x}')|. \tag{H.6}$$

Tomando otro punto($\mathbf{x}''$) que pertenezca al hiperplano se puede escribir cualquier vector que pertenezca al hiperplano como: $(\mathbf{x}' - \mathbf{x}'')$. Utilizando que $\mathbf{w}^T\mathbf{x} = -b$ se tiene que:

$$\mathbf{w}^T(\mathbf{x}'' - \mathbf{x}') = \mathbf{w}^T\mathbf{x}'' - \mathbf{w}^T\mathbf{x}' = -b + b = 0. \tag{H.7}$$

Es decir, $\mathbf{w}$ es ortogonal a cada vector del hiperplano h y por lo tanto el vector normal $\mathbf{u}$ se puede escribir como:

$$\mathbf{u} = \frac{\mathbf{w}}{\|\mathbf{w}\|}. \tag{H.8}$$

De esta manera se obtiene la distancia como:

$$d(\mathbf{x}, h) = |\mathbf{u}^T(\mathbf{x} - \mathbf{x}')| = \frac{\mathbf{w}^T\mathbf{x} - \mathbf{w}^T\mathbf{x}'}{\|\mathbf{w}\|} = \frac{|\mathbf{w}^T\mathbf{x} + b|}{\|\mathbf{w}\|}. \tag{H.9}$$

Incorporando la información de la etiqueta(y_n), se tiene que la distancia puede ser escrita como

$$d(\mathbf{x}_n, h) = \frac{y_n(\mathbf{w}^T\mathbf{x}_n + b)}{\|\mathbf{w}\|}, \tag{H.10}$$

por lo tanto el punto más cercano a h debe cumplir que:

$$\min_{n=1,\dots,N} \quad d(\mathbf{x}_n, h) = \min_{n=1,\dots,N} \quad y_n(\mathbf{w}^T\mathbf{x}_n + b)\frac{1}{\|\mathbf{w}\|} \tag{H.11}$$

$$= \frac{1}{\|\mathbf{w}\|}. \tag{H.12}$$

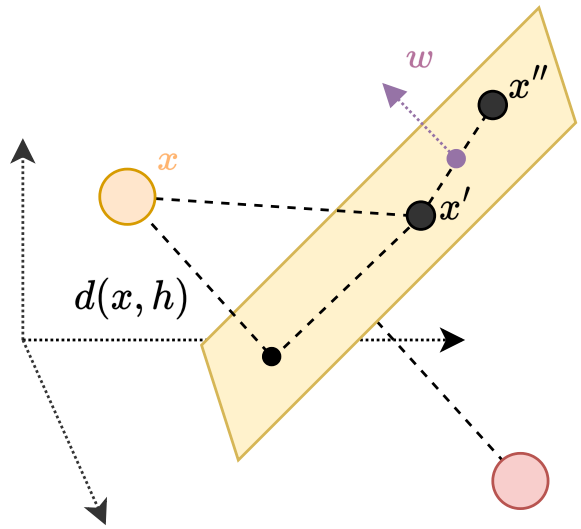


Figura H.3: En amarillo se muestra el hiperplano y cómo se computa la distancia a un punto x(naranja), siendo ésta $d(x, h)$. Se grafican dos puntos del hiperplano, x' y x'' y se muestra que w tiene dirección perpendicular al hiperplano.

Esta igualdad se cumple porque el hiperplano $h(b, w)$ es el mismo hiperplano que $h(b/\rho, \mathbf{w}/\rho)$ para cualquier $\rho > 0$ y por lo tanto, seleccionando

$$\rho = \min_{n=1,\dots,N} \quad y_n(\mathbf{w}^T\mathbf{x}_n + b) \tag{H.13}$$

y reescalando los pesos se puede obtener que

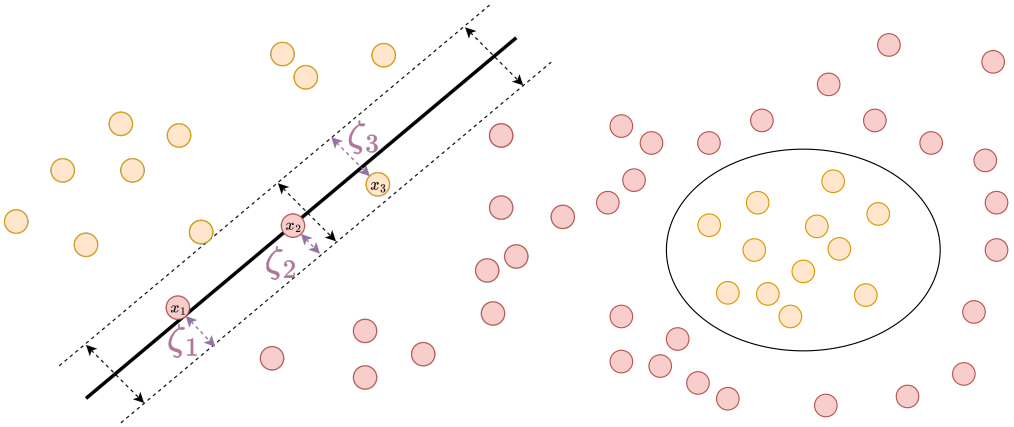
$$\min_{n=1,\dots,N} \quad y_n\left(\frac{\mathbf{w}^T}{\rho}\mathbf{x}_n + \frac{b}{\rho}\right) = \min_{n=1,\dots,N} \quad y_n(\mathbf{w}^T\mathbf{x}_n + b)\frac{1}{\rho} = \frac{\rho}{\rho} = 1. \tag{H.14}$$

Para obtener la predicción de una clase para la instancia $\mathbf{x}$ se obtiene computando $\mathbf{w}^T\mathbf{x} + b = w_1x_1 + \cdots + w_nx_n + b$, donde si el resultado es positivo se predice una clase y si es negativo, la otra. Para entrenar se define el problema para el caso de margen rígido (no se permite ningún dato dentro del margen) en la Ecuación (H.16), donde la restricción viene impuesta por la Ecuación (H.14) y la minimización surge de que:

$$\max_{\mathbf{w},b} \quad \frac{1}{\|\mathbf{w}\|} = \min_{\mathbf{w},b} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} \tag{H.15}$$

$$\begin{aligned} &\min_{\mathbf{w},b} \quad \frac{1}{2}\mathbf{w}^t\mathbf{w} \\ &sujeto\ a:\ y_n(\mathbf{w}^T\mathbf{x}_n + b) \geq 1 \quad para \quad n = 1, 2, \dots, N \end{aligned} \tag{H.16}$$

siendo $y_n = -1$ para datos con etiqueta 0 y $y_1 = 1$ para datos con etiqueta 1. Por lo tanto, para encontrar el hiperplano óptimo se tiene que resolver éste problema



(a) Separación de los subconjuntos con el hiperplano óptimo obtenido del problema de margen "suave". Se muestran las distintas violaciones al margen con ζ_1, ζ_2 y ζ_3.

(b) Separación de los subconjuntos que no son linealmente separables pero que podrían serlo luego de una transformación no lineal.

Figura H.4: En color naranja y rojo los datos correspondiente a clases distintas, en negro el hiperplano que separa ambos conjuntos.

de optimización. Cuando se minimiza una función cuadrática restringidas por una desigualdad lineal se puede utilizar *programación cuadrática* [83]. El análisis se ha acotado a asumir que los datos son linealmente separables, de otro modo, el problema planteado no tendría solución. Se podrían dar situaciones donde la función objetivo sea no lineal(H.4b) o existan *outliers*(H.4a) y eso no permita resolver el problema de optimización. El desarrollo del caso de margen "suave" y de las transformaciones no lineales no se expondrá aquí pero el lector puede encontrarlo en [79].

El problema de optimización para el caso de margen "suave" se muestra en la ecuación (H.17).

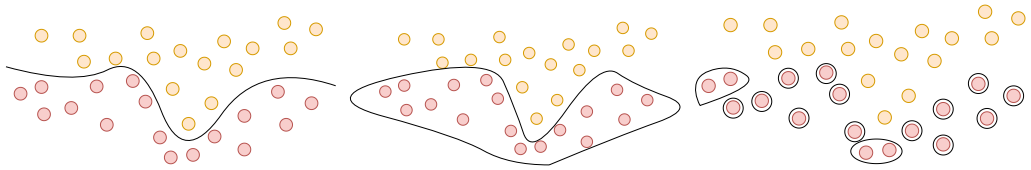
$$\begin{aligned} &\min_{\mathbf{w},b} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{n=1}^{N}\zeta_n \\ &sujeto\ a:\ y_n(\mathbf{w}^T\mathbf{x}_n + b) \geq 1 - \zeta_n \quad para \quad n = 1, 2, \ldots, N \quad y \quad \zeta_n \geq 0, \end{aligned} \tag{H.17}$$

donde ζ_n es una medida de cuanto se vulnera el margen para cada x_n. Debido a que se desea que la cantidad de vulneraciones al margen sean pocas, se incorpora un término C para penalizar la sumatoria $\sum_{n=1}^{N}\zeta_n$.

Para el caso donde se tiene una función objetivo no lineal, se pueden utilizar transformaciones no lineales. Siendo $\Phi : \mathcal{X} \to \mathcal{Z}$ la transformación no lineal, $\mathcal{X}$ el dominio de los datos y $\mathcal{Z}$ el dominio donde los datos pueden ser linealmente separables, se puede definir el *kernel* como sigue:

$$K_\Phi(\mathbf{x}, \mathbf{x}') = \Phi(\mathbf{x})^T\Phi(\mathbf{x}'). \tag{H.18}$$

Para resolver de forma eficiente este problema se utiliza el *kernel trick*, que se basa en la idea calcular el producto interno entre los datos en una dimensión superior sin realizar la transformación de cada dato independientemente, se puede encontrar



(a) $\gamma = 1$ (b) $\gamma = 10$ (c) $\gamma = 100$

Figura H.5: Se muestran tres elecciones de γ, donde a medida que aumenta se evidencia el sobreajuste. En negro se dibuja la frontera de decisión obtenida.

detallado en [84]. Dos *kernel* que se benefician del *kernel trick* son el polinomial y el *Gaussian-Radial Basis Function(RBF)*. Se define a continuación la transformación polinomial de orden 2,

$$\Phi(\mathbf{x}) = (1, x_1, x_2, \ldots, x_N, x_1x_1, x_1x_2, \ldots, x_Nx_N), \tag{H.19}$$

que deriva en el *kernel*:

$$K(\mathbf{x}, \mathbf{x}') = 1 + (\mathbf{x}^T\mathbf{x}') + (\mathbf{x}^T\mathbf{x}')^2. \tag{H.20}$$

El *kernel* del RBF se define como:

$$K(\mathbf{x}, \mathbf{x}') = e^{-\gamma\|\mathbf{x}-\mathbf{x}'\|^2} \tag{H.21}$$

El parámetro γ controla el ancho del *kernel gaussiano*. En la Figura H.5 se muestran tres ejemplo de γ, donde para $\gamma = 100$ hay un claro sobreajuste, mostrando que la selección de los hiperparámetros es crucial para cualquier modelo.

Métodos por ensamble - Random Forest

Los métodos por ensamble son modelos de aprendizaje automático que utilizan las predicciones de varios predictores para inferir el resultado. La combinación de métodos puede ser una buena solución incluso cuando se combinan modelos que son apenas mejor que los algoritmos que adivinan aleatoriamente. Estos modelos se les denomina predictores débiles y cuentan con poco sesgo y gran varianza. Otro elemento que favorece el entrenamiento de estos modelos es que exista independencia entre los predictores base, si es posible tener distintos conjuntos de entrenamiento o modelos distintos, puede favorerecer la generalización. Random Forest es una modelo que utiliza *bootstrap* (muestreo con reposición) con árboles de decisión, los árboles son idénticamente distribuidos con sesgo b y varianza σ^2. Para el entrenamiento se utiliza *Bagging*, utilizando M predictores, cada uno entrenado con un subconjunto de entrenamiento, las predicciones se combinan por mayoría. En la Figura H.6 se muestra un esquema de ejemplo con tres árboles de decisión.

Bajo las condiciones anteriores el sesgo del promedio de los árboles es el sesgo de cada árbol, es decir: $b_{rf} = b$. Para la varianza vale:

$$\sigma^2_{rf} = \rho\sigma^2 + \frac{1-\rho}{M}\sigma^2, \tag{H.22}$$

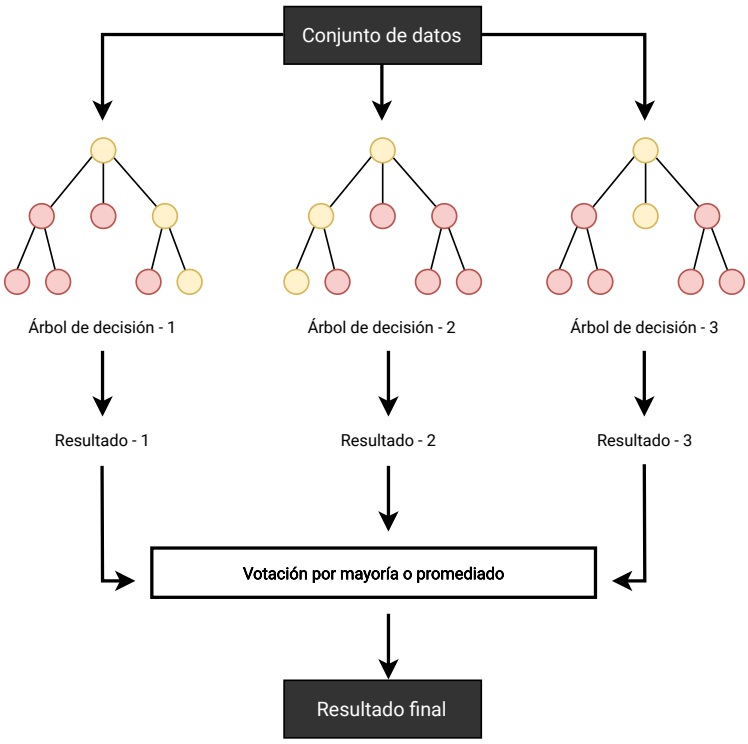


Figura H.6: Random Forest con tres árboles de decisión, se muestran en verde y azul la etiqueta a asignar en caso de que el nuevo dato se corresponda con esa hoja del árbol.

siendo ρ el coeficiente de correlación de Pearson entre un par de árboles. Al aumentar M se reduce el segundo término que implica reducir la varianza del modelo. El Random Forest, tiene los mismos hiperparámetros que los árboles de decisión para evitar el sobreajuste e incorpora nuevos por ser un método de ensamble, como la cantidad de estimadores a utilizar.

Una virtud interesante del Random Forest es que permite medir la importancia relativas de las características utilizadas para el entrenamiento.

Esta página ha sido intencionalmente dejada en blanco.

Referencias

[1] B. R. Masters, J. G. Fujimoto, and D. L. Farkas, "Biomedical optical imaging," *Journal of Biomedical Optics*, vol. 15, no. 5, pp. 3 – 28, 2010.

[2] A. López-Macay, J. Fernández-Torres, A. Zepeda, and U. Nacional, "Principios y aplicaciones de la microscopia láser confocal en la investigación biomédica Principles and applications of laser confocal microscopy in biomedical research," *Investigación en Discapacidad*, vol. 5, pp. 139–145, 2016. [Online]. Available: www.medigraphic.org.mxhttp://www.medigraphic.com/ridwww.medigraphic.org.mx

[3] "Imagina." [Online]. Available: https://www.imagina.ei.udelar.edu.uy/

[4] "Classic maximum likelihood estimation." [Online]. Available: https://svi.nl/MaximumLikelihoodEstimation

[5] "Good roughness maximum likelihood estimation." [Online]. Available: https://svi.nl/GoodRoughnessMaximumLikelihoodEstimation

[6] M. Boulakroune, D. Benatia, N. Slougui, and A. E. Oualkadi, "Tikhonov-Miller regularization with a denoisy and deconvolved signal as model of solution for improvement of depth resolution in SIMS analysis," *2008 3rd International Conference on Information and Communication Technologies: From Theory to Applications, ICTTA*, no. May, 2008.

[7] M. K. Khan, S. Morigi, L. Reichel, and F. Sgallari, "Iterative methods of Richardson-Lucy-type for image deblurring," *Numerical Mathematics*, vol. 6, no. 1, pp. 262–275, 2013.

[8] "Richardson-Lucy algorithm with total variation regularization for 3D confocal microscope deconvolution," *Microscopy Research and Technique*, vol. 69, no. 4, pp. 260–266, 2006.

[9] N. S. Punn and S. Agarwal, *Modality specific U-Net variants for biomedical image segmentation: a survey.* Springer Netherlands, 2022, no. 0123456789. [Online]. Available: https://doi.org/10.1007/s10462-022-10152-1

[10] J. Schindelin, I. Arganda-Carreras *et al.*, "Fiji: an open-source platform for biological-image analysis," *Nature Methods*, vol. 9, no. 7, pp. 676–682, Jul 2012. [Online]. Available: https://doi.org/10.1038/nmeth.2019

[11] C. T. Rueden, J. Schindelin, M. C. Hiner, B. E. DeZonia, A. E. Walter, E. T. Arena, and K. W. Eliceiri, "Imagej2: Imagej for the next generation of scientific image data," *BMC Bioinformatics*, vol. 18, no. 1, p. 529, Nov 2017. [Online]. Available: https://doi.org/10.1186/s12859-017-1934-z

[12] D. Sage *et al.*, "DeconvolutionLab2: An open-source software for deconvolution microscopy," *Methods*, vol. 115, pp. 28–41, 2017. [Online]. Available: http://dx.doi.org/10.1016/j.ymeth.2016.12.015

[13] K. W. Dunn, C. Fu, D. J. Ho, S. Lee, S. Han, P. Salama, and E. J. Delp, "Deepsynth: Three-dimensional nuclear segmentation of biological images using neural networks trained with synthetic data," *Scientific Reports*, vol. 9, no. 1, p. 18295, Dec 2019.

[14] M. Zanin, B. Santos, and P. A. *et al*, "Mitochondria interaction networks show altered topological patterns in Parkinson's disease," *npj Systems Biology and Applications*, vol. 6, no. 38, 2020. [Online]. Available: http://dx.doi.org/10.1038/s41540-020-00156-4

[15] "Microscopía de fluorescencia." [Online]. Available: https://www.leica-microsystems.com/es/aplicaciones/ciencias-biologicas/fluorescencia/

[16] C. Greb, "Fluorescent dyes," Jan 2022. [Online]. Available: https://www.leica-microsystems.com/science-lab/fluorescent-dyes/

[17] "Confocal microscopes." [Online]. Available: https://www.britannica.com/technology/microscope/Confocal-microscopes

[18] J. Waters, "The point spread function," Dec 2018. [Online]. Available: https://www.youtube.com/watch?v=Tkc_GOCjx7E

[19] D. B. Schmolze, C. Standley, K. E. Fogarty, and A. H. Fischer, "Advances in microscopy techniques," *Archives of Pathology and Laboratory Medicine*, vol. 135, no. 2, pp. 255–263, 2011.

[20] M. Wilson, "Collecting light: The importance of numerical aperture in microscopy," Aug 2017. [Online]. Available: https://www.leica-microsystems.com/science-lab/collecting-light-the-importance-of-numerical-aperture-in-microscopy/

[21] ——, "Immersion objectives: Using oil, glycerol, or water to overcome some of the limits of resolution," Nov 2021. [Online]. Available: https://www.leica-microsystems.com/science-lab/immersion-objectives-using-oil-glycerol-or-water-to-overcome-some-of-the-limits-of-resolution/

[22] "Psf generator." [Online]. Available: http://bigwww.epfl.ch/algorithms/psfgenerator/

[23] E. W. M.Born, *Principles of Optics*, Cambridge University Press, 2003, vol. 7th edition.

[24] H. W. Dan, W. C. Hung, Y. S. Tsai, and C. C. Lin, "3D image processing for mitochondria morphology variation analysis," *2014 IEEE International Symposium on Bioelectronics and Bioinformatics, IEEE ISBB 2014*, pp. 0–3, 2014.

[25] D. J. Ho, C. Fu, P. Salama, K. W. Dunn, and E. J. Delp, "Nuclei detection and segmentation of fluorescence microscopy images using three dimensional convolutional neural networks," *Proceedings - International Symposium on Biomedical Imaging*, vol. 2018-April, pp. 418–422, 2018.

[26] S. Nesmachnow and S. Iturriaga, "Cluster-uy: Collaborative scientific high performance computing in uruguay," in *Supercomputing*, M. Torres and J. Klapp, Eds. Cham: Springer International Publishing, 2019, pp. 188–202.

[27] I. Goodfellow, J. Pouget-Abadie, M. Mirza *et al.*, "Generative adversarial networks." *Communications of the ACM*, vol. 63, no. 11, pp. 139 – 144, 2020.

[28] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[29] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *CVPR*, 2017.

[30] T. Z. A. A. E. Phillip Isola, Jun-Yan Zhu, "Image-to-image translation with conditional adversarial nets." [Online]. Available: https://phillipi.github.io/pix2pix/

[31] C. Fu, S. Lee, D. J. Ho, S. Han, P. Salama, K. W. Dunn, and E. J. Delp, "Three dimensional fluorescence microscopy image synthesis and segmentation," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, June 2018, pp. 2302–23 028.

[32] T. Falk, D. Mai, R. Bensch, Çiçek, A. Abdulkadir, Y. Marrakchi, A. Böhm, J. Deubner, Z. Jäckel, K. Seiwald, A. Dovzhenko, O. Tietz, C. Dal Bosco, S. Walsh, D. Saltukoglu, T. L. Tay, M. Prinz, K. Palme, M. Simons, I. Diester, T. Brox, and O. Ronneberger, "U-net: deep learning for cell counting, detection, and morphometry." *Nature methods*, vol. 16, no. 1, pp. 67 – 70, 2019. [Online]. Available: http://proxy.timbo.org.uy/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=cmedm&AN=30559429&lang=es&site=eds-live

[33] J.-B. Sibarita, "Deconvolution microscopy," *Microscopy Techniques*, pp. 201–243, 2005.

[34] J. G. McNally, T. Karpova, J. Cooper, and J. A. Conchello, "Three-dimensional imaging by deconvolution microscopy," *Methods*, vol. 19, no. 3, pp. 373–385, 1999.

[35] P. Sarder and A. Nehorai, "Deconvolution methods for 3-d fluorescence microscopy images," *IEEE Signal Processing Magazine*, vol. 23, no. 3, pp. 32–45, 2006.

[36] N. Wiener, "Extrapolation, interpolation, and smoothing of stationary time series, vol. 2," 1949.

[37] S. Horii, "A gentle introduction to maximum likelihood estimation and maximum a posteriori estimation," Oct 2019. [Online]. Available: https://towardsdatascience.com/a-gentle-introduction-to-maximum-likelihood-estimation-and-maximum-a-posteriori-estimation-d7c318f9d22d

[38] J. Pillow, "Lecture 22: Linear Shift-Invariant (LSI) Systems and Convolution Linear Shift-Invariant (aka "time-invariant") Systems," *Mathematical Tools for Neuroscience*, no. Neu 314, 2016.

[39] R. M. Gray, "Toeplitz and circulant matrices: A review," *Foundations and Trends in Communications and Information Theory*, vol. 2, no. 3, pp. 155–239, 2006.

[40] "Pyimagej." [Online]. Available: https://pypi.org/project/pyimagej/

[41] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2005.

[42] J. P. Pluim, J. A. Maintz, and M. A. Viergever, "Mutual-information-based registration of medical images: a survey," *IEEE transactions on medical imaging*, vol. 22, no. 8, pp. 986–1004, 2003.

[43] A. Gron, *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, 2nd ed. O'Reilly Media, Inc., 2019.

[44] K. Miura, "An Introduction to Maximum Likelihood Estimation and Information Geometry," *Interdisciplinary Information Sciences*, vol. 17, no. 3, pp. 155–174, 2011.

[45] C. E. Shannon, "A mathematical theory of communication Bell Syst," *tech. J*, vol. 27, no. 379, p. 623, 1948.

[46] L.-K. H. Wang and M.-J. J, "Image thresholding by minimizing the measure of fuzziness." *Pattern Recognition Letters*, vol. 28, no. 1, pp. 41–45, 1995.

[47] C. H. Li and P. K. Tam, "An iterative algorithm for minimum cross entropy thresholding," *Pattern Recognition Letters*, vol. 19, no. 8, pp. 771–776, 1998.

[48] T. Ridler and S. Calvard, "Picture Thresholding Using An Interactive Selection Method," *IEEE Transactions on Systems, Man and Cybernetics*, vol. smc-8, no. 8, pp. 630–632, 1978.

[49] T. Pun, "A new method for grey-level picture thresholding using the entropy of the histogram," pp. 223–237, 1980.

[50] Y. C. Liang and J. R. Cuevas, "An automatic multilevel image thresholding using relative entropy and meta-heuristic algorithms," *Entropy*, vol. 15, no. 6, pp. 2181–2209, 2013.

[51] "Analisis de redes mitocondriales." [Online]. Available: https://prueba-timag.webnode.com.uy/

[52] "Algoritmo k-means: Clustering de forma sencilla," May 2021. [Online]. Available: https://www.themachinelearners.com/k-means/

[53] O. Ronneberger, P. Fischer, and T. Brox, "2015-U-Net," *arXiv*, pp. 1–8, 2015. [Online]. Available: http://lmb.informatik.uni-freiburg.de/0Aarxiv:1505.04597v1

[54] R. C. Gonzalez and R. E. Woods, 2008, vol. 3rd Editio, no. 3.

[55] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-December, pp. 770–778, 2016.

[56] "Unet with resblock for semantic segmentation." [Online]. Available: https://medium.com/@nishanksingla/unet-with-resblock-for-semantic-segmentation-dd1766b4ff66c

[57] L. R. Dice, "Measures of the amount of ecologic association between species," vol. 3, 1945.

[58] A. A. Taha and A. Hanbury, "Metrics for evaluating 3D medical image segmentation: Analysis, selection, and tool," *BMC Medical Imaging*, vol. 15, no. 1, 2015. [Online]. Available: http://dx.doi.org/10.1186/s12880-015-0068-x

[59] "Skeletonize." [Online]. Available: https://scikit-image.org/docs/stable/auto_examples/edges/plot_skeleton.html

[60] Z. Yaniv, B. C. Lowekamp, H. J. Johnson, and R. Beare, "SimpleITK image-analysis notebooks: a collaborative environment for education and reproducible research," *Journal of Digital Imaging*, vol. 31, no. 3, pp. 290–303, Nov. 2017. [Online]. Available: https://doi.org/10.1007/s10278-017-0037-8

[61] P. Kollmannsberger, M. Kerschnitzki, F. Repp, W. Wagermaier, R. Weinkamer, and P. Fratzl, "The small world of osteocytes: Connectomics of the lacuno-canalicular network in bone," *New Journal of Physics*, vol. 19, no. 7, 2017.

[62] G. Lehmann, "Label object representation and manipulation with itk," *Insight J.*, 01 2008.

[63] V. Latora and M. Marchiori, "Efficient behavior of small-world networks," *Physical Review Letters*, vol. 87, no. 19, oct 2001. [Online]. Available: https://doi.org/10.1103%2Fphysrevlett.87.198701

[64] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 10, p. P10008, oct 2008. [Online]. Available: https://doi.org/10.1088%2F1742-5468%2F2008%2F10%2Fp10008

[65] M. E. J. Newman, "Assortative mixing in networks," *Physical Review Letters*, vol. 89, no. 20, oct 2002. [Online]. Available: https://doi.org/10.1103%2Fphysrevlett.89.208701

[66] A. Hagberg, P. Swart, and D. Chult, "Exploring network structure, dynamics, and function using networkx," 01 2008.

[67] M. Gustineli, "A survey on recently proposed activation functions for Deep Learning," pp. 1–4, 2022. [Online]. Available: http://arxiv.org/abs/2204.02921

[68] "A gentle introduction to mini-batch gradient descent and how to configure batch size." [Online]. Available: https://machinelearningmastery.com/gentle-introduction-mini-batch-gradient-descent-configure-batch-size/

[69] P. Munro *et al.*, "Backprop," pp. 69–73, 2011.

[70] "Lecture 4: Backpropagation and neural networks." [Online]. Available: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture4.pdf

[71] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2015 International Conference on Computer Vision, ICCV 2015, pp. 1026–1034, 2015.

[72] K. M. A. Adweb, N. Cavus, and B. Sekeroglu, "Cervical Cancer Diagnosis Using Very Deep Networks over Different Activation Functions," *IEEE Access*, vol. 9, pp. 46 612–46 625, 2021.

[73] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," pp. 1–18, 2012. [Online]. Available: http://arxiv.org/abs/1207.0580

[74] B. Mele and G. Altarelli, "Lepton spectra as a measure of b quark polarization at LEP," *Physics Letters B*, vol. 299, no. 3-4, pp. 345–350, 1993.

[75] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *32nd International Conference on Machine Learning, ICML 2015*, vol. 1, pp. 448–456, 2015.

[76] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[77] Universidad de Sevilla, "Procesamiento de imágenes con MATLAB," *Lectura de Imagenes*, vol. 6, no. lenguaje M, p. 20, 2014. [Online]. Available: http://asignatura.us.es/imagendigital/Matlab_PID_1314.pdf%0Ahttp://lonely113.blogspot.com

[78] R. P. Grimaldi, *Matemáticas discretas y combinatoria : una introducción con aplicaciones*. Pearson Educación, 1998. [Online]. Available: https://books.google.com.uy/books?id=lHqqjoR0b1YC

[79] Y. S. Abu-Mostafa, M. Magdon-Ismail, and H. Lin, *Learning from Data: A Short Course*. AMLBook, 2012.

[80] M. Banko and E. Brill, "Scaling to very very large corpora for natural language disambiguation," in *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, ser. ACL '01. USA: Association for Computational Linguistics, 2001, p. 26–33.

[81] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1-3, pp. 37–52, 1987.

[82] H. B. Curry, "The method of steepest descent for non-linear minimization problems," *Quarterly of Applied Mathematics*, vol. 2, no. 3, pp. 258–261, 1944.

[83] S. Wright, J. Nocedal *et al.*, "Numerical optimization," *Springer Science*, vol. 35, no. 67-68, p. 7, 1999.

[84] M. Hofmann, "Support vector machines-kernels and the kernel trick," *Notes*, vol. 26, no. 3, pp. 1–16, 2006.

Esta página ha sido intencionalmente dejada en blanco.

Índice de tablas

Índice de figuras

Esta es la última página.
Compilado el lunes 12 diciembre, 2022.
http://iie.fing.edu.uy/