Trabajo Final de Grado
Ingeniería en Informática

Redes neuronales convolucionales para el procesamiento de imágenes médicas 3D

Autor: Franco Ercoli
Director: Lic. Agustín Amalfitano
Codirector: Ing. Juan Ignacio Iturriaga

Facultad de Ingeniería
Universidad Nacional de Mar del Plata
2025

Autorización Repositorio Institucional -RINFI

Se presenta conjuntamente con la versión final del Trabajo Final

Repositorio Institucional RINFI, Facultad de Ingeniería, UNMDP

En calidad de TITULARES de los derechos de autor de la obra que se detalla a continuación, y sin infringir según mi conocimiento derechos de terceros, por la presente informo a la Facultad de Ingeniería de la UNMDP mi decisión de concederle en forma gratuita, no exclusiva y por tiempo ilimitado la autorización para:

1) Publicar el texto del trabajo más abajo indicado, exclusivamente en medio digital, en el sitio web de la Facultad y/o Universidad, por Internet, a título de divulgación gratuita de la producción científica generada por la Facultad, a partir de la fecha especificada.
2) Permitir a la Biblioteca que, sin producir cambios en el contenido, establezca los formatos de publicación en la web para su más adecuada visualización y la realización de copias digitales y migraciones de formato necesarias para la seguridad, resguardo y preservación a largo plazo de la presente obra:

Autor : Franco Ercoli Documento: 43.054.843 Teléfono: 2236944699 E-mail: franco.ercoli7@gmail.com	
Director/a: Agustín Amalfitano Documento: 32383916 Leg. 16074	 Firma Director/a
Codirector/a: Juan Ignacio Iturriaga Documento: 28728614 Leg. 18199	 Firma Codirector/a

2. Título obtenido: Ingeniero en Informática

3. Identificación/Título de la Obra: Redes Neuronales Convolucionales para el procesamiento de imágenes médicas 3D.

4.  AUTORIZO la publicación bajo con la licencia Creative Commons BY-NC-ND Atribución-NoComercial-Sin Obra Derivada.

5. **Nota de Embargo:** Para aquellas obras que NO pueden ser de acceso a texto completo por razones de acuerdos previos con empresas o instituciones; por razones de índole comercial u otras razones; se procederá según lo establecido en Art. 6 de la Ley 26899 de Repositorios digitales institucionales de acceso abierto:

> **ARTICULO 6º —** En caso que las producciones científico-tecnológicas y los datos primarios estuvieran protegidos por derechos de propiedad industrial y/o acuerdos previos con terceros, los autores deberán proporcionar y autorizar el acceso público a los metadatos de dichas obras intelectuales y/o datos primarios, comprometiéndose a proporcionar acceso a los documentos y datos primarios completos a partir del vencimiento del plazo de protección de los derechos de propiedad industrial o de la extinción de los acuerdos previos antes referidos.
>
> Asimismo, podrá excluirse la difusión de aquellos datos primarios o resultados preliminares y/o definitivos de una investigación no publicada ni patentada que deban mantenerse en confidencialidad, requiriéndose a tal fin la debida justificación institucional de los motivos que impidan su difusión. Será potestad de la institución responsable en acuerdo con el investigador o equipo de investigación, establecer la pertinencia del momento en que dicha información deberá darse a conocer. A los efectos de la presente ley se entenderá como "metadato" a toda aquella información descriptiva sobre el contexto, calidad, condición o características de un recurso, dato u objeto, que tiene la finalidad de facilitar su búsqueda, recuperación, autentificación, evaluación, preservación y/o interoperabilidad.

En razón de lo expuesto, si el Trabajo se encuentra comprendido en el caso de que su producción esté protegida por derechos de Propiedad Industrial y/o acuerdos previos con terceros que implique la confidencialidad de los mismos, el/la directora/a debe indicar a continuación motivos y fecha de finalización del embargo.

Trabajo Final de Grado
Ingeniería en Informática

Redes neuronales convolucionales para el procesamiento de imágenes médicas 3D

Autor: Franco Ercoli
Director: Lic. Agustín Amalfitano
Codirector: Ing. Juan Ignacio Iturriaga

Facultad de Ingeniería
Universidad Nacional de Mar del Plata
2025

Agradecimientos

A mis padres:
Por su apoyo incondicional a lo largo de toda mi carrera universitaria. Su fortaleza emocional, su ejemplo de perseverancia y su confianza en mí.

A mi hermano:
Por su perspectiva fresca y su capacidad para escuchar activamente.

A mi novia:
Por su apoyo emocional, su paciencia infinita y por su manera de convertir el estrés en risas.

A mis amigos:
Por recordarme la importancia de mantener el equilibrio entre el estudio y la vida personal.

A la Facultad de Ingeniería:
Por brindarme una formación académica sólida y recursos esenciales para mi desarrollo profesional.

A mis profesores:
Por su dedicación en la transmisión de conocimientos y su pasión por la ingeniería, que inspiraron mi curiosidad y compromiso con la disciplina.

A los directores del proyecto:
Por sus aportes y su confianza en mis capacidades para alcanzar los objetivos planteados.

Índice

Índice....3
Capítulo 1: Introducción....6
1.1 Objetivos....7
1.2 Estructura del informe....7
Capítulo 2: Marco Teórico....9
2.1 Planteo del Problema....9
2.2 Estado del Arte....9
2.3 Imágenes Digitales....11
2.4 Imágenes Médicas....13
2.4.1 Imágenes Médicas 3D....16
2.4.2 Formato DICOM....20
2.5 Redes Neuronales Artificiales....24
2.6 Aprendizaje Profundo....27
2.6.1 Estructura de las Redes Neuronales Profundas....28
2.6.2 Características del Aprendizaje Profundo....30
2.7 Redes Neuronales Convolucionales....32
2.7.1 Ventajas de las CNN....34
2.7.2 Regularización en CNN 3D....36
2.7.3 Transfer Learning....38
2.8 Desafíos y Consideraciones....39
Capítulo 3: Desarrollo de métodos y algortimos....41
3.1 Bases de datos médicas....41
3.1.1 Aspectos a considerar de los datos médicos....41
3.1.2 Datos de entrenamiento: Tomografías de pulmón....44
3.2 Preprocesamiento de datos....46
3.3 Modelos de CNN 3D....48
3.3.1 Modelo 1....49
3.3.2 Modelo 2....50
3.3.3 Modelo 3....51
3.3.4 Modelo de Transfer Learning....52
3.4 Configuración de los entrenamientos....55
3.5 Tecnologías....56
3.6 Metodología de trabajo y experimentación....58
Capítulo 4: Resultados....63
4.1 Modelos de redes Ad Hoc....64
4.2 Modelos de Transfer Learning....68
Capítulo 5: Conclusiones....71
5.1: Gestión del proyecto....72
Bibliografía....76

Apéndice A: Definiciones previas.....82
Apéndice B: Código.....87

Resumen

Las imágenes médicas 3D son herramientas fundamentales en el diagnóstico moderno, especialmente en patologías complejas como lesiones pulmonares, anomalías cardíacas o tumores cerebrales. No obstante, el análisis manual de estos volúmenes por parte de especialistas no solo es demandante en tiempo, sino que también está sujeto a variabilidad y posibles errores humanos. En este contexto, la inteligencia artificial, y en particular las redes neuronales convolucionales, han demostrado un potencial transformador en el procesamiento automatizado de imágenes. Sin embargo, si bien su uso está ampliamente difundido en imágenes 2D, su adaptación a datos volumétricos en 3D presenta nuevos desafíos tanto computacionales como metodológicos.

A través del uso de CNN 3D, se buscó explorar el potencial de estas tecnologías, evaluando su rendimiento sobre un conjunto de datos públicos y reflexionando sobre su aplicabilidad, limitaciones y posibilidades de mejora. La propuesta se construye desde una perspectiva práctica y metodológica, poniendo el foco en la validación experimental y en el diseño de soluciones alineadas con necesidades concretas del ámbito médico.

Concretamente, se utilizaron tomografías computarizadas 3D de pulmón con el objetivo de entrenar modelos capaces de clasificar si una imagen correspondía a un caso de cáncer o no. Dado que el acceso a datos médicos reales es una de las principales barreras en el desarrollo de IA en salud, este trabajo pone en valor el uso de conjuntos de datos abiertos y las prácticas reproducibles. Se implementaron múltiples experimentos comparativos, incluyendo pruebas con diferentes arquitecturas de CNN 3D y variantes de un modelo preentrenado, cuyos resultados aportan evidencia concreta y replicable que puede servir como base para investigaciones futuras.

Los resultados obtenidos fueron comparables con los alcanzados en trabajos anteriores, incluso frente a propuestas que emplearon bases de datos privadas adicionales. El trabajo desarrollado representa un aporte concreto al dejar sentadas herramientas reutilizables y una base experimental validada para futuras investigaciones en el ámbito de imágenes médicas 3D asistidas por inteligencia artificial.

Este proyecto se enmarca en la continuidad de las investigaciones realizadas en el Laboratorio de Procesamiento de Imágenes (LPI - ICyTE) de la Facultad de Ingeniería de la Universidad Nacional de Mar del Plata, donde se desarrollaron herramientas aplicadas a imágenes médicas 2D. La presente propuesta extiende esas capacidades al análisis tridimensional, incorporando técnicas y modelos adaptados al procesamiento de volúmenes.

Capítulo 1: Introducción

La detección temprana de enfermedades es un pilar fundamental para mejorar las tasas de supervivencia y elevar la calidad de vida de los pacientes. En este contexto, las imágenes médicas 3D, como las tomografías computarizadas (TC) y las resonancias magnéticas (RM), desempeñan un papel crucial al proporcionar una visión detallada de las estructuras anatómicas y patológicas. Estas imágenes permiten a los especialistas explorar el cuerpo humano en su complejidad tridimensional, mejorando la exactitud del diagnóstico y la planificación del tratamiento [1].

Sin embargo, el análisis de estas imágenes presenta desafíos debido a la gran cantidad de datos involucrados y la necesidad de identificar patrones sutiles. Tradicionalmente, este proceso ha dependido de la experiencia y pericia de radiólogos altamente capacitados. Aunque este enfoque es efectivo, también introduce limitaciones, como la subjetividad en la interpretación y la posibilidad de que pequeños detalles pasen desapercibidos, además de tiempos prolongados de evaluación. Estas limitaciones generan cuellos de botella en el diagnóstico y posibles retrasos en la toma de decisiones clínicas [2].

Con el objetivo de superar estos obstáculos, el procesamiento automatizado de imágenes digitales ha surgido como una solución para apoyar a los profesionales de la salud en el análisis de imágenes. Los algoritmos de procesamiento digital permiten identificar patrones de manera consistente y objetiva, reduciendo el margen de error humano y acelerando el proceso diagnóstico [3].

Dentro de este ámbito, las redes neuronales convolucionales (CNN) han emergido como una herramienta poderosa y revolucionaria en la automatización del procesamiento de imágenes. Inicialmente desarrolladas para imágenes bidimensionales (2D), las CNN han superado el estado del arte en tareas como la clasificación, segmentación y detección de objetos en diversas aplicaciones, incluidas las imágenes médicas. Estas redes se destacan por su capacidad para aprender automáticamente las características más relevantes de las imágenes sin necesidad de intervención manual, optimizando tanto la precisión como la eficiencia [4].

A pesar de los importantes avances en el procesamiento de imágenes médicas 2D mediante redes neuronales convolucionales, el análisis de imágenes tridimensionales (3D) representa un desafío aún en evolución. Las redes neuronales convolucionales 3D comenzaron a desarrollarse en la década de 2010. Actualmente, el uso de CNN 3D constituye un área activa de investigación, con propuestas metodológicas en constante desarrollo y validación, tanto desde la perspectiva técnica como clínica [5].

Este proyecto, titulado "Redes Neuronales Convolucionales para el Procesamiento de Imágenes Médicas 3D", surge ante la necesidad de resolver problemas médicos complejos mediante el análisis de imágenes tridimensionales. Para ello, se investigarán, propondrán, implementarán y validarán metodologías basadas en CNN que permitan un análisis preciso

y eficiente de imágenes médicas en 3D. Tomando como base el conocimiento adquirido en el procesamiento de imágenes 2D desarrollado en el Laboratorio de Procesamiento de Imágenes (LPI - ICyTE) de la Facultad de Ingeniería de la Universidad Nacional de Mar del Plata, este trabajo busca extender estas capacidades al dominio tridimensional.

1.1 Objetivos

Este proyecto tiene como objetivo principal desarrollar métodos y algoritmos basados en CNN capaces de identificar patologías en imágenes médicas 3D, en particular tomografías computarizadas 3D de pulmón. Para alcanzar este objetivo general, se han establecido los siguientes objetivos específicos:

- Estudiar teóricamente modelos de redes de aprendizaje profundo, considerando diferentes paradigmas, tipo de aplicación y entrenamiento.
- Relevar trabajos específicos de redes de aprendizaje profundo en imágenes médicas 3D.
- Estudiar estrategias de diseño de redes profundas.
- Desarrollar modelos profundos para tareas de clasificación y segmentación en imágenes médicas, considerando problemas abiertos y bases de datos públicas.
- Llegar a resultados experimentales comparables a los del estado del arte que sirvan de base para futuras investigaciones en el laboratorio.

1.2 Estructura del informe

El informe se divide en cinco capítulos, cada uno de los cuales trata de forma organizada los diferentes aspectos del trabajo realizado.

En el primer capítulo se presenta el objetivo general del proyecto, así como los objetivos específicos que guían el desarrollo del trabajo.

El segundo capítulo proporciona una revisión del marco teórico relacionado con las imágenes médicas, las redes neuronales artificiales y el aprendizaje profundo. Establece los conceptos teóricos necesarios para comprender los experimentos y resultados presentados en el informe.

En el tercer capítulo se describe el desarrollo del proyecto, incluyendo las bases de datos médicas utilizadas, los desafíos asociados al manejo de los datos y las técnicas de preprocesamiento aplicadas. Se detallan las arquitecturas de las redes neuronales utilizadas, así como la configuración de los entrenamientos y las tecnologías empleadas en la implementación de los modelos.

El cuarto capítulo presenta los resultados obtenidos de los experimentos realizados.

En el quinto capítulo se exponen las conclusiones generales del trabajo, incluyendo los logros alcanzados, las limitaciones del estudio, las posibles líneas futuras de investigación. y la gestión del proyecto, donde se detallan los procesos, la planificación y los recursos utilizados durante el desarrollo.

Además, el informe incluye apéndices con definiciones previas y todo el código generado para la implementación de las técnicas desarrolladas durante la investigación.

Capítulo 2: Marco Teórico

2.1 Planteo del Problema

El procesamiento de imágenes médicas tridimensionales se encuentra en una etapa emergente dentro del campo del aprendizaje profundo, enfrentando retos técnicos y prácticos que limitan su adopción y estandarización en entornos clínicos. A pesar de los avances logrados en el uso de CNN para análisis volumétrico, aún persisten barreras significativas que dificultan la integración de estas tecnologías en el diagnóstico médico de rutina. Estas dificultades incluyen, entre otras, la optimización de modelos para manejar grandes volúmenes de datos, la adaptación de arquitecturas de redes diseñadas originalmente para imágenes 2D al dominio tridimensional y la interpretación de los resultados generados por las CNN en el contexto de imágenes médicas 3D [6].

El desarrollo de herramientas especializadas para la clasificación, segmentación e interpretación de datos médicos volumétricos constituye un área de investigación activa y de importancia para el progreso de la precisión diagnóstica y la toma de decisiones clínicas basadas en imágenes. El problema radica en la necesidad de diseñar y validar metodologías que no sólo implementen técnicas de CNN 3D de manera eficiente, sino que también garanticen un rendimiento robusto y comprensible desde una perspectiva médica.

A pesar del creciente desarrollo de modelos basados en CNN 3D en distintas áreas, incluyendo el ámbito de las imágenes médicas, aún persisten desafíos importantes relacionados con su estandarización, validación y adopción clínica. Esta falta de integración plantea la necesidad de continuar estudiando, explorando e implementando arquitecturas de CNN 3D. Alcanzar un desempeño comparable al de modelos reportados en la literatura ya representa un aporte valioso frente a esta situación [7].

El presente proyecto busca obtener resultados comparables a los del estado del arte, utilizando un conjunto de datos público y estandarizado de tomografías computarizadas 3D de pulmón, como así también proporcionar herramientas que sean escalables y aplicables en escenarios clínicos reales. En este contexto, la exactitud (accuracy) se refiere a la capacidad del modelo para predecir correctamente los resultados

2.2 Estado del Arte

El procesamiento y análisis de imágenes médicas tridimensionales es un gran desafío debido a la falta de herramientas especializadas y eficaces. Sin embargo, en los últimos años, la incorporación de técnicas de aprendizaje profundo ha transformado significativamente este campo. Estas técnicas han abierto nuevas posibilidades para el análisis de imágenes 3D, como las obtenidas mediante tomografía computarizada,

resonancia magnética y tomografías por emisión de positrones (PET). A pesar de estos progresos, la adopción y estandarización de estas tecnologías en el ámbito médico presentan varios desafíos, que incluyen la implementación técnica y la comprensión profunda de los algoritmos subyacentes para adaptarlos a las necesidades clínicas específicas.

Un trabajo reconocido en esta evolución es la propuesta del 3D U-Net por Çiçek et al. [8]. Este modelo extiende la clásica arquitectura U-Net a datos tridimensionales mediante la utilización de convoluciones 3D y una ruta expansiva que permite recuperar detalles espaciales perdidos. La capacidad del modelo para aprender representaciones jerárquicas a partir de anotaciones dispersas posibilita la segmentación y posterior identificación de tumores y estructuras anatómicas complejas.

Otra propuesta de CNN 3D presentada por Kamnitsas et al. [9] consiste en un modelo que procesa imágenes a diferentes escalas simultáneamente. Esta arquitectura permite analizar tanto los detalles finos como el contexto más amplio del cerebro. Además, incorpora un modelo probabilístico que mejora la precisión al reducir falsos positivos en la identificación de lesiones. Este enfoque demuestra ser eficiente y efectivo en la segmentación de diversas lesiones cerebrales.

Se han desarrollado variantes y mejoras sobre las arquitecturas 3D para optimizar tanto su rendimiento como su eficiencia computacional. Por ejemplo, se han integrado mecanismos de atención y se han empleado estrategias de transfer learning, que permiten inicializar modelos con pesos preentrenados en grandes conjuntos de datos de imágenes naturales, adaptándose posteriormente a tareas específicas del ámbito médico [10]. Estas innovaciones han contribuido a que las arquitecturas 3D evolucionen en términos de precisión diagnóstica y robustez, facilitando su implementación en entornos clínicos donde la disponibilidad de datos etiquetados y recursos computacionales puede ser limitada [11, 12].

En el ámbito de la clasificación de patologías, estudios han demostrado que las redes 3D pueden distinguir entre diferentes condiciones. Por ejemplo, se han empleado para clasificar enfermedades neurodegenerativas, diferenciando entre Alzheimer y otras formas de demencia, o para identificar subtipos de cáncer, como en el caso de la detección de adenocarcinoma de pulmón. Un estudio representativo realizado por Kim et al. [13] desarrolla un modelo 3D basado en transfer learning que fusiona imágenes PET/TC y datos clínicos para predecir la presencia de mutaciones en pacientes con adenocarcinoma de pulmón. Este enfoque no solo demostró altos índices de precisión, sino que también destacó la importancia de integrar información multimodal para obtener diagnósticos más robustos.

Otro estudio hecho por Isensee et al. [14] desarrolla nnU-Net, un framework que automatiza la configuración de la red y la optimización de hiperparámetros para cada conjunto de datos específico. Esta herramienta adaptable demuestra robustez y una adecuada capacidad de generalización en diversas tareas de segmentación médica 3D, lo cuál facilita la implementación clínica de modelos basados en U-Net sin la necesidad de una extensa personalización manual.

2.3 Imágenes Digitales

Una imagen digital se puede entender como una matriz de píxeles, donde cada valor numérico en la matriz corresponde a la intensidad de un píxel en una posición específica. En el caso de una imagen en escala de grises de tamaño m×n, la matriz es bidimensional y tiene m filas y n columnas, donde cada elemento de la matriz representa la intensidad de un píxel, con valores que generalmente van de 0 (negro) a 255 (blanco) para imágenes de 8 bits de profundidad.

Por ejemplo, una imagen en escala de grises de 6x6 píxeles se puede representar como se observa en la Figura 1.

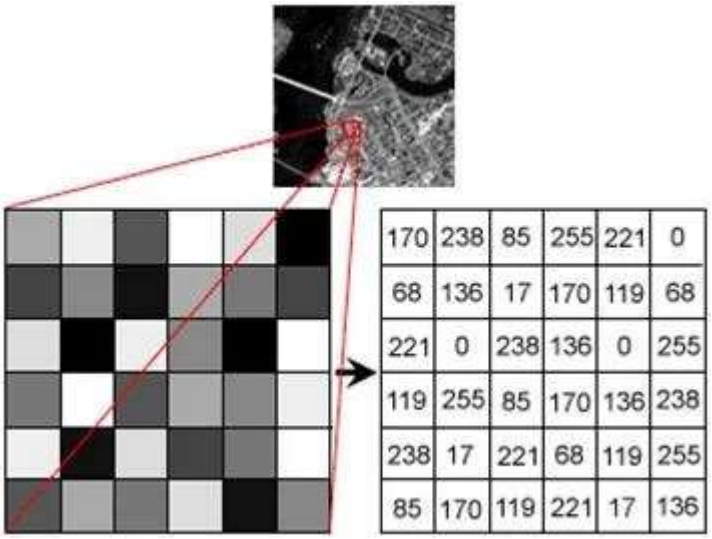


Figura 1: Representación en una matriz de una imagen en escala de grises. [15]

En esta matriz, cada número representa la intensidad de un píxel en su respectiva posición de la imagen. En el caso de una imagen a color, la representación es más compleja y se utiliza un tensor tridimensional (una estructura de datos que extiende el concepto de matriz a múltiples dimensiones), donde cada capa o canal (rojo, verde y azul) se representa con una matriz similar. De esta forma, la imagen se describe por la combinación de estas tres matrices.

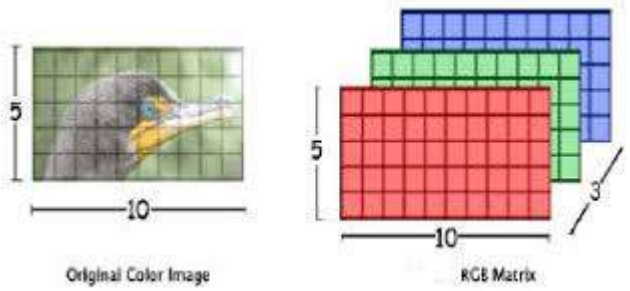


Figura 2: Representación esquemática en una matriz de una imagen a color. [16]

Las imágenes a color se representan mediante una estructura tridimensional con dimensiones m × n × 3, donde m y n denotan respectivamente el ancho y la altura de la imagen en píxeles. El tercer valor, 3, corresponde a los canales de color que la componen. Habitualmente, se emplea el modelo de color RGB (Rojo, Verde, Azul), basado en la combinación aditiva de estos tres colores primarios. Así, la posición (x,y) de una imagen de tamaño 100×100 se describe por tres valores de intensidad correspondientes a los colores rojo, verde y azul de ese píxel.

Filtro (Kernel)

En una CNN 2D, los filtros (o kernels) son matrices que recorren la imagen en dos dimensiones (ancho y alto), extrayendo características como bordes o texturas [17]. El tamaño del kernel, determina el área de la imagen que el filtro abarca a medida que recorre la imagen.

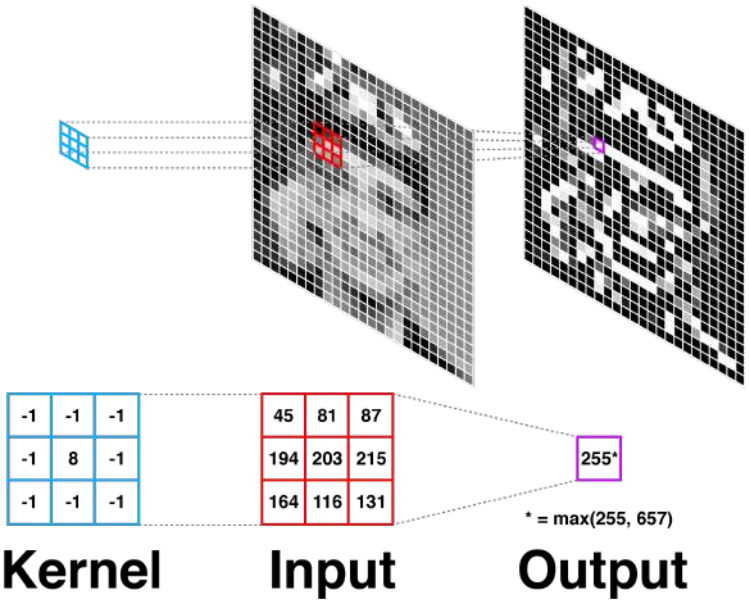


Figura 3: Ejemplo de Kernel 2D y el resultado tras su aplicación. [18]

Operación de Convolución

La operación de convolución es el proceso mediante el cual un filtro se aplica a una imagen. Consiste en desplazar el kernel sobre la imagen y, en cada posición, realizar lo siguiente:

- Multiplicación Elemento a Elemento: Cada valor del filtro se multiplica por el valor del píxel correspondiente en la región de la imagen.
- Suma de los Productos: Se suman todos los productos obtenidos para generar un único valor.
- Asignación en el Mapa de Características: Este valor resultante se asigna a la posición correspondiente en el mapa de características de salida.

Este proceso permite que la red detecte patrones y características específicas en diferentes partes de la imagen, facilitando el aprendizaje de representaciones complejas durante el entrenamiento. La Figura 3 muestra cómo un kernel 2D se aplica sobre la imagen de entrada para generar una salida, destacando características relevantes como pueden ser los bordes.

2.4 Imágenes Médicas

Las imágenes médicas han sido fundamentales en la práctica clínica y la investigación durante décadas, permitiendo a los profesionales de la salud visualizar estructuras internas del cuerpo humano de manera no invasiva. Tradicionalmente, la mayoría de las imágenes médicas se han capturado en dos dimensiones, proporcionando vistas planas de la anatomía del paciente. Sin embargo, con el avance de la tecnología, se ha hecho posible capturar imágenes tridimensionales, que revelan con mayor precisión las relaciones espaciales, la forma y el volumen de los órganos y tejidos. Esta capacidad ha abierto nuevas posibilidades en el diagnóstico y tratamiento de enfermedades.

Las imágenes médicas 2D incluyen modalidades como la radiografía, la tomografía computarizada (TC). la ecografía y algunas modalidades de resonancia magnética (RM). Estas imágenes representan información bidimensional (ancho y alto) y se utilizan ampliamente en la práctica clínica por su accesibilidad, velocidad y eficacia en el diagnóstico médico [19].

Planos de Referencia Anatómicos

Los planos axial, coronal y sagital son planos de referencia fundamentales para describir la ubicación y orientación de estructuras corporales.

El plano Axial es horizontal y divide el cuerpo en una parte superior (craneal) e inferior (caudal). Es el más utilizado en imágenes médicas como las TC, donde se observan cortes secuenciales del cuerpo desde arriba hacia abajo. Se utiliza para evaluar la anatomía interna en secciones transversales, facilitando el análisis de órganos como el cerebro, el abdomen y las extremidades.

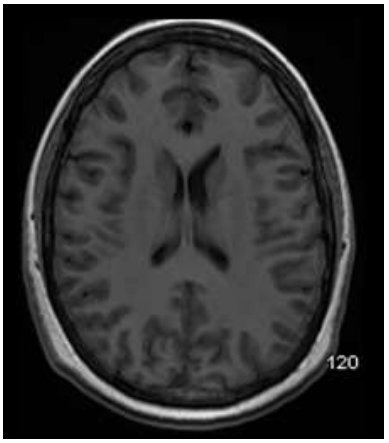


Figura 4: Plano Axial en RM de cerebro.

El plano Coronal es vertical y divide el cuerpo en una parte anterior (frontal) y posterior (dorsal). Es comúnmente empleado para evaluar estructuras en la vista frontal, como en estudios que analizan la cara, el tórax o el abdomen, brindando una perspectiva clara de la disposición de tejidos blandos y órganos en relación con la superficie corporal.

Figura 5: Plano Coronal en RM de cerebro.

El plano Sagital es también vertical y divide el cuerpo en mitades derecha e izquierda. Si la división es exactamente por la línea media, se llama plano sagital medio o mediano; si es paralelo a esta línea, se denomina parasagital. Este plano es útil en imágenes que evalúan la simetría corporal y las estructuras del sistema nervioso central, como la médula espinal y el cerebro, en vistas laterales.

Figura 6: Plano Sagital en RM de cerebro.

Radiografía

Es la forma más común de imágenes 2D, utilizando rayos X para capturar una imagen plana de los tejidos internos, principalmente huesos. Es una técnica rápida y de bajo costo, útil para detectar fracturas óseas, infecciones pulmonares y otras condiciones.

Figura 7: Radiografía de pie, plano sagital.

Tomografía Computarizada

Aunque las imágenes TC pueden generar volúmenes 3D, muchas veces se utilizan cortes individuales o conjuntos de cortes que se interpretan como imágenes 2D. Las imágenes TC ofrecen una mayor resolución que las radiografías y son útiles para detectar lesiones internas, tumores y otras anomalías.

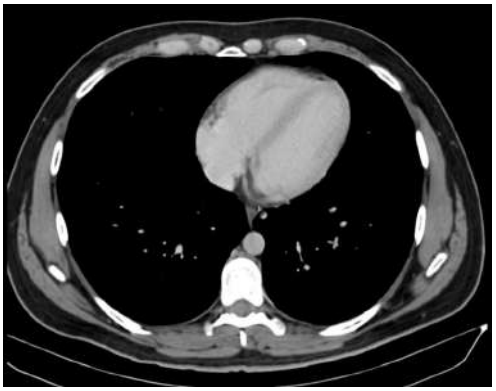


Figura 8: Tomografía Axial Computarizada de tórax.

Ecografía

Utiliza ondas de sonido para crear imágenes 2D de los órganos internos, los tejidos blandos y el flujo sanguíneo. Es particularmente útil en la obstetricia y la evaluación de órganos abdominales.

Figura 9: Ecografía de embarazo.

2.4.1 Imágenes Médicas 3D

A pesar del éxito de las CNN en imágenes 2D, la naturaleza tridimensional de la anatomía humana y de las patologías hace que su representación y análisis se vean limitados en un espacio bidimensional, lo que ha impulsado el desarrollo y la adopción de imágenes médicas 3D. Estas imágenes permiten una visualización completa y detallada de estructuras internas, lo que facilita la planificación quirúrgica, la detección de tumores y la evaluación de la progresión de enfermedades. [20].

Las imágenes médicas 3D se obtienen utilizando tecnologías avanzadas como la tomografía computarizada en su modalidad completa, la resonancia magnética y la tomografía por

emisión de positrones (PET, del inglés *positron emission tomography*), entre otras. A diferencia de las imágenes 2D, que ofrecen solo una vista plana, las imágenes 3D proporcionan información volumétrica, permitiendo la visualización de las estructuras anatómicas desde múltiples ángulos.

Tomografía Computarizada

Utiliza rayos X para crear imágenes detalladas de las estructuras internas del cuerpo. Los rayos X giran alrededor del cuerpo y crean imágenes transversales que se combinan para formar una imagen tridimensional. La TC es especialmente útil para visualizar huesos, tejidos blandos y vasos sanguíneos.

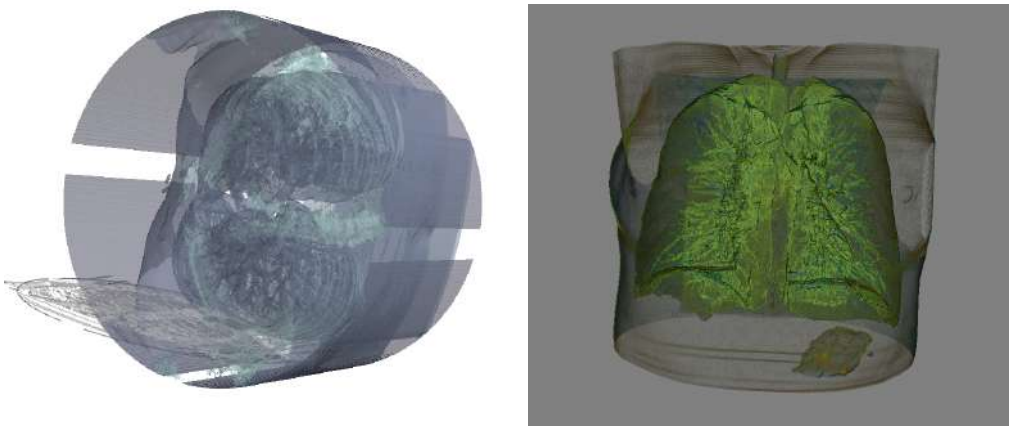


Figura 10: Tomografías 3D de pulmón.

Resonancia magnética

Emplea campos magnéticos potentes y ondas de radio para producir imágenes detalladas de los órganos y tejidos internos. A diferencia de la TC, la RM no utiliza radiación ionizante. Es particularmente eficaz para observar el cerebro, la médula espinal, las articulaciones y los tejidos blandos.

Ecografía

Utiliza ondas sonoras de alta frecuencia para generar imágenes de las estructuras internas del cuerpo. Es una técnica segura y no invasiva, comúnmente utilizada para monitorizar el desarrollo fetal.

Tomografía por Emisión de Positrones

Es una modalidad de imagen nuclear que utiliza pequeñas cantidades de material radiactivo para observar los procesos metabólicos en el cuerpo. Es especialmente útil para la detección y monitoreo de enfermedades como el cáncer, así como para evaluar la función del corazón y el cerebro.

Voxels: La Unidad de Datos de las imágenes 3D

La transición a imágenes 3D implica un aumento significativo en la complejidad de los datos. Las imágenes 3D se representan utilizando *voxels*, que son elementos volumétricos análogos a los píxeles en 2D. Cada *voxel* contiene información sobre la intensidad en un punto específico dentro del volumen 3D. Esta representación volumétrica permite un análisis más detallado y preciso de las estructuras anatómicas.

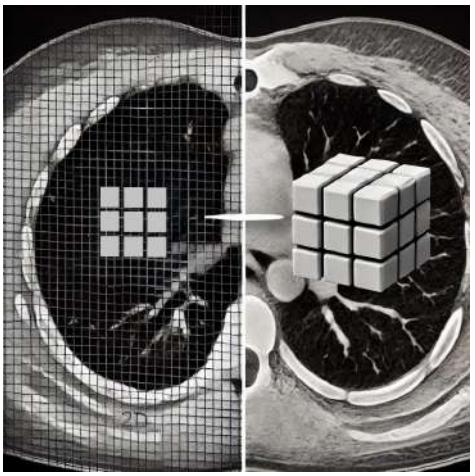


Figura 11: Ejemplo de conjuntos: pixel vs voxel.

Las imágenes médicas se adquieren con diferentes principios de imagenología. Las características de estas imágenes difieren en términos de resolución espacial, rango de intensidad de imagen, tamaño de la imagen y ruido. Además, varían en términos de dimensionalidad y representación de datos. Los datos 2D utilizan principalmente una representación euclidiana relativamente simple, que describe cada punto de datos utilizando valores de intensidad de píxel. En las imágenes médicas, estos valores de píxel representan el estado de la estructura anatómica en consideración. Por ejemplo, la intensidad del píxel de las imágenes de rayos X varía con la absorción de radiación, mientras que depende de la presión acústica en las imágenes de ultrasonido o de la amplitud de la señal de radiofrecuencia en la resonancia magnética [21].

La representación de datos 3D es más compleja. Sin embargo, se han reportado varias representaciones estándar en la literatura, como proyecciones, representaciones volumétricas, nubes de puntos y grafos [22]. La mayoría de los datos de imágenes médicas 3D pertenecen a la representación volumétrica para modelar datos 3D describiendo cómo se distribuye el objeto 3D a través de los tres ejes perpendiculares. Al adquirir datos médicos 3D, el dispositivo escanea las partes del cuerpo en cualquiera de los tres planos: axial (de adelante hacia atrás), coronal (de arriba hacia abajo) y sagital (de lado a lado). Normalmente, se adquieren múltiples cortes 2D en el área en consideración y se apilan para producir volúmenes de imágenes 3D utilizando técnicas adecuadas de registro de imágenes.

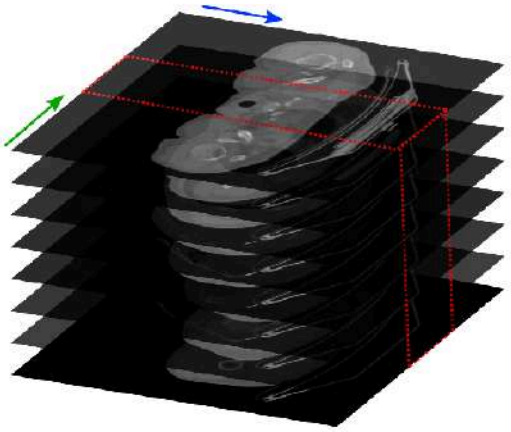


Figura 12: Pila de cortes de una TC.

Aunque las técnicas de imágenes 3D tienen numerosas ventajas sobre las imágenes 2D, también presentan varias limitaciones que pueden restringir su adopción en ciertos contextos clínicos. Una de las principales desventajas es el aumento significativo en los requisitos de almacenamiento de datos. Las imágenes tridimensionales, al capturar información en múltiples planos, generan volúmenes de datos considerablemente mayores que las imágenes bidimensionales. Esto implica la necesidad de infraestructuras de almacenamiento robustas, lo que puede incrementar los costos de implementación y mantenimiento en hospitales y centros de investigación. Además, el manejo de estos grandes volúmenes de datos puede ralentizar los procesos de análisis y visualización, especialmente si no se cuenta con sistemas computacionales adecuados para procesar la información de manera eficiente.

Otra limitación importante es el costo asociado a la adquisición de imágenes 3D. Las modalidades de imagenología que permiten la captura de imágenes tridimensionales, como la tomografía computarizada de alta resolución, la resonancia magnética 3D y la tomografía por emisión de positrones, suelen requerir equipos especializados que son significativamente más costosos que aquellos utilizados para imágenes 2D. Estos equipos también requieren un mantenimiento constante y la operación por parte de técnicos y profesionales capacitados, lo que aumenta los gastos operativos. Además, los tiempos de adquisición de imágenes suelen ser más largos para las modalidades 3D, lo que puede reducir el flujo de pacientes y aumentar los costos por estudio. Este aspecto también puede ser incómodo para los pacientes, ya que algunos procedimientos de imagenología 3D pueden requerir que permanezcan inmóviles durante períodos más prolongados.

A nivel computacional, el procesamiento de imágenes 3D es más complejo y exige mayores recursos de hardware y software. Los algoritmos de procesamiento y análisis, como los utilizados en la clasificación o en el entrenamiento de redes neuronales convolucionales tridimensionales, requieren una mayor capacidad de cómputo y memoria para manejar los tensores tridimensionales, que son estructuras de datos que representan volúmenes con dimensiones de ancho, alto y profundidad. Esto puede convertirse en un desafío en entornos donde la infraestructura computacional es limitada, dificultando la aplicación de estas

técnicas de manera accesible y eficiente. En contraste, las imágenes 2D pueden ser analizadas con mayor rapidez y con menores requisitos computacionales, lo que facilita su implementación en una amplia gama de escenarios clínicos.

2.4.2 Formato DICOM

El formato DICOM (*Digital Imaging and Communications in Medicine*) es un estándar internacional para el intercambio, almacenamiento y transmisión de imágenes médicas digitales, así como de información relacionada con la salud. Fue desarrollado por el Comité Europeo de Normalización (CEN) y el Comité Nacional Americano de Estándares (ANSI) en respuesta a la necesidad de unificar y estandarizar el manejo de imágenes médicas digitales en entornos de salud [23].

DICOM organiza los datos en una estructura jerárquica basada en elementos de datos. Cada elemento de datos contiene información específica sobre la imagen o el estudio médico. Estos elementos de datos están organizados en un formato de diccionario y pueden contener información como la identificación del paciente, datos del estudio, parámetros de adquisición de la imagen, y los píxeles de la propia imagen.

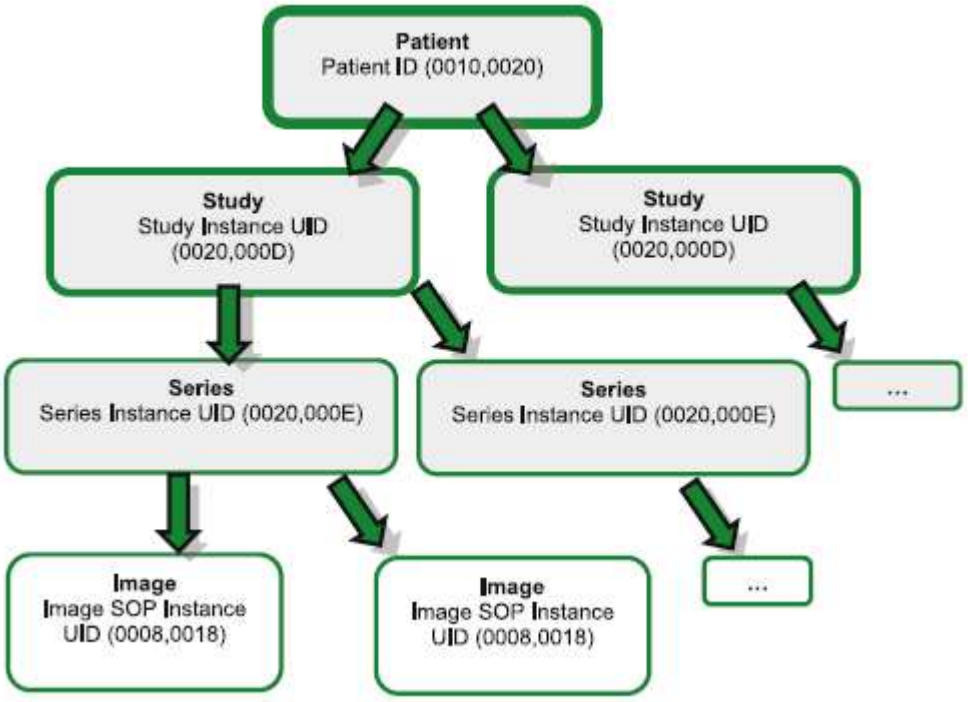


Figura 13: Niveles de la jerarquía de datos en DICOM. [23]

Una de las características más importantes de DICOM es su capacidad para almacenar una amplia gama de metadatos junto con la imagen médica. Estos metadatos proporcionan información detallada sobre la imagen y el estudio médico asociado. Esto incluye información sobre la modalidad de imagen utilizada (por ejemplo, radiografía, tomografía

computarizada, resonancia magnética), el fabricante del equipo de imágenes, la configuración de adquisición de la imagen, y cualquier procesamiento realizado en la imagen.

El estándar DICOM es compatible con una amplia variedad de modalidades de imagen médica. Esto incluye radiografías, tomografías computarizadas, resonancias magnéticas, ecografías, mamografías, imágenes nucleares y más. La capacidad de DICOM para manejar múltiples modalidades de imagen asegura que las imágenes médicas de diferentes dispositivos y fabricantes puedan ser almacenadas y compartidas de manera interoperable.

DICOM facilita el intercambio de datos entre diferentes sistemas de imágenes médicas. Esto incluye dispositivos de imágenes médicas, estaciones de trabajo de radiología, sistemas de información hospitalaria (HIS) y sistemas de información de imágenes médicas (PACS). El estándar DICOM permite que los profesionales de la salud accedan y compartan fácilmente las imágenes y la información clínica del paciente, lo que es fundamental para la colaboración interdisciplinaria y la prestación de atención de calidad al paciente.

Por último, este formato incluye características de seguridad y privacidad para proteger la confidencialidad de los datos del paciente. Esto puede incluir el cifrado de datos, la autenticación de usuarios y el control de acceso a la información médica sensible. Estas características ayudan a garantizar que las imágenes y la información del paciente estén protegidas contra accesos no autorizados y cumplan con las regulaciones de privacidad de datos en el ámbito de la salud, como la Ley de Portabilidad y Responsabilidad del Seguro Médico (HIPAA) en los Estados Unidos.

Metadatos DICOM

En el contexto de la imagenología 3D, varios atributos de los metadatos DICOM son esenciales para comprender las relaciones espaciales y las dimensiones físicas de los cortes de imagen.

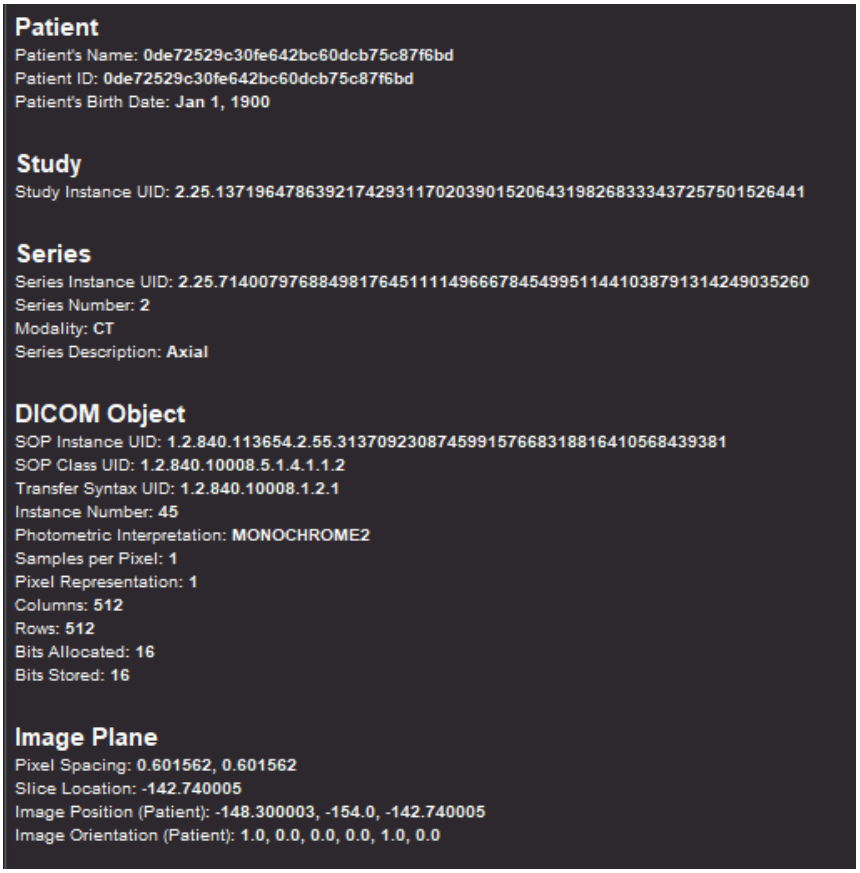


Figura 14: Atributos de DICOM.

Espaciado de píxeles

El espaciado de píxeles, se refiere a la distancia física entre los centros de píxeles adyacentes en la imagen de un paciente. Esta distancia se expresa mediante un par de valores numéricos, que indican el espaciado entre filas adyacentes y entre columnas adyacentes, ambos medidos en milímetros (mm). Esto permite determinar la resolución espacial de la imagen en las dimensiones horizontales y verticales.

Orientación de la imagen

La orientación de la imagen, describe la relación espacial entre la imagen y el paciente Esto define cómo se orienta la imagen en relación con las coordenadas físicas del paciente, lo cual es fundamental para garantizar la correcta interpretación y alineación de las imágenes en los estudios tridimensionales.

Posición de la imagen

La posición de la imagen, proporciona las coordenadas tridimensionales (x, y, z) de la esquina superior izquierda de la imagen, que corresponde al centro del primer voxel transmitido. Estas coordenadas se miden en milímetros y son esenciales para localizar la imagen dentro del espacio físico del paciente.

Grosor del corte

El grosor del corte hace referencia al grosor de cada sección transversal de la imagen, medido en milímetros. Este parámetro es clave para evaluar la calidad de la reconstrucción tridimensional de las imágenes, ya que un menor grosor de corte suele proporcionar una mayor resolución en el eje axial.

Espaciado entre cortes

El espaciado entre cortes indica la distancia entre cortes adyacentes, medida de centro a centro, en milímetros. Este atributo influye en la calidad de la reconstrucción tridimensional, ya que determina la densidad de información a lo largo del eje longitudinal del paciente.

Ubicación del corte

Por último, la ubicación del corte describe la posición relativa del plano de la imagen dentro del espacio tridimensional del paciente, expresada en milímetros. Esta ubicación es importante para correlacionar el plano de cada imagen con la anatomía real del paciente, facilitando la navegación a través de las diferentes secciones del estudio.

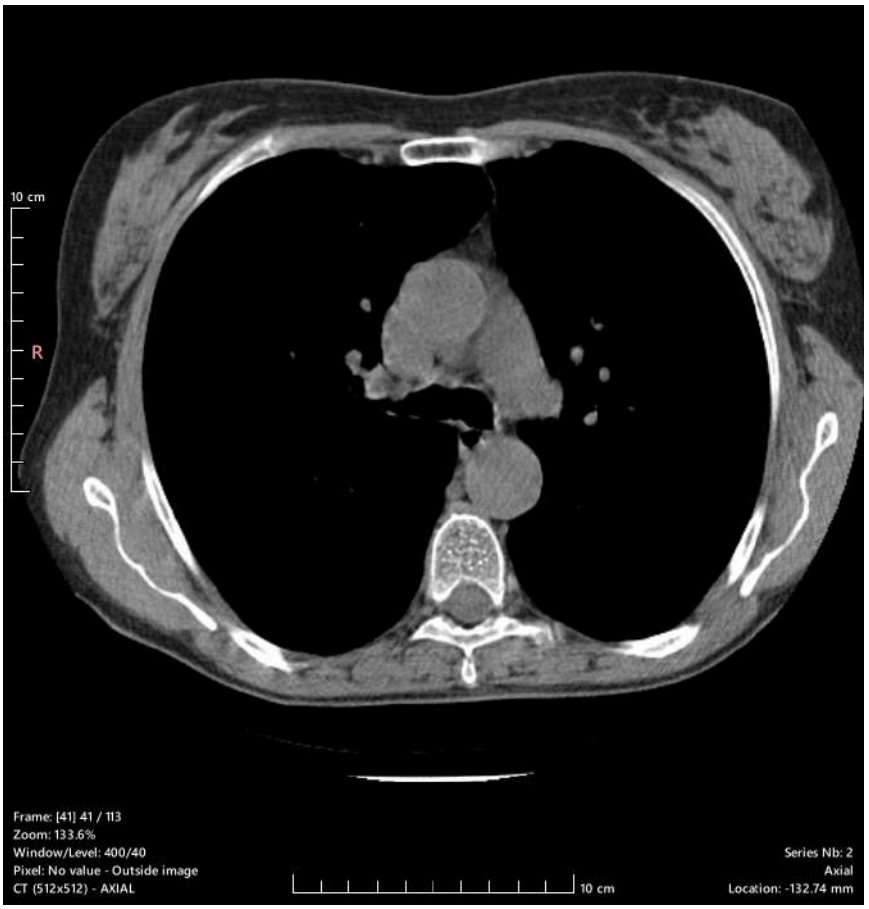


Figura 15: Visualización de un corte axial de una TC 3D con algunos atributos DICOM.

2.5 Redes Neuronales Artificiales

Las redes neuronales artificiales son un tipo de modelo computacional inspirado en la arquitectura y el funcionamiento del cerebro humano. Estas redes son utilizadas para modelar relaciones complejas entre entradas y salidas, y son especialmente efectivas en tareas donde la relación entre los datos no es lineal [24].

Este tipo de modelo puede entrenarse bajo distintos paradigmas de aprendizaje, entre los que se destacan el aprendizaje supervisado, el no supervisado y el por refuerzo. En el aprendizaje supervisado, los modelos son entrenados utilizando conjuntos de datos etiquetados, donde cada entrada tiene asociada una salida deseada, lo que permite optimizar la red para minimizar el error entre la predicción y la verdad conocida. En cambio, el aprendizaje no supervisado trabaja con datos sin etiquetas, buscando patrones o estructuras latentes, como sucede en la agrupación (*clustering*) o la reducción de dimensionalidad. Por su parte, el aprendizaje por refuerzo se basa en la interacción con un entorno, donde el agente aprende mediante recompensas o penalizaciones asociadas a sus acciones [25, 26].

En este trabajo se adopta el enfoque de aprendizaje supervisado, dado que el objetivo es entrenar modelos que clasifiquen imágenes médicas 3D en base a etiquetas clínicas predefinidas, lo que requiere una correspondencia explícita entre datos de entrada y salida durante el proceso de entrenamiento.

Las redes neuronales están compuestas por una serie de unidades básicas llamadas neuronas o nodos. Estas neuronas están organizadas en capas, las cuales se pueden clasificar en tres tipos principales:

- **Capa de Entrada (Input Layer)**: Esta es la primera capa de la red, y su función es recibir los datos de entrada en formato vectorial. Cada nodo en esta capa representa una característica del conjunto de datos de entrada.

- **Capas Ocultas (Hidden Layers):** Estas capas están situadas entre la capa de entrada y la capa de salida. Una red neuronal puede tener una o varias capas ocultas, dependiendo de su complejidad. Las capas ocultas son responsables de realizar la mayor parte del procesamiento de la red, transformando las entradas recibidas en representaciones más abstractas y de alto nivel. Cada neurona en una capa oculta recibe entradas ponderadas de todas las neuronas en la capa anterior, aplica una función de activación y pasa el resultado a la siguiente capa.

- **Capa de Salida (Output Layer):** Esta es la última capa de la red y produce el resultado final, que puede ser una clasificación, una predicción o cualquier otra forma de salida deseada. La cantidad de nodos en la capa de salida depende de la naturaleza del problema; por ejemplo, en una tarea de clasificación binaria, la capa de salida tendría un solo nodo.

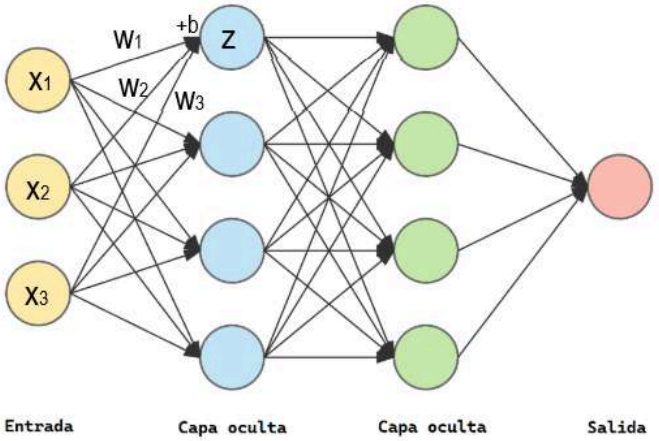


Figura 16: Capas de una red neuronal artificial destinada a clasificación binaria.

Cada neurona en una red neuronal realiza una serie de operaciones matemáticas sobre las entradas que recibe. Este proceso puede describirse en los siguientes pasos:

1. **Cálculo de la Suma Ponderada:** Cada entrada a la neurona se multiplica por un peso asociado. Los pesos son valores que determinan la importancia de cada entrada. Luego, todas las entradas ponderadas se suman junto con un término de sesgo (bias), que es un valor constante que se añade para ajustar la salida de la neurona.

$$z = \sum_{i=1}^{n} w_i \cdot x_i + b$$

   Donde z es la suma ponderada, w_i son los pesos, x_i son las entradas y b es el sesgo.

2. **Función de Activación:** La suma ponderada z se pasa a través de una función de activación que introduce no linealidades en el modelo. Esta no linealidad permite que la red aprenda y represente relaciones complejas en los datos. La función de activación transforma el valor z en una salida a, que luego se pasa a la siguiente capa de la red o se usa como resultado final si está en la capa de salida.

3. **Propagación hacia Adelante:** El proceso anterior se repite capa por capa desde la capa de entrada hasta la capa de salida. Este flujo de información hacia adelante es conocido como propagación hacia adelante (forward propagation).

Entrenamiento de Redes Neuronales Artificiales

El entrenamiento de una red neuronal artificial es un proceso que se basa en la optimización de los parámetros internos de la red, como los pesos y sesgos de las conexiones entre

neuronas, para que la red aprenda a producir salidas precisas a partir de un conjunto de entradas. Este proceso se realiza de manera iterativa, a través de un ciclo de ajustes que implica varios componentes clave, como la función de pérdida, el algoritmo de retropropagación y el método de optimización basado en descenso de gradiente [24].

Función de pérdida: La función de pérdida mide qué tan cerca están las predicciones de la red de los valores reales esperados en el entrenamiento de una red neuronal. La elección de la función de pérdida depende del tipo de problema que se esté abordando. Por ejemplo, el error cuadrático medio (MSE, del inglés *root mean square error*) es comúnmente utilizado en problemas de regresión, ya que calcula el promedio de los cuadrados de las diferencias entre las predicciones y los valores esperados. Para problemas de clasificación, se suele emplear la Entropía Cruzada, que mide la disimilitud entre las distribuciones de probabilidad de las predicciones y las etiquetas reales.

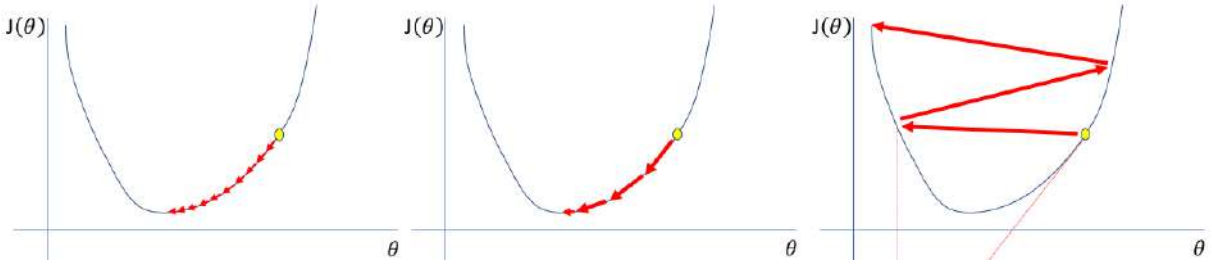
El algoritmo de retropropagación, o *backpropagation* es uno de los métodos más utilizados para ajustar los pesos y sesgos de una red neuronal. Este método calcula el gradiente de la función de pérdida con respecto a cada peso y sesgo de la red, utilizando la regla de la cadena para distribuir el error desde las capas de salida hacia las capas más profundas. De este modo, cada conexión en la red se ajusta de acuerdo con su contribución al error total. La retropropagación no solo facilita la corrección de los errores durante el proceso de entrenamiento, sino que también permite que la red aprenda patrones complejos en los datos.

Descenso de Gradiente: Una vez calculados los gradientes mediante la retropropagación, el ajuste de los pesos se realiza utilizando el método de descenso por gradiente. Este algoritmo busca minimizar la función de pérdida iterativamente ajustando los parámetros de la red en la dirección que reduce el error. La magnitud de cada ajuste está determinada por la tasa de aprendizaje (*learning rate*), un hiperparámetro que controla el tamaño de los pasos que da el algoritmo para alcanzar el mínimo de la función de pérdida. La actualización de cada peso,

$$w_i = w_i - \eta \frac{\partial L}{\partial w_i}$$

donde η es la tasa de aprendizaje, y $\frac{\partial L}{\partial w_i}$ es el gradiente de la función de pérdida con respecto al peso w_i.

La elección de una tasa de aprendizaje adecuada es determinante: una tasa demasiado alta puede provocar oscilaciones y evitar que la red converja al mínimo, mientras que una tasa demasiado baja puede hacer que el entrenamiento sea extremadamente lento.



a)

Tasa de aprendizaje demasiado baja

b)

Tasa de aprendizaje óptima

c)

Tasa de aprendizaje demasiado alta

Figura 17: Convergencia de una red neuronal de acuerdo a la tasa de aprendizaje. [27]

El proceso de entrenamiento de una red neuronal artificial se repite a lo largo de múltiples épocas, lo que significa que el modelo pasa por el conjunto de datos varias veces, ajustando los pesos y sesgos en cada iteración para mejorar su rendimiento. Sin embargo, uno de los desafíos principales durante este proceso es el sobreajuste (*overfitting*), que ocurre cuando el modelo aprende no solo las características generales de los datos, sino también los detalles específicos y el ruido de los mismos. En consecuencia, el modelo pierde capacidad de generalización y tiene un rendimiento deficiente cuando es aplicado en datos no utilizados en el entrenamiento. Para mitigar este problema, una técnica comúnmente utilizada es el early stopping, que consiste en detener el entrenamiento de manera anticipada cuando el rendimiento sobre un conjunto de validación deja de mejorar, evitando así que el modelo continúe ajustándose a aspectos irrelevantes de los datos de entrenamiento.

Para mitigar este problema, se emplean técnicas de regularización, que consisten en introducir restricciones o penalizaciones en el modelo para controlar su complejidad. La regularización ayuda a equilibrar la capacidad del modelo de ajustarse a los datos de entrenamiento sin sacrificar su desempeño en datos nuevos, logrando un mejor balance entre sesgo y varianza.

2.6 Aprendizaje Profundo

El aprendizaje profundo (*deep learning*) es una rama avanzada de la inteligencia artificial que se basa en redes neuronales artificiales de múltiples capas, también conocidas como redes profundas, para modelar y analizar datos complejos.

A diferencia de las redes neuronales clásicas, que suelen estar limitadas a unas pocas capas, el aprendizaje profundo emplea redes neuronales profundas con muchas capas ocultas. Esta mayor profundidad permite a las redes aprender representaciones jerárquicas y características de alto nivel directamente de los datos, algo que no es posible con

arquitecturas más simples. En otras palabras, las redes neuronales profundas son capaces de extraer automáticamente características complejas de los datos sin necesidad de preprocesamiento o extracción de características manuales [28].

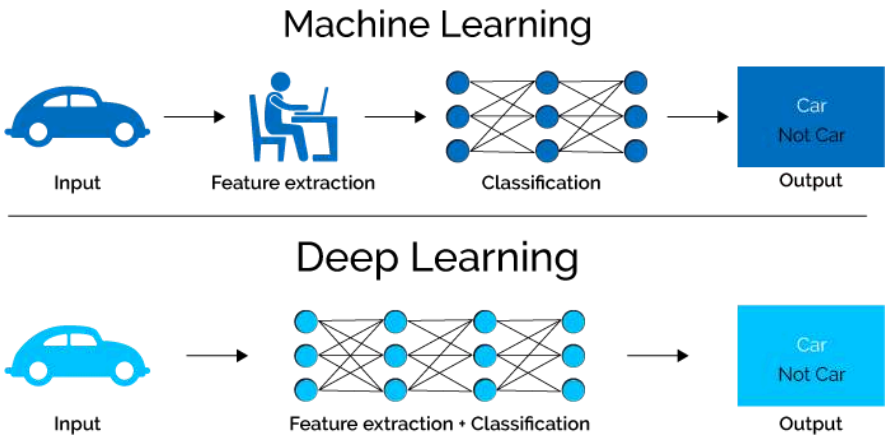


Figura 18: Diferencia entre modelos de Machine Learning y Deep Learning en una clasificación binaria. [29]

A lo largo de los últimos años, este enfoque ha revolucionado el campo de la visión por computadora y el procesamiento del lenguaje natural, y ha demostrado un rendimiento superior en diversas áreas de aplicación en comparación con técnicas tradicionales. Entre sus aplicaciones más destacadas se encuentran la simulación médica, la conducción autónoma, la robótica, la generación de imágenes, y los sistemas de recomendación, entre otros [30].

En comparación con los métodos tradicionales de análisis de imágenes, el aprendizaje profundo permite una automatización de la fase de extracción de características, lo que facilita el análisis de datos de alta dimensión y volúmenes complejos como los que se encuentran en las imágenes médicas tridimensionales. Esta capacidad de realizar automáticamente la extracción de características es una de las razones por las que las CNN han alcanzado un rendimiento superior en tareas de clasificación y segmentación de imágenes [31].

2.6.1 Estructura de las Redes Neuronales Profundas

Las redes neuronales profundas (DNN, del inglés *deep neural networks*) se distinguen de las redes neuronales tradicionales por la profundidad de su arquitectura, es decir, por el número de capas ocultas que contienen. Estas capas adicionales permiten a las DNN aprender representaciones jerárquicas de los datos, donde cada capa sucesiva capta características cada vez más abstractas y de alto nivel. A continuación se describen los distintos tipos de capas existentes, para una red de tipo convolucional:

- **Capas Convolucionales (Conv Layers)**: Estas capas aplican filtros a las entradas para detectar características locales, como bordes, texturas y formas, en las imágenes. Este procedimiento se realiza mediante la operación matemática convolución, la cuál combina la entrada con un filtro para producir un mapa de características. Un mapa de características es una representación intermedia que contiene la respuesta del filtro aplicado sobre una región específica de la imagen. Es decir, muestra las características relevantes detectadas por el filtro en diferentes posiciones de la imagen. A medida que los datos pasan por múltiples capas convolucionales, la red es capaz de identificar características cada vez más complejas, lo que le permite reconocer patrones, formas e incluso objetos completos.

- **Capas de Agrupación (Pooling Layers):** Estas capas se utilizan para reducir la dimensionalidad de los mapas de características generados por las capas convolucionales. El *MaxPooling*, por ejemplo, selecciona el valor máximo en una región específica del mapa de características, lo que ayuda a reducir el tamaño del mapa sin perder información crítica. Esta reducción no solo disminuye el costo computacional, sino que también ayuda a hacer la red más robusta frente a pequeñas variaciones en los datos de entrada. Al realizar esta reducción, la red se vuelve menos sensible a pequeñas variaciones en los datos, como desplazamientos o cambios en la posición de los patrones dentro de la imagen. De esta forma, la red es capaz de reconocer patrones independientemente de su ubicación en la imagen, mejorando su capacidad de generalización.

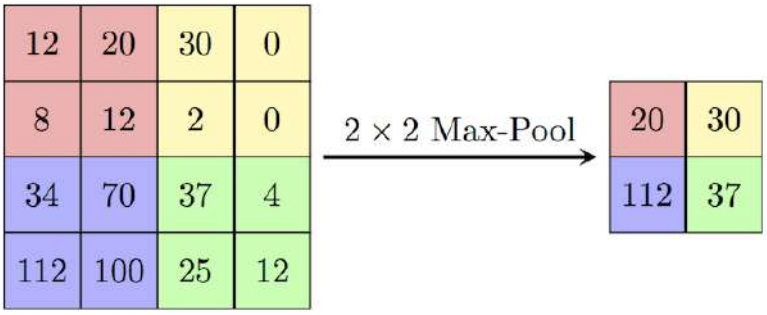


Figura 19: Ejemplo de MaxPooling.

- **Capas Completamente Conectadas (*Fully Connected Layers*):** Estas capas, descritas en la subsección anterior, conectan todas las neuronas de una capa a todas las neuronas de la siguiente capa. Son comunes en las etapas finales de una red neuronal profunda, donde combinan todas las características aprendidas por las capas anteriores para hacer una predicción final o clasificación.

- **Capas de Normalización y Regularización:** Para mejorar la estabilidad y el rendimiento de las redes profundas, se utilizan técnicas como la normalización por lotes, que normaliza las entradas de cada capa para acelerar el entrenamiento y reducir la sensibilidad a la inicialización de los parámetros. La regularización incluye técnicas como *dropout*, que apaga aleatoriamente neuronas durante el entrenamiento para prevenir el sobreajuste.

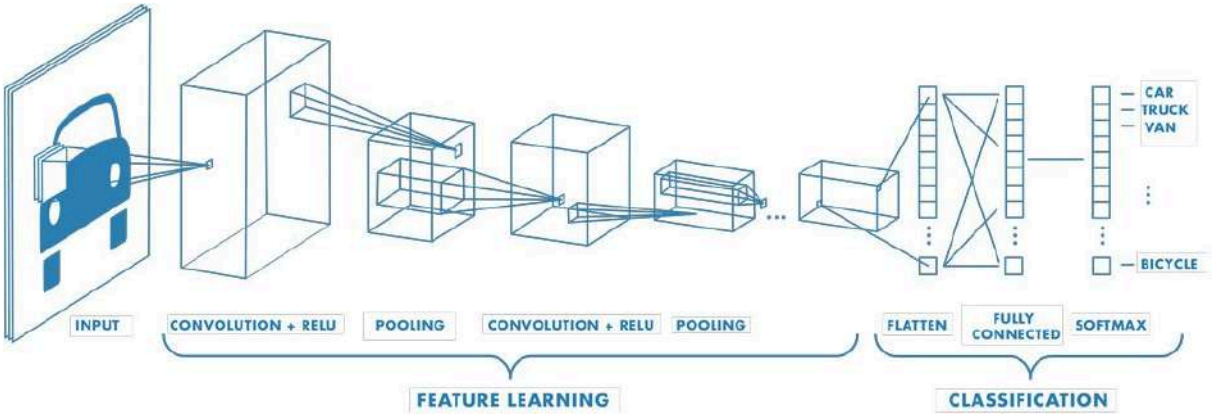


Figura 20: Diagrama de CNN.[18]

El proceso de análisis de una imagen se estructura en dos fases: la fase de extracción de características y la fase de clasificación. Luego de pasar por varias capas convolucionales y de pooling, que se encargan de identificar patrones relevantes y reducir la dimensionalidad de los datos, los mapas de características obtenidos capturan representaciones jerárquicas de la imagen, destacando detalles desde los más simples hasta los más complejos.

En la fase de clasificación, estos mapas de características se aplanan, transformándose en un vector unidimensional que sirve como entrada para las capas completamente conectadas. Estas capas actúan como una red neuronal tradicional y combinan la información extraída previamente, procesándola para identificar relaciones no lineales y patrones complejos presentes en los datos. A medida que los datos atraviesan estas capas, se ajustan los pesos de las conexiones para aprender a diferenciar entre las diferentes clases basándose en las características detectadas.

Finalmente, la red utiliza una capa de salida con una función de activación adecuada. En problemas de clasificación binaria se suele utilizar la función sigmoide, que produce un valor entre 0 y 1, representando la probabilidad de que la imagen pertenezca a una de las dos clases. En problemas de clasificación multiclase se emplea la función softmax, que genera una distribución de probabilidades para cada clase posible. De este modo, se obtiene la predicción final del modelo, indicando la clase a la que pertenece la imagen de entrada con la mayor probabilidad calculada.

2.6.2 Características del Aprendizaje Profundo

El poder del aprendizaje profundo radica en su capacidad para aprender automáticamente características relevantes directamente de los datos. Esta capacidad se debe a:

- **Aprendizaje Jerárquico de Características:** Las redes profundas aprenden características en múltiples niveles de abstracción. Las primeras capas aprenden características básicas, como bordes en una imagen, mientras que las capas más profundas combinan estas características simples para identificar patrones más complejos, como formas, objetos o incluso escenas completas en imágenes.

- **Generalización en Tareas Complejas:** Gracias a su capacidad para procesar y aprender de grandes cantidades de datos, el aprendizaje profundo puede generalizar bien en tareas complejas. Esto significa que puede hacer predicciones precisas en casos que no estaban presentes en el conjunto de entrenamiento, lo que es particularmente útil en aplicaciones como la detección de enfermedades en imágenes médicas o la conducción autónoma.

- **Reducción del Esfuerzo Humano en el Preprocesamiento:** A diferencia de las técnicas de aprendizaje automático tradicionales, que a menudo requieren un preprocesamiento intensivo de datos y la extracción manual de características, el aprendizaje profundo puede trabajar con datos crudos, como imágenes sin procesar, y aún así extraer características útiles.

Existen varias arquitecturas de aprendizaje profundo que han demostrado ser eficaces en diferentes tipos de tareas. Entre las más comunes se encuentran:

- **Redes Neuronales Convolucionales**: Son ampliamente utilizadas en tareas de visión por computadora, como clasificación de imágenes y detección de objetos. Las capas convolucionales permiten aprender características espaciales locales de las imágenes, mientras que las capas de agrupación reducen la dimensionalidad y preservan la información relevante.

- **Redes Neuronales Recurrentes (RNN):** Son adecuadas para el procesamiento de secuencias, como texto y audio. Las conexiones recurrentes permiten que la red tenga memoria, lo que la hace capaz de modelar dependencias temporales en los datos.

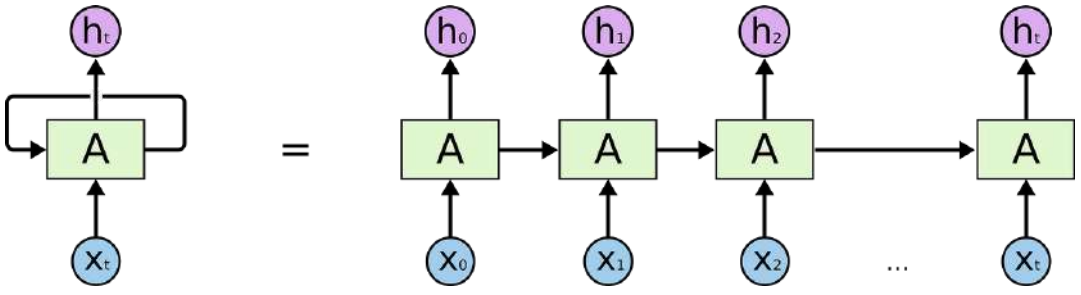


Figura 21: Diagrama de RNN [32].

- **Redes Generativas Adversariales (GAN):** Consisten en dos redes neuronales enfrentadas entre sí: un generador, que crea datos nuevos, y un discriminador, que intenta distinguir entre datos reales y generados. Esta arquitectura se utiliza en la generación de imágenes realistas y en la síntesis de datos.

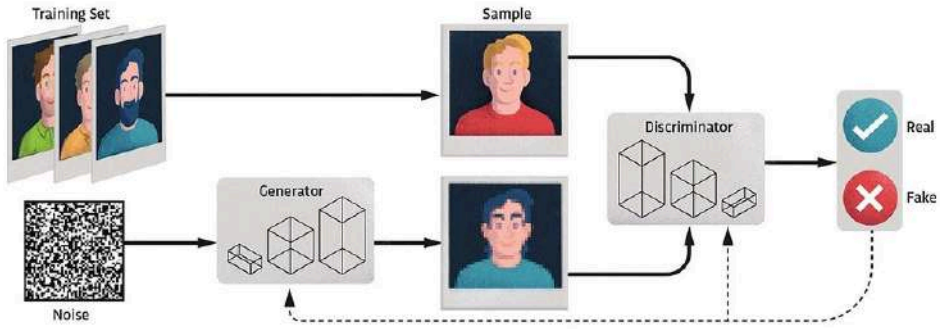


Figura 22: Diagrama de GAN. [33]

2.7 Redes Neuronales Convolucionales

Las CNN, inspiradas en la arquitectura visual del sistema nervioso de los mamíferos, han demostrado su eficacia en la clasificación, segmentación y detección de objetos en imágenes. Su potencial capacidad para capturar información contextual a diferentes niveles de abstracción las convierte en una elección natural para el procesamiento de imágenes médicas.

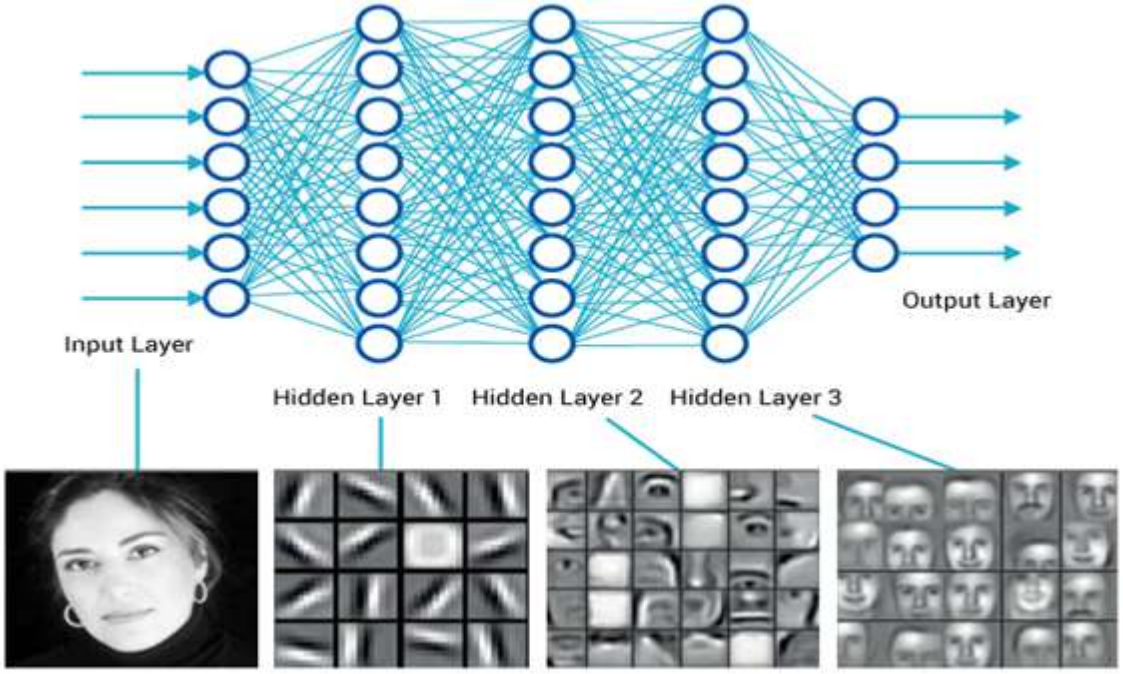


Figura 23: Ejemplo de detección de características en las distintas capas de una DNN. [34]

Funciones de activación

Las funciones de activación son componentes esenciales en las redes neuronales, ya que permiten introducir no linealidades en el modelo. Esta propiedad es fundamental para que la red pueda aprender representaciones complejas de los datos, más allá de lo que permitiría una simple combinación lineal de las entradas.

Entre las funciones más utilizadas se encuentra la ReLU (Rectified Linear Unit), esta función anula los valores negativos y deja pasar los positivos, lo que facilita una rápida convergencia durante el entrenamiento.

La elección de la función de activación depende del tipo de tarea y arquitectura utilizada. En redes convolucionales profundas, ReLU suele ser la elección por defecto debido a su simplicidad y efectividad.

Tras aplicar una función de activación, se suele emplear la normalización por lotes (Batch Normalization) para estabilizar y acelerar el proceso de entrenamiento. Posteriormente, se utiliza MaxPooling para reducir la dimensionalidad de los mapas de activación, extrayendo las características más relevantes en regiones condensadas de la imagen. Finalmente, se aplica dropout para prevenir el sobreajuste, desactivando aleatoriamente un porcentaje de neuronas durante el entrenamiento.

En una CNN 3D, estos mismos conceptos se extienden al espacio tridimensional. Los filtros ahora recorren la imagen 3D (ancho, alto y profundidad) lo que permite captar patrones volumétricos. La activación ReLU y *Batch Normalization* siguen siendo cruciales para la estabilidad del modelo. *MaxPooling3D* reduce la dimensionalidad en las tres dimensiones. Finalmente, se aplica un *dropout* para prevenir el sobreajuste.

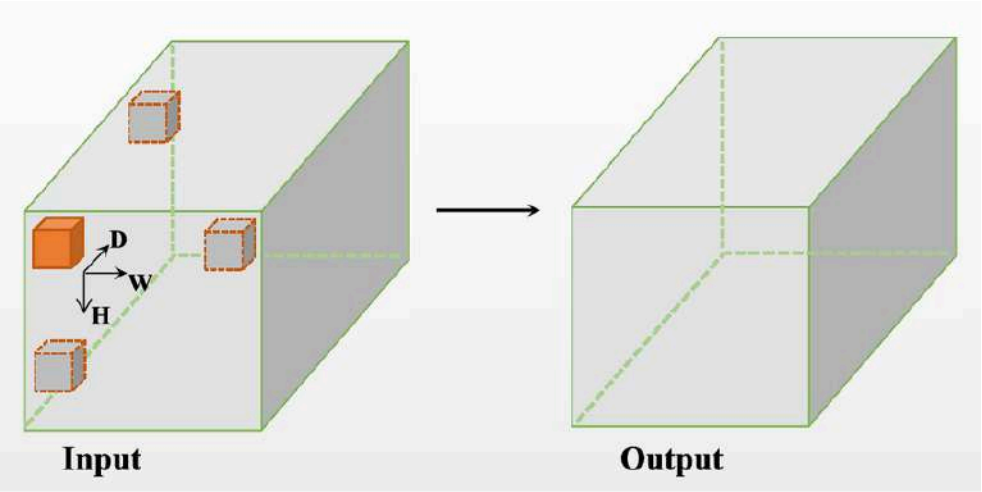


Figura 24: Recorrido de un filtro o kernel 3D. [35]

Las CNN han demostrado un rendimiento notable en la clasificación de imágenes 2D, la detección de anomalías y la segmentación de órganos. Por ejemplo, en mamografías, las

CNN se utilizan para detectar signos tempranos de cáncer de mama, mientras que en radiografías de tórax, se aplican para identificar neumonía, tuberculosis y otras enfermedades pulmonares. Estas redes convolucionales procesan imágenes en dos dimensiones, extrayendo características de bajo nivel, como bordes y texturas, y combinándolas para formar una representación rica que permite realizar tareas complejas de reconocimiento de patrones [36].

2.7.1 *Ventajas de las CNN*

Automatización de la Extracción de Características

Una de las ventajas más significativas de las CNN es su capacidad para automatizar la extracción de características, un proceso que anteriormente requería un conocimiento profundo de las características relevantes para cada tarea específica y de las técnicas adecuadas para extraerlas. En los métodos tradicionales, se debían diseñar manualmente descriptores de características, como SIFT (Scale-Invariant Feature Transform) o HOG (*Histogram of Oriented Gradients*), que pudieran captar información relevante de las imágenes [37, 38].

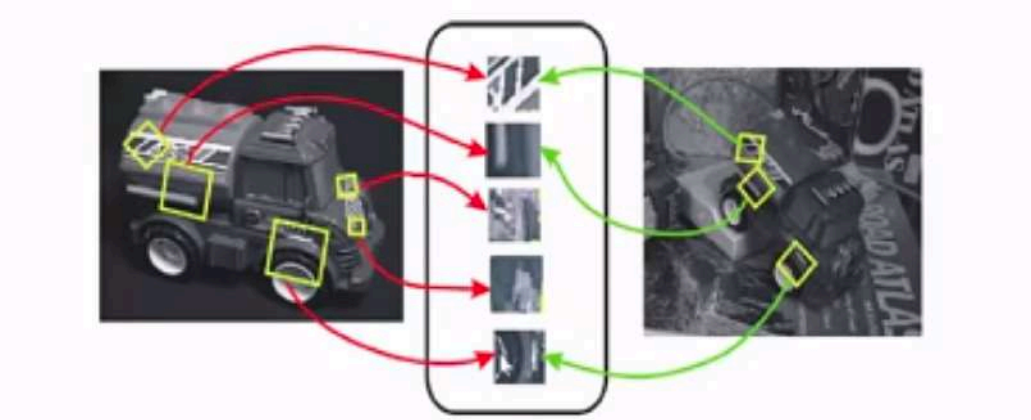


Figura 25: Descriptores de características SIFT. [39]

Figura 26: Descriptores de características HOG. [40]

En contraste, las CNN aprenden de manera autónoma cuáles son las características más importantes a partir de los datos de entrenamiento. Durante el proceso de entrenamiento, los filtros de las capas convolucionales se ajustan para captar patrones de bajo nivel, como bordes y texturas, en las capas iniciales, y patrones de alto nivel, como formas y objetos, en las capas más profundas. Esta capacidad de aprendizaje jerárquico es clave para su éxito en una amplia gama de aplicaciones.

Invarianza Espacial

Las CNN son naturalmente invariantes a pequeñas traslaciones y transformaciones en las imágenes de entrada, gracias a la combinación de operaciones de convolución y *pooling*. La invarianza a la traslación es importante en tareas de visión por computadora, ya que los objetos en las imágenes pueden aparecer en diferentes posiciones, escalas y orientaciones. Esta propiedad permite que las CNN reconozcan objetos incluso cuando no están perfectamente alineados o escalados.

El pooling contribuye a la invarianza a la traslación al seleccionar o combinar valores dentro de regiones, disminuyendo la dependencia de la posición exacta de las características.

Reducción de la Dimensionalidad y Eficiencia Computacional

Otra ventaja de las CNN es su capacidad para reducir la dimensionalidad de las imágenes de entrada sin perder información relevante. Esto se logra mediante el uso de capas de pooling y la estructura jerárquica de la red, lo que permite una representación más compacta de los datos. La reducción de dimensionalidad no solo facilita el manejo de imágenes de alta resolución, sino que también reduce la carga computacional y el almacenamiento necesario, haciendo posible entrenar redes más profundas y complejas en tiempos razonables.

Flexibilidad y Generalización a Diferentes Tareas

Las CNN no están limitadas a un tipo específico de tarea de visión por computadora. Gracias a su arquitectura modular, pueden ser fácilmente adaptadas y personalizadas para una variedad de tareas, incluyendo clasificación de imágenes, detección de objetos, segmentación semántica, reconocimiento de acción en videos y más. Este enfoque de "fin a fin" (*end-to-end*) significa que la misma arquitectura básica puede ser reutilizada y ajustada para resolver diferentes problemas, cambiando únicamente el diseño de las capas finales y ajustando los hiperparámetros de entrenamiento. [41]

La capacidad de las CNN para generalizar a diferentes tareas ha sido demostrada en diversos estudios y aplicaciones. Por ejemplo, la arquitectura de la red ResNet (*Residual Network*) ha sido ampliamente utilizada en tareas de clasificación de imágenes, detección de objetos en tiempo real y segmentación semántica con resultados de vanguardia. [42]

Adaptabilidad a Nuevas Tecnologías y Paradigmas

El diseño de las CNN las hace altamente compatibles con nuevas tecnologías y paradigmas de computación, como las Unidades de Procesamiento Gráfico (GPU, del inglés *Graphical Processing Units*) y las Unidades de Procesamiento Tensorial (TPU, del inglés *Tensor Processing Units*). Estas tecnologías permiten un entrenamiento mucho más rápido de redes profundas al proporcionar paralelización masiva y operaciones optimizadas para los cálculos matriciales que dominan el funcionamiento de las CNN.

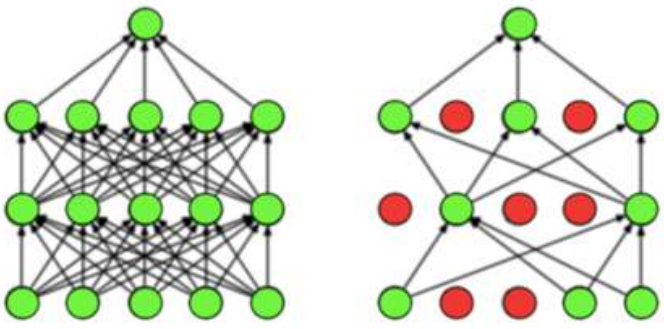
Además, las arquitecturas de CNN pueden combinarse con otras técnicas avanzadas, como los *transformers* y los modelos de atención, para aprovechar lo mejor de ambos mundos. Por ejemplo, las *Vision Transformers* (ViT) utilizan una combinación de convoluciones para capturar características locales y mecanismos de atención para captar relaciones a largo plazo en la imagen, demostrando un rendimiento sobresaliente en tareas de visión por computadora a gran escala [43, 44, 45].

2.7.2 Regularización en CNN 3D

La regularización es un método común para mitigar el sobreajuste e incrementar la generalización del modelo. Estas técnicas penalizan la complejidad del modelo, promoviendo un mejor desempeño ante datos no previamente observados [46]**.**

Dropout

El dropout es una técnica de regularización que implica la "desactivación" aleatoria de una fracción de neuronas durante el entrenamiento. En cada paso de entrenamiento, cada neurona tiene una probabilidad de ser omitida. Esta técnica fuerza a la red a aprender representaciones redundantes, lo que mejora la robustez y la generalización del modelo.



a) *b)*

Figura 27: a) Red Neuronal Estándar, b) Red Neuronal con Dropout.

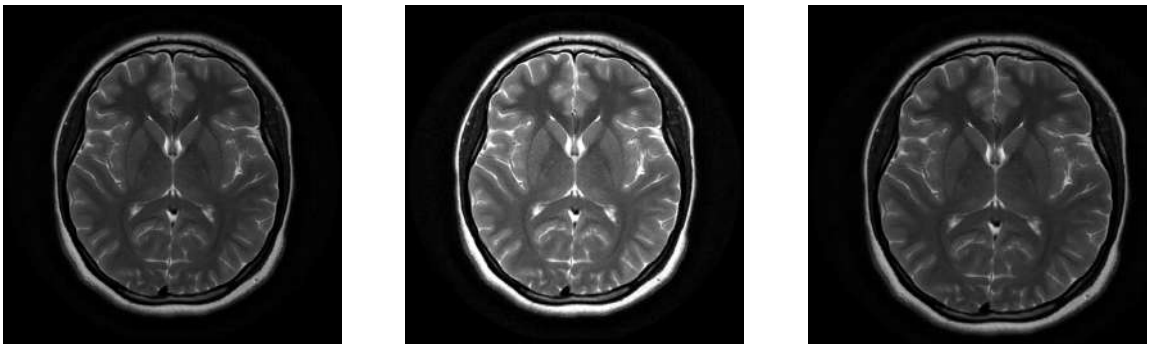
Batch Normalization

La normalización por lotes es una técnica que estandariza las activaciones dentro de cada mini-lote durante el entrenamiento de redes neuronales. Esto se logra sustrayendo la media

del lote y dividiendo por su desviación estándar, lo que resulta en activaciones con media cercana a cero y varianza cercana a uno. Posteriormente, se aplica un escalado y desplazamiento mediante parámetros aprendibles, permitiendo que la red optimice la escala y el sesgo de estas activaciones normalizadas.

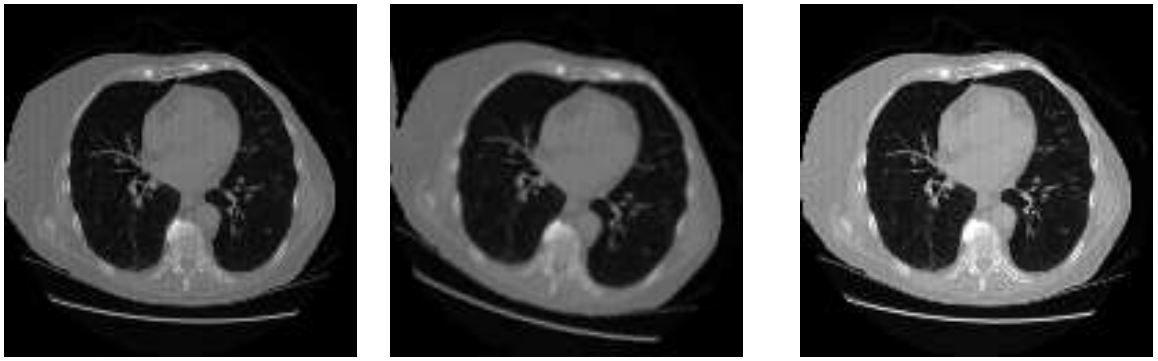
Data Augmentation

El aumento de datos (*data augmentation*) es una técnica que amplía el conjunto de datos de entrenamiento mediante la creación de nuevas muestras a partir de transformaciones de las existentes. En imágenes médicas 3D, esto puede incluir rotaciones, traslaciones, escalados, recortes aleatorios y cambios en la intensidad. Aunque técnicamente no es una técnica de regularización en sí misma, el data augmentation mejora la capacidad del modelo para generalizar al aumentar la diversidad del conjunto de entrenamiento. Esto es especialmente útil en aplicaciones médicas, donde los datos etiquetados pueden ser escasos y costosos de obtener.



a) Corte original. *b) Aumento del brillo.* *c) "zoom" del 10%.*

Figura 28: Ejemplo del resultado de data augmentation sobre cortes de una resonancia magnética 3D de cerebro.



a) Corte original. *b) Rotación de 10°.* c) *Aumento del brillo.*

Figura 29: Ejemplo del resultado de data augmentation sobre cortes de una TC 3D de tórax.

Figura 30: Ejemplo del resultado de data augmentation (rotación + ajuste de brillo) sobre el volumen generado por una TC 3D de tórax.

2.7.3 *Transfer Learning*

El *transfer learning* es una técnica que permite aprovechar el conocimiento adquirido por modelos previamente entrenados en tareas similares, acelerando el proceso de entrenamiento y mejorando la capacidad de generalización de los modelos en nuevas tareas. Su utilización es especialmente útil cuando se trabaja con conjuntos de datos limitados [47].

El funcionamiento del *transfer learning* implica varios pasos:

1. **Selección de un modelo base:** Se elige un modelo preentrenado con un conjunto de datos. Las CNN aprenden características de forma jerárquica: las capas iniciales detectan patrones simples (bordes, texturas), mientras que las capas profundas aprenden características complejas y específicas (estructuras anatómicas, patrones patológicos). Esta jerarquía justifica la práctica de congelar las capas iniciales durante el fine-tuning, preservando sus filtros genéricos, y permitiendo que solo las capas finales se adapten a la nueva tarea.

2. **Ajuste fino (*fine-tuning*):** Después de cargar el modelo preentrenado, se realiza un ajuste fino en el nuevo conjunto de datos objetivo. Esto puede implicar descongelar algunas capas superiores, modificar la arquitectura de salida (por ejemplo, cambiar el número de clases).

3. **Entrenamiento del modelo:** Una vez preparada la arquitectura, se entrena el modelo en el nuevo conjunto de datos. Dado que el modelo ya ha sido entrenado en un conjunto de datos similar, el tiempo de entrenamiento se reduce considerablemente, y el modelo puede converger más rápidamente.

4. **Evaluación y validación:** Se evalúa el modelo ajustado utilizando tanto un conjunto de validación como un conjunto de evaluación. El conjunto de validación se utiliza durante el entrenamiento para monitorear el rendimiento del modelo y ajustar hiperparámetros, proporcionando una estimación de su capacidad de generalización antes de finalizar el entrenamiento. Por su parte, el conjunto de evaluación o testeo se emplea exclusivamente al finalizar el proceso de entrenamiento, con el objetivo de medir el rendimiento final del modelo sobre datos completamente no vistos, simulando su comportamiento en condiciones reales.

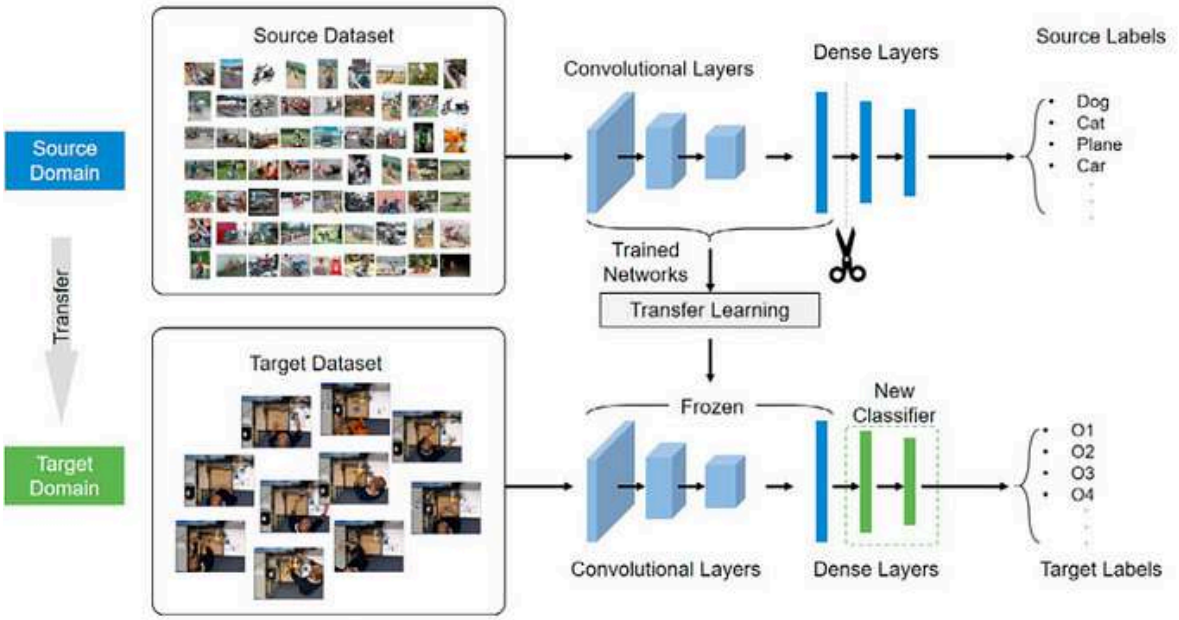


Figura 31: Ejemplo del funcionamiento del transfer learning. [48]

En el contexto médico, las CNN 3D preentrenadas tienen la capacidad de capturar características anatómicas generales que pueden ser útiles para una variedad de aplicaciones. Por ejemplo, modelos entrenados para la segmentación de estructuras pulmonares pueden ser adaptados para la clasificación de lesiones específicas, aprovechando características que son comunes entre ambas tareas.

De hecho, estudios han mostrado cómo el uso de redes preentrenadas reduce significativamente el tiempo de entrenamiento y mejora el rendimiento en tareas de clasificación de imágenes médicas [49].

2.8 Desafíos y Consideraciones

El uso de CNN en el procesamiento de imágenes médicas 3D presenta una serie de desafíos que pueden clasificarse en intrínsecos y extrínsecos. Los desafíos intrínsecos están relacionados directamente con las características técnicas y complejidades inherentes

a las imágenes 3D, mientras que los extrínsecos están vinculados a factores externos, como la implementación y aceptación de estas tecnologías en entornos clínicos.

Entre los desafíos intrínsecos, el tamaño y la cantidad de datos que representan las imágenes médicas 3D suponen una dificultad significativa. Las imágenes volumétricas son mucho más grandes que las imágenes en dos dimensiones, lo que exige un mayor poder computacional y recursos de almacenamiento. Esto no solo aumenta los tiempos de procesamiento y entrenamiento de las CNN, sino que también plantea problemas de memoria que deben ser resueltos para optimizar su rendimiento [50].

Además, la segmentación y anotación de imágenes 3D requiere un gran esfuerzo. En la mayoría de los casos, este trabajo es realizado manualmente por especialistas, lo que implica no sólo una inversión de tiempo considerable, sino también una posible variabilidad en los resultados debido a las diferencias inter-observador. Aunque existen herramientas automáticas y semiautomáticas que ayudan en este proceso, su exactitud sigue siendo limitada en escenarios complejos.

Por último, la visualización de imágenes médicas 3D constituye un desafío intrínseco, ya que representarlas de manera efectiva en pantallas 2D dificulta la comprensión del volumen y las estructuras internas. Incluso con tecnologías avanzadas como pantallas 3D o visualizaciones mediante transparencias, la interpretación sigue siendo un problema debido a la complejidad de los datos volumétricos.

Por otro lado, los desafíos extrínsecos se centran en la aplicación y aceptación de las CNN en el ámbito médico. Una de las principales barreras es la falta de transparencia en el funcionamiento interno de estas redes, lo que las convierte en una “caja negra” para los usuarios finales. Dada la criticidad de las decisiones clínicas, los modelos deben ofrecer precisión e interpretabilidad. La fundamentación de las inferencias de las redes neuronales resulta necesaria para la validación por parte de médicos y especialistas.

Otro desafío es la escasez de datos etiquetados. A diferencia de otros campos donde los datos abundan, en el ámbito médico la recopilación de imágenes 3D etiquetadas es costosa y lenta, ya que requiere la intervención de expertos. Este problema ha llevado a la adopción de técnicas como el transfer learning, que permiten utilizar modelos pre entrenados en conjuntos de datos amplios y adaptarlos a tareas específicas con conjuntos de datos reducidos.

A pesar de estas dificultades, el campo de las CNN aplicadas a imágenes médicas en 3D sigue evolucionando rápidamente. Tecnologías emergentes como la realidad aumentada (AR) y la realidad virtual (VR) prometen revolucionar la forma en que los especialistas visualizan y analizan estas imágenes, mejorando la planificación quirúrgica y la comprensión anatómica. Al mismo tiempo, la personalización de tratamientos basada en las capacidades de las CNN para identificar patrones específicos en las imágenes médicas se perfila como una herramienta clave para la medicina de precisión, promoviendo una atención más eficaz y adaptada a las características únicas de cada paciente [51].

Capítulo 3: Desarrollo de métodos y algortimos

En este capítulo se presenta el núcleo de la implementación del proyecto, proporcionando una descripción del proceso que abarca desde la obtención y preparación de los datos hasta la configuración y entrenamiento de los modelos de redes neuronales convolucionales 3D. El objetivo principal de este capítulo es detallar, de manera sistemática y organizada, cada fase del desarrollo, con un enfoque en las decisiones técnicas y metodológicas para la resolución del problema.

Se abordan las diferentes etapas, desde la selección y preprocesamiento de una base de datos médica, pasando por el diseño y ajuste de las arquitecturas de las redes neuronales, hasta la implementación de técnicas de transfer learning y la configuración de entrenamientos. Asimismo, se exploran las tecnologías y herramientas empleadas, proporcionando un marco integral que permita comprender la interrelación entre los distintos componentes desarrollados. Se busca no sólo ilustrar el proceso seguido, sino también justificar las elecciones realizadas en cada etapa, con el fin de ofrecer técnicas, herramientas, metodologías y experiencias como recursos valiosos para futuros trabajos de clasificación de imágenes 3D utilizando redes neuronales.

3.1 Bases de datos médicas

Las bases de datos médicas contienen colecciones de imágenes y datos clínicos que permiten entrenar y validar algoritmos de segmentación y clasificación. En esta sección, se abordan las características, beneficios y desafíos asociados con el uso de estas bases de datos en el contexto de este proyecto. La calidad, variedad y cantidad de las imágenes son determinantes para la efectividad de los modelos. Asimismo, el etiquetado de los datos es fundamental para el aprendizaje supervisado, ya que permite que los modelos identifiquen patrones y características relevantes en las imágenes.

Sin embargo, la obtención de datos médicos también implica retos significativos, como la necesidad de garantizar la privacidad y la ética en el manejo de información sensible. Es esencial cumplir con normativas y regulaciones para proteger los datos de los pacientes, lo que puede complicar el acceso y la disponibilidad de conjuntos de datos [52].

3.1.1 Aspectos a considerar de los datos médicos

Uno de los mayores obstáculos para el desarrollo efectivo de modelos de Deep Learning en medicina es la adquisición y el manejo de datos de imágenes médicas de calidad. Existen cuestiones a resolver relacionadas a:

Acceso Limitado a Datos: El acceso a datos médicos está sujeto a estrictas regulaciones de privacidad y seguridad, como la Ley de Portabilidad y Responsabilidad del Seguro de Salud (HIPAA) en los Estados Unidos [52]. De manera similar, en Argentina rige la Ley N.º

25.326 de Protección de los Datos Personales, que considera los datos de salud como datos sensibles y establece restricciones sobre su recolección, tratamiento y transferencia, especialmente sin consentimiento explícito [53].

Estas regulaciones imponen restricciones significativas en la recopilación y el intercambio de datos médicos entre instituciones, debido a preocupaciones sobre la confidencialidad y la protección de la información del paciente. Además, la obtención de datos médicos requiere el consentimiento informado de los pacientes, lo que puede ser complicado de obtener, especialmente para conjuntos de datos retrospectivos. Asimismo, la propiedad de los datos médicos puede ser un problema, ya que estos suelen ser propiedad de las instituciones médicas o de los propios pacientes, lo que puede dificultar el acceso y el uso compartido de datos para fines de investigación y desarrollo de modelos.

Calidad y Variabilidad de los Datos: Las imágenes médicas pueden ser adquiridas a través de una variedad de dispositivos de captura, como resonancias magnéticas, tomografías computarizadas, rayos X, ecografías, entre otros. Cada uno de estos dispositivos tiene sus propias características técnicas y configuraciones, lo que puede resultar en diferencias significativas en la calidad y el formato de las imágenes. Además, la calidad de las imágenes médicas puede estar influenciada por las condiciones físicas y fisiológicas de los pacientes, como el movimiento involuntario, la obesidad o la presencia de metal en el cuerpo, lo que puede introducir artifacts en las imágenes y afectar su interpretabilidad.

Etiquetado y Anotación: El etiquetado y la anotación de datos médicos son procesos fundamentales para el entrenamiento de modelos de Deep Learning. Sin embargo, estos procesos requieren la experiencia de profesionales médicos capacitados, como radiólogos y patólogos, lo que puede resultar en costos adicionales y tiempos de procesamiento más largos. Además, la calidad de las anotaciones puede variar entre diferentes anotadores, lo que afecta la consistencia y precisión de los datos etiquetados. La falta de estándares de anotación también puede dificultar la comparación y combinación de datos de diferentes fuentes.

Desbalance de clases: El desbalance de clases ocurre cuando el número de muestras de una clase en un conjunto de datos es significativamente mayor que el número de muestras de otra clase. En el contexto de imágenes médicas, esto se traduce en tener muchas más imágenes de tejido sano que de tejido con patología, como tumores o lesiones. Este desequilibrio puede ser extremadamente problemático, ya que las redes neuronales tienden a ser sesgadas hacia las clases dominantes, lo que lleva a un rendimiento subóptimo en la detección y clasificación de las clases minoritarias, que son a menudo las más críticas desde el punto de vista clínico. El desbalance de clases afecta negativamente varios aspectos del entrenamiento de las CNN 3D:

- **Funciones de pérdida y Métricas**: Las funciones de pérdida estándar, como la entropía cruzada, pueden ser inadecuadas ya que no penalizan suficientemente los errores en las clases minoritarias. Las métricas de exactitud global pueden ser

engañosas, ya que un modelo podría mostrar una alta exactitud simplemente prediciendo siempre la clase mayoritaria.

- **Convergencia del Modelo:** El modelo puede converger a soluciones subóptimas, donde ignora las clases minoritarias debido a la falta de suficiente retroalimentación de error de estas clases durante el entrenamiento.

- **Capacidad de Generalización:** La capacidad del modelo para generalizar a nuevos datos puede verse comprometida.

Varios enfoques pueden mitigar los efectos del desbalance de clases en el entrenamiento de CNN 3D. Estos incluyen técnicas de preprocesamiento de datos, modificación de la función de pérdida y enfoques basados en la arquitectura del modelo.

Técnicas de Balanceo de Datos:

El sobremuestreo y el submuestreo son técnicas utilizadas para abordar el problema del desequilibrio de clases en conjuntos de datos, una situación común en el análisis de imágenes médicas.

El submuestreo consiste en reducir el número de muestras de la clase mayoritaria para equilibrar la proporción entre clases. Si bien esta técnica puede simplificar el modelo y reducir el tiempo de entrenamiento, conlleva el riesgo de eliminar información valiosa, especialmente en escenarios donde el conjunto de datos es limitado.

Mientras que el submuestreo consiste en reducir el número de muestras de la clase mayoritaria, el sobremuestreo implica aumentar el número de muestras de la clase minoritaria.

El sobremuestreo se lleva a cabo a través de técnicas de aumento de datos como rotación, traslación y escalado de imágenes 3D. Estas técnicas permiten la creación de nuevas variaciones de las imágenes existentes, lo que enriquece el conjunto de datos y ayuda a mejorar el rendimiento del modelo. Además, se pueden emplear métodos más avanzados, como las GAN, que sintetizan nuevas muestras realistas, contribuyendo así a un mejor aprendizaje del modelo al representar de manera más efectiva la variabilidad de la clase minoritaria. [54]

Técnicas de sobremuestreo:

- Rotaciones con Límites Definidos: Las rotaciones son una transformación comúnmente utilizada en el aumento de datos, donde las imágenes médicas 3D se rotan alrededor de un eje específico. Dado que en las imágenes médicas, las orientaciones anatómicas son importantes, se establecen límites para estas rotaciones. Por ejemplo, las rotaciones se limitaron a ±10 grados en los planos axial, sagital y coronal. Esto asegura que las características anatómicas críticas se mantengan reconocibles mientras se introduce variabilidad en la orientación.

En las TC de tórax, estas rotaciones limitadas permiten al modelo aprender a reconocer estructuras pulmonares y mediastinales desde diferentes ángulos. Esta técnica es especialmente útil para mejorar la detección de nódulos que pueden aparecer en distintas orientaciones debido a variaciones en la posición del paciente durante el escaneo.

- Pequeños Zooms: Otra técnica eficaz es aplicar pequeños zooms a las imágenes médicas 3D. Al realizar zooms con factores entre 1.0 y 1.1, se introduce variabilidad en la escala de las estructuras anatómicas sin distorsionar significativamente su apariencia. Este enfoque es particularmente útil para simular pequeñas variaciones en la distancia de adquisición de las imágenes y para ayudar al modelo a aprender características multiescalares.

- Cambio de Brillo: El cambio de brillo es una técnica que ajusta la intensidad de los píxeles en las imágenes médicas 3D. Al aplicar distintos factores de brillo, se simulan diferentes condiciones de adquisición y variaciones en la exposición.

Todas las técnicas mencionadas se enmarcan dentro del enfoque de data augmentation, el cual consiste en aplicar transformaciones geométricas y/o fotométricas sobre los datos originales para generar versiones alternativas que mantengan la información esencial.

3.1.2 Datos de entrenamiento: Tomografías de pulmón

El conjunto de datos utilizado en el proyecto proviene de la competencia *Data Science Bowl 2017*, organizada en apoyo a la iniciativa "Cancer Moonshot" del gobierno de los EE.UU. Esta competencia reunió a la comunidad de ciencia y medicina con el fin de desarrollar algoritmos para la detección temprana del cáncer de pulmón. El National Cancer Institute proporcionó miles de escaneos de pulmón de alta resolución, que fueron utilizados para entrenar modelos destinados a mejorar la precisión en la detección y reducir las tasas de falsos positivos [55].

En este conjunto de datos se proporcionan 1595 estudios de tomografía computarizada (TC) de baja dosis en formato DICOM, correspondientes a pacientes de alto riesgo. Cada estudio contiene un número variable de cortes axiales 2D, lo cual varía según el equipo de adquisición y las características del paciente. El conjunto no se encuentra particionado previamente, por lo que fue necesario realizar una división manual en subconjuntos de entrenamiento, validación y prueba. Se priorizó destinar el 80% al entrenamiento para maximizar el aprendizaje del modelo, reservando el 10% restante para validación y otro 10% para prueba, manteniendo un equilibrio entre ambas etapas de evaluación.

Además, se incluye un archivo con las etiquetas (labels) asociadas a cada TC: mediante un identificador único por estudio, es posible saber si el paciente presenta o no cáncer, permitiendo así utilizar el conjunto en tareas de clasificación supervisada.

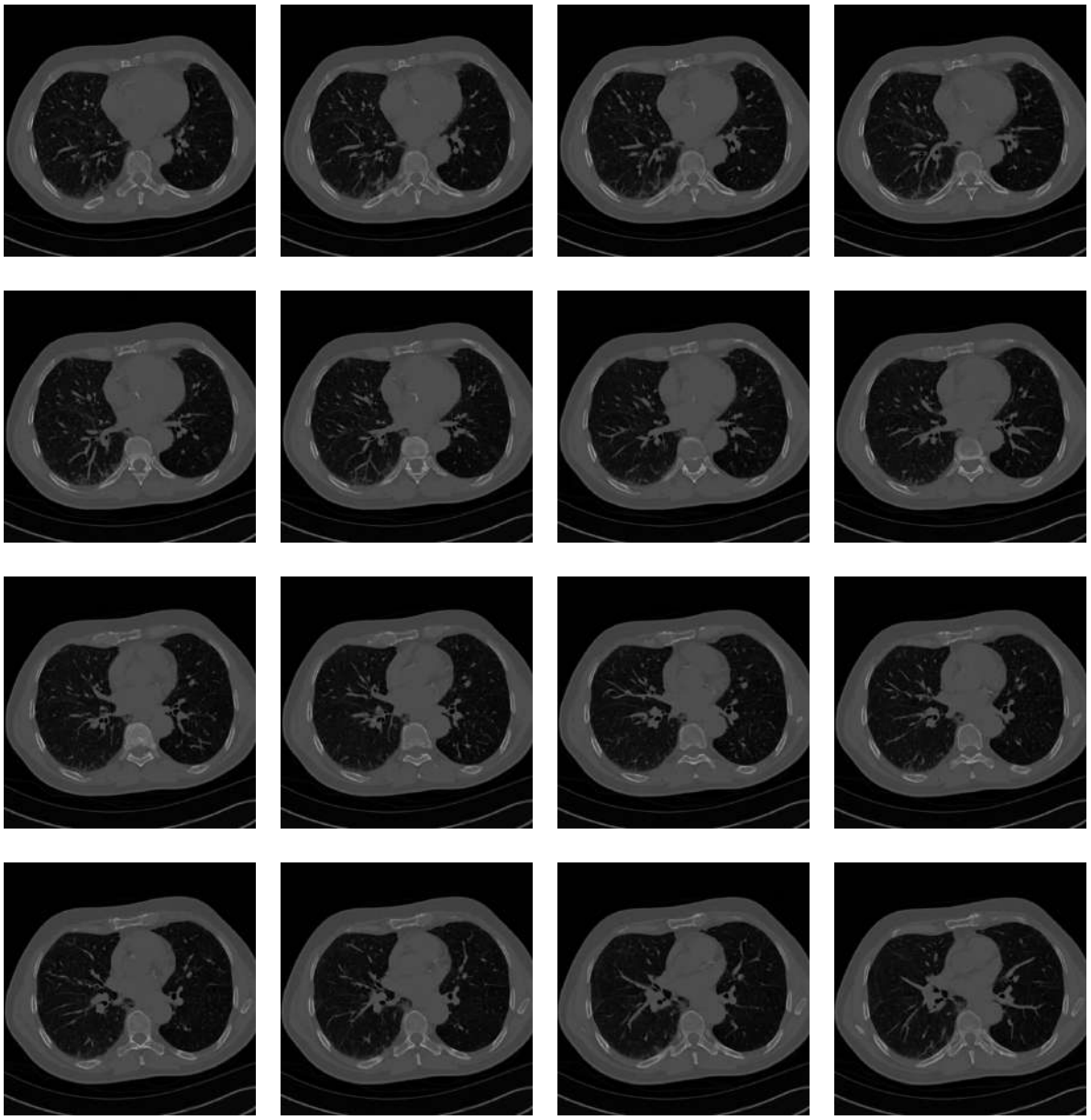


Figura 32: Cortes 2D de un ejemplar del conjunto de datos utilizado.

El conjunto de datos presenta un desbalance significativo entre clases, contiene 1176 TC 3D sin cáncer y 419 con cáncer, lo que representa una proporción considerablemente inclinada hacia la clase negativa. Para mitigar este desbalance, se aplicó una estrategia de data augmentation sobre las TC 3D con cáncer del subconjunto de entrenamiento. Dado que el 80% de los datos se destina al entrenamiento, por partición aproximadamente se trabaja con unas 335 TC 3D con cáncer, sobre las cuales se generaron dos nuevas variantes por cada estudio original. De esta manera, se incorporan alrededor de 670 datos adicionales, logrando una distribución más equilibrada entre ambas clases en el entrenamiento.

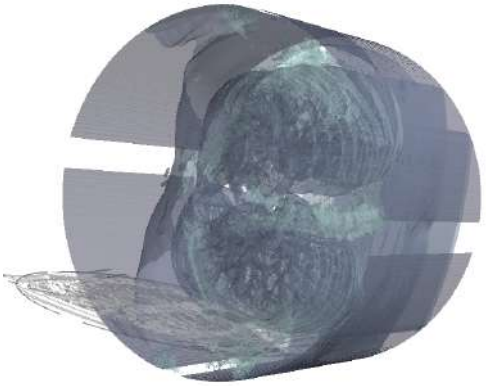


Figura 33: Volumen 3D generado de un ejemplar del conjunto de datos utilizado.

3.2 Preprocesamiento de datos

El redimensionamiento de imágenes es un paso en el preprocesamiento de los datos, las imágenes adquiridas mediante modalidades como tomografía computarizada, resonancia magnética y ultrasonido presentan diversas resoluciones y dimensiones, lo cual complica su análisis y procesamiento. El redimensionamiento es un proceso que estandariza el tamaño de las imágenes, garantizando así la uniformidad de los datos y permitiendo su procesamiento y análisis. Al mantener un tamaño estándar, se asegura que todas las imágenes puedan ser procesadas conjuntamente sin necesidad de ajustes adicionales, mejorando así la eficiencia y efectividad del entrenamiento del modelo.

El redimensionamiento de imágenes médicas en el ámbito tridimensional no solo implica ajustar las dimensiones de las imágenes en dos dimensiones (ancho y alto), sino también en la tercera dimensión (profundidad), que representa la cantidad de cortes (*slices*) en un volumen de imagen médica.

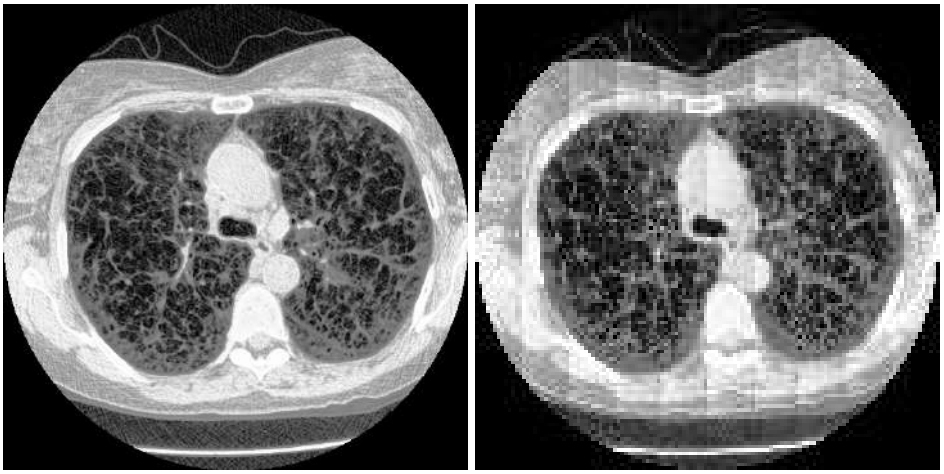


Figura 34: Redimensionamiento de 512x512 a 128x128 pixeles, con ecualización de histograma.

Entrenar modelos de CNN 3D utilizando volúmenes de imágenes médicas con dimensiones originales de 512×512×128 resulta computacionalmente costoso. Por ejemplo, trabajar con un conjunto de datos compuesto por 1595 estudios de TC 3D con estas dimensiones utilizando el equipo disponible (*ver sección 3.5*) implicaría tiempos de entrenamiento del orden de varios días por experimento.

Ante esta situación, reducir la complejidad computacional es una necesidad práctica para permitir el entrenamiento de múltiples arquitecturas y configuraciones en tiempos razonables. Por ello, se realiza un redimensionamiento de las imágenes originales, pasando de 512×512 a 128×128 píxeles para cada slice de la TC 3D.

Este redimensionamiento se lleva a cabo mediante una técnica de interpolación lineal [56]. El proceso calcula los factores de escala a partir del tamaño original del volumen y ajusta la resolución en los ejes espaciales teniendo en cuenta el espacio entre píxeles. El tamaño reducido a 128x128 píxeles, aunque presenta menos detalle en comparación con la resolución original de 512x512 píxeles, sigue siendo adecuado para capturar las características anatómicas principales necesarias para el análisis médico. Si bien es cierto que al disminuir la resolución se pueden perder detalles más finos, las estructuras esenciales y patrones relevantes suelen mantenerse.

Además, las técnicas avanzadas de aprendizaje profundo permiten compensar esta reducción al enfocarse en los aspectos más importantes del volumen de datos para las tareas de diagnóstico. Por ejemplo, métodos de interpolación avanzada pueden mejorar la calidad de las imágenes reducidas, mientras que la extracción jerárquica de características en redes neuronales convolucionales permite captar información relevante a diferentes escalas [57].

Este redimensionamiento también permite la implementación de modelos con una cierta profundidad y complejidad, que pueden beneficiarse de una mayor capacidad de generalización, siempre que se mantenga un equilibrio con la resolución de entrada. En resumen, la elección de un tamaño de 128x128 píxeles para los cortes de TC 3D equilibra eficientemente la necesidad de mantener información anatómica relevante con la demanda de recursos computacionales manejables, optimizando así el rendimiento del modelo en aplicaciones clínicas [58].

Por otro lado, la cantidad de cortes en las imágenes médicas puede variar entre diferentes estudios. Para garantizar la uniformidad en la entrada de datos al modelo, se utiliza la interpolación para ajustar el número de cortes a un valor específico. Esta elección se fundamenta en un equilibrio entre la resolución necesaria para capturar suficiente detalle anatómico y la manejabilidad computacional, considerando el tamaño original de las imágenes [59].

Desde el punto de vista médico y anatómico, este ajuste tridimensional asegura la continuidad espacial y la coherencia de las estructuras anatómicas a lo largo del volumen de la imagen. La interpolación lineal, por ejemplo, ajusta suavemente las imágenes entre

cortes, preservando la integridad de características anatómicas y patológicas como los alvéolos, bronquios y lesiones pulmonares.

Existen varios métodos de interpolación, como la lineal, cúbica y por splines. En este contexto, la interpolación lineal suele ser suficiente para ajustar el número de cortes, proporcionando un equilibrio adecuado entre exactitud y eficiencia computacional. Este método reduce el riesgo de introducir artifacts o distorsiones en las imágenes, manteniendo la integridad de los datos médicos durante el preprocesamiento.

El proceso de interpolación ajusta las imágenes a una dimensión uniforme en profundidad, normalizando todos los volúmenes a 64 cortes, lo cual facilita el procesamiento por parte de los modelos de aprendizaje profundo. Esto permite una mejor generalización y precisión en la detección y diagnóstico de patologías pulmonares a partir de las imágenes de tomografía. Desde un punto de vista médico, la interpolación adecuada asegura la captura de todos los detalles relevantes para el diagnóstico, manteniendo la continuidad de estructuras críticas como vasos sanguíneos, bronquios y posibles lesiones a lo largo del volumen de la imagen. Esta precisión es esencial para el análisis detallado y la toma de decisiones clínicas basadas en los datos procesados por modelos de inteligencia artificial.

Otra técnica que se aplicó durante el preprocesamiento es la ecualización del histograma, para mejorar el contraste de las imágenes, ajustando la distribución de los niveles de intensidad. Esto permitió resaltar estructuras anatómicas importantes que podrían no ser visibles en la imagen original.

3.3 Modelos de CNN 3D

Esta sección presenta un análisis detallado de las arquitecturas desarrolladas para los modelos de CNN empleados en la clasificación binaria de imágenes volumétricas. Las tres arquitecturas diseñadas están fundamentadas en la literatura y estudios previos de deep learning aplicados a imágenes médicas. Si bien se inspiran en modelos de investigación existentes, estas configuraciones no son una réplica directa, cada modelo ha sido ajustado y optimizado específicamente para el contexto de este proyecto, adaptando los parámetros y estructura de acuerdo a los objetivos y características particulares del conjunto de datos utilizado.

Además de estas arquitecturas, se presenta un análisis de modelos preentrenados de *transfer learning*. Estas arquitecturas han sido ampliamente utilizadas en el campo de la visión computacional y ofrecen un enfoque complementario al entrenamiento desde cero, permitiendo aprovechar las características previamente aprendidas en grandes conjuntos de datos. La combinación de arquitecturas entrenadas desde cero y modelos de transfer learning permite comparar enfoques y optimizar el rendimiento general en la clasificación de imágenes médicas, explorando tanto los beneficios de la personalización completa como los de la reutilización de conocimientos preexistentes en redes profundas.

3.3.1 Modelo 1

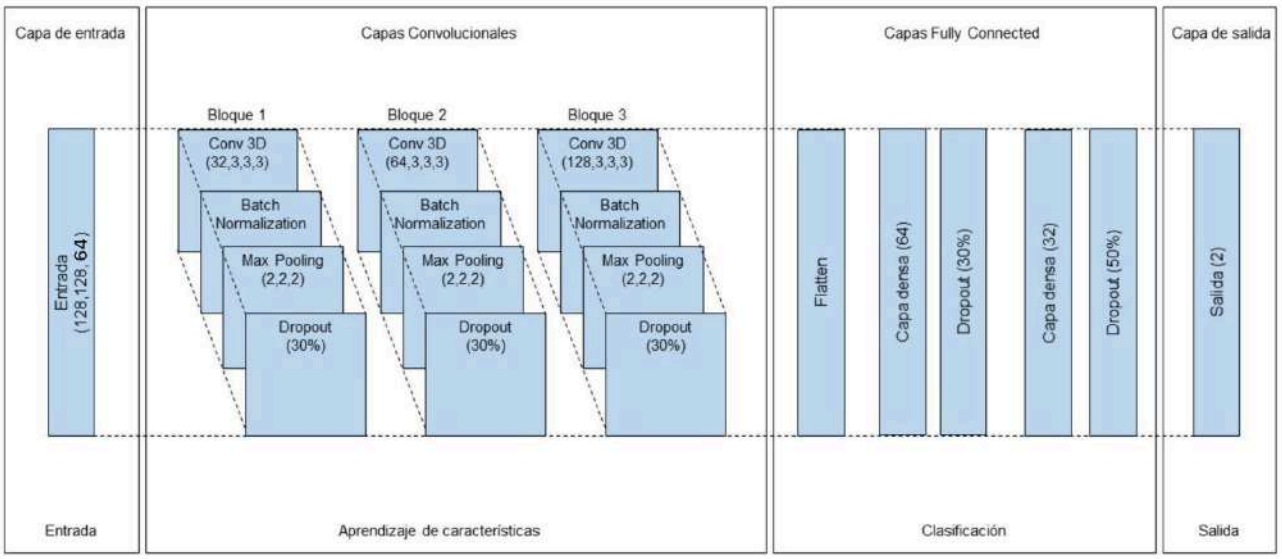


Figura 35: Modelo 1 de arquitectura de CNN.

El primer modelo fue diseñado como una arquitectura base siguiendo una estrategia de prueba y ajuste, con el objetivo de establecer una línea de base experimental a partir de una configuración simple y liviana. Esta red está compuesta por tres bloques convolucionales básicos, cada uno formado por una capa de convolución con filtros 3x3, activación ReLU y una capa de max pooling que reduce progresivamente la resolución espacial del volumen de entrada. El número de filtros por bloque va de 32 a 128.

Luego del proceso de extracción de características se aplica un aplanamiento (*flatten*) de los mapas tridimensionales, convirtiéndolos en un vector unidimensional que alimenta dos capas densas intermedias con 64 y 32 neuronas respectivamente, ambas activadas con ReLU. Estos tamaños se definieron tras explorar varias configuraciones y seleccionar aquellas que mostraron convergencia estable y buen *trade-off* entre capacidad de modelo y riesgo de sobreajuste.

Después de cada una de estas capas densas, se aplica dropout con tasas de 30% y 50%, tras observar que tasas menores no eran suficientes para reducir el riesgo de sobreajuste. Finalmente, una única neurona con activación sigmoide en la capa de salida permite realizar la clasificación binaria.

Esta arquitectura base resultó de un proceso sistemático de iteraciones, en cada una del cual se experimentó obteniendo nuevos resultados y se ajustaron parámetros, hasta consolidar esta configuración inicial.

3.3.2 Modelo 2

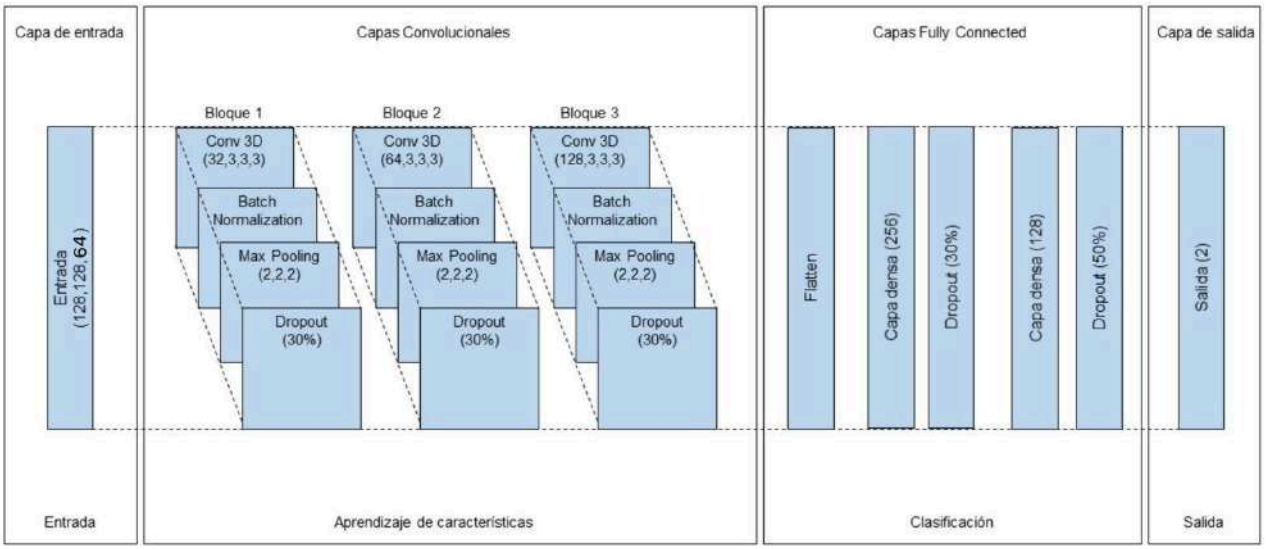


Figura 36: Modelo 2 de arquitectura de CNN.

El segundo modelo introduce una fase de clasificación más robusta, incorporando una mayor cantidad de neuronas en las capas densas, lo que representa un incremento en la capacidad de aprendizaje con respecto al modelo base, pero aún así manteniendo un balance entre simplicidad y capacidad de aprendizaje. La fase de extracción de características se mantiene igual al primer modelo.

Tras aplanar las características extraídas, el modelo incluye dos capas densas intermedias de 256 y 128 neuronas, respectivamente, ambas activadas con la función ReLU. Se aplica dropout con tasas del 30% y 50% después de cada capa densa para reducir el sobreajuste. Al igual que el primer modelo, la capa de salida cuenta con una única neurona con activación sigmoide para la clasificación binaria.

Este modelo busca aumentar la capacidad de generalización al incorporar más neuronas en las capas densas intermedias, optimizando el equilibrio entre eficiencia computacional y precisión.

3.3.3 Modelo 3

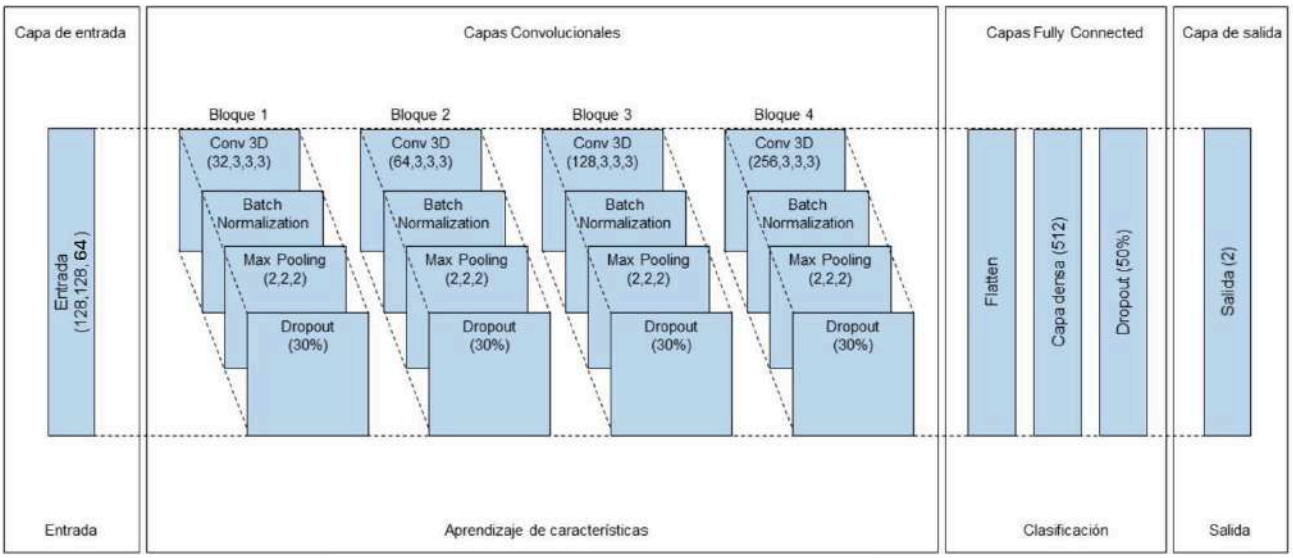


Figura 37: Modelo 3 de arquitectura de CNN.

El tercer modelo representa la arquitectura más profunda y compleja de las tres propuestas. Su diseño se centró en potenciar la capacidad de extracción de características al incorporar un cuarto bloque convolucional, en contraste con los tres bloques presentes en los modelos anteriores. Cada bloque está compuesto por una capa convolucional, seguida de BatchNormalization, MaxPooling y Dropout. El número de filtros se incrementa progresivamente en cada bloque, partiendo de 32 y alcanzando 256 en el último, decisión que responde a la necesidad de capturar patrones cada vez más abstractos y de mayor nivel a medida que avanza la red.

A diferencia de los modelos anteriores, en este caso se optó por simplificar la fase de clasificación a una única capa densa de 512 neuronas con activación ReLU. Esta decisión se tomó con el objetivo de concentrar la capacidad de representación en una sola capa profunda, compensando así la mayor carga computacional introducida en la etapa convolucional. Las tasas de dropout utilizadas son del 30% en los bloques convolucionales y del 50% en la capa totalmente conectada, con el fin de prevenir el sobreajuste dada la mayor complejidad del modelo.

En comparación con las arquitecturas anteriores, este modelo incrementa significativamente la profundidad en la etapa de extracción de características, mientras que reorganiza la fase de clasificación para mantener un balance entre capacidad de aprendizaje y riesgo de sobreajuste. Esta configuración tuvo como objetivo evaluar la posibilidad de entrenar una arquitectura compleja con la cantidad limitada de datos disponible y analizar los resultados bajo estas condiciones.

3.3.4 Modelo de Transfer Learning

Durante el desarrollo del proyecto, se plantea inicialmente la intención de buscar modelos preentrenados específicos para el caso particular con el que se experimenta la detección de cáncer en TC de pulmón. Este enfoque surge de la idea de aprovechar el conocimiento ya existente y encapsulado en modelos que hayan sido entrenados previamente en un contexto similar al problema en cuestión. Sin embargo, debido a las limitaciones actuales en la disponibilidad de este tipo de modelos, se extendió la búsqueda hacia modelos diseñados para imágenes médicas 3D en general y, finalmente, a modelos preentrenados en dominios más amplios de imágenes 3D.

El enfoque escalonado de búsqueda se fundamenta en la premisa teórica de que las capas iniciales de las CNN suelen detectar características genéricas como bordes, texturas y patrones básicos que pueden ser útiles en diversas tareas. Este fenómeno se explica porque estas capas operan a un nivel más abstracto, capturando estructuras que son comunes en una amplia variedad de imágenes, independientemente de su dominio. Por lo tanto, incluso en modelos que no fueron entrenados específicamente para imágenes médicas o para el problema del cáncer de pulmón, las capas iniciales pueden contribuir significativamente al aprendizaje del modelo en una etapa de transferencia.

Desafíos en la búsqueda de modelos preentrenados

A pesar del potencial que representa la técnica de transfer learning, no es sencillo acceder a modelos preentrenados para imágenes médicas 3D. En primer lugar, el número de modelos disponibles públicamente es reducido, lo que dificulta encontrar redes adecuadas para problemas específicos. Además, la gran mayoría de los escasos modelos que se pueden encontrar suelen carecer de documentación detallada. Las bases de datos utilizadas para el entrenamiento, las configuraciones específicas de los modelos, y los resultados alcanzados (como exactitud, sensibilidad o especificidad) no siempre están disponibles o claramente descritos. Esta falta de transparencia limita la posibilidad de evaluar adecuadamente la idoneidad de dichos modelos para una nueva tarea.

Ante las limitaciones mencionadas, se optó por utilizar un modelo preentrenado de CNN 3D que, si bien no fue diseñado específicamente para tomografías de pulmón, proviene de un dominio relacionado dentro del ámbito de imágenes médicas. Este modelo fue entrenado sobre un conjunto de datos que incluye estudios de resonancia magnética multimodal (FLAIR, T1, T1CE y T2) de pacientes con tumores cerebrales. [60].

Dicha base de datos fue diseñada para tareas de clasificación y proporciona volúmenes 3D con distintas secuencias de RM cerebrales. El modelo preentrenado representa una alternativa viable para reutilizar conocimientos adquiridos en otro contexto clínico, permitiendo aprovechar características generales de las imágenes médicas tridimensionales.

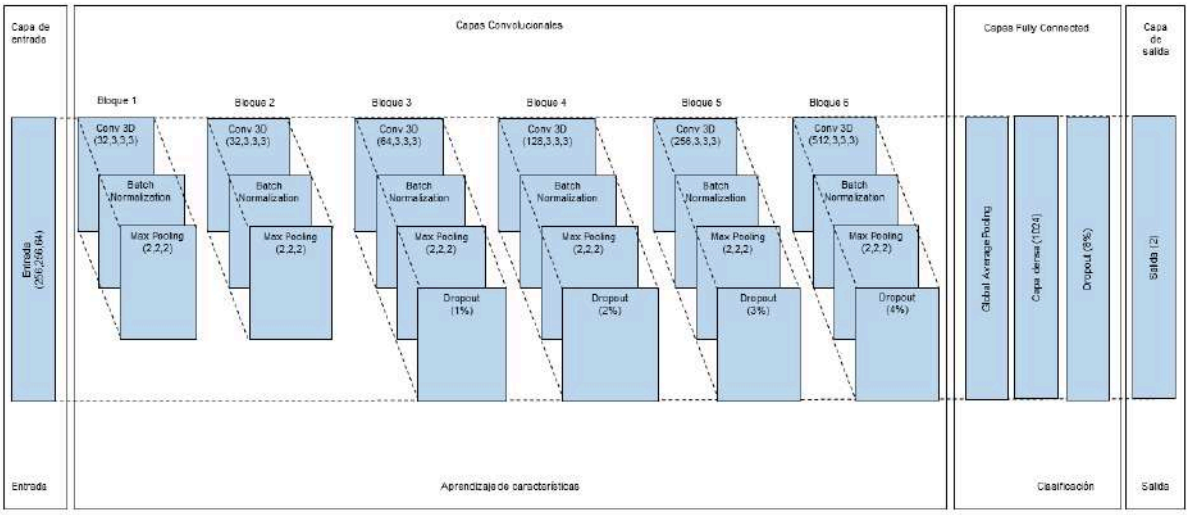


Figura 38: Modelo de Transfer Learning de arquitectura de CNN.

Arquitectura del modelo preentrenado

Este modelo consta de seis bloques convolucionales, los dos primeros bloques incluyen una capa convolucional, seguida de una capa de BatchNormalization y una de MaxPooling. Luego, los siguientes cuatro bloques añaden una capa de Dropout. El número de filtros aumenta progresivamente, comenzando con 32 filtros en las dos primeras capas convolucional y alcanzando los 512 filtros en el sexto bloque.

La profundad de este modelo permite capturar características cada vez más complejas y específicas del volumen de datos. Cada capa de normalización por lotes ayuda a estabilizar el aprendizaje y acelera la convergencia al normalizar las activaciones. Además, las capas de dropout con tasas que van del 1% al 4% mitigan el sobreajuste.

Finalmente, la red culmina con una capa que condensa las características aprendidas en un vector de 512 dimensiones. Este vector se pasa a una capa densa que genera 1024 unidades con activación ReLU y una segunda capa densa que produce la salida final con una activación sigmoide, adecuada para problemas de clasificación binaria.

Entrenamiento y ajuste del modelo preentrenado

La congelación de capas iniciales, una técnica habitual en Transfer Learning, reduce el tiempo de entrenamiento. Esto se logra mediante la reutilización del conocimiento adquirido por un modelo base entrenado con datos similares, enfocando el ajuste en las capas superiores para aprender las características específicas de la tarea de clasificación actual.

La experimentación incluyó la exploración de diferentes configuraciones de la red preentrenada y la variación de la cantidad de capas ajustadas durante el entrenamiento. Cada configuración experimental se diseña para evaluar cómo diferentes grados de ajuste fino afectan la capacidad del modelo para adaptarse a los nuevos datos.

Se realizaron pruebas donde se descongelaron diferentes cantidades de capas, desde el ajuste exclusivo de las capas densas hasta la descongelación de capas convolucionales profundas. Esto permitió evaluar el impacto de retener características generales frente a la necesidad de especializar la red para los patrones específicos del conjunto de datos. Un mayor ajuste de capas puede ser beneficioso cuando la variabilidad de los datos es alta, aunque también aumenta el riesgo de sobreajuste si el tamaño del conjunto de datos es limitado.

Los experimentos se realizaron en varias fases, ajustando los hiperparámetros de manera iterativa, con especial atención a la cantidad de capas descongeladas. Cada iteración fue cuidadosamente monitoreada para evaluar la estabilidad del entrenamiento y la convergencia del modelo hacia un desempeño adecuado, evitando el riesgo de sobreajuste al utilizar un conjunto de datos limitado.

Configuraciones de redes preentrenedas.

Se diseñaron tres arquitecturas con diferentes enfoques de adaptación del modelo preentrenado. Cada configuración fue concebida para explorar distintos aspectos del proceso de Transfer Learning:

- Modelo 4
    - Descongelación selectiva de la última capa convolucional.
        - Capas densas de adaptación: 64 → 32 unidades

    Objetivo Experimental:

    Evaluar una estrategia mínimamente invasiva donde solo las capas finales de alto nivel se ajustan a la nueva tarea, manteniendo la mayor parte de los patrones aprendidos originalmente. La estructura de clasificación reducida busca prevenir sobreajuste en conjuntos de datos pequeños.

- Modelo 5
    - Descongelación selectiva de la última capa convolucional.
        - Arquitectura clasificadora expandida: 256 → 128 → 64 unidades

    Objetivo Experimental:

    Investigar si el incremento de complejidad en las capas de clasificación permite capturar relaciones más sofisticadas en los descriptores espaciales extraídos.

- Modelo 6
    - Descongelación extendida a capas medias y profundas
        - Misma estructura clasificadora.

    Objetivo Experimental:

    Analizar el efecto de permitir un ajuste más granular del modelo base, combinando patrones de bajo nivel (bordes, texturas) con características de alto nivel (formas complejas) específicas del dominio médico.

La triple configuración permite evaluar:

- Impacto de la profundidad de ajuste (4 vs 6)

- Efecto de la complejidad del clasificador (4 vs 5)

- Interacción entre profundidad de ajuste y capacidad clasificadora (5 vs 6)

3.4 Configuración de los entrenamientos

La configuración de los entrenamientos en los experimentos con CNN 3D se basó en la selección y variación del número de épocas, tamaño del batch, uso de early stopping y la selección de algoritmos de optimización. Las distintas combinaciones fueron elegidas teniendo como objetivo la optimización de la eficiencia computacional y el rendimiento del modelo.

Para evaluar de manera robusta el desempeño de los modelos, se construyeron diez particiones distintas del conjunto de datos, generadas de forma aleatoria. Cada partición se dividió en un 80% para entrenamiento, 10% para validación y 10% para prueba, asegurando una distribución balanceada de clases y permitiendo una evaluación cruzada del comportamiento de los modelos en distintos subconjuntos.

Dada la limitada cantidad de datos disponibles, se determinó un rango de entrenamiento entre 100 y 150 épocas para permitir que los modelos convergieran de manera adecuada. Durante el proceso de entrenamiento, se implementa la técnica de early stopping con un criterio de paciencia de entre 20 y 30 épocas. Esto implica que el entrenamiento se detuviera si la pérdida en el conjunto de validación no mejoraba dentro del rango especificado, mitigando así el riesgo de sobreentrenamiento.

El tamaño del batch utilizado fue de 16, que representa el valor máximo permitido por los recursos computacionales disponibles, específicamente la memoria de la GPU empleada. Aunque esta configuración es más alta que en estudios previos con limitaciones similares, permitió alcanzar un balance entre la estabilidad en la estimación de los gradientes y el

aprovechamiento de los recursos computacionales. Masters y Luschi (2018) sugieren que tamaños de batch moderados son beneficiosos, ya que permiten una exploración eficiente de la superficie de pérdida y una estimación precisa de los gradientes durante el entrenamiento de modelos complejos [61].

Para la optimización de los pesos de las redes, se empleó el algoritmo Adam, configurado con una tasa de aprendizaje inicial de 0,0001. Adam es una variante del descenso de gradiente estocástico que ajusta adaptativamente la tasa de aprendizaje de cada parámetro. Su elección se basó en estudios que destacan su capacidad para balancear rapidez de convergencia y estabilidad en escenarios con datos limitados [62]. En los experimentos realizados, Adam facilita una reducción más rápida de la pérdida en las primeras épocas de entrenamiento, optimizando así el tiempo disponible para realizar múltiples pruebas.

3.5 Tecnologías

Se presenta un resumen detallado de las tecnologías empleadas a lo largo del desarrollo del proyecto. La configuración tecnológica abarca tanto el hardware como el software, optimizando el desempeño del servidor en tareas de procesamiento intensivo. En primer lugar, se describe la configuración del servidor, incluyendo componentes de hardware como el procesador, memoria RAM, GPU, y almacenamiento.

El software y las herramientas específicas empleadas en el proyecto también se detallan en esta sección. El sistema operativo, los drivers y configuraciones, como CUDA y los controladores de la GPU, que permiten aprovechar al máximo los recursos del hardware. Además, se hace un repaso de los lenguajes de programación y librerías especializadas en deep learning y procesamiento de imágenes que fueron seleccionadas para el desarrollo de las redes neuronales y la manipulación de datos.

Finalmente, se describen los entornos virtuales configurados para gestionar las dependencias y versiones de las librerías y herramientas de manera ordenada y eficiente. Esta estructura tecnológica no solo facilita el proceso de desarrollo, sino que también garantiza la reproducibilidad de los experimentos y la estabilidad en el funcionamiento de los modelos entrenados, proporcionando así un entorno optimizado y confiable para el entrenamiento y evaluación de las redes CNN 3D en el análisis de imágenes médicas.

Configuración y Datos del Servidor de Desarrollo

Elemento	Detalles
Procesador	Intel Core i9 10900
Memoria RAM	64 GB
Unidad de Procesamiento Gráfico (GPU)	NVIDIA RTX 3070
Almacenamiento	SSD 480GB, HDD 2TB
UPS	800 VA
Sistema Operativo	Windows 10 Pro, 64 bits
Drivers NVIDIA	516.94
CUDA	11.7

Lenguaje de programación

- Python ha sido elegido como el lenguaje de programación principal debido a su sintaxis clara, versatilidad y su amplia adopción en la comunidad científica. Su capacidad para integrar bibliotecas avanzadas de aprendizaje automático y visión por computadora lo convierte en una opción idónea para proyectos de investigación. La extensa comunidad de desarrolladores y la amplia gama de bibliotecas disponibles en Python facilitan tanto el desarrollo como el soporte técnico, garantizando un entorno robusto y bien documentado.

Librerías

- **TensorFlow y Keras:** Este marco de trabajo, desarrollado por Google, es ampliamente reconocido en la comunidad de aprendizaje automático por su versatilidad y eficiencia en el entrenamiento de modelos de redes neuronales profundas. TensorFlow soporta la implementación de modelos de redes convolucionales y permite la aceleración de su entrenamiento mediante el uso de GPU a través de la plataforma CUDA de NVIDIA. Su capacidad de integración con herramientas avanzadas de machine learning lo hace indispensable para experimentos complejos y entornos de producción [63].

- **NumPy y Pandas:** Estas bibliotecas son ampliamente utilizadas en el procesamiento y análisis de datos en Python. NumPy proporciona estructuras de datos eficientes para el manejo de matrices y tensores, facilitando operaciones matemáticas complejas y cálculos vectorizados. Pandas, por su parte, ofrece herramientas

robustas para la manipulación y limpieza de datos estructurados. Ambas bibliotecas optimizan el rendimiento en el manejo de grandes volúmenes de datos, permitiendo una manipulación rápida y eficiente [64].

Drivers

- **CUDA:** Para aprovechar el poder computacional de la GPU en el entrenamiento de modelos de redes neuronales, se emplea la plataforma de computación paralela CUDA, desarrollada por NVIDIA. CUDA acelera de forma significativa las operaciones matemáticas intensivas requeridas en el entrenamiento de modelos de deep learning, resultando en tiempos de entrenamiento más cortos y una escalabilidad mayor para la implementación de modelos más complejos. La compatibilidad de CUDA con TensorFlow y otras bibliotecas de aprendizaje automático asegura una integración fluida y eficiente en el proceso de desarrollo [65].

Entornos Virtuales

Para garantizar un entorno de desarrollo organizado y controlado, se ha implementado el uso de entornos virtuales, los cuales ofrecen una serie de beneficios importantes:

- **Aislamiento y Gestión de Dependencias:** Los entornos virtuales aseguran que cada proyecto mantenga su conjunto único de bibliotecas y herramientas, evitando conflictos de versiones y asegurando que cada experimento utilice las dependencias correctas sin interferencias externas.
- **Reproducibilidad:** La definición explícita de dependencias y versiones en un entorno virtual permite la replicación precisa del proyecto en diferentes sistemas y momentos, facilitando la validación de resultados en contextos diversos.
- **Flexibilidad y Experimentación:** Al trabajar en entornos virtuales, es posible realizar pruebas y modificaciones sin afectar el sistema principal de desarrollo, permitiendo una mayor libertad en la experimentación y ajustes de configuraciones.
- **Facilidad de Distribución:** Al almacenar las dependencias en un archivo, se facilita el despliegue y la distribución del proyecto en otros sistemas, tanto para la colaboración con otros investigadores como para su implementación en entornos de producción.
- **Gestión de Versiones:** La capacidad de gestionar versiones específicas de bibliotecas y herramientas permite asegurar la compatibilidad entre distintas partes del sistema, garantizando la estabilidad del proyecto y facilitando futuras actualizaciones y correcciones.

3.6 Metodología de trabajo y experimentación

Se presenta una guía metodológica del proceso realizado para el desarrollo de la herramienta basada en CNN 3D orientada al análisis y procesamiento de imágenes médicas

tridimensionales. Esta herramienta ha sido diseñada con el propósito de ofrecer un enfoque eficiente y reproducible para el preprocesamiento y organización de los datos y el entrenamiento y evaluación de modelos de clasificación de imágenes médicas volumétricas. La misma está destinada a investigadores y profesionales del ámbito médico que requieren herramientas robustas para el análisis de datos complejos en medicina.

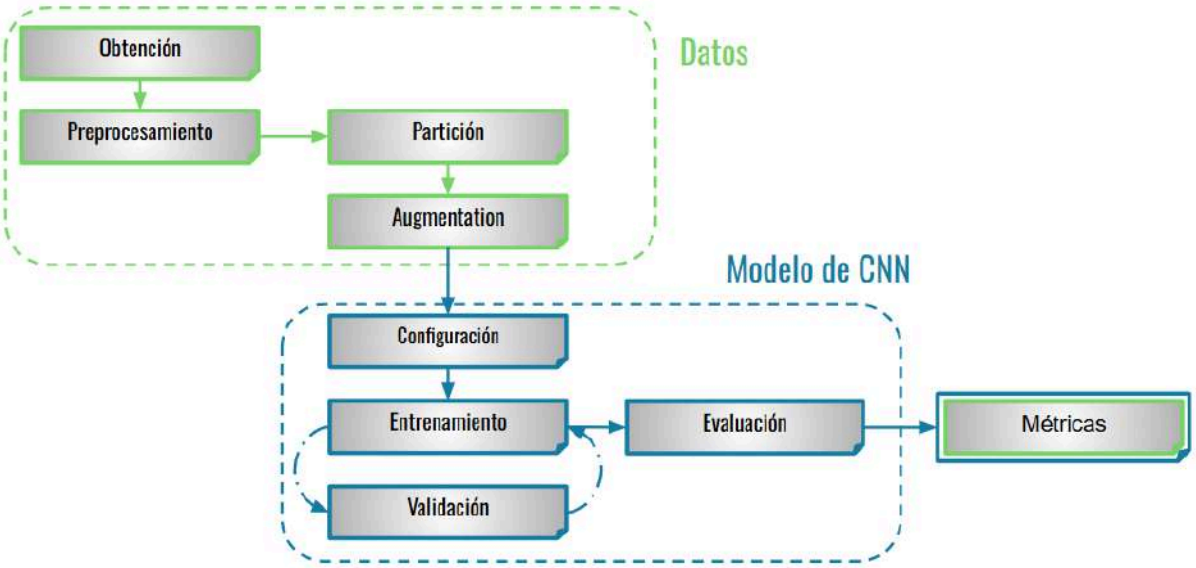


Figura 39: Diagrama de la metodología desarrollada. Las imágenes 3D pasan por una etapa de preprocesamiento y balanceo (en caso de ser necesario) antes del entrenamiento y la validación de cada modelo.

En el flujo de trabajo inicial, una vez adquirido un conjunto de imágenes médicas a clasificar, el proceso comienza con una fase de preprocesamiento para preparar los datos en un formato adecuado para el análisis con CNN 3D. Las imágenes originales suelen estar en formato DICOM. En esta etapa, se realiza una lectura de los archivos DICOM, convirtiéndolos a formato PNG para facilitar la manipulación de imágenes en procesamiento avanzado y permitir una mayor compatibilidad con bibliotecas de deep learning. Para asegurar la uniformidad, cada imagen es redimensionada a un tamaño estándar de 128x128 píxeles. Dado que las imágenes volumétricas pueden presentar diferentes cantidades de cortes o slices en cada estudio, el proceso también incluye la interpolación de estos cortes, normalizando todos los estudios a la cantidad seleccionada de slices. Este paso permite un procesamiento más coherente y simplificado en la arquitectura 3D de la red.

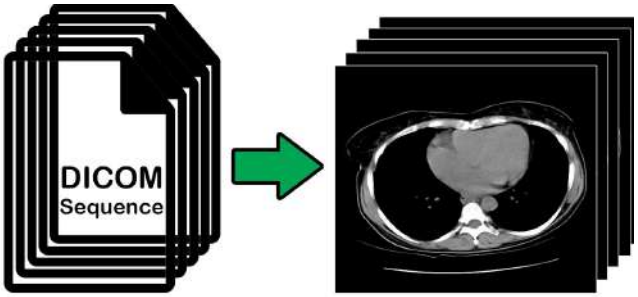


Figura 40: Conversión de formato DICOM a PNG.

Con las imágenes ya en formato PNG y estandarizadas se procede a la ecualización del histograma de cada imagen, una técnica que ajusta los niveles de intensidad de los píxeles para mejorar el contraste. La ecualización de histograma contribuye a resaltar detalles en las imágenes médicas que pueden ser relevantes para la detección y clasificación de patrones en los estudios clínicos.

Posteriormente, el conjunto de imágenes preprocesadas se organiza en un directorio específico denominado “dataset”, el cual contiene subcarpetas individuales para cada imagen 3D; dentro de cada subcarpeta, se almacenan los cortes en formato PNG, que en conjunto constituyen el volumen tridimensional completo.

Para asociar los datos de entrada con sus respectivas etiquetas de clasificación, se crea un archivo de etiquetas en formato CSV. Este archivo contiene información de identificación para cada estudio y su respectiva etiqueta de clasificación, facilitando así la asociación entre cada volumen de imagen y su clase. La estructura del conjunto de datos se complementa con una partición en subconjuntos de entrenamiento, validación y prueba. Para ampliar el espectro de resultados y reducir los sesgos relacionados a la elección del conjunto de entrenamiento, se realizan múltiples particiones distintas, permitiendo un enfoque de evaluación cruzada (cross-validation) basada en 10 particiones diferentes del mismo conjunto de datos. En cada partición, el 80 % de los datos se destinó al entrenamiento, y el 20 % restante se dividió equitativamente entre validación y prueba.. Además, en los casos donde el conjunto de datos presenta un desbalance significativo en la representación de clases, se incorpora una técnica de aumento de datos que permite equilibrar las clases mediante transformaciones aplicadas sobre el conjunto de entrenamiento, aumentando así la robustez y exactitud del modelo.

En el presente proyecto se realizaron 10 particiones diferentes del conjunto de datos en entrenamiento, validación y prueba. Este método permite evaluar el modelo en distintas configuraciones del conjunto de datos, reduciendo la dependencia de los resultados a una única partición de entrenamiento o prueba. Además, permite obtener resultados más representativos, incluyendo métricas promedio y su desviación estándar, lo cual ofrece una visión más robusta y generalizable sobre el desempeño del modelo.

Una vez finalizada esta etapa, se utiliza el 100% de los datos disponibles para entrenar un modelo final, que representa la red definitiva. Las métricas de este modelo se estiman considerando las configuraciones previas y los resultados obtenidos durante el proceso de evaluación cruzada, asegurando que el modelo sea capaz de generalizar adecuadamente en datos no vistos.

Luego se pasa a la definición del modelo de CNN 3D, seleccionando alguna de las arquitecturas definidas anteriormente. Estos modelos varían en profundidad y complejidad, siendo el Modelo 1 el más simple, mientras que el Modelo 3 es el más profundo y sofisticado. La elección de un modelo sobre otro dependerá del tipo de problema, en este trabajo, la detección de cancer en TC 3D de pulmón, como así también del tamaño y características del conjunto de datos, así como de la capacidad computacional disponible.

Antes de comenzar el entrenamiento, se eligen algunos hiperparámetros clave para mejorar el rendimiento del modelo. Estos incluyen el tamaño del lote (batch size), la tasa de aprendizaje (learning rate), la cantidad de épocas (epochs), los parámetros de regularización para prevenir el sobreajuste (regularization) y el tipo de optimizador que se usará (optimizer). La elección de estos valores se realiza mediante una metodología basada en pruebas iterativas y ajustes progresivos, evaluando el impacto de cada parámetro en la estabilidad del entrenamiento y la capacidad del modelo para generalizar a datos no vistos

Durante el entrenamiento, los resultados se registran en un archivo general que proporciona una visión global del rendimiento del modelo en cada configuración. Paralelamente, en otro archivo detallado, se almacenan los resultados de cada época, incluyendo métricas de exactitud y otros parámetros relevantes, con el objetivo de permitir un análisis exhaustivo de la evolución del modelo a lo largo del entrenamiento y obtener conclusiones detalladas sobre su desempeño en cada configuración.

Esta metodología asegura la reproducibilidad del proceso y facilita la obtención de resultados confiables y sólidos para el análisis y clasificación de imágenes médicas tridimensionales en el ámbito de las redes neuronales profundas.

El trabajo fue diseñado de manera modular, lo que permite abordar cada etapa del procesamiento de imágenes médicas de manera independiente y adaptable a diferentes escenarios. Estos módulos se implementan de forma secuencial en un flujo lógico, cubriendo desde la configuración inicial hasta la visualización final de resultados. Cada módulo está diseñado para integrarse sin problemas con los demás, pero también pueden usarse de forma independiente, dependiendo de los requerimientos específicos del usuario. A continuación, se presenta una descripción detallada de cada módulo:

Módulo de Configuración General

- Gestión de rutas: Definición de directorios de entrada y salida para los datos procesados.
- Selección de GPU: Configuración de la unidad de procesamiento gráfico para acelerar los cálculos intensivos.

Módulo de Carga de Datos

- Compatibilidad con DICOM: Funciones para cargar, visualizar y convertir imágenes desde estos formatos.
- Manejo de metadatos: Extracción y manipulación de metadatos asociados a los estudios, como información del paciente, parámetros de adquisición, etc.
- Optimización de memoria: Técnicas para manejar eficientemente el uso de memoria, especialmente útil al trabajar con volúmenes grandes.

Módulo de Preprocesamiento de Datos

- Normalización: Ajuste de la intensidad de los píxeles para asegurar una distribución uniforme y mejorar la convergencia de los modelos de aprendizaje profundo.
- Filtrado y reducción de ruido: Aplicación de filtros como Gaussian o median para eliminar artifacts y ruido en las imágenes.
- Segmentación básica: Herramientas para segmentar regiones de interés utilizando técnicas como umbralización o k-means clustering.
- Interpolación: Redimensionamiento de volúmenes a una resolución estándar, utilizando métodos como interpolación lineal o spline cúbico.

Módulo de Definición de Modelos y Arquitecturas 3D

- Constructor de modelos: Herramientas para definir arquitecturas desde cero, permitiendo la personalización de capas, funciones de activación, y esquemas de regularización.
- Modelos predefinidos: Implementación de arquitecturas estándar que pueden ser ajustadas según las necesidades del usuario.
- Entrenamiento: Funciones para entrenar los modelos con conjuntos de datos personalizados, incluyendo opciones de ajuste de hiperparámetros y técnicas de data augmentation.
- Evaluación y validación: Herramientas para la evaluación del rendimiento de los modelos, como métricas de exactitud y sensibilidad.

Módulo de Visualización

- Visualización 3D: Herramientas para la visualización interactiva de volúmenes, permitiendo la exploración en tiempo real de cortes axiales, coronales y sagitales
- Gráficos de métricas de rendimiento

Capítulo 4: Resultados

En esta sección se detallan los resultados obtenidos de los experimentos realizados sobre las arquitecturas de redes neuronales convolucionales 3D. Los experimentos se realizaron utilizando las configuraciones con mayor probabilidad de impactar positivamente en el rendimiento del modelo, tomando como base lo relevado en el estado del arte.

Dado el tamaño reducido del dataset original, que consistía en 1176 TC 3D sin cáncer y 419 con cáncer, se aplicó una estrategia de data augmentation para ampliar la cantidad de muestras disponibles. En particular, sobre el subconjunto de entrenamiento que contiene aproximadamente el 80% de las TC 3D con cáncer, es decir alrededor de 335 TC 3D con cáncer, se generaron dos nuevos ejemplares por cada una mediante rotaciones, cambios de brillo y pequeños zooms, sumando un total aproximado de 670 TC 3D adicionales. Esto representa un aumento aproximado del 42% respecto al conjunto de datos original total, contribuyendo a equilibrar la representación de clases y mejorar la robustez del modelo. El enfoque general se centró en encontrar un balance entre la profundidad de la red y las técnicas de regularización. Los resultados indicaron que, aunque es posible agregar múltiples capas para aumentar la capacidad de aprendizaje, esto no siempre implica una mejora en la exactitud del modelo cuando se trabaja con conjuntos de datos limitados.

En los trabajos que emplearon el mismo conjunto de datos, se recurrió sistemáticamente a bases de datos privadas para enriquecer la cantidad de imágenes. Por ejemplo, LUNA16 [66] fue utilizado por Liao et al. para lograr una accuracy del 82.1 % [67] y también fue empleado por Wit & Hammack [68]. Por otro lado, la base de datos NLST [69] fue incluida por Causey et al. (2019) para su modelo, que alcanzó una accuracy del 78.2 % [70]. El uso de LUNA16 y NLST no solo aportó más datos, sino que también permitió entrenar modelos más profundos y variados, lo que favorece un desempeño elevado.

Este trabajo se desarrolló íntegramente con el dataset público del Data Science Bowl 2017, sin utilizar datos externos. A pesar de esta restricción, los resultados obtenidos se encuentran muy próximos a los valores reportados por los trabajos que sí usaron datos privados adicionales. Este logro demuestra que es posible alcanzar un rendimiento comparable al del estado del arte. Además, al depender exclusivamente de datos públicos, este trabajo ofrece mayor transparencia, reproducibilidad y una base sólida para futuras investigaciones.

Métricas de validación

Al evaluar el desempeño de modelos de clasificación en el contexto del diagnóstico médico, es fundamental utilizar métricas que reflejen tanto la capacidad del modelo para detectar correctamente las enfermedades como su habilidad para evitar falsos diagnósticos. A continuación, se describen las principales métricas utilizadas en este estudio:

	Etiqueta positiva	Etiqueta negativa
Predicción positiva	Verdaderos Positivos (VP)	Falsos Positivos (FP)
Predicción negativa	Falsos Negativos (FN)	Verdaderos Negativos (VN)

Exactitud (Accuracy) = $\frac{VP+VN}{VP+VN+FP+FN}$

Es la proporción total de predicciones correctas sobre todas las predicciones realizadas. Esta métrica refleja qué tan bien el modelo clasifica tanto los casos positivos como negativos. Sin embargo, en conjuntos de datos desequilibrados, donde una clase puede ser mucho más prevalente que la otra, la exactitud puede no ser un indicador suficiente del rendimiento.

Sensibilidad = $\frac{VP}{VP+FN}$

También conocida como recall o tasa de verdaderos positivos, mide la capacidad del modelo para identificar correctamente los casos positivos, es decir, aquellos que presentan la condición o enfermedad. Es especialmente importante en aplicaciones médicas, ya que un valor bajo de sensibilidad podría implicar que el modelo no detecta a muchos pacientes enfermos.

Especificidad = $\frac{VN}{VN+FP}$

Representa la proporción de verdaderos negativos correctamente identificados, es decir, mide la capacidad del modelo para identificar correctamente los pacientes que no presentan la enfermedad. Un valor alto de especificidad es clave para evitar diagnósticos falsos positivos, que pueden llevar a tratamientos innecesarios o ansiedad en los pacientes.

4.1 Modelos de redes Ad Hoc

En esta sección, se presentan los resultados obtenidos por los tres modelos de CNN 3D sin la aplicación de *Transfer Learning* considerando las 10 particiones detalladas en la sección metodología.

Arquitectura	Exactitud (%)	Sensibilidad (%)	Especificidad (%)	Tiempo de entrenamiento (horas)
Modelo 1	73.2 ± 2.4	72.2 ± 2.1	73.8 ± 2.5	5.1 ± 0.8
Modelo 2	74.2 ± 2.3	72.9 ± 1.9	75.0 ± 2.4	6.3 ± 1.1
Modelo 3	71.9 ± 2.1	71.2 ± 1.8	71.6 ± 2.4	8.1 ± 1.3

Tabla 4.1: Resultados con Arquitecturas sin Transfer Learning

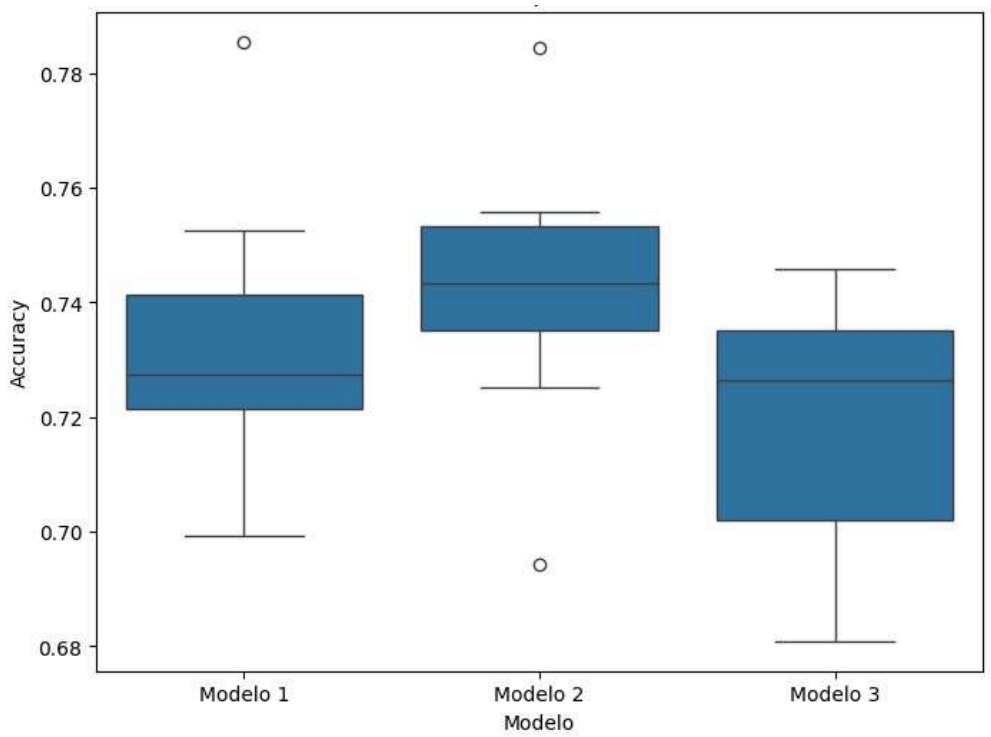


Figura 41: Exactitud por modelo.

En el gráfico se comparan las distribuciones de exactitud de tres modelos. El eje vertical muestra los valores de exactitud, mientras que cada caja representa el rango intercuartílico (es decir, los valores entre el primer y el tercer cuartil). La línea horizontal dentro de cada caja indica la mediana de la distribución y los círculos por encima o por debajo de los bigotes señalan valores atípicos (outliers).

- **Modelo 1:** Presenta una mediana de exactitud ligeramente inferior a la de Modelo 2, con un rango intercuartílico moderado y outliers tanto en la parte superior como

inferior. Esto indica cierta variabilidad en su desempeño, aunque mantiene un nivel de exactitud cercano al de los otros modelos.

- **Modelo 2:** Muestra la mediana más alta entre los tres, lo que sugiere que, en promedio, logra un mejor rendimiento. Además, el rango intercuartílico es relativamente estrecho, reflejando una consistencia mayor en los valores de exactitud y menos variación en comparación con los otros modelos.

- **Modelo 3:** Tiene una mediana de exactitud ligeramente menor que la de los otros dos, con un rango de valores más amplio y algún valor atípico que indica fluctuaciones considerables en su desempeño.

En conjunto, el análisis del diagrama sugiere que Modelo 2 obtiene la mejor exactitud promedio y menor dispersión, mientras que Modelo 1 y Modelo 3 presentan mayor variabilidad y valores de exactitud ligeramente menores.

Este comportamiento puede estar relacionado con la capacidad de las arquitecturas para generalizar. El Modelo 2 parece encontrar un balance adecuado entre simplicidad y profundidad, lo que podría explicar su rendimiento superior en precisión respecto a las otras arquitecturas. En contraste, el Modelo 3, a pesar de ser más complejo, no supera al Modelo 2, lo que sugiere que una mayor complejidad no necesariamente garantiza un mejor desempeño en términos de precisión.

La sensibilidad mide la capacidad del modelo para identificar correctamente los casos positivos, estos valores indican que las tres arquitecturas tienen un desempeño similar al identificar casos positivos, aunque el Modelo 2 muestra una ligera ventaja. El Modelo 3, nuevamente, no supera al Modelo 2, posiblemente debido a un sobreajuste o a dificultades para aprovechar su mayor complejidad en un conjunto de datos limitado.

La especificidad, que evalúa la capacidad del modelo para identificar correctamente los casos negativos.Aquí se observa nuevamente que el Modelo 2 tiene un desempeño ligeramente superior al de las otras dos arquitecturas, lo que refuerza la idea de que su diseño podría estar mejor optimizado para este conjunto de datos específico. La disminución en la especificidad del Modelo 3 podría estar vinculada a una mayor dificultad para manejar casos negativos debido a su complejidad.

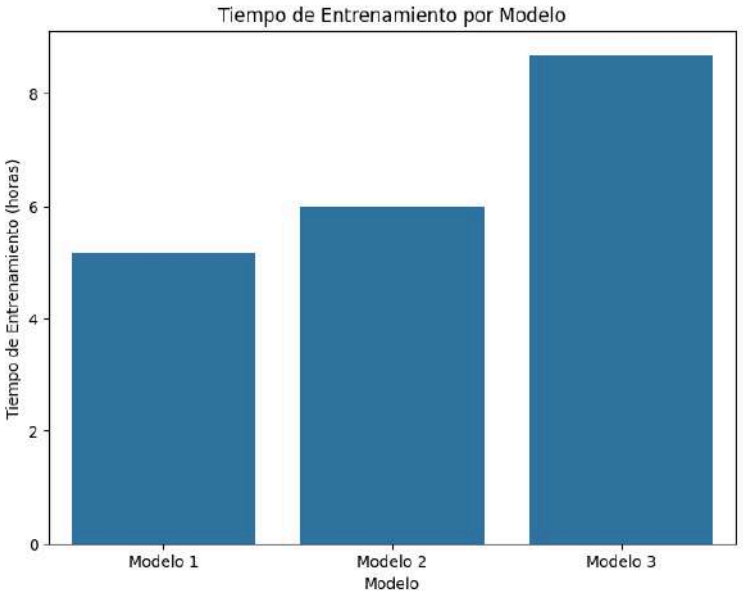


Figura 42: Tiempo de entrenamiento por modelo.

El tiempo de entrenamiento aumenta significativamente con la complejidad de las arquitecturas. El Modelo 1, siendo el más simple, requirió 5.1 ± 0.8 horas, mientras que el Modelo 2, más complejo, necesitó 6.3 ± 1.1 horas. Por otro lado, el Modelo 3 presentó el mayor tiempo de entrenamiento, con 8.1 ± 1.3 horas.

Este incremento es consistente con el número de parámetros y la profundidad de cada arquitectura. La mayor complejidad del Modelo 3 no solo implica un tiempo de entrenamiento más largo, sino también un mayor riesgo de subajuste, especialmente cuando el tamaño del conjunto de datos es limitado.

En general, los resultados muestran que las tres arquitecturas tienen un desempeño comparable, con diferencias sutiles en cada métrica. El Modelo 2 destaca como un punto intermedio, logrando el mejor balance entre exactitud, sensibilidad y especificidad, a costa de un tiempo de entrenamiento moderadamente mayor que el del Modelo 1. El Modelo 3, aunque más complejo y costoso en términos computacionales, no logra superar a los otros dos en ninguna métrica clave.

Este análisis permite entender cómo la complejidad arquitectónica afecta el desempeño de los modelos en distintos aspectos, proporcionando información valiosa para decisiones futuras en el diseño de redes neuronales. Las conclusiones específicas sobre estos resultados se presentarán en el capítulo correspondiente.

4.2 Modelos de Transfer Learning

En este apartado, se presentan los resultados obtenidos al aplicar transfer learning considerando las 10 particiones detalladas en la sección metodologìa. Se han considerado únicamente los resultados de las configuraciones que emplean tanto data augmentation como early stopping, con el objetivo de mejorar la generalización de los modelos en contextos de datos limitados.

Modelo Preentrenado	Exactitud(%)	Sensibilidad (%)	Especificidad (%)	Tiempo de entrenamiento (horas)
Modelo 4	76.4 ± 1.6	77.8 ± 1.5	75.1 ± 1.7	0.7 ± 0.3
Modelo 5	77.9 ± 1.6	79.5 ± 1.3	76.3 ± 1.6	1.4 ± 0.3
Modelo 6	74.2 ± 1.7	76.5 ± 1.9	71.8 ± 2.1	2.7 ± 0.5

Tabla 4.2: Resultados de los modelos pre entrenados con transfer learning

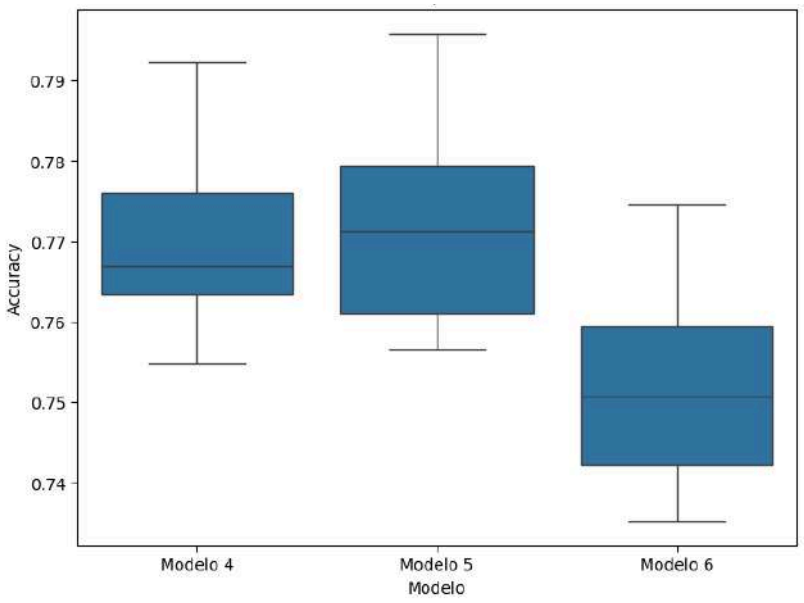


Figura 43: Exactitud por modelo de Transfer Learning.

Los modelos 4 y 5 demuestran un desempeño más consistente, mostrando una concentración de resultados altos y poca variabilidad en comparación con el Modelo 6. El modelo 5 se diferencia por presentar una distribución de resultados que se concentra en valores altos y de manera consistente, lo que sugiere un rendimiento robusto y confiable en múltiples ejecuciones. Por el contrario, el Modelo 6 exhibe una mayor dispersión en sus resultados, evidenciando variabilidad y, en promedio, una exactitud inferior.

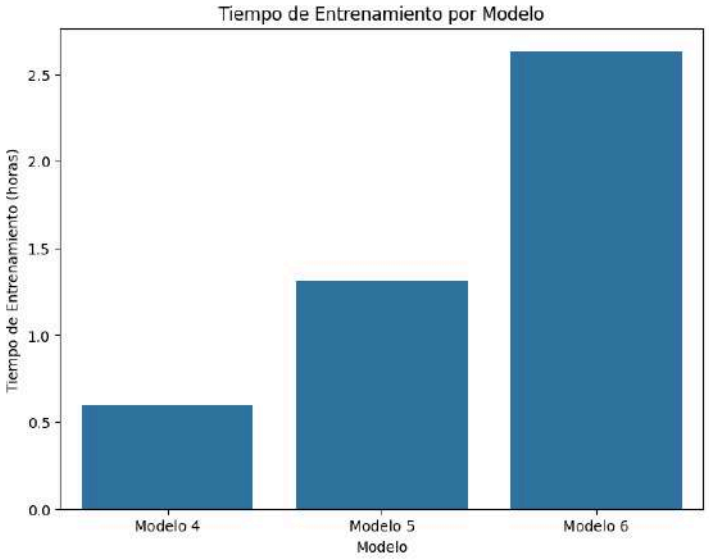


Figura 44: Tiempo de entrenamiento por modelo de Transfer Learning.

El Modelo 4 muestra un tiempo de entrenamiento significativamente menor en comparación con los otros dos, reflejando un proceso de ajuste más rápido. El Modelo 5 requiere más tiempo que el Modelo 4, pero sigue siendo moderado. Por otro lado, el Modelo 6 registra el mayor tiempo de entrenamiento, casi duplicando el de los modelos anteriores, lo cual sugiere una arquitectura o configuración más compleja, o bien la necesidad de más iteraciones para alcanzar su nivel de rendimiento final.

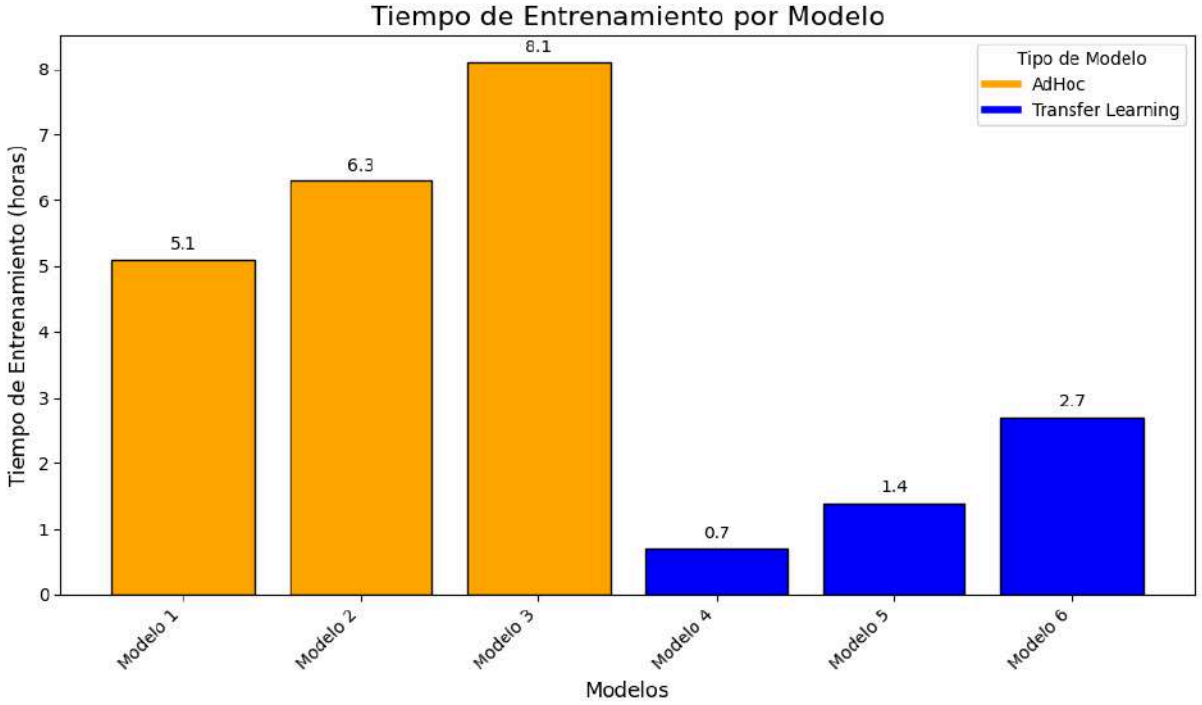


Figura 45: Tiempo de entrenamiento de modelos AdHoc y de Transfer Learning.

La Figura 45 muestra una comparación clara entre los tiempos de entrenamiento de los modelos diseñados desde cero (Ad Hoc, Modelos 1 a 3) y aquellos construidos mediante Transfer Learning (Modelos 4 a 6). Todos los modelos abordan el mismo problema de clasificación de TC 3D, distinguiendo entre imágenes con o sin cáncer. Se observa que los modelos Ad Hoc requieren considerablemente más tiempo de entrenamiento, alcanzando hasta más de 8 horas en el caso del Modelo 3. En contraste, los modelos basados en Transfer Learning demandan tiempos notablemente menores, siendo el Modelo 4 el más rápido con tan solo 0.7 horas. Esta diferencia resalta una de las principales ventajas del enfoque de transfer learning: la eficiencia en el entrenamiento, especialmente valiosa cuando se dispone de recursos computacionales limitados o se necesita una rápida iteración sobre modelos.

Capítulo 5: Conclusiones

Este trabajo aborda la creciente demanda de herramientas que faciliten el análisis y la gestión de datos médicos volumétricos, permitiendo una transición eficiente hacia el uso de imágenes 3D en investigación y aplicaciones clínicas. En este contexto, el trabajo desarrollado se posiciona como una solución robusta y funcional, respondiendo a las necesidades actuales del laboratorio, aportando una continuidad natural de las líneas de investigación previamente desarrolladas en imagenes mèdicas 2D.

Más allá de los desafíos técnicos y organizativos que implica este trabajo, la experiencia general es enriquecedora. Haber llevado adelante este proyecto resultò no solo un gran desafío, sino también una oportunidad de crecimiento personal y académico. A lo largo del proceso, se pudieron aplicar y consolidar conocimientos adquiridos durante toda la carrera y también disfrutar del camino de investigaciòn y experimentaciòn. El entusiasmo por aprender y superar obstáculos fue constante, incluso en las etapas más exigentes. Finalizar este proyecto genera una gran satisfacción, por los resultados obtenidos y por el trayecto formativo recorrido que deja aprendizajes duraderos y motiva a seguir investigando y formándose en el futuro.

El desarrollo de este trabajo se llevó a cabo exclusivamente utilizando un único conjunto de datos públicos. Esta limitación impuso un desafío importante en cuanto a la cantidad de imágenes disponibles para el entrenamiento y evaluación de los modelos. Aun así, los resultados obtenidos lograron aproximarse a los de trabajos que utilizaron conjuntos de datos complementarios como LUNA16 o NLST, que aportaron una mayor diversidad y volumen de datos. En este contexto, alcanzar mètricas de accuracy similares otorga un valor significativo al trabajo, tanto por su reproducibilidad como por su potencial para servir de base a futuras investigaciones con recursos abiertos y accesibles.

El código desarrollado combina capacidades de preprocesamiento, carga y gestión de datos, modelado y visualización adaptadas a imágenes volumétricas. Esta integración de funcionalidades proporciona una experiencia de usuario eficiente y permite un impacto positivo en aplicaciones prácticas. En cuanto a la gestión de datos, se automatiza el procesamiento de archivos DICOM, convirtiéndolos en imágenes listas para el análisis. Además, es adaptable a otros formatos de datos médicos, asegurando su compatibilidad futura con distintas fuentes de información.

En el preprocesamiento, se incorporan técnicas avanzadas como la interpolación volumétrica, que estandariza los datos a una cantidad definida de cortes, así como la normalización y ecualización de histogramas en cada slice, mejorando la calidad y consistencia de las imágenes para el aprendizaje profundo. También se incluyen modelos personalizados de redes neuronales convolucionales 3D, facilitando el uso de *transfer learning* para aprovechar modelos pre entrenados y mejorar el rendimiento con datos limitados.

En cuanto al mediano plazo, la posibilidad de acceder a más datos permitirá una mejor generalización de los modelos y un mayor impacto clínico. A largo plazo, se podría evolucionar hacia la incorporación de redes neuronales más avanzadas y aplicaciones de apoyo al diagnóstico, contribuyendo al avance de la medicina personalizada.

Los resultados experimentales mostraron que el rendimiento de los modelos no siempre depende de la complejidad de la arquitectura, sino de un balance adecuado entre la profundidad del modelo, la cantidad y calidad de los datos y los recursos computacionales. Se evidenció que configuraciones técnicas de regularización pueden superar modelos excesivamente complejos, lo que destaca la importancia de una selección estratégica de los híperparámetros y arquitecturas. El sobreajuste y la baja generalización son riesgos comunes cuando se trabaja con una cantidad de datos limitada. Este desafío subraya la necesidad de fomentar colaboraciones para expandir los conjuntos de datos disponibles y mejorar la calidad de los resultados.

Además, la metodología utilizada en este trabajo puede servir como base para futuros desarrollos similares en el estudio de imágenes 3D. Se propone como un modelo de trabajo estructurado, con lineamientos replicables que pueden guiar nuevas investigaciones que apliquen técnicas de Deep Learning en imágenes médicas 3D. Esta base metodológica permite adaptar el código desarrollado a nuevas aplicaciones y contribuye a la continuidad científica en un campo en constante crecimiento.

5.1: Gestión del proyecto

Realizar este proyecto de forma individual supuso un desafío significativo. El acompañamiento de los directores fue una gran ayuda para sostener el rumbo del proyecto. Uno de los aspectos más complejos fue encontrar tiempos consistentes para avanzar en las distintas etapas, en paralelo con otras responsabilidades académicas y personales. Esta dinámica generó desvíos respecto a lo planificado inicialmente, algo difícil de manejar en un esquema unipersonal. No obstante, estas circunstancias permitieron desarrollar una gran autonomía y consolidar habilidades como la toma de decisiones técnicas, la gestión del tiempo y la resiliencia ante imprevistos, aportando un valor formativo integral al proceso.

Resumen general de horas y semanas.

Resumen	Planificación	Ejecución
Horas	580	760
Semanas	29	53

Se planificaron 580 horas de trabajo, distribuidas en 29 semanas (20 horas por semana).
La ejecución del trabajo llevó 760 horas de trabajo, distribuídas en 53 semanas.

Detalle de tareas: planificación y ejecución

A continuación se muestra una tabla comparativa para cada tarea:

Tarea	Planificación		Ejecución		Diferencia (horas)
	Semanas	Horas	Semanas	Horas	
1. Estudio teórico de modelos de redes de aprendizaje profundo	8	100	8	100	0
2. Reuniones y entrevistas con médicos especialistas	6	12 (aprox. 2 horas por encuentro)	6	12	0
3. Pruebas y análisis de algoritmos de redes convolucionales existentes	10	80	12	100	+20
4. Revisión de Trabajos Específicos en Imágenes Médicas 3D	9	60	9	60	0
5. Estudio de Estrategias de Diseño de Redes Profundas	8	70	10	100	+30
6. Desarrollo de los Modelos y Algoritmos	9	180	16	280	+100
7. Redacción y Confección del Informe	8	90	12	120	+30

Se realizó una planificación inicial de 580 horas distribuidas en 29 semanas, basándose en un promedio de 20 horas semanales. Sin embargo, a lo largo del proyecto se presentaron diversos desvíos y contingencias que alteraron el ritmo de trabajo previsto.

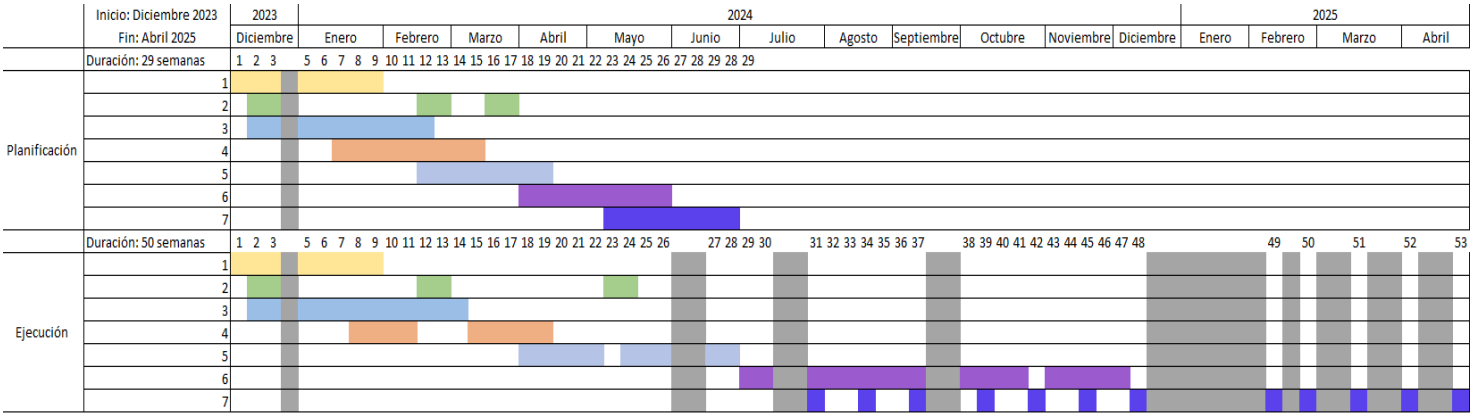
En primer lugar, se evidenció que, pese a respetar inicialmente la estimación de 20 horas semanales, conforme avanzaba el proyecto la dedicación semanal se redujo por situaciones personales, entre las que destacó la incorporación a un nuevo empleo. Esta situación, junto con otros imprevistos, como periodos de inactividad atribuibles a cuestiones de salud y viajes, impactaron directamente en la productividad semanal.

Como resultado de estos factores, la ejecución real del proyecto ascendió a 760 horas, lo que representa un incremento de 180 horas sobre lo planificado. Asimismo, la extensión en

la duración del proyecto pasó de las 29 semanas inicialmente previstas a 53 semanas en total.

Este escenario ilustra cómo, a pesar de la intención de mantener un ritmo constante de 20 horas semanales, las circunstancias personales y los imprevistos inherentes a proyectos complejos pueden requerir ajustes significativos en la planificación original para garantizar la calidad y la integridad del trabajo realizado.

A continuación se muestra el diagrama de Gantt final del proyecto en comparación al inicial:



En resumen, las principales causas del desvío identificadas fueron:

1. Inicio de actividades laborales.
2. Periodos de inactividad por motivos de salud personal.
3. Falta de experiencia en la planificación de proyectos de esta magnitud.
4. Ajustes técnicos de software / hardware.
5. Viajes imprevistos.

Durante el desarrollo del proyecto, se evidenció que, a pesar de contar con una planificación inicial sólida, es importante disponer de márgenes de maniobra que permitan responder de forma ágil a situaciones adversas, como lo fueron los cambios en nuevos compromisos laborales, problemas de salud y viajes imprevistos. Esta capacidad de adaptación facilitó la implementación de medidas correctivas oportunas y el reajuste de los tiempos de ejecución, lo que resultó fundamental para sortear los obstáculos y mantener la integridad del trabajo realizado.

Asimismo, la gestión del proyecto resaltó la relevancia de una comunicación fluida y un seguimiento constante de los avances. El uso de herramientas de monitoreo y el establecimiento de revisiones constantes permitieron identificar tempranamente la

desviación del plan original, posibilitando ajustes precisos y efectivos. De esta manera, se consolidó un ambiente de trabajo colaborativo y proactivo, donde cada desafío se transformó en una oportunidad para aprender y mejorar.

Aprendizajes en la gestión de proyectos

- Flexibilidad en la planificación: La capacidad de ajustar el cronograma ante imprevistos ha sido esencial para preservar la calidad del trabajo sin desviar significativamente el esfuerzo total planificado.
- Importancia del seguimiento continuo: las reuniones intermedias, permitieron identificar desviaciones a tiempo y aplicar medidas correctivas efectivas.
- Equilibrio entre calidad y tiempo: La experiencia demuestra que, aunque la ejecución requirió una extensión en la duración total del proyecto, la estrategia adoptada permitió mantener una calidad alta en el análisis y desarrollo, logrando un balance adecuado entre compromisos personales y exigencias técnicas.

Esta gestión adaptable y proactiva no solo aseguró el cumplimiento de los objetivos, sino que también proporcionó valiosas lecciones para la conducción de futuros proyectos en entornos complejos y dinámicos.

Bibliografía

[1] Yi, J., Lee, S., & Kim, H. (2023). Advancements in 3D Medical Imaging for Early Disease Detection: Implications for Survival Rates and Clinical Outcomes. Journal of Medical Imaging and Health Informatics.

[2] Doi, K. (2007). Computer-aided diagnosis in medical imaging: Historical review, current status and future potential. Computerized Medical Imaging and Graphics, 31(4–5), 198–211.

[3] Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A. A., Ciompi, F., Ghafoorian, M., van der Laak, J. A. W. M., van Ginneken, B., & Sánchez, C. I. (2017). A survey on deep learning in medical image analysis. Medical Image Analysis, 42, 60–88.

[4] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. Communications of the ACM, 60(6), 84–90.

[5] Ji, S., Xu, W., Yang, M., & Yu, K. (2013). 3D convolutional neural networks for human action recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 35(1), 221–231..

[6] A. Esteva et al., "A guide to deep learning in healthcare," Nature Medicine, 2019.

[7] J. De Fauw et al., "Clinically applicable deep learning for diagnosis in medical imaging," Nature Medicine, 2018.

[8] Çiçek, Ö., et al. 3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation. Medical Image Computing and Computer-Assisted Intervention (MICCAI), 2016.

[9] Kamnitsas, K., et al. Efficient Multi-Scale 3D CNN with Fully Connected CRF for Accurate Brain Lesion Segmentation. IEEE Transactions on Medical Imaging, 2017.

[10] Shin, H. C., Roth, H. R., Gao, M., Lu, L., Xu, Z., Nogues, I., Yao, J., Mollura, D., & Summers, R. M. (2016). Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning. IEEE Transactions on Medical Imaging,

[11] Hassanzadeh T., Essam D., Sarker R. S. (2021). Evolutionary Deep Attention Convolutional Neural Networks for 2D and 3D Medical Image Segmentation. J Digit Imaging.

[12] Chen S., Ma K., Zheng Y. (2019). Med3D: Transfer Learning for 3D Medical Image Analysis.

[13] Xiaonan Shao, Xinyu Ge, Jianxiong Gao, Rong Niu, Yunmei Shi, Xiaoliang Shao, Zhenxing Jiang, Renyuan Li & Yuetao Wang. (2024). Transfer learning–based PET/CT three-dimensional convolutional neural network fusion of image and clinical information for prediction of EGFR mutation in lung adenocarcinoma. BMC Medical Imaging.

[14] Isensee, F., Petersen, J., Klein, A., Zimmerer, D., Jaeger, P. F., Kohl, S., Wasserthal, J., Koehler, G., Norajitra, T., Wirkert, S., & Maier-Hein, K. H. (2018). nnU-Net: Self-adapting Framework for U-Net-Based Medical Image Segmentation.

[15] Soto, P. (n.d.). Convolución Matricial Aplicado al Procesamiento de Imágenes. Instituto Tecnológico de Costa Rica.
https://www.tec.ac.cr/sites/default/files/media/doc/presentacion_pablosoto.pdf

[16] Gundersen, G. (2017, 24 de febrero). Convolutional neural networks: An overview. Gregory Gundersen. https://gregorygundersen.com/blog/2017/02/24/cnns/

[17] Gonzalez, R. C., & Woods, R. E. (2008). Digital Image Processing (3rd Edition). Pearson Education.

[18] MathWorks. (n.d.). Convolutional Neural Network. MathWorks.
https://la.mathworks.com/discovery/convolutional-neural-network.html

[19] S. K. Bhargava, "Textbook of Radiology and Imaging," 3rd edition, Elsevier, 2006.

[20] Lisle, D. A., & Hacking, C. (2023). *Imaging for Students* (5th ed.). Routledge. Recuperado de Taylor & Francis.

[21] Mettler, F. A. (2019). *Essentials of Radiology* (3rd ed.). Elsevier. Recuperado de Elsevier.

[22] Thengane, V., Zhu, X., Bouzerdoum, S., Phung, S. L., & Li, Y. (2025). Foundational models for 3D point clouds: A survey and outlook.

[23] Pianykh, O. S. (2008). Digital Imaging and Communications in Medicine (DICOM): A Practical Introduction and Survival Guide. Springer.

[24] Roza Dastres, Mohsen Soori. Artificial Neural Network Systems. International Journal of Imaging and Robotics (IJIR), 2021, 21 (2), pp.13-25.

[25] NVIDIA. (2021). Difference Between Supervised, Unsupervised, & Reinforcement Learning. NVIDIA Blog.

[26] Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction (2ª ed.). MIT Press.

[27] Jordan, J. (2018). Neural network learning rate - explanation and experimentation. Jeremy Jordan. https://www.jeremyjordan.me/nn-learning-rate/

[28] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.

[29] Software Testing Help. (2023l). Data Mining vs Machine Learning vs AI - Key Differences. Software Testing Help. https://www.softwaretestinghelp.com/data-mining-vs-machine-learning-vs-ai/

[30] LeCun, Y., Bengio, Y., Hinton, G. (2015). Deep learning. Nature, 521(7553), 436–444. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.

[31] McBee, M. P., et al. (2018). Deep learning in radiology. Academic Radiology,

[32] Suárez Castro, R. M., Ladino Vega, I. D. (2023). Redes neuronales aplicadas al control estadístico de procesos con cartas de control EWMA (Neural networks applied to statistical process control with EWMA control charts)

[33] Science Focus. (n.d.). How do machine learning GANs work? Science Focus. https://www.sciencefocus.com/future-technology/how-do-machine-learning-gans-work

[34] ResearchGate. (n.d.). Layers and their abstraction in deep learning. ResearchGate. https://www.researchgate.net/figure/Layers-and-their-abstraction-in-deep-learning-Image-recognition-as-measured-by-ImageNet_fig17_326531654

[35] ResearchGate. (n.d.). 2D and 3D convolution operations. ResearchGate. https://www.researchgate.net/figure/2D-and-3D-convolution-operations-a-Applying-2D-convolution-on-image-results-in-an-image_fig1_374780566

[36] Zhang, H., & Qie, Y. (2023). Applying deep learning to medical imaging: A review. Applied Sciences, 13(18), 10521.

[37] Bay, H., Tuytelaars, T., & Van Gool, L. (2006). SURF: Speeded Up Robust Features. *European Conference on Computer Vision (ECCV)*.

[38] Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*.

[39] ResearchGate. (n.d.). Various geometric and photometric transformations. ResearchGate. https://www.researchgate.net/figure/Various-geometric-and-photometric-transformations-65_fig3_346490848

[40] ResearchGate. (n.d.). Original image and incomplete image with 75 image blocks of size 4x4 missing. ResearchGate. https://www.researchgate.net/figure/aOriginal-image-bIncomplete-image-with-75-image-blocks-of-size-4-4-missing_fig3_312655192

[41] Kokkinos, I. (2016). UberNet: Training a 'Universal' Convolutional Neural Network for Low-, Mid-, and High-Level Vision using Diverse Datasets and Limited Memory.

[42] He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition.

[43] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... & Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. International Conference on Learning Representations (ICLR).

[44] Wu, H., Xiao, B., Codella, N., Liu, M., Dai, X., Yuan, L., & Zhang, L. (2021). CvT: Introducing convolutions to vision transformers. Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 22, 22-31.

[45] Yeganeh, Y., Farshad, A., Weinberger, P., Ahmadi, S.-A., Adeli, E., & Navab, N. (2023). Transformers Pay Attention to Convolutions Leveraging Emerging Properties of ViTs by Dual Attention-Image Network. Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops, 2304-2315.

[46] Salehin, I., & Kang, D. K. (2023). A Review on Dropout Regularization Approaches for Deep Neural Networks within the Scholarly Domain. Electronics, 12(14), 3106.

[47] S. J. H. et al. "Transfer learning in medical imaging: a systematic review," Journal of Healthcare Informatics Research, vol. 4, no. 1, pp. 18-42, 2020.

[48] ResearchGate. (n.d.). Real-time assembly operation recognition with fog computing and transfer learning for human-centered intelligent manufacturing. ResearchGate. https://www.researchgate.net/publication/342400905_Real-Time_Assembly_Operation_Recognition_with_Fog_Computing_and_Transfer_Learning_for_Human-Centered_Intelligent_Manufacturing

[49] A. M. et al. "Transfer learning for medical image classification: a review," IEEE Access, vol. 8, pp. 133555-133573, 2020.

[50] Tajbakhsh, N., Jeyaseelan, L., Li, Q., Chiang, J. N., Wu, Z., & Ding, X. (2020). Embracing Imperfections: Semi-Supervised Learning with Noisy Data in Medical Image Analysis. Medical Image Analysis, 61, 101693.

[51] Lundervold, A. S., & Lundervold, A. (2021). An Overview of Deep Learning in Medical Imaging Focusing on MRI. Zeitschrift für Medizinische Physik, 31(1), 104-127.

[52] U.S. Department of Health and Human Services. (1996). Health Insurance Portability and Accountability Act (HIPAA). Retrieved from https://www.hhs.gov/hipaa

[53] Ley 25.326/2000, de Protección de los Datos Personales (Argentina). Boletín Oficial de la República Argentina, 30 de octubre de 2000. Autoridad de Aplicación: Dirección Nacional de Protección de Datos Personales, AAIP.

[54] Shin, H.-C., Tenenholtz, N. A., Rogers, J. K., Schwarz, C. G., Senjem, M. L., Gunter, J. L., Andriole, K., & Michalski, M. (2018). Medical Image Synthesis for Data Augmentation and Anonymization using Generative Adversarial Networks.

[55] Data Science Bowl 2017 (sin fecha). Kaggle. Recuperado el 12 de abril de 2024. https://www.kaggle.com/c/data-science-bowl-201

[56] Gonzalez, R. C., & Woods, R. E. (2018). Digital Image Processing (4th ed.). Pearson.

[57] Jia, X., Chang, H., & Tuytelaars, T. (2017). Super-Resolution with Deep Adaptive Image Resampling.

[58] Survey: interpolation methods in medical image processing. TAU Image Processing, 2021.

[59] Bushberg, J. T., Seibert, J. A., Leidholdt, E. M., & Boone, J. M. (2011). The Essential Physics of Medical Imaging (3rd ed.). Lippincott Williams & Wilkins.

[60] Brain Tumor 3D [Training]. (sin fecha). Kaggle. Recuperado el 27 de abril de 2024. https://www.kaggle.com/code/brain-tumor-3d-training

[61] Masters, D., & Luschi, C. (2018). Revisiting small batch training for deep neural networks.

[62] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization.

[63] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, Zheng, X. (2016). TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems.

[64] Harris, C. R., Millman, K. J., van der Walt, S. J., et al. (2020). Array programming with NumPy. Nature, 585, 357–362.

[65] Nickolls, J., Buck, I., Garland, M., & Skadron, K. (2008). Scalable parallel programming with CUDA. ACM Queue, 6(2), 40–53.

[66] Papers With Code. (s. f.). LUNA16 Dataset. Recuperado el 12 de abril de 2024, de https://paperswithcode.com/dataset/luna16.

[67] Liao, F., Liang, M., Li, Z., Hu, X., & Song, S. (2017). Evaluate the malignancy of pulmonary nodules using the 3D deep leaky noisy-or network.

[68] de Wit, J., & Hammack, D. (2017). Second-place solution, Data Science Bowl 2017: Predicting lung cancer. Blog.

[69] National Cancer Institute. (s. f.). Learn about NLST CT images. Cancer Data Access System. Recuperado el 12 de abril de 2024, de https://cdas.cancer.gov/learn/nlst/images/

[70] Causey, J. L., Guan, Y., Dong, W., Walker, K., Qualls, J. A., Prior, F., & Huang, X. (2019). DeepScreener: A deep learning model for lung cancer screening. arXiv. https://arxiv.org/abs/1906.00240

Apéndice A: Definiciones previas

Activación Softmax: Función utilizada en la capa final de una red neuronal para convertir los valores de salida en probabilidades, comúnmente utilizada en clasificación.

Activation function (Función de Activación): Función matemática aplicada a la salida de una neurona para introducir no linealidades en el modelo.

Arquitectura de Red Neuronal: Estructura de una red neuronal que determina cómo se organizan y conectan las capas de nodos en el modelo.

Backpropagation: Algoritmo de optimización que ajusta los pesos de una red neuronal mediante la propagación del error hacia atrás desde la capa de salida hasta la capa de entrada.

Batch Normalization: Técnica de normalización utilizada en redes neuronales para estabilizar el entrenamiento y mejorar la velocidad de convergencia.

Capa Fully Connected (FC): Capa de una red neuronal en la que cada neurona está conectada a todas las neuronas de la capa anterior.

Class Imbalance (Desbalanceo de Clases): Problema que ocurre cuando una clase en un conjunto de datos tiene significativamente más ejemplos que otra, lo que puede afectar la precisión del modelo.

CNN 2D: Red neuronal convolucional diseñada para procesar datos de imágenes bidimensionales, en contraste con su versión 3D, que maneja datos volumétricos.

CNN 3D: Red neuronal convolucional diseñada específicamente para trabajar con datos en tres dimensiones, como imágenes médicas volumétricas.

Convolución: Operación matemática en redes neuronales convolucionales que se utiliza para extraer características locales de una imagen.

Convolución 3D: Operación matemática en redes neuronales convolucionales 3D que permite extraer características espaciales y temporales de datos volumétricos.

Convolutional Layer (Capa Convolucional): Capa de una red neuronal que aplica operaciones de convolución para extraer características locales de las imágenes.

Data Augmentation (Aumento de Datos): Técnica que implica modificar los datos de entrenamiento mediante transformaciones (rotaciones, recortes, etc.) para aumentar la cantidad de datos disponibles.

Deep Learning: Rama del aprendizaje automático que utiliza redes neuronales profundas para modelar y resolver tareas complejas.

Dropout: Técnica de regularización que consiste en "apagar" aleatoriamente algunas neuronas durante el entrenamiento para prevenir el sobreajuste.

Entrenamiento por Mini-Batches: Técnica de entrenamiento donde los datos se dividen en pequeños lotes o "mini-batches", en lugar de procesar todo el conjunto de datos de una vez.

Época (Epochs): Una iteración completa a través de todo el conjunto de datos de entrenamiento durante el proceso de entrenamiento de un modelo.

Escalado de Imagen (Image Scaling): Técnica que ajusta el tamaño de las imágenes para que se adecuen a las dimensiones requeridas por el modelo de red neuronal.

Exactitud: Porcentaje de predicciones correctas realizadas por un modelo de clasificación.

F1-Score: Métrica que combina precisión y recall en una única medida para evaluar la calidad de un modelo de clasificación.

Falso Negativo: Resultado de una predicción incorrecta en la que un modelo clasifica erróneamente una instancia positiva como negativa.

Falso Positivo: Resultado de una predicción incorrecta en la que un modelo clasifica erróneamente una instancia negativa como positiva.

Fase de Entrenamiento: Proceso en el cual un modelo ajusta sus parámetros mediante el aprendizaje de un conjunto de datos etiquetados.

Feature Extraction: Proceso de identificar y extraer características relevantes de los datos de entrada para alimentar a un modelo de aprendizaje automático.

Hiperparámetro: Parámetro que se ajusta antes del entrenamiento de un modelo y controla el comportamiento del proceso de aprendizaje.

Histograma de intensidad: Representación gráfica de la distribución de los valores de píxeles en una imagen, útil para técnicas de ecualización y ajuste de contraste.

Imagen Axial: Imagen médica obtenida a partir de cortes transversales del cuerpo en dirección horizontal, comúnmente utilizada en tomografías computarizadas.

Imagen Coronal: Imagen médica obtenida a partir de cortes verticales que dividen al cuerpo en la parte anterior y posterior.

Imagen DICOM: Formato estándar para el almacenamiento y transmisión de imágenes médicas digitales, utilizado en radiología y otras disciplinas clínicas.

Imagen de Alta Resolución: Imagen que tiene un número elevado de píxeles o voxeles, permitiendo una mayor claridad y detalle.

Imagen de Resonancia Magnética Funcional (fMRI): Tipo de imagen médica que mide la actividad cerebral mediante la detección de cambios en el flujo sanguíneo.

Imagen Sagital: Imagen médica obtenida de cortes longitudinales que dividen al cuerpo en lados izquierdo y derecho, utilizada en resonancias magnéticas.

Keras: Framework de alto nivel para redes neuronales que facilita la construcción y entrenamiento de modelos de aprendizaje profundo.

Mapeo de Características: Proceso de transformación de los datos de entrada en un espacio de características de mayor dimensión para facilitar el aprendizaje por parte de la red neuronal.

Medición de Sensibilidad: Métrica que evalúa la capacidad de un modelo para identificar correctamente los casos positivos.

Métrica de Especificidad: Medida que indica la capacidad de un modelo para identificar correctamente las instancias negativas.

Métricas de Evaluación: Conjunto de medidas utilizadas para evaluar la precisión y el rendimiento de un modelo de aprendizaje automático, como la precisión, sensibilidad, y especificidad.

Muestreo: Proceso de seleccionar un subconjunto de datos de un conjunto más grande para realizar experimentos o validación.

Normalización de Imagen: Proceso de ajustar los valores de los píxeles de una imagen a un rango específico, como [0, 1], para facilitar el entrenamiento del modelo.

Optimización: Proceso de ajustar los parámetros de un modelo para minimizar el error de predicción.

Overfitting: Situación en la que un modelo de aprendizaje automático se ajusta demasiado a los datos de entrenamiento, perdiendo capacidad de generalización.

Pad (Relleno): Técnica en la que se añaden ceros (o valores específicos) alrededor de los bordes de las imágenes para mantener las dimensiones constantes tras aplicar convoluciones.

Pérdida (Loss Function): Función que mide la discrepancia entre las predicciones del modelo y las etiquetas reales durante el entrenamiento.

Pooling: Operación en redes neuronales que reduce las dimensiones espaciales de las imágenes, conservando las características más importantes.

Preprocesamiento de datos: Fase del flujo de trabajo en la que se limpian, transforman y normalizan los datos antes de ser alimentados al modelo.

Preprocesamiento de imágenes: Conjunto de técnicas utilizadas para mejorar la calidad y utilidad de las imágenes antes de que sean analizadas por un modelo.

Reentrenamiento (Fine-Tuning): Técnica de transferencia de aprendizaje en la que se ajustan los parámetros de un modelo preentrenado a un nuevo conjunto de datos para mejorar su rendimiento.

Red Neuronal Convolucional (CNN): Tipo de red neuronal que utiliza capas convolucionales para procesar datos de imágenes de manera eficiente.

Red Neuronal Feedforward: Tipo de red neuronal en la que las conexiones entre las neuronas siguen un flujo unidireccional, desde la capa de entrada hasta la capa de salida.

Redes Generativas Antagónicas (GANs): Modelos de aprendizaje profundo formados por dos redes que compiten entre sí: una genera datos y la otra los discrimina, mejorando la calidad de las predicciones.

Redes Neuronales Recurrentes (RNN): Tipo de red neuronal que tiene conexiones de retroalimentación, permitiendo procesar secuencias de datos.

Regularización: Técnica utilizada para prevenir el sobreajuste en un modelo al penalizar configuraciones excesivamente complejas.

ReLU (Rectified Linear Unit): Función de activación utilizada en redes neuronales que convierte todos los valores negativos a cero y deja los valores positivos sin cambios.

Resampling: Técnica que ajusta la resolución o tamaño de una imagen para que se adapte a los requisitos del modelo o de la plataforma de análisis.

Resolución Espacial: Medida de la calidad de una imagen en términos de la precisión con la que se representan los detalles espaciales.

RM (Resonancia Magnética): Técnica de imagen médica que utiliza campos magnéticos y ondas de radio para obtener imágenes detalladas del interior del cuerpo.

ROI (Region of Interest): Área de una imagen médica seleccionada para su análisis, generalmente correspondiente a un órgano o anomalía.

Segmentación semántica: Tarea en visión por computadora en la que cada píxel de una imagen es asignado a una categoría específica, como "tumor" o "tejido normal".

Segmentation: Proceso de identificar y marcar las regiones de interés en una imagen médica, como tumores u órganos.

Slice: Capa o corte de una imagen médica en 2D, generalmente utilizado en tomografías computarizadas (TC) y resonancias magnéticas (RM).

Tamaño de Kernel: Dimensiones de la ventana de convolución utilizada en una capa convolucional para extraer características locales de la imagen.

Tasa de Aprendizaje: Parámetro que controla el tamaño de los pasos que da el modelo al ajustar los pesos durante el entrenamiento.

TC (Tomografía Computarizada): Técnica de imagen que utiliza rayos X para crear imágenes transversales del cuerpo, comúnmente usada en diagnóstico médico.

Tensor: Objeto de datos multidimensional utilizado en las redes neuronales para representar las entradas, salidas y parámetros del modelo.

TensorFlow: Framework de código abierto para el desarrollo de modelos de aprendizaje automático, ampliamente utilizado en redes neuronales profundas.

Trade-off: Compromiso entre dos o más características en conflicto, donde mejorar una implica sacrificar otra.

Transfer Learning: Técnica en la que un modelo entrenado en un conjunto de datos se utiliza como base para un nuevo modelo en otro conjunto de datos relacionado.

Underfitting: Situación en la que un modelo no logra capturar las características importantes del conjunto de datos, resultando en un rendimiento deficiente.

Validación Cruzada: Técnica de evaluación que consiste en dividir el conjunto de datos en varios subconjuntos y entrenar el modelo en diferentes combinaciones de esos subconjuntos.

Volumen de Imágenes: Conjunto de slices que componen una imagen médica 3D, que representa una estructura volumétrica.

Volumen de Interés (VOI): Área seleccionada dentro de una imagen médica 3D para análisis detallado.

Voxel: Elemento volumétrico en una imagen médica 3D, equivalente a un píxel en una imagen 2D, pero con dimensión adicional.

Voxelización: Proceso de convertir una imagen médica 3D en un conjunto de voxeles, que son las unidades volumétricas que representan el volumen total de la imagen.

Apéndice B: Código

En esta sección del apéndice se presenta la estructura general del código completo, por módulo y por función, incluyendo una breve descripción para cada una de ellas.

Módulo de carga de datos

Carga y organización de volúmenes DICOM:

Carga las imágenes DICOM de una carpeta específica, organiza las imágenes en un volumen tridimensional y extrae el espaciado de píxeles en los ejes X, Y y Z.

```
def load_dicom_volume(dicom_folder):
    dicom_files = glob(dicom_folder + '/*.dcm')
    slices = [pydicom.dcmread(f) for f in dicom_files]

    # Ordenar los cortes según su posición en Z
    slices.sort(key=lambda x: float(x.ImagePositionPatient[2]))

    # Crear un array 3D a partir de los cortes
    images = np.stack([s.pixel_array for s in slices], axis=-1)

    # Obtener espaciado original (X, Y)
    pixel_spacing = slices[0].PixelSpacing

    # Obtener el espaciado en Z
    if hasattr(slices[0], 'SliceThickness'):
        slice_thickness = slices[0].SliceThickness
    else:
        z_positions = [float(s.ImagePositionPatient[2]) for s in slices]
        slice_thickness = np.mean(np.diff(z_positions))

    return images, pixel_spacing, slice_thickness
```

Interpolación del volumen a un tamaño objetivo:

Redimensiona un volumen tridimensional a un tamaño fijo utilizando interpolación lineal.

```
def resize_volume(volume, pixel_spacing, slice_thickness, target_size=(128,
128, 128)):
    current_spacing = [pixel_spacing[0], pixel_spacing[1], slice_thickness]
```

```
    new_shape = target_size

    # Calcular los factores de escala para X, Y, Z
    scale_factors = [
        volume.shape[0] / new_shape[0],  # Factor de escala en X
        volume.shape[1] / new_shape[1],  # Factor de escala en Y
        volume.shape[2] / new_shape[2]   # Factor de escala en Z
    ]

    # Redimensionar el volumen utilizando la interpolación
    resized_volume = zoom(volume, (1/scale_factors[0], 1/scale_factors[1],
1/scale_factors[2]), order=1)

    return resized_volume
```

Procesamiento, interpolación y guardado de imágenes en formato PNG:

Procesa volúmenes DICOM de pacientes: los carga, interpola, normaliza a un rango de 0-255 y guarda los cortes resultantes como imágenes PNG.

```
def load_preprocess_and_adjust_images(patient_id,
base_path='D:/FrancoErcoli/dataset/', target_slices=128, img_size=(128,
128)):
    patient_folder = os.path.join(base_path, patient_id)

    # Cargar el volumen DICOM e interpolar
    volume, pixel_spacing, slice_thickness =
load_dicom_volume(patient_folder)
    interpolated_volume = resize_volume(volume, pixel_spacing,
slice_thickness, target_size=(128, 128, 128))

    # Normalizar el volumen (valores entre 0 y 255 para PNG)
    interpolated_volume = (interpolated_volume -
np.min(interpolated_volume)) / (np.max(interpolated_volume) -
np.min(interpolated_volume))
    interpolated_volume = (interpolated_volume * 255).astype(np.uint8)

    # Crear carpeta de destino para cada paciente
    output_folder = os.path.join(datos_listos_dir, patient_id)
    os.makedirs(output_folder, exist_ok=True)

    # Guardar cada slice como PNG
    for i in range(interpolated_volume.shape[2]):
```

```
        slice_img = interpolated_volume[:, :, i]
        output_path = os.path.join(output_folder, f'slice_{i:03d}.png')
        img = Image.fromarray(slice_img)
        img.save(output_path)

    print(f'Volumen de {patient_id} procesado y guardado en
{output_folder}')
```

Asignación de conjunto de datos (80%, 10%, 10%):

Distribuye aleatoriamente las muestras en tres conjuntos: 80% para entrenamiento, 10% para validación y 10% para prueba.

```
def asignar_conjunto():
    rand_num = random.random()
    if rand_num < 0.8:
        return 1  # 80% para el conjunto 1
    elif rand_num < 0.9:
        return 2  # 10% para el conjunto 2
    else:
        return 3  # 10% para el conjunto 3
```

Generación de particiones de datos:

Genera 10 archivos CSV que contienen las particiones aleatorias de datos para experimentos repetidos, respetando la proporción 80%-10%-10%.

```
for i in range(1, 11):
    archivo_csv = f'particion_{i}.csv'
    with open(archivo_csv, mode='w', newline='') as file:
        writer = csv.writer(file)
        writer.writerow(['id', 'conjunto'])  # Escribir las cabeceras
        for nombre in nombres_subcarpetas:
            conjunto = asignar_conjunto()
            writer.writerow([nombre, conjunto])
```

Carga de lotes de datos

Crea un lote de imágenes y sus correspondientes etiquetas (labels) a partir de un conjunto de índices proporcionados, que se utilizan para seleccionar un subconjunto específico de pacientes de un archivo CSV de etiquetas.

```
def cargar_lote(ruta_datos, batch_indices):
    imagenes = []
    labels = []

    # Leer el archivo CSV de labels
    labels_df = pd.read_csv('D:/FrancoErcoli/labels_balanceado.csv')

    # Obtener los IDs de pacientes para este lote
    pacientes_ids = labels_df['id'].iloc[batch_indices].tolist()
    contador = 0
    # Iterar sobre los IDs de pacientes en este lote
    for paciente_id in pacientes_ids:
        # Leer las imágenes de la carpeta del paciente
        paciente_path = os.path.join(ruta_datos, paciente_id)
        slices = []
        for filename in sorted(os.listdir(paciente_path)):
            if filename.endswith(".png"):
                img = cv2.imread(os.path.join(paciente_path, filename),
cv2.IMREAD_GRAYSCALE)
                if img is None:
                    print(f"Error al cargar la imagen
{os.path.join(paciente_path, filename)}")
                img = img.astype('float32') / 255.0  # Normaliza
                slices.append(img)
        if len(slices) > 0:
            slices = np.array(slices)
            imagenes.append(slices)
            # Obtener el label correspondiente y agregarlo
            label = labels_df.loc[labels_df['id'] == paciente_id,
'cancer'].values[0]
            labels.append(label)

    imagenes = np.array(imagenes)
    if imagenes.ndim == 4:  # Asegurar de que la dimensión sea (batch_size,
depth, height, width, channels)
        imagenes = np.expand_dims(imagenes, axis=-1)

    return imagenes, np.array(labels)
```

Generación de lotes de datos

Actúa como un generador que produce lotes de imágenes de manera indefinida para su uso en entrenamiento, validación o pruebas.

```
def generador_lotes(ruta_datos, indices, batch_size):
    np.random.shuffle(indices)
    while True:  # Bucle infinito para repetir el dataset
        for batch_indices in np.array_split(indices, len(indices) //
batch_size):
            yield cargar_lote(ruta_datos, batch_indices)
```

Carga de ID de acuerdo al conjunto a donde pertenezca (entrenamiento, validación, prueba)

Organiza los IDs de los pacientes en tres conjuntos distintos: entrenamiento, validación y prueba, basándose en las clases especificadas en un archivo CSV.

```
def cargar_ids_por_clase(archivo_csv):
    train_ids = []
    val_ids = []
    test_ids = []

    # Leer el archivo CSV
    with open(archivo_csv, mode='r') as file:
        reader = csv.DictReader(file)
        for row in reader:
            id_val = row['id']
            clase = int(row['conjunto'])
            # Clasificar los ids según su clase
            if clase == 1:
                train_ids.append(id_val)
            elif clase == 2:
                val_ids.append(id_val)
            elif clase == 3:
                test_ids.append(id_val)

    return train_ids, val_ids, test_ids
```

Obtención de indices por id

Encuentra los índices correspondientes a un conjunto de IDs en un archivo CSV de etiquetas.

```
def obtener_indices_por_id(archivo_labels, ids):
    indices = []

    # Leer el archivo labels
    with open(archivo_labels, mode='r') as file:
        reader = csv.reader(file)
        labels_list = [row[0] for row in reader]  # Asume que el archivo
labels tiene una columna con los ids

        # Buscar el índice de cada id en la lista labels_list
        for id_val in ids:
            if id_val in labels_list:
                indices.append(labels_list.index(id_val)-1)

    return indices
```

Carga de imagenes desde la carpeta

Carga todas las imágenes en formato PNG desde una carpeta específica, asegurando que se leen en el orden adecuado.

```
def load_images_from_folder(folder_path):
    # Obtener la lista de archivos en el directorio, filtrando solo los
archivos .png
    image_files = [f for f in os.listdir(folder_path) if
f.endswith('.png')]

    # Ordenar los archivos para asegurar que se lean en el orden correcto
    image_files.sort()

    # Inicializar una lista para almacenar las imágenes
    slices = []

    # Leer cada imagen y almacenarla en la lista
    for image_file in image_files:
        image_path = os.path.join(folder_path, image_file)
        image = cv2.imread(image_path, cv2.IMREAD_UNCHANGED)
        slices.append(image)

    # Convertir la lista de imágenes en un array de numpy
    slices = np.array(slices)
```

```
    return slices
```

Eliminación de datos aumentados de la carpeta de datos

Elimina las carpetas que contienen el sufijo "_aug", lo que se utiliza comúnmente para identificar carpetas de datos aumentados.

```
def remove_aug_folders(directory):
    # Obtener todas las carpetas en el directorio
    folders = [folder for folder in os.listdir(directory) if
os.path.isdir(os.path.join(directory, folder))]

    # Filtrar y eliminar carpetas que terminan con "_aug"
    for folder in folders:
        if folder.endswith('_aug'):
            folder_path = os.path.join(directory, folder)
            shutil.rmtree(folder_path)
            print(f'Carpeta eliminada: {folder_path}')

    # Comprobar la cantidad de carpetas restantes
    remaining_folders = [folder for folder in os.listdir(directory) if
os.path.isdir(os.path.join(directory, folder))]
    remaining_count = len(remaining_folders)

    if remaining_count == 1595:
        print("El proceso se completó correctamente. Quedan 1595
carpetas.")
    else:
        print(f"Proceso completado, pero la cantidad de carpetas restantes
es {remaining_count}, no 1595.")
```

Módulo de preprocesamiento

Data Augmentation

Aplica una serie de transformaciones de aumento de datos sobre un volumen 3D, como rotaciones con ángulos pequeños, zoom aleatorio, y ajustes de brillo, con el fin de generar versiones modificadas del volumen original para aumentar la diversidad del conjunto de datosy mejorar el entrenamiento del modelo.

```
def augment_volume(volume):
```

```
    # Funcion de Data Augmentation

    augmented_volumes = []
    volume_augmented = volume / 255.0

    # Rotación con ángulos aleatorios pequeños
    angle = random.uniform(-15, 15)
    volume_augmented = scipy.ndimage.rotate(volume_augmented, angle,
axes=(1, 2), reshape=False, mode='reflect')

    # Zoom aleatorio (factor entre 0% y 5%)
    if random.choice([True, False]):
        zoom_factor = random.uniform(1, 1.05)
        volume_augmented = scipy.ndimage.zoom(volume_augmented, (1,
zoom_factor, zoom_factor), order=1)
        # Recortar o pad para mantener el tamaño original
        if volume_augmented.shape[1:] != volume.shape[1:]:
            volume_augmented = volume_augmented[:, :volume.shape[1],
:volume.shape[2]]

    # Cambio de brillo aleatorio
    if random.choice([True, False]):
        brightness_factor = random.uniform(0.8, 1.2)
        volume_augmented = volume_augmented * brightness_factor
        # Clip para asegurar que los valores estén en el rango válido
        volume_augmented = np.clip(volume_augmented, 0, 1)

    augmented_volumes.append(volume_augmented)

    #_____________________________________________________________
    #_____________________________________________________________
    volume_augmented = volume / 255.0

    # Rotación con ángulos aleatorios pequeños
    angle = random.uniform(-15, 15)
    volume_augmented = scipy.ndimage.rotate(volume_augmented, angle,
axes=(1, 2), reshape=False, mode='reflect')

    # Zoom aleatorio (factor entre 1 y 1.05)
    if random.choice([True, False]):
        zoom_factor = random.uniform(1, 1.05)
        volume_augmented = scipy.ndimage.zoom(volume_augmented, (1,
zoom_factor, zoom_factor), order=1)
        # Recortar o pad para mantener el tamaño original
        if volume_augmented.shape[1:] != volume.shape[1:]:
```

```
            volume_augmented = volume_augmented[:, :volume.shape[1],
:volume.shape[2]]

    # Cambio de brillo aleatorio
    if random.choice([True, False]):
        brightness_factor = random.uniform(0.8, 1.2)
        volume_augmented = volume_augmented * brightness_factor
        # Clip para asegurar que los valores estén en el rango válido
        volume_augmented = np.clip(volume_augmented, 0, 1)

    augmented_volumes.append(volume_augmented)

    return augmented_volumes
```

Balanceo de datos

Maneja el proceso de balanceo de clases en un conjunto de datos, mediante la generación de datos aumentados de la clase minoritaria, utilizando la función de aumento de datos. Los nuevos datos aumentados se agregan a las etiquetas y los índices de entrenamiento para equilibrar las clases y mejorar la representación del dataset.

```
def balancear_datos(ruta_datos, train_indices, labels_df):
    # Lista para guardar los datos aumentados
    augmented_data = []
    labels_df_train = labels_df.loc[train_indices].copy()

    labels_df_majority = labels_df_train[labels_df_train['cancer'] == 0]
    labels_df_minority = labels_df_train[labels_df_train['cancer'] == 1]

    cantidad_a_balancear = len(labels_df_majority) -
len(labels_df_minority)

    # Aumentar la clase minoritaria
    print("Generando datos aumentados para la clase minoritaria...")
    for index, row in labels_df_minority.iterrows():
        patient_id = row['id']
        base_path = ruta_datos
        patient_folder = os.path.join(base_path, patient_id)

        volume = load_images_from_folder(patient_folder)

        augmented_volumes = augment_volume(volume)
```

```
        for i, aug_volume in enumerate(augmented_volumes):
            aug_patient_id = f'{patient_id}_{i}_aug'
            aug_patient_dir = os.path.join(ruta_datos, aug_patient_id)
            os.makedirs(aug_patient_dir, exist_ok=True)
            for j, slice_img in enumerate(aug_volume):
                filename = f'slice_{j:03d}.png'
                cv2.imwrite(os.path.join(aug_patient_dir, filename),
slice_img * 255)  # Guardar como PNG

            # Agregar el nuevo dato a labels_df
            new_row = {'id': aug_patient_id, 'cancer': 1}
            labels_df = labels_df.append(new_row, ignore_index=True)

            # Agregar el índice del nuevo dato a train_indices
            train_indices.append(labels_df.index[-1])

            # Detener cuando se alcance la cantidad necesaria
            if len(augmented_data) == cantidad_a_balancear:
                break
        if len(augmented_data) == cantidad_a_balancear:
            break

    print("Datos balanceados con éxito.")
    return labels_df, train_indices
```

Generación de nuevos labels para los nuevos datos

Crea un archivo CSV con las etiquetas balanceadas para un conjunto de datos, incluyendo tanto las imágenes originales como las generadas por aumento de datos. Genera una nueva entrada en el archivo CSV para cada imagen aumentada, asegurando que el conjunto de datosesté balanceado en cuanto a las clases.

```
def genero_labels(csv_path, ruta_datos, balanceado_csv_path):

    # Cargar el DataFrame original desde el archivo CSV
    labels_df = pd.read_csv(csv_path)

    # Crear una lista para almacenar los datos balanceados
    labels_balanceado = []

    # Agregar las imágenes originales al nuevo CSV
    print("Agregando las imágenes originales al nuevo archivo CSV...")
    for index, row in labels_df.iterrows():
        patient_id = row['id']
```

```
        label = row['cancer']
        labels_balanceado.append({'id': patient_id, 'cancer': label})
        #print(f"Agregado: {patient_id} con label {label}")

    # Agregar los nuevos labels para las imágenes aumentadas
    print("Generando nuevos labels para las imágenes aumentadas...")
    contador = 0
    for patient_folder in os.listdir(ruta_datos):
        if patient_folder.endswith('_aug'):
            contador = contador + 1
            labels_balanceado.append({'id': patient_folder, 'cancer': 1})
            print(f"Generado nuevo label para: {patient_folder}")

    if contador == 0:
        print("No hizo falta balancear el conjunto de datospor medio de
data augmentation.")

    # Crear un DataFrame con los datos balanceados
    labels_balanceado_df = pd.DataFrame(labels_balanceado)

    labels_balanceado_df.to_csv(balanceado_csv_path, index=False)
    print(f"Nuevo archivo CSV con labels balanceados guardado en:
{balanceado_csv_path}")
```

Ecualización del histograma

Aplica una técnica de ecualización de histograma sobre una imagen en escala de grises, lo que mejora el contraste de la imagen al redistribuir los valores de píxel a través de todo el rango disponible.

```
def ecualizar_histograma(img):
    # Aplicar ecualización de histograma a una imagen en escala de grises
    return cv2.equalizeHist(img)
```

Creación del nuevo directorio

Verifica si un directorio dado existe, y si no es así, lo crea, asegurando que el espacio de almacenamiento necesario esté disponible para guardar archivos o resultados.

```
def crear_directorio_si_no_existe(directorio):
    # Crear el directorio si no existe
    if not os.path.exists(directorio):
```

```
        os.makedirs(directorio)
```

Organización de las imagenes ecualizadas

Recorre un conjunto de subcarpetas que contienen imágenes, crea nuevas carpetas para guardar las imágenes procesadas, y aplica ecualización de histograma a cada imagen dentro de las subcarpetas, guardando los resultados en una nueva carpeta específica para estudios 3D ecualizados.

```
def procesar_estudio_3d(ruta_carpeta):
    # Crear nueva carpeta con sufijo "_ecu" para los resultados
    ruta_nueva_carpeta = ruta_carpeta + "_ecu"
    crear_directorio_si_no_existe(ruta_nueva_carpeta)

    # Obtener todas las subcarpetas en la ruta original
    subcarpetas = [os.path.join(ruta_carpeta, d) for d in
os.listdir(ruta_carpeta) if os.path.isdir(os.path.join(ruta_carpeta, d))]

    for subcarpeta in subcarpetas:
        nombre_subcarpeta = os.path.basename(subcarpeta)
        nueva_subcarpeta = os.path.join(ruta_nueva_carpeta,
nombre_subcarpeta)

        # Crear nueva subcarpeta en la carpeta "_ecu"
        crear_directorio_si_no_existe(nueva_subcarpeta)
        print(f"Procesando subcarpeta: {subcarpeta}")

        imagenes = sorted([f for f in os.listdir(subcarpeta) if
f.endswith('.png')])

        for imagen in imagenes:
            # Cargar cada imagen en escala de grises
            ruta_imagen = os.path.join(subcarpeta, imagen)
            img = cv2.imread(ruta_imagen, cv2.IMREAD_GRAYSCALE)

            if img is None:
                print(f"Error al cargar imagen: {ruta_imagen}")
                continue

            # Ecualizar el histograma de la imagen
            img_ecualizada = ecualizar_histograma(img)

            # Definir la ruta de guardado en la nueva carpeta
```

```
            ruta_nueva_imagen = os.path.join(nueva_subcarpeta, imagen)

            # Guardar la imagen ecualizada en la nueva subcarpeta
            cv2.imwrite(ruta_nueva_imagen, img_ecualizada)

        print(f"Estudio {subcarpeta} procesado con éxito. Resultados
guardados en: {nueva_subcarpeta}")
```

Módulo de configuración

Ajuste de memoria

Configura el límite de memoria que un GPU virtual puede usar en un entorno de TensorFlow. Establece un límite de memoria de 32 GB (32 * 1024 MB) para un GPU físico, y luego asigna ese límite a un GPU lógico específico. Si el entorno ya tiene GPUs disponibles, se ajusta la memoria que pueden utilizar. Si ocurre algún error durante este proceso, como intentar configurar un dispositivo después de que el GPU se haya inicializado, se imprime un mensaje de error.

```
def ajustar_memoria():
    # Limitar la memoria máxima que puede usar (por medio de un GPU virtual
asociado al físico)
    Limite_Memoria = 32 * 1024  # Lo ponemos en MB
    # Saco la lista de dispositivos:
    gpus = tf.config.list_physical_devices('GPU')
    if gpus:
        # Restringimos el espacio en memoria que puede utilizar:
        try:
                        tf.config.set_logical_device_configuration(gpus[0],
[tf.config.LogicalDeviceConfiguration(memory_limit=Limite_Memoria)])
            logical_gpus = tf.config.list_logical_devices('GPU')
             print(len(gpus), "Physical GPUs,", len(logical_gpus), "Logical
GPUs")
        except RuntimeError as e:
                # Si ocurrió un error (en particular porque se creo el
dispositivo
            # virtual luego de inicializar el GPU) lo indico:
            print(e)
```

Módulo de visualización

Lectura de la serie DICOM

Lee todos los archivos DICOM en una carpeta dada, ordenándolos en base a su posición en el eje Z (profundidad) para preservar la coherencia espacial del volumen. Extrae los datos de píxeles de cada slice y los apila en una única matriz tridimensional representando el volumen completo.

```
def read_dicom_series(folder_path):
    dicom_files = [pydicom.dcmread(os.path.join(folder_path, filename)) for
filename in os.listdir(folder_path)]
    dicom_files.sort(key=lambda x: float(x.ImagePositionPatient[2]))
    slices = [dicom_file.pixel_array for dicom_file in dicom_files]
    return np.stack(slices)
```

Proyección de intensidad máxima (MIP)

Genera una proyección de intensidad máxima del volumen a lo largo de un eje especificado ('x', 'y' o 'z'). Esta técnica destaca las estructuras de mayor intensidad, facilitando la visualización de regiones densas como huesos o nódulos en las imágenes médicas.

```
def maximum_intensity_projection(volume, axis):
    if axis not in ['x', 'y', 'z']:
        raise ValueError("Axis must be 'x', 'y', or 'z'")

    if axis == 'x':
        mip = np.max(volume, axis=0)
    elif axis == 'y':
        mip = np.max(volume, axis=1)
    else:  # axis == 'z'
        mip = np.max(volume, axis=2)

    return mip
```

Visualización 3D del volumen

Integra el proceso completo de lectura, normalización y visualización de un volumen DICOM. Calcula las proyecciones MIP en los tres ejes principales y utiliza la librería Mayavi para renderizar tanto el volumen completo en 3D como sus proyecciones. Esto permite al usuario explorar visualmente la anatomía contenida en el estudio de manera intuitiva y detallada.

```
def see_3d_volume(dicom_folder):
    # Ruta al directorio que contiene las imágenes en formato DICOM
    dicom_volume = read_dicom_series(dicom_folder)
```

```
    # Normalizar el volumen (opcional)
    dicom_volume = (dicom_volume - np.min(dicom_volume)) /
(np.max(dicom_volume) - np.min(dicom_volume))

    # Calcular las proyecciones de intensidad máxima en los ejes x, y, y z
    mip_x = maximum_intensity_projection(dicom_volume, 'x')
    mip_y = maximum_intensity_projection(dicom_volume, 'y')
    mip_z = maximum_intensity_projection(dicom_volume, 'z')

    # Crear un plot 3D con Mayavi
    mlab.figure(bgcolor=(1, 1, 1))
    mlab.contour3d(dicom_volume, colormap='bone', opacity=0.5)
    mlab.contour_surf(mip_x, colormap='gray', opacity=0.5)
    mlab.contour_surf(mip_y, colormap='gray', opacity=0.5)
    mlab.contour_surf(mip_z, colormap='gray', opacity=0.5)
    mlab.show()
```

Módulo de modelos

Definicion de las arquitecturas

```
def definir_modelo_01(input_shape):
    print("Definiendo modelo...")
    model = Sequential()

    model.add(Conv3D(32, kernel_size=(3, 3, 3), activation='relu',
input_shape=input_shape))
    model.add(BatchNormalization())
    model.add(MaxPooling3D(pool_size=(2, 2, 2)))
    model.add(Dropout(0.3))

    model.add(Conv3D(64, kernel_size=(3, 3, 3), activation='relu'))
    model.add(BatchNormalization())
    model.add(MaxPooling3D(pool_size=(2, 2, 2)))
    model.add(Dropout(0.3))

    model.add(Conv3D(128, kernel_size=(3, 3, 3), activation='relu'))
    model.add(BatchNormalization())
    model.add(MaxPooling3D(pool_size=(2, 2, 2)))
    model.add(Dropout(0.3))

    model.add(Flatten())
    model.add(Dense(64, activation='relu'))
```

```
    model.add(Dropout(0.3))

    model.add(Dense(32, activation='relu'))
    model.add(Dropout(0.5))

    model.add(Dense(1, activation='sigmoid'))

    print("Modelo definido.")
    return model


def definir_modelo_02(input_shape):
    print("Definiendo modelo...")
    model = Sequential()

    model.add(Conv3D(32, kernel_size=(3, 3, 3), activation='relu',
input_shape=input_shape))
    model.add(BatchNormalization())
    model.add(MaxPooling3D(pool_size=(2, 2, 2)))
    model.add(Dropout(0.3))

    model.add(Conv3D(64, kernel_size=(3, 3, 3), activation='relu'))
    model.add(BatchNormalization())
    model.add(MaxPooling3D(pool_size=(2, 2, 2)))
    model.add(Dropout(0.3))

    model.add(Conv3D(128, kernel_size=(3, 3, 3), activation='relu'))
    model.add(BatchNormalization())
    model.add(MaxPooling3D(pool_size=(2, 2, 2)))
    model.add(Dropout(0.3))

    model.add(Flatten())
    model.add(Dense(256, activation='relu'))
    model.add(Dropout(0.3))

    model.add(Dense(128, activation='relu'))
    model.add(Dropout(0.5))

    model.add(Dense(1, activation='sigmoid'))

    print("Modelo definido.")
    return model


def definir_modelo_03(input_shape):
```

```
    print("Definiendo modelo...")
    model = Sequential()

    model.add(Conv3D(32, kernel_size=(3, 3, 3), activation='relu',
input_shape=input_shape))
    model.add(BatchNormalization())
    model.add(MaxPooling3D(pool_size=(2, 2, 2)))
    model.add(Dropout(0.3))

    model.add(Conv3D(64, kernel_size=(3, 3, 3), activation='relu'))
    model.add(BatchNormalization())
    model.add(MaxPooling3D(pool_size=(2, 2, 2)))
    model.add(Dropout(0.3))

    model.add(Conv3D(128, kernel_size=(3, 3, 3), activation='relu'))
    model.add(BatchNormalization())
    model.add(MaxPooling3D(pool_size=(2, 2, 2)))
    model.add(Dropout(0.3))

    model.add(Conv3D(256, kernel_size=(3, 3, 3), activation='relu'))
    model.add(BatchNormalization())
    model.add(MaxPooling3D(pool_size=(2, 2, 2)))
    model.add(Dropout(0.3))

    model.add(Flatten())
    model.add(Dense(512, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(1, activation='sigmoid'))

    print("Modelo definido.")
    return model

def definir_modelo_transfer_learning_04(input_shape):
    print("Definiendo modelo...")

    # Cargar el modelo preentrenado (formato .h5)
    model_path = 'D:/FrancoErcoli/TOOLBOX/3D_CNN_toolbox/brain_tumor_3d.h5'
    base_model = load_model(model_path)

    # Congelar capas del modelo base para evitar que se reentrenen
    for layer in base_model.layers:
        layer.trainable = False

    for layer in ['conv3d_5', 'batch_normalization_5', 'dropout_3',
'global_average_pooling3d']:
```

```
        base_model.get_layer(layer).trainable = True

    layer_name = 'global_average_pooling3d'  # Reemplazar con el nombre de
la capa deseada
    intermediate_layer = base_model.get_layer(layer_name).output

    # Crear un nuevo modelo a partir de la entrada original y la salida de
la capa intermedia
    model = Model(inputs=base_model.input, outputs=intermediate_layer)

    # Definir la entrada del nuevo modelo
    inputs = base_model.input

    # Añadir nuevas capas con nombres únicos
    #x = base_model.output
    x = base_model.get_layer(layer_name).output
    x = Dense(64, activation='relu', name='new_dense_64')(x)
    x = Dropout(0.04, name='new_dropout_3')(x)
    x = Dense(32, activation='relu', name='new_dense_32')(x)
    x = Dropout(0.08, name='new_dropout_4')(x)
    outputs = Dense(1, activation='sigmoid', name='new_output_sigmoid')(x)

    # Crear el nuevo modelo
    modelo = Model(inputs=inputs, outputs=outputs)
    model = modelo
    print("Modelo definido.")
    return model

def definir_modelo_transfer_learning_05(input_shape):
    print("Definiendo modelo...")

    # Cargar el modelo preentrenado (formato .h5)
    model_path = 'D:/FrancoErcoli/TOOLBOX/3D_CNN_toolbox/brain_tumor_3d.h5'
    base_model = load_model(model_path)

    # Congelar capas del modelo base para evitar que se reentrenen
    for layer in base_model.layers:
        layer.trainable = False

    for layer in ['conv3d_5', 'batch_normalization_5', 'dropout_3',
'global_average_pooling3d']:
        base_model.get_layer(layer).trainable = True

    layer_name = 'global_average_pooling3d'  # Reemplazar con el nombre de
la capa deseada
```

```
    intermediate_layer = base_model.get_layer(layer_name).output

    # Crear un nuevo modelo a partir de la entrada original y la salida de
la capa intermedia
    model = Model(inputs=base_model.input, outputs=intermediate_layer)

    # Definir la entrada del nuevo modelo
    inputs = base_model.input

    # Añadir nuevas capas con nombres únicos
    #x = base_model.output
    x = base_model.get_layer(layer_name).output
    x = Dense(256, activation='relu', name='new_dense_64')(x)
    x = Dropout(0.04, name='new_dropout_3')(x)
    x = Dense(128, activation='relu', name='new_dense_32')(x)
    x = Dropout(0.04, name='new_dropout_4')(x)
    x = Dense(64, activation='relu', name='new_dense_32')(x)
    x = Dropout(0.08, name='new_dropout_4')(x)
    outputs = Dense(1, activation='sigmoid', name='new_output_sigmoid')(x)

    # Crear el nuevo modelo
    modelo = Model(inputs=inputs, outputs=outputs)
    model = modelo
    print("Modelo definido.")
    return model

def definir_modelo_transfer_learning_06(input_shape):
    print("Definiendo modelo...")

    # Cargar el modelo preentrenado (formato .h5)
    model_path = 'D:/FrancoErcoli/TOOLBOX/3D_CNN_toolbox/brain_tumor_3d.h5'
    base_model = load_model(model_path)

    # Congelar capas del modelo base para evitar que se reentrenen
    for layer in base_model.layers:
        layer.trainable = False

    # Descongelar solo las capas necesarias por nombre
    layers_to_train = [
        'conv3d_4', 'batch_normalization_4', 'dropout_2',
        'conv3d_5', 'batch_normalization_5', 'dropout_3',
        'global_average_pooling3d'
    ]

    for layer_name in layers_to_train:
```

```
        base_model.get_layer(layer_name).trainable = True


    layer_name = 'global_average_pooling3d'  # Reemplazar con el nombre de
la capa deseada
    intermediate_layer = base_model.get_layer(layer_name).output

    # Crear un nuevo modelo a partir de la entrada original y la salida de
la capa intermedia
    model = Model(inputs=base_model.input, outputs=intermediate_layer)

    # Definir la entrada del nuevo modelo
    inputs = base_model.input

    # Añadir nuevas capas con nombres únicos
    #x = base_model.output
    x = base_model.get_layer(layer_name).output
    x = Dense(256, activation='relu', name='new_dense_64')(x)
    x = Dropout(0.04, name='new_dropout_3')(x)
    x = Dense(128, activation='relu', name='new_dense_32')(x)
    x = Dropout(0.04, name='new_dropout_4')(x)
    x = Dense(64, activation='relu', name='new_dense_32')(x)
    x = Dropout(0.08, name='new_dropout_4')(x)
    outputs = Dense(1, activation='sigmoid', name='new_output_sigmoid')(x)

    # Crear el nuevo modelo
    modelo = Model(inputs=inputs, outputs=outputs)
    model = modelo
    print("Modelo definido.")
    return model
```

Módulo de entrenamiento

Entrenamiento del modelo

Entrena un modelo de redes neuronales utilizando un generador de datos de entrenamiento y validación. Primero, compila el modelo con el optimizador 'adam' y la función de pérdida 'binary_crossentropy', seguida de la métrica de exactitud ('accuracy'). Define dos callbacks: uno para guardar el mejor modelo basado en la mínima pérdida de validación (ModelCheckpoint) y otro para detener el entrenamiento temprano si no mejora la pérdida de validación durante 20 épocas consecutivas (EarlyStopping). Luego, entrena el modelo durante el número de épocas especificado, utilizando los generadores para los datos de

entrenamiento y validación, y devuelve el historial del entrenamiento, que contiene las métricas de rendimiento.

```
def entrenar_modelo(modelo, generador_train, generador_val,
steps_per_epoch, validation_steps, epochs, nombre):
    print("Entrenando modelo...")
    modelo.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])

    # Definir callbacks para early stopping y guardar el mejor modelo
    checkpoint = ModelCheckpoint(nombre, monitor='val_loss', verbose=1,
save_best_only=True, mode='min')
    early_stopping = EarlyStopping(monitor='val_loss', patience=20,
verbose=1, restore_best_weights=True)

    # Entrenar el modelo
    historia = modelo.fit(
        generador_train,
        steps_per_epoch=steps_per_epoch,
        epochs=epochs,
        validation_data=generador_val,
        validation_steps=validation_steps,
        callbacks=[checkpoint, early_stopping]
    )

    print("Modelo entrenado.")
    return historia
```

Evaluación del modelo

Carga un modelo previamente entrenado y lo evalúa utilizando un conjunto de datos de prueba. Para cada paciente en el conjunto de prueba, reconstruye el volumen 3D a partir de las imágenes individuales (slices) en escala de grises, normaliza los datos y los agrupa en un arreglo listo para la predicción. Luego, genera predicciones binarias utilizando un umbral de 0.5, calcula la métrica de exactitud y guarda los resultados en un archivo CSV que incluye los valores reales, predichos y las probabilidades asociadas para cada muestra.

```
def evaluar_modelo(model_path, test_indices, test_labels, ruta_datos,
output_csv):
    modelo = load_model(model_path)
    X = []
    y = []

    size = (128, 128, 64)
```

```
    for i, paciente_id in enumerate(test_indices):
        carpeta_path = os.path.join(ruta_datos, paciente_id)
        slices = sorted(os.listdir(carpeta_path))
        volumen = []

        for nombre_imagen in slices:
            ruta_imagen = os.path.join(carpeta_path, nombre_imagen)
            imagen = load_img(ruta_imagen, color_mode='grayscale',
target_size=size[:2])
            imagen_array = img_to_array(imagen).squeeze()
            volumen.append(imagen_array)

        volumen = np.stack(volumen, axis=-1)  # [128,128,64]
        volumen = volumen.astype('float32') / 255.0
        X.append(volumen)
        y.append(test_labels[i])

    X = np.array(X)
    X = np.expand_dims(X, axis=-1)  # [N,128,128,64,1]
    y_true = np.array(y)

    y_pred_prob = modelo.predict(X)
    y_pred = (y_pred_prob > 0.5).astype(int).flatten()

    acc = accuracy_score(y_true, y_pred)
    print(f"Accuracy: {acc:.4f}")

    resultados = pd.DataFrame({
        'id': test_indices,
        'y_true': y_true,
        'y_pred': y_pred,
        'y_pred_prob': y_pred_prob.flatten()
    })

    resultados.to_csv(output_csv, index=False)
    print(f"Resultados guardados en: {output_csv}")

    return acc
```