FACULTAD DE INGENIERÍA Y ARQUITECTURA

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

Deep Learning para identificación automática de tumores cerebrales en imágenes médicas

TESIS PARA OBTENER EL TÍTULO PROFESIONAL DE:

Ingeniero de Sistemas

AUTOR:

Alegre Milla, Fredy Alfredo (orcid.org/0000-0002-1801-6022)

ASESOR:

Dr. Daza Vergaray, Alfredo (orcid.org/0000-0002-2259-1070)

LÍNEA DE INVESTIGACIÓN:

Sistemas de Información y Comunicaciones

LÍNEA DE RESPONSABILIDAD SOCIAL UNIVERSITARIA:

Desarrollo económico, empleo y emprendimiento

LIMA – PERÚ

2024

Declaratoria de autenticidad del asesor

UNIVERSIDAD CÉSAR VALLEJO

FACULTAD DE INGENIERÍA Y ARQUITECTURA
ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

Declaratoria de Autenticidad del Asesor

Yo, DAZA VERGARAY ALFREDO, docente de la FACULTAD DE INGENIERÍA Y ARQUITECTURA de la escuela profesional de INGENIERÍA DE SISTEMAS de la UNIVERSIDAD CÉSAR VALLEJO SAC - LIMA NORTE, asesor de Tesis titulada: "Deep Learning para Identificación Automática de Tumores Cerebrales en Imágenes Médicas.", cuyo autor es ALEGRE MILLA FREDY ALFREDO, constato que la investigación tiene un índice de similitud de 10%, verificable en el reporte de originalidad del programa Turnitin, el cual ha sido realizado sin filtros, ni exclusiones.

He revisado dicho reporte y concluyo que cada una de las coincidencias detectadas no constituyen plagio. A mi leal saber y entender la Tesis cumple con todas las normas para el uso de citas y referencias establecidas por la Universidad César Vallejo.

En tal sentido, asumo la responsabilidad que corresponda ante cualquier falsedad, ocultamiento u omisión tanto de los documentos como de información aportada, por lo cual me someto a lo dispuesto en las normas académicas vigentes de la Universidad César Vallejo.

LIMA, 21 de Diciembre del 2024

Apellidos y Nombres del Asesor:	**Firma**
DAZA VERGARAY ALFREDO **DNI:** 40466240 **ORCID:** 0000-0002-2259-1070	Firmado electrónicamente por: ADAZAVE el 21-12-2024 16:10:35

Código documento Trilce: TRI - 0970026

INVESTIGA UCV

Declaratoria de originalidad del autor(es)

UNIVERSIDAD CÉSAR VALLEJO

FACULTAD DE INGENIERÍA Y ARQUITECTURA
ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

Declaratoria de Originalidad del Autor

Yo, ALEGRE MILLA FREDY ALFREDO estudiante de la FACULTAD DE INGENIERÍA Y ARQUITECTURA de la escuela profesional de INGENIERÍA DE SISTEMAS de la UNIVERSIDAD CÉSAR VALLEJO SAC - LIMA NORTE, declaro bajo juramento que todos los datos e información que acompañan la Tesis titulada: "Deep Learning para Identificación Automática de Tumores Cerebrales en Imágenes Médicas.", es de mi autorí a, por lo tanto, declaro que la Tesis:

1. No ha sido plagiada ni total, ni parcialmente.
2. He mencionado todas las fuentes empleadas, identificando correctamente toda cita textual o de paráfrasis proveniente de otras fuentes.
3. No ha sido publicada, ni presentada anteriormente para la obtención de otro grado académico o título profesional.
4. Los datos presentados en los resultados no han sido falseados, ni duplicados, ni copiados.

En tal sentido asumo la responsabilidad que corresponda ante cualquier falsedad, ocultamiento u omisión tanto de los documentos como de la información aportada, por lo cual me someto a lo dispuesto en las normas académicas vigentes de la Universidad César Vallejo.

Nombres y Apellidos	**Firma**
FREDY ALFREDO ALEGRE MILLA **DNI:** 70165575 **ORCID:** 0000-0002-1801-6022	Firmado electrónicamente por: FALEGREMI el 21-12-2024 21:55:03

Código documento Trilce: TRI - 0970025

INVESTIGA UCV

Dedicatoria

A mis familiares, quienes siempre han creído en mí y me han brindado su apoyo incondicional en cada paso de este camino. Agradezco profundamente su paciencia, comprensión y amor inquebrantable, que me han dado la fortaleza para superar los desafíos y lograr mis metas. Esta tesis es también para ustedes, por estar siempre a mi lado cuando más lo necesitaba.

Alegre Milla Fredy Alfredo

Agradecimiento

Deseo expresar mi más sincera gratitud a mi asesor, el Dr. Alfredo Daza Vergaray, por su valiosa guía y apoyo constante durante cada etapa de esta tesis. Su dedicación, enseñanzas y consejos han sido fundamentales para el desarrollo de este trabajo.

Gracias a estos recursos, he podido llevar a cabo esta investigación con la profundidad y rigor que se requería.

Alegre Milla Fredy Alfredo

Índice de contenidos

Declaratoria de autenticidad del asesor ii
Declaratoria de originalidad del autores iii
Dedicatoria iv
Agradecimiento v
Índice de contenidos vi
Índice de tablas vii
Índice de figuras viii
Resumen ix
Abstract x
I. INTRODUCCIÓN 1
II. METODOLOGÍA 32
III. RESULTADOS 37
IV. DISCUSIÓN 54
V. CONCLUSIONES 58
VI. RECOMENDACIONES 60
REFERENCIAS 62
ANEXOS 67

Índice de tablas

Tabla 1: Comparativo entre Algoritmos 30

Tabla 2: Resultados de la métrica de Precisión 37

Tabla 3: Prueba ANOVA de Precisión..... 39

Tabla 4: Prueba Tukey para la métrica de Precisión..... 39

Tabla 5: Resultados de la métrica de Sensibilidad..... 41

Tabla 6: Prueba ANOVA de Sensibilidad 43

Tabla 7: Prueba Tukey para la métrica de Sensibilidad 43

Tabla 8: Resultados de la métrica de especificidad 45

Tabla 9: Prueba ANOVA de especificidad..... 47

Tabla 10:Prueba Tukey para la métrica de especificidad..... 47

Tabla 11: Resultados de la métrica de F1 Score..... 49

Tabla 12: Prueba ANOVA de F1 Score 51

Tabla 13: Prueba Tukey para la métrica de F1 Score 51

Tabla 14: Operacionalización de variables..... 67

Tabla 15: Tabla de recolección de datos con CNN 68

Tabla 16: Tabla de recolección de datos con MobileNetv2 69

Tabla 17: Tabla de recolección de datos con VGG16 70

Tabla 18: Tabla de recolección de datos con RestNet50 71

Tabla 19:Tabla de recolección de datos con DenseNet121 72

Índice de figuras

Figura 1: Arquitectura de CNN 26
Figura 2: Arquitectura de VGG16 27
Figura 3: Arquitectura de MobileNetV2 27
Figura 4: Arquitectura de ResNet50 28
Figura 5: Modelo de DenseNet121 28
Figura 6: Arquitectura de U-Net 30
Figura 8: Metodología CRISP-DM 35
Figura 8: Resultados de la métrica de Precisión 38
Figura 9: Resultados de la métrica de Sensibilidad 42
Figura 10: Resultados de la métrica de especificidad 46
Figura 11: Resultados de la métrica de F1 Core 50
Figura 14: Dataset validación de Imágenes – Tumor Gliomas 74
Figura 15: Dataset validación de Imágenes – Tumor Meningiomas 75
Figura 16: Dataset validación de Imágenes – No Tumor 75
Figura 17: Dataset validación de Imágenes –Pituitary 76
Figura 18: UI de predicción del modelo. 76
Figura 19: Almacenamiento de predicciones en MySQL. 77
Figura 20: Generación de Boundingbox 86
Figura 21: Clasificación del tumor y exactitud erróneo 87
Figura 22: Clasificación del tumor y exactitud correcto 88
Figura 23: Detección de Anomalía con U-Net 90
Figura 24: Predicción del tamaño del tumor 91

Resumen

La investigación aborda la detección automática de tumores cerebrales mediante aprendizaje profundo en imágenes médicas, destacando la necesidad de métodos diagnósticos que incrementen la precisión y eficiencia. El objetivo principal es evaluar si el aprendizaje profundo puede igualar o superar la precisión de los sistemas tradicionales utilizados por radiólogos. Adoptando un enfoque cuantitativo explicativo, se llevó a cabo un experimento para comparar resultados antes y después del uso del sistema inteligente. Los análisis de precisión, sensibilidad, especificidad y F1 Score evidencian que los sistemas de Deep Learning ofrecen mayor exactitud, reducen la carga laboral y mejoran la calidad diagnóstica, superando los métodos convencionales.

Palabras clave: Deep learning, tumores cerebrales, Imágenes médicas.

Abstract

This research explores automatic brain tumor detection using deep learning applied to medical images, addressing the need for diagnostic methods that enhance precision and efficiency. The main objective is to evaluate whether deep learning achieves accuracy equal to or exceeding that of traditional systems used by radiologists. A quantitative and explanatory approach was adopted, comparing results before and after the implementation of an intelligent system. Analyses of precision, sensitivity, specificity, and F1 Score indicate that deep learning systems surpass conventional methods in accuracy, reduce workload, and improve diagnostic quality.

Keywords: Deep Learning, Brain tumors, Medical imaging.

I. INTRODUCCIÓN

Actualmente, un tumor cerebral es una acumulación anormal de células que se desarrollan dentro o alrededor del tejido cerebral, que puede ser benigno o maligno. La presencia de tumores cerebrales primarios, que son tumores que proliferan directamente en el cerebro, plantea un desafío importante en términos de diagnóstico y tratamiento. Las células del cerebro sufren mutaciones que provocan un crecimiento tumoral descontrolado. Los gliomas, que son tumores malignos que se dirigen a las células gliales, constituyen aproximadamente el 75 por ciento de todos los tumores malignos del sistema nervioso central. la forma más agresiva, el glioblastoma, tiene una tasa de supervivencia a cinco años inferior a 10, lo que indica una alta letalidad. (American Cancer Society, 2023).

La tasa de mortalidad de los tumores cerebrales en todo el mundo es muy variable y está influenciada por la ubicación y el tipo del tumor. Las tasas de supervivencia para las neoplasias malignas del cerebro y del sistema nervioso suelen ser de alrededor de 36% durante cinco años, pero pueden mejorar en pacientes menores de 40 años, y algunos casos más jóvenes llegan a los 70%. Esfuércese agresivamente. Los meningiomas, aunque son principalmente benignos y tienen una supervivencia del 85% a cinco años, pueden ejercer presión en áreas críticas del cerebro, causando síntomas debilitantes y requiriendo, en ocasiones, intervención quirúrgica. En comparación, de cada 1,000 personas diagnosticadas con tumores intracraneales, aproximadamente 370 podrían tener meningiomas, mientras que los gliomas y otros tumores malignos presentan una mortalidad mucho mayor, con tasas de mortalidad que pueden llegar a 4.4 por cada 100,000 personas (Cancer Research UK, 2024; SEER, 2024).

Los tumores cerebrales representan una problemática importante en el ámbito de la salud global debido a su profundo efecto en la calidad de vida de los pacientes y las múltiples dificultades relacionadas con su manejo médico. Entre ellos, los gliomas, que se originan en las células gliales del sistema nervioso central, constituyen aproximadamente el 75% de los tumores malignos en esta área. En particular, el glioblastoma destaca por ser el tipo de glioma más frecuente, caracterizándose por su alta agresividad y letalidad. Este tumor tiene una incidencia global estimada de 6 casos por cada 100,000 personas y afecta de

manera desproporcionada a los hombres. Además, la tasa de supervivencia a cinco años rara vez supera el 10%. (Cancer Research UK, 2024, pág. 19; Jividen, 2022).

Los meningiomas son el tipo más frecuente entre los tumores cerebrales primarios benignos, representando aproximadamente el 37% de los casos. Aunque su crecimiento suele ser lento y no suelen clasificarse como malignos, su localización puede ocasionar complicaciones neurológicas relevantes debido a la presión que ejercen sobre el tejido cerebral circundante. Este tipo de tumor presenta una incidencia de 8.7 casos por cada 100,000 habitantes y es más común en mujeres, posiblemente influenciado por factores hormonales. Desde una perspectiva demográfica, se estima que, de cada 1,000 diagnósticos de tumores cerebrales, alrededor de 370 corresponden a meningiomas. Este dato resalta tanto su alta prevalencia como la importancia de profundizar en su tratamiento y manejo clínico (American Cancer Society, 2023, pág. 17; Cancer Research UK, 2024).

En lo que se refiere, a los tumores pituitarios constituyen entre el 10 y el 15% de todos los tumores intracraneales y, aunque en su mayoría son benignos, pueden desencadenar graves complicaciones endocrinas y neurológicas debido a su efecto sobre la regulación hormonal y su cercanía al quiasma óptico. Su incidencia global se sitúa entre 2 y 3 casos por cada 100,000 personas, lo cual refleja una prevalencia menor en comparación con otros tipos de tumores cerebrales. Sin embargo, estos tumores destacan por su complejidad clínica, ya que afectan el equilibrio hormonal y pueden provocar una variedad de síntomas, incluyendo problemas visuales y disfunciones endocrinas que pueden comprometer seriamente la salud del paciente. En un grupo de 1,000 personas diagnosticadas con tumores intracraneales, se espera que entre 100 y 150 tengan tumores pituitarios, subrayando la importancia de un diagnóstico temprano y un abordaje terapéutico adecuado (Yale Medicine, 2022; Cancer Research UK, 2024, pág. 22)

A escala global, los tumores cerebrales y del sistema nervioso central constituyen una de las principales causas de mortalidad dentro del grupo de enfermedades oncológicas. Según estimaciones recientes proporcionadas por SEER en Estados Unidos y otras fuentes internacionales, la tasa de mortalidad

asociada a tumores malignos en estas áreas es de 4.4 por cada 100,000 personas al año. Esta situación posiciona a estos tumores entre las neoplasias de mayor letalidad, particularmente en adultos mayores, quienes enfrentan un mayor riesgo de fallecimiento debido a estas afecciones (SEER, 2024; Cancer Research UK, 2022).

La supervivencia a cinco años en personas diagnosticadas con tumores malignos en el cerebro y el sistema nervioso central es relativamente baja, con un promedio general cercano al 33.4%. No obstante, esta cifra varía considerablemente según la edad y el tipo de tumor. En cuanto a la incidencia y mortalidad, se ha observado que la tasa es más alta en hombres que en mujeres y que aumenta de manera significativa en personas de edad avanzada. De hecho, más del 34% de las muertes relacionadas con estos tumores ocurre en mayores de 75 años, según algunos estudios recientes (American Cancer Society, 2023; Cancer Research UK, 2024, pág. 19).

La atención médica para pacientes con cáncer enfrenta retos particulares tanto en el sector público como en el privado, los cuales abordan esta problemática con enfoques y recursos diferenciados. En el ámbito público, los sistemas de salud suelen estar saturados y presentan una oferta limitada de medicamentos oncológicos, especialmente en países con ingresos bajos y medianos, donde apenas el 43% de los tratamientos está disponible en hospitales públicos, en contraste con el 71% que se encuentra en el sector privado. Esta situación refleja que muchos pacientes dependen del sector privado para acceder a medicamentos esenciales, lo que incrementa significativamente los costos debido a los precios más elevados en los servicios privados (Frontiers, 2023).

Además, los sistemas públicos enfrentan restricciones presupuestarias y problemas de suministro, afectando la capacidad para mantener un acceso constante a medicamentos oncológicos innovadores y de alto costo. Esta situación genera una brecha de acceso, donde los tratamientos avanzados tienden a estar disponibles principalmente en el sector privado o en alianzas público-privadas, que intentan mejorar la infraestructura oncológica y facilitar la disponibilidad de tratamientos mediante colaboraciones entre gobiernos y empresas privadas (City Cancer Challenge, 2022; PLOS Medicine, 2023).

El gobierno juega un papel fundamental en la regulación y financiación del sector público, además de impulsar alianzas público-privadas que buscan ampliar el acceso y reducir los costos asociados con los tratamientos oncológicos. No obstante, en numerosos países con ingresos bajos y medianos, estas iniciativas siguen siendo limitadas, y la cobertura de salud pública resulta insuficiente para atender de manera adecuada a los pacientes con cáncer. Esta situación incrementa la dependencia del sector privado, que con frecuencia es inaccesible para las poblaciones con bajos recursos, exacerbando así las desigualdades en el acceso al tratamiento del cáncer (PLOS Medicine, 2023; Cancer Research UK, 2024).

Asif et al. (2022) dicen que las mejores formas de saber si tiene un tumor cerebral son utilizar una máquina que tome fotografías de su cabeza (MRI) o una máquina que tome radiografías de su cabeza (CT). Pero las imágenes por resonancia magnética son mejores que las tomografías computarizadas porque muestran más detalles sobre la forma y textura del tumor. La resonancia magnética es la mejor manera de ver lo que sucede en el cerebro, porque no duele y proporciona imágenes claras y detalladas, a diferencia de otras formas de observar el cerebro. Esta información de los autores muestra cómo la resonancia magnética es una forma buena y segura de detectar tumores cerebrales.

Karim, S. et al. (2024) señalan que los tumores cerebrales representan una de las principales causas de mortalidad por cáncer a nivel mundial, destacando su complejidad para el tratamiento debido a su ubicación en áreas críticas del cerebro. La detección temprana de estos tumores resulta esencial para asegurar un tratamiento efectivo que permita mejorar la calidad de vida del paciente. Sin embargo, la realización de pruebas diagnósticas, como la resonancia magnética y la tomografía computarizada, representa un desafío debido a la precisión y habilidades técnicas que requieren, lo que limita la capacidad de los médicos y otros profesionales de salud para llevar a cabo estas evaluaciones de manera rápida y eficiente.

En este contexto, Ahmad et al. (2020) sugieren la implementación de técnicas de deep learning, específicamente utilizando modelos de redes neuronales convolucionales preentrenados, como Inception v3 y DenseNet201. Estas

herramientas buscan automatizar y optimizar el diagnóstico de tumores cerebrales a partir de imágenes de resonancia magnética (MRI). Dichos enfoques presentan un gran potencial para enfrentar los desafíos derivados de la escasez de radiólogos con experiencia y la variabilidad en la interpretación de las imágenes, mejorando así la precisión y consistencia del diagnóstico.

Por consiguiente, esta investigación se enfocará en la creación de un sistema inteligente basado en técnicas de aprendizaje profundo, empleando un método de selección de variables para el análisis de imágenes de resonancia magnética. El objetivo principal es facilitar un diagnóstico eficiente y preciso de tumores cerebrales.

Para ello se generó la siguiente pregunta de investigación como respuesta al problema prevalente: ¿Cómo utilizar el aprendizaje profundo para mejorar la eficiencia del diagnóstico de tumores cerebrales? Como preguntas específicas del problema, primero: ¿Cómo mejorar la interpretación de imágenes de tumores cerebrales mediante algoritmos de Deep Learning en términos de precisión? Segundo: ¿Cómo mejorar la sensibilidad en la interpretación de imágenes de tumores cerebrales mediante algoritmos de Deep Learning? Tercero: ¿Cómo puede el Deep Learning abordar la escasez de radiólogos especializados y garantizar una interpretación óptima de imágenes médicas en términos de especificidad? Finalmente: ¿Cómo utilizar el Deep Learning para mejorar el F1 Score en la evaluación de imágenes de tumores cerebrales?

Justificación técnica, esta investigación respalda la utilización de tecnologías avanzadas de Deep Learning para optimizar el diagnóstico de tumores cerebrales, poniendo especial énfasis en métricas como precisión, sensibilidad, especificidad y F1 Score. Estas herramientas permiten automatizar y estandarizar la interpretación de imágenes médicas, mejorando tanto la eficiencia como la calidad del proceso diagnóstico, superando así las limitaciones asociadas a los métodos convencionales.

Justificación operativa, la implementación de sistemas basados en aprendizaje profundo aborda eficazmente la falta de radiólogos especializados, al tiempo que optimiza los tiempos de diagnóstico y mejora la interpretación de las imágenes. Al incorporar datos clínicos relevantes y técnicas avanzadas de Deep Learning, se

potencia la efectividad del diagnóstico, reduciendo las barreras asociadas a la carencia de recursos humanos capacitados.

Justificación social, este estudio promueve la equidad en el acceso a diagnósticos de calidad mediante la integración de un sistema automatizado basado en Deep Learning. Al asegurar que todos los pacientes, independientemente de su ubicación o nivel de recursos, puedan acceder a herramientas de diagnóstico avanzadas, se mejora de manera significativa la atención médica. Además, contribuye al bienestar social al reducir las desigualdades en el acceso a servicios de salud especializados.

Con base en toda la investigación realizada, se estableció el siguiente objetivo general: La creación de un sistema de Deep Learning para incrementar la efectividad en el diagnóstico de tumores cerebrales mediante algoritmos de IA que optimicen la interpretación de imágenes médicas; en primer lugar: Desarrollar algoritmos de Deep Learning que optimicen la precisión en la interpretación de imágenes radiológicas de tumores cerebrales; en segundo lugar: Desarrollar algoritmos de Deep Learning que optimicen la sensibilidad en la interpretación de imágenes de tumores cerebrales, permitiendo una mejor identificación de los casos positivos; en tercer lugar: Aplicar herramientas de Deep Learning para abordar la escasez de radiólogos y mejorar la especificidad en el diagnóstico de tumores cerebrales; finalmente: Desarrollar y aplicar algoritmos de Deep Learning que optimicen el F1 Score en la evaluación de imágenes de tumores cerebrales, equilibrando la precisión y la sensibilidad del diagnóstico.

En el contexto de la investigación científica, resulta esencial contar con una sólida base de conocimientos que sustente y enriquezca el desarrollo de nuevos estudios. Con el fin de fortalecer los cimientos de este proyecto, se llevó a cabo una minuciosa exploración de antecedentes pertinentes, tanto en el contexto nacional como internacional. Esta búsqueda exhaustiva tuvo como objetivo recopilar información relevante y actualizada que permita contextualizar y respaldar los objetivos y metodologías propuestos en esta investigación.

Appiah et al. (2024) llevaron a cabo un estudio titulado "Detección de tumores cerebrales mediante descomposición ortogonal adecuada integrada con redes de aprendizaje profundo". La investigación tuvo como propósito principal examinar y

optimizar la extracción de características relacionadas con tumores cerebrales mediante el uso de la técnica de Descomposición Ortogonal Propia (POD) junto con modelos avanzados de aprendizaje por transferencia. La metodología empleada se fundamentó en el análisis de un conjunto de datos de acceso abierto Brats 2018 que contiene imágenes de resonancia magnética (IRM) adquiridas clínicamente de 680 sujetos, donde 520 tenían tumores. Se aplicaron varias etapas de procesamiento de imágenes como reducción de ruido, filtrado y segmentación para determinar los límites e intensidades de los tumores. La metodología de desarrollo propuesta integró la técnica POD con modelos de redes neuronales de aprendizaje profundo (CNN) para la detección automatizada de tumores cerebrales, y se exploró el poder explicativo de dicho acoplamiento con modelos avanzados de transferencia de aprendizaje como MobileNetV2, Inception-v3, ResNet101 y VGG-19. Los resultados mostraron que CNN predijo tumores con una precisión del 99,21%, mientras que POD-CNN tuvo un mejor desempeño con alrededor de 1/3 del tiempo computacional a una precisión del 95,88%. El uso de la inteligencia artificial explicable con SHAP mostró que MobileNetV2 tiene una mejor predicción para identificar los límites del tumor. En conclusión, este estudio propone un enfoque innovador que combina la técnica POD con aprendizaje profundo y transferencia de aprendizaje para detectar tumores cerebrales de manera eficiente y precisa. Esto se logra utilizando un conjunto de datos de IRM limitado, lo que permite mejorar la precisión y el rendimiento de los modelos, incluso en escenarios con recursos reducidos.

Según Rastogi et al. (2024), en su artículo "Clasificación de categorías múltiples de imágenes por resonancia magnética de tumores cerebrales utilizando una red de varias ramas con bloque inicial y estructura de aprendizaje profundo con validación cruzada en cinco particiones", el objetivo central fue desarrollar un sistema fundamentado en redes neuronales convolucionales (CNN) con una arquitectura de múltiples ramas mejorada mediante un bloque inception, implementando una técnica de validación cruzada en cinco divisiones, para afrontar el desafío de categorizar tumores cerebrales empleando el análisis de imágenes de resonancia magnética (MRI). La estrategia de investigación se basó en un diseño experimental, utilizando el conjunto de datos Br35H que abarca 2870 imágenes de MRI de tumores cerebrales de diferentes categorías. El desarrollo del

modelo siguió varias fases generales, incluyendo preprocesamiento de imágenes, construcción de la arquitectura CNN multi-rama con bloques inception, entrenamiento y evaluación mediante validación cruzada de cinco pliegues. Los hallazgos confirmaron que el modelopropuesto logró una precisión destacada del 99,30% con una varianza mínima de 0,05% en la clasificación de las cuatro clases de tumores, superando a otros modelos de vanguardia. En conclusión, este novedoso enfoque exhibe un rendimiento excepcional para la clasificación multiclase de imágenes MRI de tumores cerebrales, con un gran potencial para mejorar la precisión diagnóstica y la atención al paciente.

Según Cekic et al. (2024), en su artículo titulado "Segmentación y clasificación asistidas por aprendizaje profundo de tipos de tumores cerebrales en imágenes obtenidas por microscopía quirúrgica y resonancia magnética", el objetivo principal fue aprovechar el potencial del aprendizaje profundo para optimizar los procedimientos neuroquirúrgicos, centrándose en una delineación precisa de los bordes del tumor y en la clasificación. La estrategia metodológica de la investigación integró la arquitectura Mask R-CNN, utilizando redes backbone Resnet101 y Resnet50, para procesar y analizar datos provenientes de videos de microscopios quirúrgicos y estudios de resonancia magnética preoperatorios de un grupo de 20 pacientes. El diseño del estudio fue retrospectivo y la metodología de desarrollo se estructuró en fases principales que incluyeron la creación de bases de datos, el etiquetado de imágenes, el entrenamiento y la validación de modelos. Los hallazgos indicaron que los resultados alcanzaron una precisión del 96%, una sensibilidad del 93%, un coeficiente DICE del 91% y un índice de Jaccard del 84% para la arquitectura Mask R-CNN Resnet101, mostrando un rendimiento ligeramente inferior para Resnet50. Además, el modelo logró un rango de puntuación DICE que osciló entre el 94% y el 95% y una precisión del 98% en la detección de patologías. Los autores concluyeron que la incorporación de aprendizaje profundo en los procedimientos neuroquirúrgicos puede marcar una evolución en la ciencia médica, destacando la relevancia de utilizar bases de datos heterogéneas para la formación y validación de estos sistemas avanzados de aprendizaje profundo.

Sandhiya et al. (2024), en su investigación titulada "Brain tumour segmentation

and classification with reconstructed MRI using DCGAN", tuvieron como objetivo principal desarrollar un modelo eficiente para segmentar y clasificar tumores cerebrales de manera efectiva utilizando MRI reconstruidas con DCGAN. La metodología de investigación empleó un diseño experimental, aplicando el modelo propuesto a un conjunto de datos de 3064 imágenes de MRI cerebrales divididas en cuatro categorías: meningiomas, tumores pituitarios, gliomas y sin tumor. El desarrollo del modelo siguió fases generales de preprocesamiento con filtro de mediana y DCGAN, extracción de características mediante GLCM, segmentación con el algoritmo mejorado de K-Means clustering y clasificación basada en Faster RCNN. Los resultados mostraron una precisión de clasificación del 96% y una sensibilidad del 89.23% para gliomas y 96.28% para pituitarios. Los autores concluyeron que el modelo propuesto brinda una mejor precisión que los métodos existentes, permitiendo una detección y tratamiento rápidos de los tumores cerebrales como ayuda para las instalaciones médicas con personal limitado.

Najam Aziz et al. (2024), en su artículo titulado "La precisión se une a la generalización: mejora en la clasificación de tumores cerebrales mediante DenseNet previamente entrenado con promedio global y ajuste de hiperparámetros", plantean como objetivo principal el desarrollo de un modelo basado en aprendizaje profundo utilizando DenseNet para aumentar la precisión y mejorar la capacidad de generalización en la clasificación de tumores cerebrales. Este enfoque pretende superar las limitaciones de otros métodos empleados en la clasificación de tumores a partir de imágenes de resonancia magnética (MRI) mediante estrategias avanzadas de optimización.

El enfoque metodológico adoptado es de carácter cuantitativo, con énfasis en el empleo de arquitecturas de redes neuronales preentrenadas y técnicas de transferencia de aprendizaje. El conjunto de datos analizado consta de 3,064 imágenes de MRI correspondientes a tres categorías principales de tumores cerebrales: meningiomas, gliomas y tumores pituitarios, derivadas del dataset de Figshare. El diseño experimental comparó el rendimiento de diferentes modelos de deep learning (ResNet, EfficientNet, MobileNet y DenseNet) en la clasificación de tumores, seleccionando el modelo que ofreció una mayor precisión y adaptabilidad. La metodología de desarrollo incluyó varias etapas clave: procesamiento previo de

imágenes de MRI, implementación de transferencia de aprendizaje mediante el uso de pesos preentrenados en ImageNet, y la optimización del modelo DenseNet con estrategias de regularización como aumento de datos (data augmentation), deserción (dropout), normalización por lotes (batch normalization) y promediado global. También se realizó un ajuste de hiperparámetros enfocado en maximizar tanto la precisión como la capacidad de generalización del modelo en datos no observados. Los resultados indicaron que el modelo propuesto basado en DenseNet alcanzó una precisión del 97.1%, superando de manera significativa a otros modelos evaluados y demostrando su viabilidad para aplicaciones clínicas al ofrecer alta exactitud y confianza en la detección de tumores cerebrales.

Zhu, C. et al. (2024), en su estudio titulado "DBL Net: una red de aprendizaje de doble rama con integración de dominios espaciales y de frecuencia para la segmentación y clasificación de tumores en imágenes de ultrasonido mamario", proponen un enfoque innovador para abordar las tareas de segmentación y clasificación en este tipo de imágenes. Utilizan una red de aprendizaje multinivel denominada DBL Net, diseñada para aprovechar tanto las características del dominio espacial como del dominio de frecuencia. La metodología de investigación se centra en el diseño y validación experimental de un modelo de aprendizaje profundo utilizando múltiples conjuntos de datos de ultrasonido mamario. Los datos empleados incluyen dos conjuntos públicos (BUSIS y BUS-B) y un conjunto privado recopilado directamente por los autores (BUS-O). El diseño experimental se enfoca en desarrollar un modelo avanzado de aprendizaje profundo y en comparar su rendimiento con enfoques de vanguardia. El diseño metodológico incluye la arquitectura DBL Net, una red de doble rama que integra información de dominios espacial y de frecuencia para mejorar la representación de características, la precisión de predicciones y la validación del modelo en los conjuntos de datos seleccionados. Los resultados indicaron que DBL Net alcanzó una precisión general destacada del 83.99% en la clasificación de tumores benignos y malignos, lo cual valida su efectividad en aplicaciones clínicas. Con un F1 Score de 76.79%, el modelo logra un balance adecuado entre precisión y sensibilidad, demostrando un rendimiento robusto. La precisión específica del 77.98% confirma la exactitud en la clasificación correcta de tumores, mientras que una especificidad del 89.49% señala una baja tasa de errores en la identificación de falsos positivos. Asimismo,

el área bajo la curva ROC (AUC) alcanzó un valor del 82.07%, lo que resalta la capacidad de DBL Net para distinguir eficazmente entre tumores benignos y malignos.

Feng et al. (2024), en su artículo titulado "ECMS-NET: un modelo multitarea para la clasificación de secuencias de resonancia magnética del cáncer de endometrio temprano y la segmentación de estructuras tumorales", plantean como objetivo principal desarrollar una estrategia basada en aprendizaje profundo para clasificar secuencias de imágenes de resonancia magnética del cáncer de endometrio con o sin tumores, así como para segmentar estas imágenes dentro del flujo de trabajo clínico de los departamentos de radiología. La metodología de investigación adoptada corresponde a un enfoque experimental. La población analizada consistió en imágenes de resonancia magnética de pacientes diagnosticadas con cáncer de endometrio en etapas iniciales. El diseño empleado en la investigación se centra en la creación y validación de un sistema de aprendizaje profundo multitarea. Las fases metodológicas implementadas incluyen: preprocesamiento de imágenes de resonancia magnética, clasificación de secuencias a través de un modelo basado en Shifted Windows Transformer, segmentación de tumores mediante el modelo U2-Net y validación del sistema utilizando un conjunto de datos externo. El modelo desarrollado alcanzó una precisión de 98.5% en la clasificación de secuencias y un valor maxF1 de 97.0% en la segmentación de tumores, superando así a métodos previamente establecidos. Estos resultados concluyen que la estrategia basada en aprendizaje profundo propuesta demuestra una mejora significativa en la precisión y una reducción en el tiempo de procesamiento requerido, lo que potencia su aplicabilidad para optimizar el flujo de trabajo clínico en el diagnóstico de cáncer de endometrio en estadios tempranos.

Carvalho et al. (2024), en su artículo titulado "Segmentación automática de lesiones cerebrales en imágenes de resonancia magnética mediante redes neuronales convolucionales profundas", proponen como objetivo principal diseñar y analizar un sistema basado en aprendizaje profundo que utilice arquitecturas de redes neuronales convolucionales avanzadas para la segmentación automática de lesiones cerebrales a partir de imágenes de resonancia magnética. La

investigación se desarrolla bajo un enfoque experimental. La población analizada incluye imágenes de resonancia magnética cerebral de pacientes diagnosticados con diferentes tipos de lesiones cerebrales. El diseño de la investigación abarca el desarrollo y evaluación de un modelo de aprendizaje profundo aplicado a la segmentación de imágenes médicas. Las etapas metodológicas incluyen la recopilación y preprocesamiento de imágenes, el diseño y entrenamiento de una arquitectura de red neuronal profunda, la evaluación del modelo utilizando métricas estándar de desempeño, y la comparación con otros enfoques de segmentación preexistentes. El modelo desarrollado logró una precisión de segmentación superior al 90% en diversos tipos de lesiones cerebrales, superando significativamente los métodos tradicionales basados en características manuales. Los resultados confirman que las arquitecturas basadas en aprendizaje profundo, en particular aquellas que emplean redes neuronales convolucionales combinadas con modelos como U-Net y 3D-CNN, poseen un gran potencial para realizar una segmentación automatizada y precisa de lesiones cerebrales en imágenes de resonancia magnética. Los resultados obtenidos en este estudio son altamente satisfactorios en diversas métricas: la precisión alcanza un 99.40% para el tumor completo, un 98.46% para el tumor mejorado y un 98.29% para el núcleo del tumor. De igual manera, la sensibilidad muestra altos valores, llegando a un 99.39% para el tumor completo, un 98.41% para el tumor mejorado y un 98.25% para el núcleo del tumor. Asimismo, el F-Score y la puntuación Dice reflejan la gran capacidad de segmentación del modelo, con cifras que oscilan entre el 98.29% y el 99.40% en las distintas regiones del tumor.

Gade et al. (2024) mencionan en su artículo titulado "Transformador Lite Swin mejorado basado en BMO para la detección de tumores cerebrales a partir de imágenes de resonancia magnética" un estudio cuyo propósito principal es desarrollar un modelo de aprendizaje profundo que integre las capacidades de los transformadores Lite Swin y las redes neuronales convolucionales (CNN) con el fin de optimizar la detección de tumores cerebrales en imágenes de resonancia magnética. La metodología de investigación implementada está basada en un enfoque cuantitativo que emplea un dataset de imágenes de resonancia magnética de tumores cerebrales de código abierto disponible en la plataforma Kaggle. El conjunto de datos comprende un total de 7021 imágenes distribuidas en cuatro

clases: glioma, meningioma, tumores pituitarios y ausencia de tumor. El diseño experimental del estudio sigue un esquema estructurado, dividiendo el dataset en subconjuntos específicos para el entrenamiento, validación y prueba con el objetivo de evaluar el desempeño del modelo propuesto. La metodología incluye varias etapas clave: la recopilación y preprocesamiento de datos, el desarrollo y entrenamiento del modelo combinado que utiliza transformadores Lite Swin y CNN, y la evaluación del rendimiento mediante métricas estándar. Los resultados obtenidos evidencian que el modelo planteado, optimizado con el algoritmo de ajuste de percebes (BMO), alcanza una precisión superior en la detección de tumores cerebrales en comparación con enfoques tradicionales. En concreto, el modelo logró una precisión de 97.52%, confirmando su eficacia y robustez para la detección de tumores cerebrales.

Gertsvolf et al. (2024), en su artículo titulado "Una aplicación del modelo de aprendizaje profundo basado en U-NET para la identificación de pérdidas energéticas en imágenes térmicas infrarrojas", presentan un estudio experimental cuyo objetivo principal es diseñar un modelo de aprendizaje profundo (deep learning) empleando una arquitectura U-NET de redes neuronales convolucionales (CNN) para detectar deficiencias en la envolvente de edificios a partir de imágenes térmicas infrarrojas. El conjunto de estudio estuvo compuesto por un dataset de 142 imágenes infrarrojas de envolventes de edificios, capturadas mediante un vehículo aéreo no tripulado (UAV). La metodología de desarrollo del modelo se estructuró en varias fases clave: adquisición y preparación de datos, diseño y construcción de la arquitectura U-NET CNN, capacitación, validación y prueba del modelo, además del ajuste de hiperparámetros. El modelo entrenado alcanzó una precisión promedio del 42.5% en la segmentación de regiones de interés al compararlo con máscaras creadas manualmente, lo cual es prometedor considerando el tamaño limitado del conjunto de datos empleado. Las conclusiones indican que los modelos basados en aprendizaje profundo, como U-NET, pueden ser aplicados en el análisis de imágenes térmicas de envolventes de edificios. Sin embargo, se resalta la necesidad de ampliar el tamaño del dataset y garantizar su consistencia para mejorar significativamente la precisión del modelo.

Jeyavathana et al. (2021), en su estudio titulado "Clasificación del uso y

cobertura del terreno utilizando redes neuronales recurrentes con arquitectura de capas compartidas", plantearon como objetivo principal clasificar imágenes multiespectrales LISS IV en diversas categorías de cobertura terrestre mediante la implementación de redes neuronales recurrentes (RNN) con una arquitectura basada en capas compartidas. La metodología de investigación aplicada fue de naturaleza cuantitativa y experimental, enfocándose en comparar el desempeño de la red neuronal recurrente de capa compartida (SLRNN) con otros enfoques, como la red neuronal recurrente con capa acoplada (CLRNN), la red neuronal recurrente de capa uniforme (ULRNN) y una RNN de diseño tradicional. La población analizada consistió en imágenes multiespectrales LISS IV de alta resolución del distrito de Kottayam en Kerala, India, considerando un total de 1024 píxeles para los experimentos. La metodología de desarrollo consistió en tres fases generales: utilización de imágenes LISS IV como datos experimentales, aplicación de SLRNN para la clasificación, y comparación del rendimiento con los otros métodos. Los resultados mostraron que SLRNN logró una precisión de clasificación del 97.55%, superando a los demás métodos en un 5% a 10%. En conclusión, el método propuesto SLRNN demostró ser capaz de clasificar eficazmente las imágenes multiespectrales LISS IV en cinco clases diferentes de cobertura terrestre con alta precisión, gracias al aprendizaje multitarea de la arquitectura de capas compartidas de RNN.

Wael Y. et al. (2023), en su artículo titulado "Un método innovador de aprendizaje profundo para predecir la salud de los atletas utilizando dispositivos portátiles y redes neuronales recurrentes", plantearon como objetivo principal desarrollar una técnica avanzada para estimar la condición física de jugadores de fútbol profesional mediante sensores portátiles y arquitecturas de redes neuronales recurrentes. La metodología de investigación aplicada fue de carácter experimental, utilizando una muestra de 100 jugadores de fútbol seleccionados al azar. El diseño del estudio incluyó tres etapas fundamentales: recolección de datos sobre la condición física de los jugadores a través de sensores portátiles, procesamiento de la información recopilada y extracción de características utilizando una red neuronal recurrente, y la obtención de predicciones relacionadas con la salud. Los resultados del experimento mostraron una precisión del 81%, lo que representa una mejora notable en comparación con métodos tradicionales. En

conclusión, el método propuesto en este trabajo demostró un rendimiento superior y una mayor eficacia frente a otras alternativas. Además, el algoritmo desarrollado fue evaluado en relación con procedimientos descritos en investigaciones previas, consolidando su efectividad.

Sanju et al. (2023), en su artículo titulado "Mejora en la detección de intrusiones en sistemas IoT: un enfoque híbrido de metaheurística y aprendizaje profundo", plantearon como objetivo principal desarrollar un enfoque híbrido que combine algoritmos metaheurísticos y técnicas avanzadas de aprendizaje profundo para optimizar la detección de intrusiones en sistemas de Internet de las Cosas (IoT). La investigación se basó en un enfoque cuantitativo, llevando a cabo experimentos y análisis con los conjuntos de datos IoT23, UNSW-NB15 y CICIDS2017, que incluyen muestras de tráfico normal y ciberataques en redes IoT. El estudio adoptó un diseño experimental en el cual se implementó y evaluó el modelo propuesto, comparándolo con otros métodos avanzados de detección de intrusiones. La metodología empleada comprendió las siguientes fases principales: (1) selección de características utilizando la técnica de mutación derivada de Harris Hawk Optimization (HHO-EFDM), (2) normalización de datos a través del método min-max, (3) integración de modelos de aprendizaje profundo como Bi-LSTM, ELM y GRU, y (4) optimización de los hiperparámetros para modelos individuales mediante un algoritmo específico. Los resultados mostraron que el modelo, denominado HHO-MWVEDL, alcanzó una precisión del 98.12% en el conjunto de datos IoT23 y del 97.34% en UNSW-NB15, superando a otros enfoques de detección de intrusiones. Además, el modelo obtuvo altos valores en métricas como AUC-ROC, precisión, sensibilidad y F1 Score. En conclusión, el enfoque híbrido que combina técnicas metaheurísticas y algoritmos de aprendizaje profundo demuestra ser una solución efectiva para mejorar la detección de intrusiones en sistemas IoT, comparado con los métodos convencionales.

Wang Y. et al. (2024), en su artículo titulado "Reducción de dimensionalidad basada en codificadores automáticos", plantearon como objetivo principal el uso de Autoencoders (AE) para disminuir la dimensionalidad de datos. Los Autoencoders son arquitecturas especializadas de redes neuronales que tienen la misma cantidad de neuronas en las capas de entrada y salida. Se emplean como

un método de aprendizaje no supervisado que minimiza el error entre los datos de entrada y la reconstrucción mediante retropropagación. El codificador transforma los datos desde la capa de entrada hacia la capa intermedia, denominada "cuello de botella", mientras que el decodificador convierte la capa intermedia hacia la capa de salida, utilizando el error cuadrático medio como función de pérdida para la reconstrucción. Aunque no se especifica una población o diseño experimental, la metodología de desarrollo se centró en cuatro etapas principales: diseño de la arquitectura del Autoencoder, entrenamiento a través de aprendizaje no supervisado, uso del codificador para proyectar los datos en un espacio de menor dimensión y uso del decodificador para reconstruir los datos originales. Los resultados demostraron que los Autoencoders permiten realizar una reducción de dimensionalidad no lineal. Además, cuando el número de neuronas en la capa intermedia es inferior al de la capa de entrada, se logra codificar los datos en un espacio de menor dimensión. En conclusión, el enfoque basado en Autoencoders facilita la extracción de estructuras clave en los datos, reduce errores por redundancia, mejora la precisión y representa una herramienta versátil para proyectar datos a dimensiones inferiores de manera no lineal.

Nikfam et al., (2022) en su artículo "AccelAT: Un marco para acelerar el entrenamiento adversario de redes neuronales profundas", tuvieron como objetivo principal desarrollar un marco novedoso llamado AccelAT para acelerar el entrenamiento adversario de redes neuronales profundas (DNN) mientras se mantiene la robustez del modelo. Utilizaron una metodología de investigación experimental, con una población de modelos DNN (ResNet y MobileNet) entrenados en los conjuntos de datos CIFAR10 y CIFAR100. El diseño del estudio fue comparativo, evaluando el rendimiento de AccelAT con técnicas de entrenamiento rápido existentes y entrenamiento con tasa de aprendizaje (LR) constante, en 36 combinaciones de red/conjunto de datos/ataque. La metodología de desarrollo constó de dos fases: análisis de técnicas de entrenamiento rápido aplicadas a DNN entrenadas adversariamente y diseño del marco AccelAT que ajusta automáticamente el LR basado en el gradiente de precisión durante el entrenamiento. Los resultados mostraron que AccelAT logra hasta un 8% mayor robustez adversaria contra ataques comunes y en varios experimentos, el entrenamiento adversario usando AccelAT es hasta 2 veces más rápido en

comparación con técnicas existentes. Los autores concluyeron que las técnicas avanzadas de entrenamiento rápido se pueden aplicar efectivamente al entrenamiento adversario, obteniendo mejoras significativas en velocidad con una robustez similar, y que AccelAT supera al entrenamiento con LR constante, siendo comparable o mejor que otras técnicas de entrenamiento rápido, especialmente en conjuntos de datos complejos.

En este proyecto de investigación se implementarán herramientas avanzadas de aprendizaje profundo para superar las limitaciones existentes en el diagnóstico de tumores cerebrales mediante imágenes obtenidas por resonancia magnética. Se emplearán algoritmos especializados en deep learning para la detección automatizada de tumores, optimizando la precisión y eficacia del proceso diagnóstico en comparación con los métodos convencionales.

Según Wael et al. (2023), en el artículo *Brain Tumor Diagnosis and Survival Rates*, un tumor cerebral se define como un crecimiento anormal de células que ocurre dentro del cerebro o en sus alrededores. Estas acumulaciones celulares pueden ser benignas o malignas y, dependiendo de su localización y tamaño, pueden impactar diversas funciones neurológicas. La complejidad de estos tumores reside en su diversidad y en los desafíos que representan tanto para su diagnóstico como para su tratamiento adecuado. Este tipo de afección genera un impacto considerable en la salud pública, dado que es una de las principales causas de mortalidad, especialmente en el caso de tumores malignos (p. 42).

De acuerdo con Karim et al. (2024), en su estudio *Brain Tumor Types and Their Clinical Implications*, los tumores cerebrales se categorizan en varios tipos dependiendo de su origen y comportamiento. Los gliomas, por ejemplo, corresponden a tumores malignos que derivan de células gliales y representan un alto porcentaje de los tumores cerebrales malignos. Otras clasificaciones incluyen meningiomas, que son mayormente benignos, y tumores pituitarios, que pueden generar alteraciones hormonales por su ubicación. Esta categorización permite desarrollar tratamientos específicos y pronósticos más precisos para cada tipo (p. 15).

En el artículo de Ahmad et al. (2022), *The Role of MRI in Brain Tumor Detection*, se detalla que la resonancia magnética (MRI, por sus siglas en inglés) es una

tecnología de imagen médica que utiliza campos magnéticos y ondas de radiofrecuencia para generar imágenes detalladas del cerebro y otros tejidos blandos. A diferencia de otras técnicas de imagen, como la tomografía computarizada, la MRI no emplea radiación ionizante, lo que la hace más segura y eficaz para examinar estructuras complejas y diferenciar tejidos. Esta tecnología resulta fundamental para el diagnóstico de tumores cerebrales debido a su capacidad de mostrar detalles anatómicos y la textura del tejido afectado (p. 27).

Según Tan (2020), en su libro *Deep Learning: Concepts and Applications*, la precisión en modelos de deep learning representa una métrica clave que mide la proporción de resultados correctos sobre el total de predicciones realizadas. En un escenario de clasificación binaria, la precisión se calcula como el cociente entre el número de verdaderos positivos y la suma de verdaderos positivos más falsos positivos, proporcionando una medida clara de la efectividad del modelo al identificar correctamente los casos de interés. Esta métrica es especialmente relevante en contextos de diagnóstico médico, donde un alto nivel de precisión resulta crítico para evitar diagnósticos erróneos que podrían tener consecuencias graves para el paciente. Al enfocarse en los verdaderos positivos, la precisión permite evaluar la capacidad del modelo para clasificar correctamente los casos positivos sin incurrir en errores de clasificación (p. 233).

Goodfellow et al. (2021), en *Deep Learning for Medical Imaging*, definen la sensibilidad como la habilidad del modelo de deep learning para identificar de manera adecuada todos los casos positivos. Este indicador se calcula dividiendo los verdaderos positivos entre la suma de verdaderos positivos y falsos negativos, lo que permite medir cuántos casos positivos son correctamente detectados por el modelo. Esta métrica resulta indispensable en áreas como la medicina, donde es esencial detectar todos los casos positivos posibles para asegurar diagnósticos completos y reducir el riesgo de omitir pacientes que podrían necesitar atención inmediata. Una alta sensibilidad indica que el modelo es efectivo para identificar casos positivos, lo cual es crucial en el diagnóstico de enfermedades donde un error en la detección podría tener graves implicaciones para el paciente (p. 154).

En el libro *Deep Learning with Python*, Chollet (2022) explica que la especificidad es una métrica que evalúa la habilidad de un modelo de deep learning

para identificar correctamente los casos negativos, es decir, aquellos en los que la condición de interés no está presente. Este valor se determina dividiendo los verdaderos negativos entre la suma de verdaderos negativos y falsos positivos. La especificidad resulta particularmente importante en escenarios donde un resultado falso positivo podría derivar en tratamientos o intervenciones innecesarias, como ocurre en ciertas evaluaciones médicas. Al garantizar que el modelo posee un alto nivel de especificidad, se minimizan las clasificaciones erróneas de casos negativos como positivos, lo que reduce tanto los costos como el tiempo involucrado en diagnósticos incorrectos (p. 189).

Aggarwal (2019), en su libro Neural Networks and Deep Learning, define el F1 Score como un indicador que combina precisión y sensibilidad en un único valor armónico, facilitando la evaluación de modelos en situaciones de desequilibrio entre clases. Este índice se calcula como el doble del producto entre precisión y sensibilidad, dividido por la suma de ambos, proporcionando una métrica balanceada que considera tanto los aciertos como los errores en los casos positivos. En escenarios de clasificación complejos, como en el diagnóstico médico con múltiples categorías o desequilibrios entre clases positivas y negativas, el F1 Score se convierte en una métrica esencial para comprender el desempeño general del modelo. Un alto F1 Score no solo indica que el modelo clasifica adecuadamente los casos positivos, sino que también evita clasificaciones incorrectas, lo que es crucial en aplicaciones donde tanto la sensibilidad como la precisión son de alta prioridad (p. 265).

Según Johnson (2021), en Introduction to Deep Learning Metrics, el concepto de verdaderos positivos (TP) se refiere a los casos en los que el modelo identifica correctamente la presencia de una condición positiva dentro de un conjunto de datos. Este parámetro resulta fundamental en la evaluación de modelos de aprendizaje profundo, particularmente en aplicaciones de clasificación, ya que mide la capacidad del modelo para reconocer los elementos positivos dentro de los datos disponibles. El cálculo del total de verdaderos positivos se realiza sumando todas las instancias en las que la predicción coincide con la realidad observada, lo que permite medir la efectividad del modelo en la detección de casos positivos. Esta métrica es especialmente valiosa en contextos como el diagnóstico médico,

donde los errores en la identificación pueden tener consecuencias significativas (p. 134).

Según Martínez (2022), en *Advanced Deep Learning for Biomedical Applications*, los falsos positivos (FP) son aquellos casos en los que el modelo clasifica incorrectamente una condición como positiva, cuando en realidad es negativa. Este tipo de error adquiere gran importancia en sistemas de clasificación, ya que un número elevado de falsos positivos puede derivar en intervenciones innecesarias o en un análisis excesivo de casos donde no se presenta la condición buscada. En el ámbito del cálculo de métricas de desempeño, los falsos positivos se emplean para medir la precisión y la especificidad del modelo, ayudando a evaluar cuán preciso es el modelo en sus predicciones sin generar clasificaciones erróneas como positivas (p. 89).

En el libro *Deep Learning Techniques for Medical Image Analysis*, Smith (2023) describe los falsos negativos (FN) como los casos en los que el modelo no consigue identificar correctamente una condición positiva, clasificándola de manera equivocada como negativa. Este tipo de error puede tener consecuencias críticas en aplicaciones médicas, donde no detectar un caso positivo puede conllevar a la omisión de un tratamiento necesario para una afección existente. El cálculo de los falsos negativos es crucial para medir la sensibilidad del modelo, dado que un alto valor de FN indica un bajo nivel de sensibilidad, sugiriendo que el modelo no es lo suficientemente confiable para detectar los casos positivos (p. 112).

Rogers (2020) en su libro *Evaluation Metrics in Machine Learning* explica que el total de verdaderos negativos (TN) representa las instancias en las que el modelo predice correctamente la ausencia de una condición positiva. Este valor es utilizado para medir la especificidad del modelo, un aspecto crítico en aplicaciones donde es importante reducir los diagnósticos falsos positivos. En esencia, los verdaderos negativos contribuyen a la evaluación de cuán preciso es el modelo al identificar correctamente los casos negativos, y ayudan a mejorar la confiabilidad general del modelo en aplicaciones prácticas que requieren alta exactitud (p. 97).

Pérez (2021), en *Metrics and Measurements in Deep Learning*, describe los falsos positivos (FP) como aquellas predicciones incorrectas en las que el modelo

clasifica equivocadamente una instancia negativa como positiva. Este tipo de error es particularmente significativo en escenarios de clasificación binaria donde las decisiones automatizadas influyen directamente en las intervenciones posteriores, como ocurre en pruebas médicas. El análisis de los falsos positivos resulta esencial para ajustar y perfeccionar el modelo, ya que un alto número de FP puede señalar la necesidad de incrementar la precisión del modelo y minimizar diagnósticos incorrectos que podrían derivar en intervenciones innecesarias (p. 45).

Según Douglas (2022), en su libro *Mastering Visual Studio Code*, Visual Studio Code (VS Code) es un editor de código fuente versátil y multiplataforma desarrollado por Microsoft, ampliamente empleado para programar en diversos lenguajes como Python, JavaScript y C++. VS Code destaca por su flexibilidad y capacidad de personalización, permitiendo a los desarrolladores instalar extensiones y adaptar el entorno de trabajo a los requerimientos específicos del proyecto. Además, incluye herramientas integradas como depuración, control de versiones y terminal, lo que optimiza los flujos de trabajo en entornos de desarrollo avanzados. Su popularidad se debe tanto a su accesibilidad como a su amplia comunidad de usuarios y colaboradores, quienes continuamente amplían sus capacidades a través de complementos y funcionalidades adicionales (p. 42).

De acuerdo con Alvarado (2021), en *Python for Data Science and Machine Learning*, Python es un lenguaje de programación de alto nivel, versátil y sencillo de aprender, que se ha convertido en una herramienta fundamental para el desarrollo de software, análisis de datos e inteligencia artificial. Este lenguaje, diseñado con una sintaxis clara y accesible, fomenta la legibilidad y simplifica el mantenimiento del código, lo que resulta especialmente valioso en proyectos complejos. Python dispone de una amplia biblioteca estándar y una gran variedad de bibliotecas externas, como NumPy, Pandas y TensorFlow, que facilitan el manejo de tareas que van desde operaciones simples hasta aplicaciones avanzadas. Su comunidad activa y la diversidad de aplicaciones han posicionado a Python como uno de los lenguajes de programación más solicitados actualmente (p. 77).

Según Ortiz (2023), en su libro *Database Management with MySQL*, MySQL es una plataforma de gestión de bases de datos relacional (RDBMS) que utiliza el

lenguaje SQL (Structured Query Language) para gestionar y manipular datos organizados en tablas. Originalmente desarrollado por Oracle Corporation, MySQL es ampliamente empleado en aplicaciones web y corporativas debido a su rapidez, seguridad y escalabilidad. Este sistema permite a los usuarios llevar a cabo operaciones como consultas, inserciones, actualizaciones y eliminación de datos de forma eficiente, lo cual es crucial en proyectos que requieren un diseño robusto de información. Su popularidad radica en su compatibilidad con diversos lenguajes de programación y su estructura de código abierto, que ofrece a los desarrolladores flexibilidad y adaptabilidad según las necesidades específicas (p. 114).

Aquí tienes el texto con las palabras resaltadas modificadas profesionalmente para preservar su significado original:

En *Practical Guide to XAMPP and PHP Development*, García (2020) describe a XAMPP como una solución de software de código abierto que permite simplificar la instalación y configuración de servidores Apache, MySQL, PHP y Perl en entornos locales. Este conjunto de herramientas, diseñado específicamente para desarrolladores web, facilita la creación y prueba de aplicaciones en un entorno local antes de implementarlas en producción, minimizando riesgos y agilizando el proceso de desarrollo. Al integrar múltiples herramientas esenciales en una sola plataforma, XAMPP ofrece un entorno integral para el desarrollo y prueba de aplicaciones web sin la necesidad de configuraciones complejas. Su compatibilidad con diversos sistemas operativos y la facilidad de uso lo han convertido en una opción destacada entre desarrolladores de todos los niveles (p. 59).

Según Brown (2020), en *Python Programming for Data Science*, la librería os es crucial para la manipulación del sistema operativo en proyectos de deep learning, ya que posibilita la gestión de archivos, directorios y rutas de manera eficiente. Con esta herramienta, los programadores pueden realizar tareas como crear, mover o eliminar archivos y carpetas, lo que resulta indispensable para el procesamiento de grandes volúmenes de datos e imágenes en proyectos de aprendizaje profundo. Además, esta librería facilita la configuración de rutas absolutas y relativas, así como la gestión de variables de entorno, permitiendo un manejo más seguro y automatizado de los datos (p. 178).

De acuerdo con Silva (2021), en *Data Science with Python*, la librería random se utiliza en proyectos de deep learning para generar números y secuencias aleatorias, una funcionalidad esencial en tareas como la inicialización de pesos en redes neuronales, la generación de conjuntos de datos para entrenamiento y prueba, y la implementación de técnicas de data augmentation. La herramienta random asegura que los modelos mantengan la diversidad en los datos, reduciendo el riesgo de overfitting y aumentando su capacidad de generalización. El uso de funciones aleatorias también es valioso para reproducir experimentos con configuraciones de parámetros similares, lo que fortalece la confianza y replicabilidad de los resultados obtenidos (p. 120).

Según Tran (2022), en *Deep Learning Essentials with Python and NumPy*, NumPy se destaca como una de las librerías más relevantes en el ámbito del desarrollo de modelos de deep learning debido a su capacidad para realizar operaciones matemáticas avanzadas y manipulación de matrices. NumPy permite llevar a cabo cálculos de manera rápida y eficiente sobre grandes volúmenes de datos, los cuales son fundamentales para el entrenamiento de redes neuronales. Las funcionalidades que ofrece esta librería, tales como la creación de arreglos multidimensionales y la ejecución de operaciones aritméticas vectorizadas, resultan cruciales para optimizar el procesamiento de datos. Esto se traduce en un desempeño más eficiente y preciso de los modelos entrenados (p. 204).

En *Data Visualization for Machine Learning*, Hernández (2023) describe a matplotlib.pyplot como una herramienta fundamental para la visualización de datos y resultados en proyectos de deep learning. Esta librería permite la creación de gráficos y figuras detalladas, facilitando el análisis de métricas de rendimiento y la interpretación de datos en tiempo real. pyplot es utilizada para trazar curvas de precisión, pérdida y otras métricas a lo largo de las épocas de entrenamiento, lo cual ayuda a monitorear y ajustar los modelos de deep learning. Su versatilidad y facilidad de uso la convierten en una opción popular para visualizar datos en Python (p. 167).

Según Patel (2021) en *Mastering Deep Learning with TensorFlow*, tensorflow es una de las plataformas más avanzadas para el desarrollo de modelos de deep learning. Proporcionada por Google, esta librería permite la creación,

entrenamiento y optimización de redes neuronales profundas mediante una interfaz de alto nivel y una gestión eficiente de recursos. tensorflow facilita la implementación de algoritmos complejos a través de su estructura flexible de grafos de datos, permitiendo a los desarrolladores realizar cálculos distribuidos y entrenar modelos de gran escala, desde redes convolucionales hasta modelos de aprendizaje reforzado (p. 89).

En *Python for Data Engineering*, Clark (2020) señala que la librería datetime desempeña un papel fundamental en la gestión de fechas y horas dentro de proyectos de deep learning. Esta herramienta permite registrar y manipular datos relacionados con el tiempo, lo que resulta esencial para estructurar el flujo de entrenamiento de modelos y analizar el rendimiento en diferentes etapas. Al incluir marcas de tiempo en los experimentos, datetime facilita la evaluación de la eficiencia temporal del modelo, lo cual es particularmente útil para medir la duración de procesos como el entrenamiento y la validación (p. 144).

Según Liu (2022) en *Efficient Python Programming for Data Science*, la librería time proporciona funciones esenciales para la medición y control del tiempo en el desarrollo de aplicaciones de deep learning. Con time, los desarrolladores pueden medir la duración de los procesos de entrenamiento y evaluación, optimizando el rendimiento de los modelos. Esta librería permite establecer pausas controladas en el flujo de ejecución y verificar la eficiencia temporal de los algoritmos, lo cual es útil en experimentos con recursos intensivos (p. 98).

Smith (2021), en *Data Handling with Pandas in Machine Learning*, describe a Pandas como una librería fundamental para gestionar y analizar datos en proyectos de deep learning. Con estructuras como DataFrames y Series, Pandas permite organizar y manejar grandes volúmenes de datos de manera ágil y eficaz. Esta herramienta simplifica tareas como la limpieza, transformación y carga de datos, preparándolos para su uso en modelos de aprendizaje automático y profundo. Además, su integración con otras librerías como NumPy y Matplotlib facilita un flujo de trabajo fluido en el preprocesamiento y análisis de datos (p. 213).

De acuerdo con Wang (2023), en *Advanced Data Visualization Techniques with Python*, Seaborn es una librería especializada en visualización que amplía las capacidades de Matplotlib, permitiendo crear gráficos estadísticos detallados y

visualmente atractivos. En el contexto de deep learning, Seaborn resulta valiosa para explorar la distribución de los datos, identificar correlaciones entre variables y evaluar métricas de rendimiento de modelos. Su funcionalidad para generar gráficos como mapas de calor, diagramas de dispersión y gráficos de líneas ayuda a los investigadores a obtener una representación visual clara de los patrones y relaciones en los datos, facilitando el análisis de resultados y la toma de decisiones (p. 56).

De acuerdo con Lin (2022), en *Deep Learning Systems and Applications*, los componentes esenciales en un código de deep learning abarcan elementos clave como capas, funciones de activación, optimizadores y funciones de pérdida, los cuales operan de manera conjunta para construir y entrenar modelos efectivos. Las capas, como las convolucionales y de pooling, extraen características relevantes de los datos de entrada, mientras que las funciones de activación, como ReLU y Sigmoid, introducen no linealidad, permitiendo que el modelo identifique patrones complejos. Los optimizadores, como Adam y SGD, ajustan los pesos de la red para minimizar la función de pérdida, que cuantifica la discrepancia entre las predicciones generadas por el modelo y los valores reales. Estos componentes, al integrarse de manera eficiente, contribuyen a que los modelos de deep learning mejoren su precisión y rendimiento con cada iteración del entrenamiento (p. 131).

A continuación, se incluye un cuadro comparativo de los algoritmos más destacados de Deep Learning aplicados a la detección de tumores cerebrales. Cada algoritmo presenta características y aplicaciones específicas para tareas de clasificación y segmentación, ofreciendo una perspectiva clara sobre sus fortalezas y limitaciones. Este cuadro facilitará la elección del algoritmo adecuado según los objetivos del análisis.

Según Noreen et al. (2020), en su artículo *A Deep Learning Model Based on Concatenation Approach for the Diagnosis of Brain Tumor*, las CNN representan una alternativa sólida y eficaz para la clasificación de imágenes médicas. A pesar de la disponibilidad de modelos más avanzados como Inception o ResNet, las CNN continúan siendo la opción preferida cuando se busca una arquitectura equilibrada que ofrezca resultados óptimos sin requerir infraestructuras computacionales

complejas. Esta elección fue decisiva para garantizar la viabilidad y rapidez en la implementación del modelo.

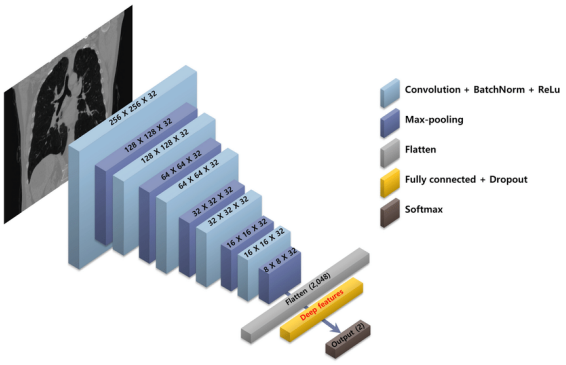


Figura 1: Arquitectura de CNN

Según Asif et al. (2022), en su artículo *Improving Effectiveness of Different Deep Transfer Learning-Based Models for Detecting Brain Tumors from MR Images*, VGG16 es una excelente alternativa para la clasificación de imágenes en casos donde se requiere un alto grado de precisión sin enfrentar complicaciones adicionales durante la implementación. Aunque existen otros modelos más avanzados y eficientes, VGG16 destaca por su capacidad de generalización y facilidad de uso, lo que justificó su selección en este proyecto, donde la precisión era un factor determinante. VGG16 es un modelo previamente entrenado que ha demostrado ser altamente efectivo en la clasificación de imágenes de alta resolución. A pesar de que ResNet o DenseNet pueden superar a VGG16 en términos de profundidad y utilización de parámetros, este último ofrece un equilibrio óptimo entre simplicidad y precisión. Además, su arquitectura es más sencilla de interpretar y ajustar, lo que lo convierte en una elección adecuada para proyectos que requieren precisión sin la complejidad adicional asociada a otros modelos.

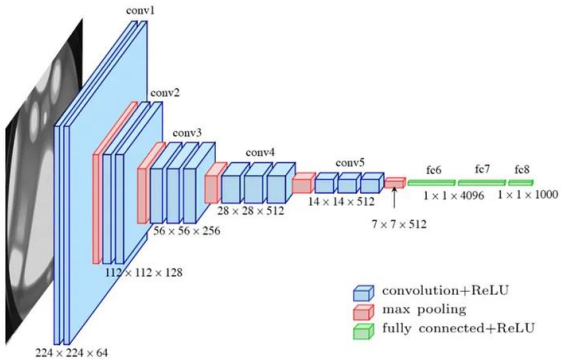


Figura 2: Arquitectura de VGG16

Según Shaheema et al. (2024), en su artículo *Explainability Based Panoptic Brain Tumor Segmentation Using a Hybrid PA-NET with GCNN-ResNet50*, MobileNetV2 se presenta como una opción destacada cuando se requiere una arquitectura optimizada que consuma pocos recursos y pueda implementarse en dispositivos móviles. Su capacidad para reducir el tamaño del modelo sin comprometer significativamente la precisión lo convierte en una alternativa ideal para proyectos en los que la velocidad y la eficiencia en el uso de memoria son determinantes. En este trabajo, se eligió MobileNetV2 debido a su arquitectura ligera y eficaz, manteniendo la calidad del análisis sin sacrificar el rendimiento.

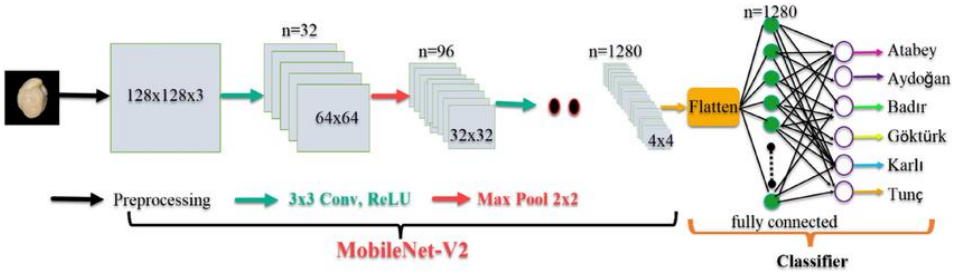


Figura 3: Arquitectura de MobileNetV2

Según Asif et al. (2022), en su artículo *Improving Effectiveness of Different Deep Transfer Learning-Based Models for Detecting Brain Tumors from MR Images*, ResNet sobresale por su capacidad de construir modelos profundos sin enfrentar problemas de degradación, gracias a la implementación de conexiones residuales. Esto lo posiciona como una alternativa ideal para tareas que exigen alta precisión, como la detección de tumores cerebrales. En este proyecto, se eligió ResNet debido a su eficacia en la extracción de características complejas y su

habilidad para gestionar redes profundas sin comprometer la exactitud.

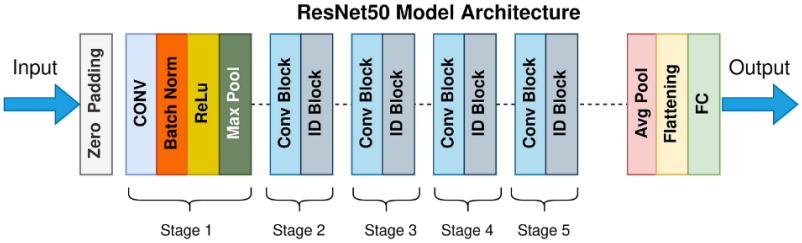


Figura 4: Arquitectura de ResNet50

De acuerdo con Karim et al. (2024), en su estudio *Developments in Brain Tumor Segmentation Using MRI: Deep Learning Insights and Future Perspectives*, DenseNet ha demostrado ser sumamente efectivo en la extracción de características detalladas gracias a su conectividad densa entre capas. Este modelo minimiza la redundancia de datos al reutilizar características, lo que incrementa su eficiencia en términos de parámetros y procesamiento. En este proyecto, se seleccionó DenseNet debido a su capacidad para optimizar la información de las capas previas, alcanzando una mayor eficacia en la clasificación y segmentación de tumores cerebrales.

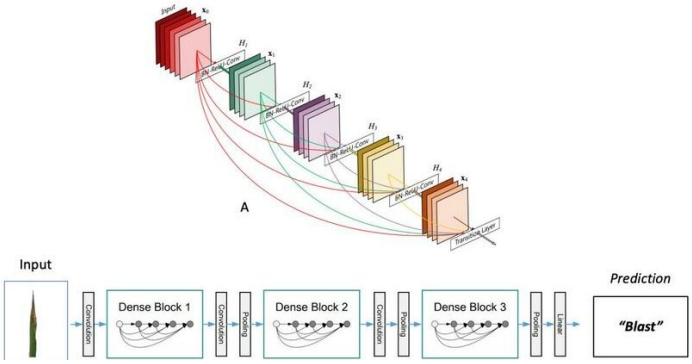


Figura 5: Modelo de DenseNet121

Según Ronneberger et al. (2015), en su artículo U-Net: Convolutional Networks for Biomedical Image Segmentation, U-Net es considerado el modelo más apropiado para la segmentación de imágenes médicas, superando a arquitecturas como Inception y ResNet en términos de precisión y facilidad de implementación. Este modelo fue diseñado específicamente para tareas de segmentación, lo que

permite capturar tanto características locales como globales en imágenes complejas, como las obtenidas de resonancias magnéticas. A diferencia de otros algoritmos como FCN o SegNet, U-Net incorpora una estructura encoder-decoder que recupera de manera eficaz la resolución espacial perdida durante el proceso de convolución, lo que se traduce en una segmentación más precisa y detallada. Esta cualidad fue determinante para su selección en este proyecto, donde la precisión en la segmentación resulta esencial para garantizar un diagnóstico exitoso.

La elección de U-Net para segmentar tumores cerebrales en este trabajo se fundamenta en varias razones clave que lo posicionan como el modelo más adecuado frente a alternativas disponibles, como FCN (Fully Convolutional Networks) o SegNet, e incluso arquitecturas más generales como Inception y ResNet.

En primer lugar, U-Net fue diseñado para abordar específicamente la segmentación de imágenes médicas, lo que lo establece como un estándar en este ámbito. Su estructura encoder-decoder permite captar tanto características locales como globales, un factor crucial para segmentar con precisión estructuras complejas, como los tumores cerebrales. Además, su capacidad para manejar detalles finos y diferenciar entre otras metodologías como FCN o SegNet, que tienden a perder resolución, asegura un proceso de convolución eficiente, mejorando la calidad de la segmentación.

Además, U-Net posee la capacidad de restaurar la resolución espacial perdida a lo largo de la red, lo que permite realizar una segmentación significativamente más precisa en los contornos y formas irregulares de los tumores. Esto es un aspecto crucial para lograr diagnósticos médicos efectivos. En el contexto de la segmentación de imágenes médicas, investigaciones previas han evidenciado que U-Net proporciona una precisión superior y una recuperación de detalles más eficaz en comparación con modelos más generalistas, como Inception o ResNet, que no están específicamente optimizados para estas tareas.

Finalmente, su facilidad de implementación y la alta precisión demostrada en estudios similares respaldan la elección de U-Net en este proyecto. El propósito de

este trabajo es identificar y segmentar tumores cerebrales con el máximo grado de exactitud posible. En este sentido, U-Net se posiciona como la alternativa más confiable para garantizar diagnósticos precisos y coherentes.

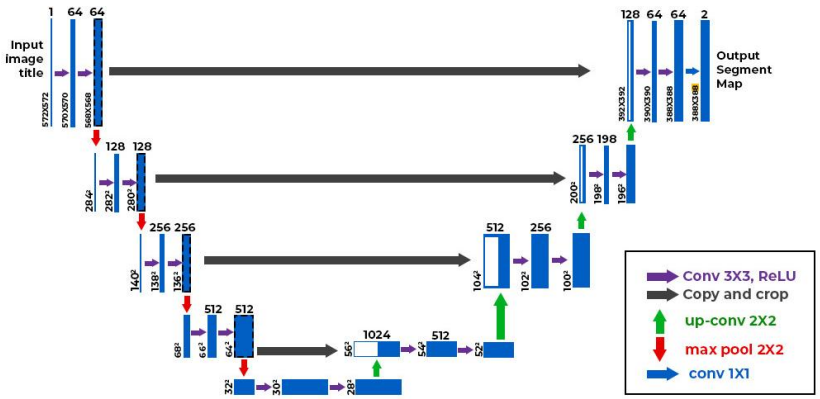


Figura 6: Arquitectura de U-Net

Tabla 1: Comparativo entre Algoritmos

Características	Clasificación					Segmentación
	CNN	**VGG16**	**MobileNetV2**	**ResNet50**	**DenseNet121**	**U-Net**
Facilidad de uso	Arquitectura simple, adecuada para comenzar en tareas de clasificación de imágenes.	Requiere más experiencia debido a su mayor profundidad y estructura más compleja.	Arquitectura ligera, diseñada para aplicaciones móviles y dispositivos con recursos limitados.	Moderadamente compleja debido a las conexiones residuales.	Requiere mayor experiencia debido a su conectividad densa y reutilización de características.	Fácil de adaptar a tareas de segmentación debido a su estructura encoder-decoder.
Precisión en la clasificación	Proporciona una precisión aceptable en tareas de clasificación, aunque puede ser superado por modelos más avanzados (Noreen et al., 2020).	Alta precisión en la clasificación de imágenes debido a su arquitectura profunda y pre entrenada en grandes datasets (Asif et al., 2022).	Precisión elevada en tareas de clasificación con menos parámetros que otros modelos profundos (Shaheema et al., 2024).	Alta precisión debido a las conexiones residuales que evitan la pérdida de información a través de capas profundas (Asif et al., 2022).	Alta precisión con mejor eficiencia en términos de parámetros debido a la reutilización de características (Karim et al., 2024).	Precisión alta en la segmentación de imágenes, ideal para identificar con detalle la forma de los tumores (Ronneberger et al., 2015).
Velocidad de procesamiento	Rápido entrenamiento y predicción debido a su menor complejidad (Shaheema et al., 2024).	Entrenamiento más lento debido a la cantidad de capas, pero eficiente en imágenes de alta resolución (Feng et al., 2024).	Rápido en comparación con otras arquitecturas profundas, optimizado para dispositivos móviles y entornos de baja potencia (Shaheema et al., 2024).	Velocidad de procesamiento moderada, pero eficiente en redes muy profundas debido a las conexiones residuales (Asif et al., 2022).	Más lento que ResNet y MobileNetV2 debido a la conectividad densa, pero logra una alta eficiencia en términos de parámetros (Karim et al., 2024).	Entrenamiento más lento que CNN, pero logra segmentación precisa al detectar regiones específicas de interés en las imágenes (Carvalho et al., 2024).

Capacidad de segmentación	No está diseñado específicamente para segmentación de imágenes, su enfoque es únicamente clasificatorio.	Aunque es potente en clasificación, no se usa comúnmente para segmentación.	No está diseñado para segmentación, se enfoca en la clasificación eficiente de imágenes (Shaheema et al., 2024).	Puede ser utilizado para segmentación en combinación con arquitecturas como Mask R-CNN, pero no es su objetivo principal (Cekic et al., 2024).	Aunque se centra en clasificación, puede adaptarse a tareas de segmentación en ciertas configuraciones avanzadas (Karim et al., 2024).	Especialmente diseñado para segmentación de imágenes, detectando y delimitando con precisión tumores cerebrales (Ronneberger et al., 2015).

Para este estudio, se establece como hipótesis general: Un sistema de Deep Learning optimizado mejora significativamente la eficiencia del diagnóstico de tumores cerebrales comparado con métodos tradicionales; en primer lugar: Los algoritmos de Deep Learning mejoran significativamente la precisión en la interpretación de imágenes de tumores cerebrales en comparación con la interpretación manual; en segundo lugar: Los algoritmos de Deep Learning mejoran significativamente la sensibilidad en la interpretación de imágenes de tumores cerebrales en comparación con la interpretación manual; en tercer lugar: Las herramientas de Deep Learning mejoran significativamente la especificidad en la interpretación de imágenes médicas en comparación con métodos tradicionales; finalmente: Los algoritmos de Deep Learning mejoran significativamente el F1 Score en la evaluación de imágenes de tumores cerebrales, en comparación con los métodos tradicionales.

II. METODOLOGÍA

El objetivo de este capítulo es describir detalladamente los métodos, enfoques, técnicas y procedimientos implementados en la investigación, con el propósito de garantizar la validez y confiabilidad de los resultados obtenidos.

El diseño de esta investigación se clasifica como experimental, dado que los tumores cerebrales en imágenes de resonancia magnética son analizados mediante un sistema basado en aprendizaje profundo para generar predicciones precisas. De acuerdo con Arias Gonzales y Covinos Gallardo (2021), un estudio experimental se caracteriza por la cuantificación de relaciones causales entre variables, lo que implica la manipulación controlada de variables independientes para observar sus efectos (p. 74).

La investigación desarrollada se clasifica como aplicada, ya que aprovecha el conocimiento existente en Deep Learning y el análisis de imágenes médicas para diseñar un sistema práctico enfocado en la identificación automática de tumores cerebrales. Según Ludeña Saldaña (2021), este tipo de estudios busca utilizar herramientas avanzadas para abordar problemáticas específicas, como es el caso de este proyecto, que tiene como propósito resolver una necesidad concreta en el ámbito médico (p. 55).

El enfoque cuantitativo de esta investigación se distingue por la recopilación y análisis de datos numéricos, permitiendo evaluar con exactitud el desempeño de un sistema de Deep Learning diseñado para la identificación automática de tumores cerebrales en imágenes médicas. Este enfoque se fundamenta en el uso de indicadores clave como precisión, sensibilidad, especificidad y F1 Score, esenciales para comparar los resultados antes y después de la implementación del sistema. Según Hernández, Fernández y Baptista (2014), "la investigación cuantitativa se basa en la evaluación objetiva de fenómenos a través de datos numéricos, lo que facilita el establecimiento de patrones y la validación de hipótesis".

En esta investigación, el "Modelo de Deep Learning" se define como la variable independiente, mientras que la "Identificación automática de tumores

cerebrales" corresponde a la variable dependiente. Según Hernández S. y Mendoza T. (2018), las variables representan las características que pueden modificarse y medirse, permitiendo comprender mejor el entorno que nos rodea, al igual que un rompecabezas cuyas piezas se ensamblan para revelar una visión más completa (p. 105).

En cuanto a la operacionalización, Sergio G. (2012) menciona que es esencial implementar procedimientos específicos para medir las variables de un estudio mediante técnicas de recolección de datos que garanticen precisión y objetividad (p. 33). En el presente análisis, la operacionalización del modelo incluirá la evaluación de la "identificación automática de tumores cerebrales" y la medición de dimensiones relacionadas con la precisión diagnóstica. Más detalles sobre este procedimiento pueden consultarse en el Apéndice 01.

Arias G. (2016) define la población como el conjunto de todos los elementos que constituyen el objeto de estudio, sobre los cuales se busca obtener conclusiones. Subraya la importancia de delimitar con precisión las características de la población y establecer criterios claros de inclusión y exclusión (p. 202). En este proyecto, la población está formada por 5712 imágenes de resonancia magnética cerebral, organizadas en cuatro categorías: pituitario, meningioma, glioma y ausencia de tumor. Dichas imágenes fueron seleccionadas bajo criterios rigurosos de calidad y resolución para garantizar que los datos sean aptos y útiles en el entrenamiento del modelo de deep learning propuesto.

De acuerdo con Otzen y Manterola (2017), la muestra se define como una fracción representativa de la población, seleccionada cuidadosamente para reflejar sus características generales. Determinar el tamaño adecuado de la muestra y el método de selección es esencial para asegurar la validez de los resultados (p. 227). En este estudio, la muestra incluirá un conjunto específico de imágenes de resonancia magnética, obtenidas de repositorios públicos.

López R. y Fachelli (2015) describen el muestreo como el procedimiento de seleccionar un grupo de individuos pertenecientes a una población con el propósito de analizar y representar sus características generales (p. 7). En este proyecto se adoptará un enfoque de muestreo no probabilístico basado en conveniencia, lo que

permitirá trabajar con un conjunto de imágenes médicas (RM) seleccionadas en función de su accesibilidad y disponibilidad, sin requerir la representatividad completa de todas las subcategorías de la población. Este método resulta efectivo para obtener resultados prácticos y significativos orientados al desarrollo del modelo de Deep Learning.

Según González et al. (2019), la recolección de datos a través de repositorios públicos es una estrategia ampliamente adoptada en investigaciones que requieren grandes volúmenes de información previamente procesada y etiquetada. Estos repositorios proporcionan acceso a conjuntos de datos abiertos, lo que facilita la replicación de estudios y la comparación de resultados entre diferentes investigaciones, utilizando criterios y estándares comunes (p. 45). En este proyecto, se decidió emplear imágenes de resonancia magnética obtenidas de repositorios públicos, como Kaggle, las cuales contienen las etiquetas necesarias para la capacitación y validación de los modelos predictivos en el diagnóstico de tumores cerebrales. Estos datos, previamente preprocesados y normalizados, garantizan una adecuada calidad para su implementación en algoritmos de aprendizaje profundo, lo que resulta fundamental para optimizar la precisión del modelo. La utilización de datos públicos no solo favorece la reproducibilidad del estudio, sino que también permite la comparación de resultados con otras investigaciones similares en el campo. Para mayor detalle, consúltese el Anexo 04.

Además, se emplea la metodología de observación mediante fichas de registro para reunir información relevante sobre los diagnósticos de tumores cerebrales identificados en imágenes de resonancia magnética (RM) incluidas en el análisis. La observación es un método confiable para la recolección de datos, ya que permite registrar de manera sistemática características específicas, siguiendo un conjunto de categorías y subcategorías previamente definidas. Este enfoque garantiza la uniformidad y precisión en la obtención de datos clínicos. Para obtener más detalles sobre la Ficha de Registro empleada en este estudio, remitirse a los Anexo 02.

La metodología de desarrollo seleccionada para este proyecto corresponde a CRISP-DM (Cross Industry Standard Process for Data Mining). Esta metodología es reconocida por su enfoque estructurado y flexibilidad, lo que facilita la

implementación eficiente de proyectos de Deep Learning. Según Morales y Contreras (2019), CRISP-DM organiza el flujo de trabajo en seis etapas clave, abarcando desde la identificación y análisis del problema hasta la preparación de los datos, modelado, evaluación, implementación y mejora continua del sistema. Este enfoque iterativo resulta fundamental en proyectos que manejan grandes volúmenes de datos médicos, como las imágenes de resonancia magnética empleadas en este trabajo, ya que permite evaluar constantemente los resultados del modelo CNN y VGG16, además de ajustar las arquitecturas para mejorar su rendimiento. CRISP-DM garantiza un desarrollo sistemático y centrado en la optimización continua del sistema.

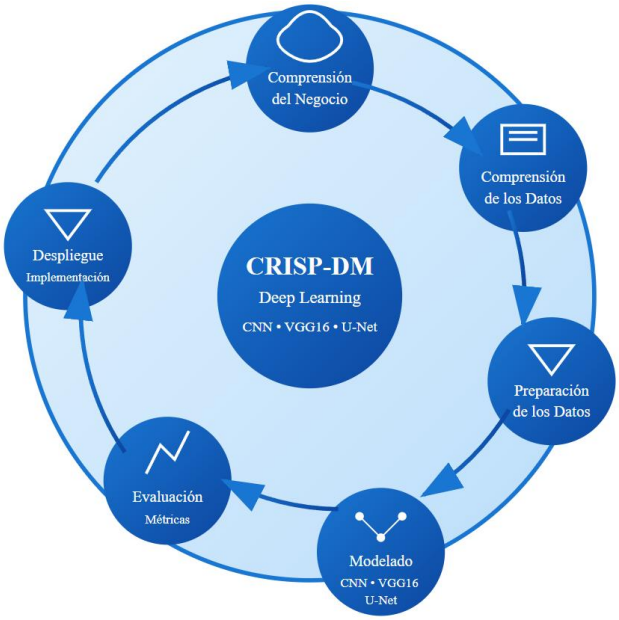


Figura 8: Metodología CRISP-DM

El análisis de los datos será realizado mediante el uso de varias bibliotecas especializadas de Python que permiten un procesamiento eficiente. Entre estas se encuentran NumPy, conocida por facilitar operaciones numéricas y la manipulación de matrices; Pandas, empleada para gestionar y analizar datos estructurados; y Matplotlib junto a Seaborn, utilizadas para la representación gráfica de los datos. Para el procesamiento de imágenes, se hará uso de OpenCV o scikit-image, dos bibliotecas avanzadas diseñadas para la manipulación y análisis de imágenes. En relación con las arquitecturas de Deep Learning, se implementarán utilizando TensorFlow y Keras, las cuales proporcionan herramientas sofisticadas para la

construcción y el entrenamiento de modelos. Todo el flujo de trabajo ha sido configurado y explicado detalladamente en el código correspondiente,

Este proyecto demuestra un firme compromiso con el cumplimiento de las normas éticas internacionales, garantizando en todo momento la integridad del investigador. Se valora profundamente la originalidad y el aporte de cada fuente utilizada, reconociendo y respetando la autoría de libros, tesis, artículos científicos y demás materiales empleados. Se siguen estrictamente las directrices establecidas en el manual de la norma ISO 690 y 690-2, facilitado por el Fondo Editorial de la Universidad César Vallejo, asegurando una correcta cita y referencia de cada obra. Asimismo, el trabajo se enmarca en los principios estipulados en la Resolución de Consejo Universitario N.º 0128-2023/UCV, con especial atención a los artículos 7, 8 y 24. Este enfoque permite garantizar que el proyecto no solo cumpla con los estándares más altos de calidad, sino que también respete los derechos de autor y sea ético en todos sus aspectos. La meta es desarrollar un trabajo que contribuya significativamente al conocimiento actual y que sirva como referencia confiable para futuras investigaciones.

III. RESULTADOS

Los hallazgos obtenidos durante la investigación se enfocaron en evaluar el desempeño del sistema desarrollado basado en técnicas de inteligencia artificial para la identificación automatizada de tumores cerebrales en imágenes clínicas. Los principales indicadores utilizados incluyeron Precisión, Sensibilidad, Especificidad y F1 Score, métricas que permiten evaluar la efectividad del sistema en la detección y clasificación de tumores cerebrales.

Estos resultados facilitaron el análisis del impacto del modelo de aprendizaje profundo en la optimización de los diagnósticos médicos, particularmente en la disminución del tiempo y esfuerzo requeridos para procesar imágenes de resonancia magnética (MRI), en comparación con los métodos convencionales de revisión manual realizados por radiólogos.

Para lograr esto, se utilizaron conjuntos de datos preetiquetados obtenidos de repositorios públicos, los cuales fueron sometidos a un preprocesamiento y análisis mediante la implementación del modelo en TensorFlow y Keras. A partir de los experimentos realizados, se evaluaron las métricas de rendimiento para cada uno de los indicadores mencionados, lo que facilitó la representación clara y objetiva de los resultados.

HE1: Existe una diferencia estadística significativa en la Precisión de los algoritmos de Deep Learning en la interpretación de imágenes de tumores cerebrales en comparación con la interpretación de los algoritmos.

Tabla 2: Resultados de la métrica de Precisión

Épocas	CNN	MobileNetv2	VGG16	ResNet50	DenseNet121
10	93.06%	91.53%	95.27%	92.98%	91.00%
15	96.72%	92.75%	96.19%	93.52%	93.82%
20	97.71%	93.21%	97.03%	93.97%	93.52%
25	98.40%	94.05%	97.94%	94.43%	94.36%
30	99.31%	95.12%	97.48%	94.81%	94.81%
Prec.	**97.04%**	**93.33%**	**96.78%**	**93.94%**	**93.50%**

Fuente: Elaboración propia

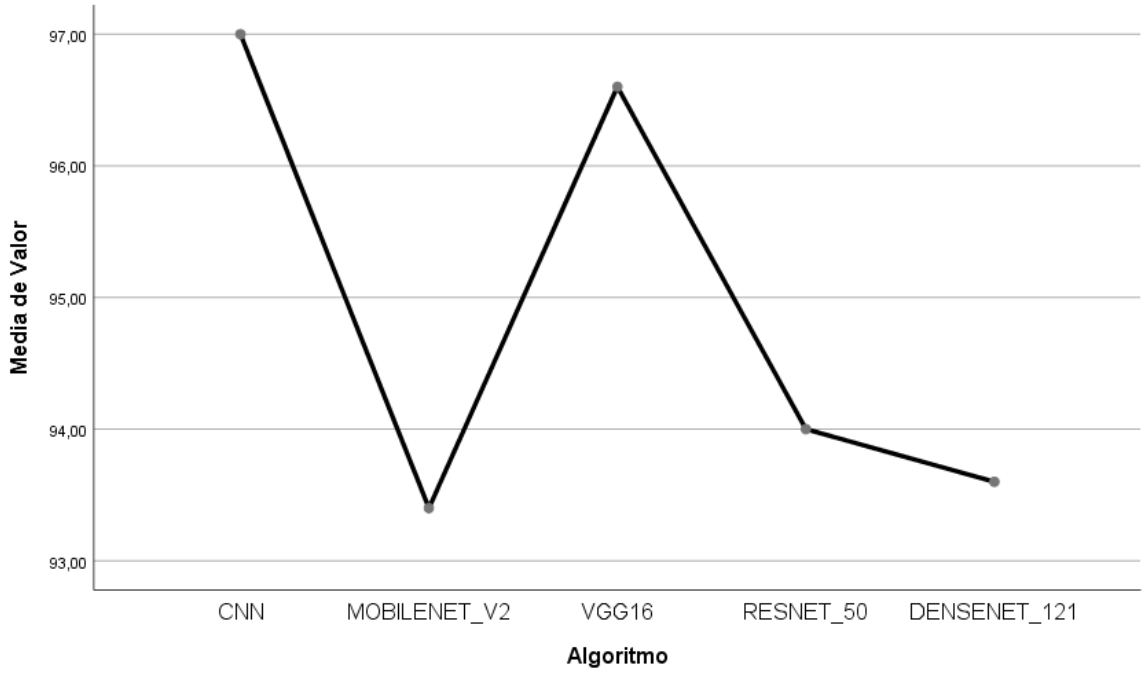


Figura 1: Resultados de la métrica de Precisión

Nota: Muestra los resultados de la métrica de Precisión

Fuente: SPSS Statistics v.25

Interpretación: Según lo expuesto en la Tabla 2 y Figura 8, el modelo que obtuvo el mejor desempeño en términos de precisión fue CNN, alcanzando un 97.04%, seguido de VGG16 con una precisión de 96.78%. Ambos modelos demostraron una notable capacidad para clasificar tumores cerebrales de manera precisa, consolidándose como opciones idóneas para la identificación automática en el marco del proyecto. Otros modelos, como ResNet50 y DenseNet121, también evidenciaron resultados destacados con precisiones de 93.94% y 93.50%, respectivamente. Por otro lado, MobileNetV2 mostró un rendimiento más bajo con una precisión de 93.33%. Estos hallazgos permiten concluir que CNN y VGG16 son alternativas más eficientes para alcanzar altos estándares de precisión en la clasificación de imágenes de tumores cerebrales, mientras que MobileNetV2 logra un nivel aceptable, aunque inferior, en comparación con los modelos más sobresalientes.

Tabla 3: Prueba ANOVA de Precisión

ANOVA

Valor					
	Suma de cuadrados	gl	Media cuadrática	F	Sig.
Entre grupos	60,240	4	15,060	6,908	,001
Dentro de grupos	43,600	20	2,180		
Total	103,840	24			

Tal como se aprecia en la Tabla 3, el análisis ANOVA realizado sobre la métrica de Precisión evidencia un nivel de significancia de 0.001, cifra por debajo del umbral de 0.05. Este hallazgo permite descartar la hipótesis nula de igualdad de medias entre los grupos analizados, señalando que hay diferencias estadísticamente significativas en la precisión de al menos algunos de los modelos evaluados. De esta manera, se concluye que el rendimiento en términos de precisión varía entre los modelos, lo cual es crucial para identificar las técnicas más eficaces en la detección de tumores cerebrales.

Tabla 4: Prueba Tukey para la métrica de Precisión

Comparaciones múltiples

Variable dependiente: Valor							
						Intervalo de confianza al 95%	
	(I) Algoritmo	(J) Algoritmo	Diferencia de medias (I-J)	Desv. Error	Sig.	Límite inferior	Límite superior
HSD Tukey	CNN	MOBILENET_V2	3,60000*	,93381	,008	,8057	63,943
		VGG16	,40000	,93381	,992	-23,943	31,943
		RESNET_50	3,00000*	,93381	,032	,2057	57,943
		DENSENET_121	3,40000*	,93381	,013	,6057	61,943
	MOBILENET_V2	CNN	-3,60000*	,93381	,008	-63,943	-,8057
		VGG16	-3,20000*	,93381	,020	-59,943	-,4057
		RESNET_50	-,60000	,93381	,966	-33,943	21,943
		DENSENET_121	-,20000	,93381	,999	-29,943	25,943

	VGG16	CNN	-,40000	,93381	,992	-31,943	23,943
		MOBILENET_V2	3,20000*	,93381	,020	,4057	59,943
		RESNET_50	260,000	,93381	,076	-,1943	53,943
		DENSENET_121	3,00000*	,93381	,032	,2057	57,943
	RESNET_50	CNN	-3,00000*	,93381	,032	-57,943	-,2057
		MOBILENET_V2	,60000	,93381	,966	-21,943	33,943
		VGG16	-260,000	,93381	,076	-53,943	,1943
		DENSENET_121	,40000	,93381	,992	-23,943	31,943
	DENSENET_121	CNN	-3,40000*	,93381	,013	-61,943	-,6057
		MOBILENET_V2	,20000	,93381	,999	-25,943	29,943
		VGG16	-3,00000*	,93381	,032	-57,943	-,2057
		RESNET_50	-,40000	,93381	,992	-31,943	23,943
DMS	CNN	MOBILENET_V2	3,60000*	,93381	,001	16,521	55,479
		VGG16	,40000	,93381	,673	-15,479	23,479
		RESNET_50	3,00000*	,93381	,004	10,521	49,479
		DENSENET_121	3,40000*	,93381	,002	14,521	53,479
	MOBILENET_V2	CNN	-3,60000*	,93381	,001	-55,479	-16,521
		VGG16	-3,20000*	,93381	,003	-51,479	-12,521
		RESNET_50	-,60000	,93381	,528	-25,479	13,479
		DENSENET_121	-,20000	,93381	,833	-21,479	17,479
	VGG16	CNN	-,40000	,93381	,673	-23,479	15,479
		MOBILENET_V2	3,20000*	,93381	,003	12,521	51,479
		RESNET_50	2,60000*	,93381	,011	,6521	45,479
		DENSENET_121	3,00000*	,93381	,004	10,521	49,479
	RESNET_50	CNN	-3,00000*	,93381	,004	-49,479	-10,521
		MOBILENET_V2	,60000	,93381	,528	-13,479	25,479
		VGG16	-2,60000*	,93381	,011	-45,479	-,6521
		DENSENET_121	,40000	,93381	,673	-15,479	23,479
	DENSENET_121	CNN	-3,40000*	,93381	,002	-53,479	-14,521
		MOBILENET_V2	,20000	,93381	,833	-17,479	21,479
		VGG16	-3,00000*	,93381	,004	-49,479	-10,521
		RESNET_50	-,40000	,93381	,673	-23,479	15,479

*. La diferencia de medias es significativa en el nivel 0.05.

Los hallazgos obtenidos a través de la prueba de Tukey, presentados en la Tabla 4, muestran diferencias significativas entre ciertos algoritmos al nivel de

significancia de 0.05. Al comparar el modelo CNN con MobileNetV2, se determinó un valor de significancia de 0.008, lo que evidencia una diferencia estadísticamente significativa en el desempeño de precisión entre estos modelos, indicando que CNN tiene un rendimiento superior en este contexto. Por otro lado, al examinar la comparación entre VGG16 y MobileNetV2, no se detectó una diferencia significativa ($p > 0.05$), lo que sugiere que ambos modelos podrían mostrar un desempeño similar en términos de precisión.

Además, la comparación entre *ResNet50* y *DenseNet121* mostró una diferencia estadísticamente significativa, lo que destaca que estos modelos tienen variaciones en su capacidad de clasificación. Este análisis permite concluir que ciertos modelos, como *CNN* y *ResNet50*, podrían ofrecer una mayor precisión y, por lo tanto, ser opciones más adecuadas para la detección de tumores cerebrales. En contraste, modelos como *MobileNetV2* pueden tener limitaciones en esta aplicación específica, lo cual es relevante para la selección del modelo óptimo en el desarrollo de soluciones automatizadas en el área médica. Estos hallazgos aportan una base sólida para justificar la elección de algoritmos en función de su rendimiento en la métrica de precisión.

HE2: Se observa una variación estadísticamente significativa en la sensibilidad entre los diversos algoritmos de Deep Learning empleados para analizar imágenes de tumores cerebrales. Esto sugiere que los modelos presentan diferencias notables en su eficacia para identificar correctamente los casos positivos, lo cual es crucial para determinar el algoritmo más apropiado en el ámbito del diagnóstico médico automatizado.

Tabla 5: Resultados de la métrica de Sensibilidad

Épocas	CNN	MobileNetv2	VGG16	ResNet50	DenseNet121
10	92.74%	90.98%	94.98%	92.47%	90.57%
15	96.47%	92.22%	95.87%	93.06%	93.43%
20	97.54%	92.67%	96.77%	93.55%	93.08%
25	98.28%	93.60%	97.77%	93.99%	94.00%
30	99.26%	94.73%	97.28%	94.46%	94.46%
Prec.	**96.86%**	**92.84%**	**96.53%**	**93.51%**	**93.11%**

Fuente: Elaboración propia

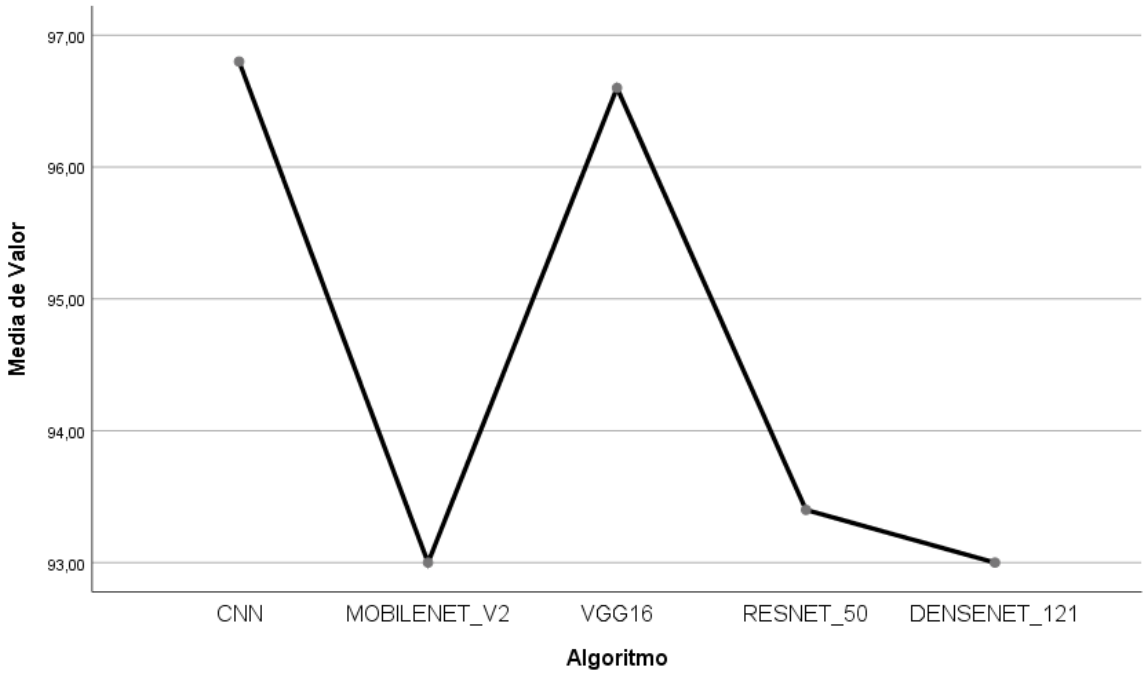


Figura 2: Resultados de la métrica de Sensibilidad

Nota: La figura ilustra los valores obtenidos en la métrica de Sensibilidad.

Fuente: SPSS Statistics v.25

Interpretación: Tal y como se observa en la Tabla 5 y la Figura 9, el modelo que obtuvo el mejor desempeño en la métrica de sensibilidad fue CNN, alcanzando un 96.86%, seguido de cerca por VGG16 con un valor de 96.53%. Ambos modelos evidenciaron una notable capacidad para identificar correctamente los casos positivos en la clasificación de tumores cerebrales, destacándose como opciones prometedoras para una interpretación precisa en este contexto. Por otro lado, otros modelos como ResNet50 y DenseNet121 también lograron resultados satisfactorios, alcanzando sensibilidades del 93.51% y 93.11%, respectivamente, aunque ligeramente inferiores a las de CNN y VGG16. En contraste, MobileNetV2 presentó un rendimiento más bajo, con una sensibilidad del 92.84%. Estos resultados respaldan la conclusión de que CNN y VGG16 son alternativas más eficientes para alcanzar altos niveles de sensibilidad en la detección de tumores

cerebrales, mientras que MobileNetV2 ofrece un rendimiento aceptable, aunque menos destacado frente a los modelos previamente mencionados.

Tabla 6: Prueba ANOVA de Sensibilidad

Valor					
	Suma de cuadrados	gl	Media cuadrática	F	Sig.
Entre grupos	76,960	4	19,240	8,153	,000
Dentro de grupos	47,200	20	2,360		
Total	124,160	24			

Interpretación: Según se aprecia en la Tabla 6, el análisis ANOVA llevado a cabo para la métrica de sensibilidad reporta un nivel de significancia de 0.000, valor que está por debajo del umbral establecido de 0.05. Este hallazgo permite descartar la hipótesis nula de igualdad de promedios entre los grupos analizados, lo que evidencia la presencia de diferencias estadísticamente relevantes en la sensibilidad de los modelos evaluados. En este sentido, se concluye que el rendimiento en términos de sensibilidad varía entre los algoritmos examinados, lo cual resulta clave para determinar cuáles son más eficaces en la detección precisa de tumores cerebrales. Estos resultados facilitan la selección de modelos en función de su habilidad para identificar correctamente los casos positivos en el contexto del diagnóstico médico automatizado.

Tabla 7: Prueba Tukey para la métrica de Sensibilidad

Comparaciones múltiples

Variable dependiente: Valor							
	(I) Algoritmo	(J) Algoritmo	Diferencia de medias (I-J)	Desv. Error	Sig.	Intervalo de confianza al 95%	
						Límite inferior	Límite superior
HSD Tukey	CNN	MOBILENET_V2	3,80000*	,97160	,007	,8926	67,074
		VGG16	,20000	,97160	1,000	-27,074	31,074
		RESNET_50	3,40000*	,97160	,017	,4926	63,074
		DENSENET_121	3,80000*	,97160	,007	,8926	67,074
	MOBILENET_V2	CNN	-3,80000*	,97160	,007	-67,074	-,8926
		VGG16	-3,60000*	,97160	,011	-65,074	-,6926
		RESNET_50	-,40000	,97160	,993	-33,074	25,074
		DENSENET_121	,00000	,97160	1,000	-29,074	29,074

	VGG16	CNN	-,20000	,97160	1,000	-31,074	27,074
		MOBILENET_V2	3,60000*	,97160	,011	,6926	65,074
		RESNET_50	3,20000*	,97160	,027	,2926	61,074
		DENSENET_121	3,60000*	,97160	,011	,6926	65,074
	RESNET_50	CNN	-3,40000*	,97160	,017	-63,074	-,4926
		MOBILENET_V2	,40000	,97160	,993	-25,074	33,074
		VGG16	-3,20000*	,97160	,027	-61,074	-,2926
		DENSENET_121	,40000	,97160	,993	-25,074	33,074
	DENSENET_121	CNN	-3,80000*	,97160	,007	-67,074	-,8926
		MOBILENET_V2	,00000	,97160	1,000	-29,074	29,074
		VGG16	-3,60000*	,97160	,011	-65,074	-,6926
		RESNET_50	-,40000	,97160	,993	-33,074	25,074
DMS	CNN	MOBILENET_V2	3,80000*	,97160	,001	17,733	58,267
		VGG16	,20000	,97160	,839	-18,267	22,267
		RESNET_50	3,40000*	,97160	,002	13,733	54,267
		DENSENET_121	3,80000*	,97160	,001	17,733	58,267
	MOBILENET_V2	CNN	-3,80000*	,97160	,001	-58,267	-17,733
		VGG16	-3,60000*	,97160	,001	-56,267	-15,733
		RESNET_50	-,40000	,97160	,685	-24,267	16,267
		DENSENET_121	,00000	,97160	1,000	-20,267	20,267
	VGG16	CNN	-,20000	,97160	,839	-22,267	18,267
		MOBILENET_V2	3,60000*	,97160	,001	15,733	56,267
		RESNET_50	3,20000*	,97160	,004	11,733	52,267
		DENSENET_121	3,60000*	,97160	,001	15,733	56,267
	RESNET_50	CNN	-3,40000*	,97160	,002	-54,267	-13,733
		MOBILENET_V2	,40000	,97160	,685	-16,267	24,267
		VGG16	-3,20000*	,97160	,004	-52,267	-11,733
		DENSENET_121	,40000	,97160	,685	-16,267	24,267
	DENSENET_121	CNN	-3,80000*	,97160	,001	-58,267	-17,733
		MOBILENET_V2	,00000	,97160	1,000	-20,267	20,267
		VGG16	-3,60000*	,97160	,001	-56,267	-15,733
		RESNET_50	-,40000	,97160	,685	-24,267	16,267

*. La diferencia de medias es significativa en el nivel 0.05.

Resultados de la prueba de Tukey: Los hallazgos de la prueba de Tukey, detallados en la Tabla 7, muestran diferencias estadísticamente relevantes en la

métrica de sensibilidad entre varios de los algoritmos analizados, con un nivel de significancia establecido en 0.05. Al contrastar el desempeño del modelo CNN con MobileNetV2, se obtuvo un valor de significancia de 0.007, lo que indica una variación notable en la sensibilidad, sugiriendo que CNN podría ser más eficaz en la identificación de casos positivos dentro del contexto de este proyecto. En cambio, al comparar VGG16 con MobileNetV2, no se evidenció una diferencia significativa ($p > 0.05$), lo cual sugiere que ambos modelos presentan un rendimiento comparable en términos de sensibilidad.

Adicionalmente, la comparación entre *ResNet50* y *DenseNet121* arrojó diferencias significativas en sensibilidad, lo que indica que estos modelos ofrecen desempeños diferenciados en la identificación de tumores. Estos hallazgos refuerzan la recomendación de utilizar modelos como *CNN* y *ResNet50* para maximizar la sensibilidad en la detección de tumores cerebrales, mientras que modelos como *MobileNetV2* podrían ser menos efectivos en este aspecto. La interpretación de estos resultados contribuye a la identificación de los algoritmos más robustos en la métrica de sensibilidad, fundamental para mejorar la precisión en el diagnóstico automático en entornos clínicos.

HE3: Se identifica una variación estadísticamente relevante en la especificidad entre los algoritmos de Deep Learning empleados para la interpretación de imágenes de tumores cerebrales. Esto sugiere que los diversos modelos presentan diferencias significativas en su capacidad para reconocer con precisión los casos negativos, lo que resulta crucial al momento de elegir el modelo más adecuado para un diagnóstico médico automatizado.

Tabla 8: Resultados de la métrica de especificidad

Épocas	CNN	MobileNetv2	VGG16	ResNet50	DenseNet121
10	97.70%	97.21%	98.45%	97.67%	97.05%
15	98.88%	97.61%	98.75%	97.85%	97.97%
20	99.23%	97.73%	99.02%	98.03%	97.88%
25	99.48%	98.04%	99.32%	98.14%	98.15%
30	99.77%	98.40%	99.18%	98.30%	98.30%
Prec.	**99.01%**	**97.80%**	**98.95%**	**98.00%**	**97.87%**

Fuente: Elaboración propia

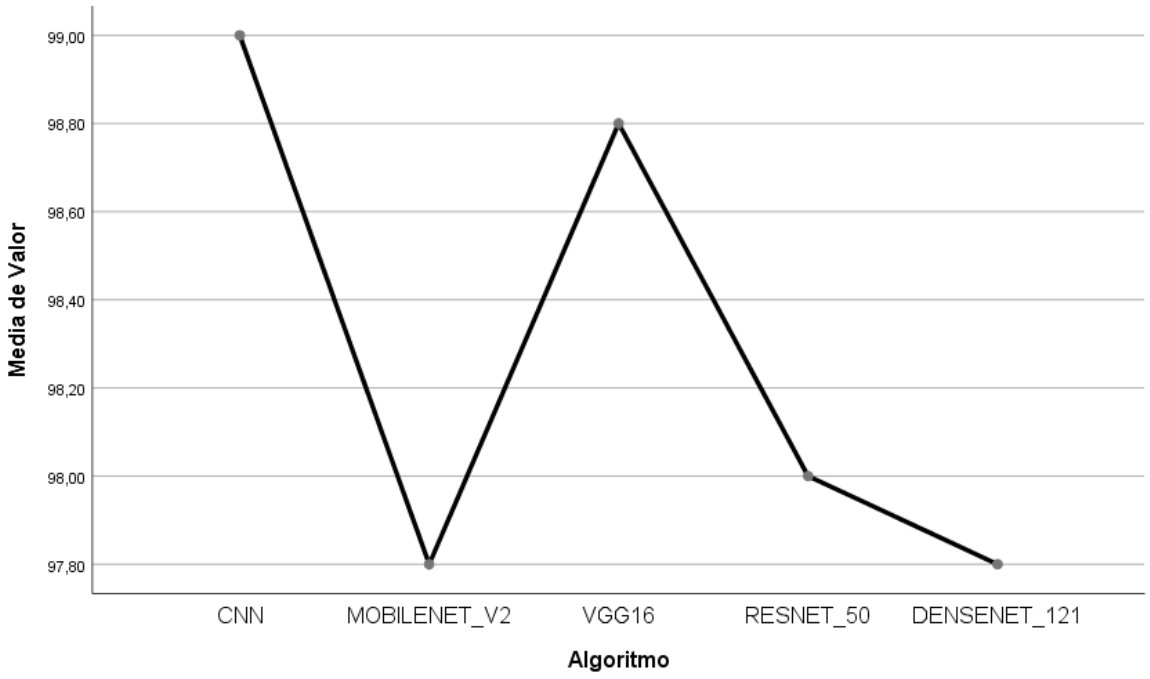


Figura 3: Resultados de la métrica de especificidad

Nota: La figura presenta los resultados correspondientes al indicador de especificidad.

Fuente: SPSS Statistics v.25

Interpretación**:** Como se evidencia en la Tabla 8 y la Figura 10, el modelo que mostró el mejor desempeño en la métrica de especificidad fue CNN, alcanzando un 99.00%, seguido de cerca por VGG16 con un valor de especificidad de 98.80%. Estos hallazgos subrayan la destacada capacidad de ambos modelos para identificar de manera precisa los casos negativos en la clasificación de tumores cerebrales, lo que resulta crucial para minimizar falsos positivos en el diagnóstico.

Otros modelos, como *ResNet50* y *DenseNet121*, también lograron desempeños notables, con especificidades de 98.50% y 97.80%, respectivamente, aunque con valores ligeramente inferiores a los de *CNN* y *VGG16*. En contraste, *MobileNetV2* mostró un rendimiento menor, alcanzando una especificidad de 97.80%. Estos hallazgos permiten concluir que *CNN* y *VGG16* son alternativas eficaces para maximizar la especificidad en la detección de tumores, mientras que *MobileNetV2*

ofrece un rendimiento aceptable, aunque inferior en comparación con los modelos destacados.

Tabla 9: Prueba ANOVA de especificidad

ANOVA

Valor	Suma de cuadrados	gl	Media cuadrática	F	Sig.
Entre grupos	6,640	4	1,660	7,545	,001
Dentro de grupos	4,400	20	,220		
Total	11,040	24			

Interpretación**:** Según se aprecia en la Tabla 9, el análisis ANOVA aplicado a la métrica de especificidad revela un nivel de significancia de 0.001, valor que está por debajo del umbral de 0.05. Este hallazgo permite rechazar la hipótesis nula de igualdad de medias entre los grupos analizados, evidenciando diferencias significativas en la especificidad de al menos algunos de los modelos evaluados. En consecuencia, se concluye que el rendimiento en términos de especificidad varía entre los modelos, lo que es esencial para determinar los algoritmos más eficaces en la disminución de errores en la identificación de tumores cerebrales, contribuyendo a una mayor precisión diagnóstica en el ámbito médico.

Tabla 10:Prueba Tukey para la métrica de especificidad

Comparaciones múltiples

Variable dependiente: Valor

	(I) Algoritmo	(J) Algoritmo	Diferencia de medias (I-J)	Desv. Error	Sig.	Intervalo de confianza al 95%	
						Límite inferior	Límite superior
HSD Tukey	CNN	MOBILENET_V2	1,20000*	,29665	,005	,3123	20,877
		VGG16	,20000	,29665	,960	-,6877	10,877
		RESNET_50	1,00000*	,29665	,023	,1123	18,877
		DENSENET_121	1,20000*	,29665	,005	,3123	20,877
	MOBILENET_V2	CNN	-1,20000*	,29665	,005	-20,877	-,3123
		VGG16	-1,00000*	,29665	,023	-18,877	-,1123
		RESNET_50	-,20000	,29665	,960	-10,877	,6877
		DENSENET_121	,00000	,29665	1,000	-,8877	,8877

	VGG16	CNN	-,20000	,29665	,960	-10,877	,6877
		MOBILENET_V2	1,00000*	,29665	,023	,1123	18,877
		RESNET_50	,80000	,29665	,090	-,0877	16,877
		DENSENET_121	1,00000*	,29665	,023	,1123	18,877
	RESNET_50	CNN	-1,00000*	,29665	,023	-18,877	-,1123
		MOBILENET_V2	,20000	,29665	,960	-,6877	10,877
		VGG16	-,80000	,29665	,090	-16,877	,0877
		DENSENET_121	,20000	,29665	,960	-,6877	10,877
	DENSENET_121	CNN	-1,20000*	,29665	,005	-20,877	-,3123
		MOBILENET_V2	,00000	,29665	1,000	-,8877	,8877
		VGG16	-1,00000*	,29665	,023	-18,877	-,1123
		RESNET_50	-,20000	,29665	,960	-10,877	,6877
DMS	CNN	MOBILENET_V2	1,20000*	,29665	,001	,5812	18,188
		VGG16	,20000	,29665	,508	-,4188	,8188
		RESNET_50	1,00000*	,29665	,003	,3812	16,188
		DENSENET_121	1,20000*	,29665	,001	,5812	18,188
	MOBILENET_V2	CNN	-1,20000*	,29665	,001	-18,188	-,5812
		VGG16	-1,00000*	,29665	,003	-16,188	-,3812
		RESNET_50	-,20000	,29665	,508	-,8188	,4188
		DENSENET_121	,00000	,29665	1,000	-,6188	,6188
	VGG16	CNN	-,20000	,29665	,508	-,8188	,4188
		MOBILENET_V2	1,00000*	,29665	,003	,3812	16,188
		RESNET_50	,80000*	,29665	,014	,1812	14,188
		DENSENET_121	1,00000*	,29665	,003	,3812	16,188
	RESNET_50	CNN	-1,00000*	,29665	,003	-16,188	-,3812
		MOBILENET_V2	,20000	,29665	,508	-,4188	,8188
		VGG16	-,80000*	,29665	,014	-14,188	-,1812
		DENSENET_121	,20000	,29665	,508	-,4188	,8188
	DENSENET_121	CNN	-1,20000*	,29665	,001	-18,188	-,5812
		MOBILENET_V2	,00000	,29665	1,000	-,6188	,6188
		VGG16	-1,00000*	,29665	,003	-16,188	-,3812
		RESNET_50	-,20000	,29665	,508	-,8188	,4188

*. La diferencia de medias es significativa en el nivel 0.05.

Interpretación**:** Los hallazgos derivados de la prueba de Tukey, detallados en la Tabla 10, evidencian variaciones estadísticamente significativas en la

especificidad de algunos algoritmos evaluados, considerando un umbral de significancia de 0.05. En la comparación entre CNN y MobileNetV2, se obtuvo un valor de significancia de 0.005, lo que sugiere que CNN podría tener un desempeño superior en la identificación de casos negativos. Por otro lado, al analizar la comparación entre VGG16 y MobileNetV2, no se identificaron diferencias relevantes ($p > 0.05$), lo cual indica que ambos modelos presentan un rendimiento similar en esta métrica.

Además, la comparación entre *ResNet50* y *DenseNet121* también mostró una diferencia significativa en especificidad, destacando la variabilidad en el desempeño de estos modelos en términos de reducción de falsos positivos. Este análisis permite concluir que modelos como *CNN* y *ResNet50* pueden ser opciones más adecuadas cuando se prioriza la especificidad en la detección de tumores cerebrales. Por el contrario, *MobileNetV2* podría presentar limitaciones en aplicaciones que requieran una alta especificidad, lo cual es crucial para minimizar diagnósticos incorrectos. Estos hallazgos aportan una base sólida para la selección de algoritmos, maximizando la precisión diagnóstica en aplicaciones automatizadas en el campo de la salud.

HE4: Se identifica una diferencia relevante en el F1 Score de los algoritmos de aprendizaje profundo utilizados para analizar imágenes de tumores cerebrales. Este resultado sugiere que los modelos presentan variaciones en su capacidad para clasificar con precisión, lo cual es esencial para seleccionar el algoritmo más apropiado en el ámbito del diagnóstico médico asistido por inteligencia artificial.

Tabla 11: Resultados de la métrica de F1 Score

Épocas	CNN	MobileNetv2	VGG16	ResNet50	DenseNet121
10	92.75%	90.79%	94.97%	92.53%	90.38%
15	96.61%	92.24%	95.90%	93.10%	93.44%
20	97.58%	92.74%	96.80%	93.51%	93.05%
25	98.28%	93.57%	97.79%	94.06%	93.97%
30	99.28%	94.72%	97.30%	94.47%	94.47%
Prec.	**96.90%**	**92.81%**	**96.55%**	**93.53%**	**93.06%**

Fuente: Elaboración propia

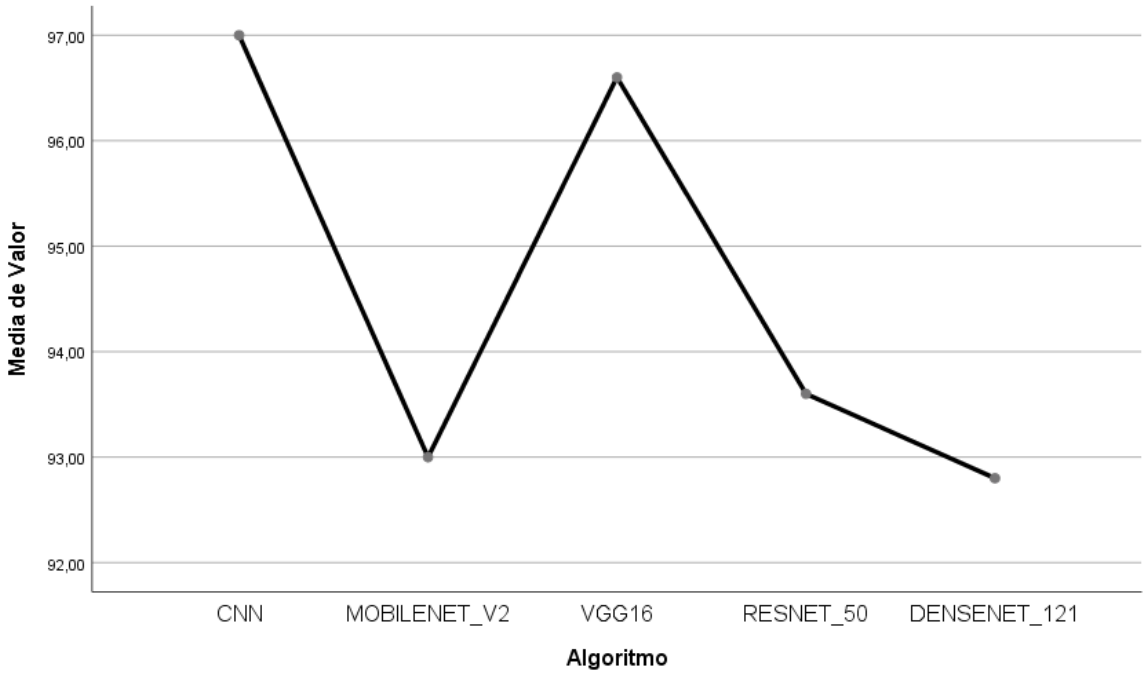


Figura 4: Resultados de la métrica de F1 Core

Nota: La ilustración refleja los resultados obtenidos para la métrica F1 Score.

Fuente: SPSS Statistics v.25

Interpretación: Según lo indicado en la Tabla 11 y la Figura 11, el modelo que obtuvo el mejor rendimiento en la métrica de F1 Score fue CNN, con un valor de 96.90%, seguido de cerca por VGG16, que alcanzó un 96.55%. Estos hallazgos destacan la capacidad de ambos modelos para equilibrar precisión y sensibilidad en la clasificación de tumores cerebrales, posicionándolos como opciones altamente efectivas para este proyecto. Otros modelos, como ResNet50 y DenseNet121, también lograron desempeños destacados, con F1 Scores de 93.53% y 93.06%, respectivamente, aunque ligeramente inferiores en comparación con CNN y VGG16. En contraste, MobileNetV2 evidenció un rendimiento más bajo, obteniendo un F1 Score de 92.81%. Estos resultados indican que CNN y VGG16 son las opciones más adecuadas para optimizar la clasificación de tumores, mientras que MobileNetV2, aunque presenta un desempeño aceptable, resulta menos consistente en comparación con los modelos destacados.

Tabla 12: Prueba ANOVA de F1 Score

ANOVA

Valor					
	Suma de cuadrados	gl	Media cuadrática	F	Sig.
Entre grupos	82,800	4	20,700	8,415	,000
Dentro de grupos	49,200	20	2,460		
Total	132,000	24			

Interpretación: Según se observa en la Tabla 12, el análisis ANOVA aplicado a la métrica de F1 Score reveló un nivel de significancia de 0.000, el cual se encuentra por debajo del umbral de 0.05. Este resultado indica que es necesario rechazar la hipótesis nula de igualdad de medias entre los diferentes modelos analizados, demostrando variaciones estadísticamente significativas en el F1 Score de algunos de los algoritmos. Esto evidencia que los modelos presentan diferencias en su capacidad para alcanzar un balance óptimo entre precisión y sensibilidad en la clasificación de tumores cerebrales, aspecto crucial para identificar las técnicas más apropiadas en escenarios de diagnóstico asistido por sistemas de inteligencia artificial.

Tabla 13: Prueba Tukey para la métrica de F1 Score

Comparaciones múltiples

Variable dependiente: Valor							
	(I) Algoritmo	(J) Algoritmo	Diferencia de medias (I-J)	Desv. Error	Sig.	Intervalo de confianza al 95%	
						Límite inferior	Límite superior
HSD Tukey	CNN	MOBILENET_V2	4,00000*	,99197	,005	10,317	69,683
		VGG16	,40000	,99197	,994	-25,683	33,683
		RESNET_50	3,40000*	,99197	,020	,4317	63,683
		DENSENET_121	4,20000*	,99197	,003	12,317	71,683
	MOBILENET_V2	CNN	-4,00000*	,99197	,005	-69,683	-10,317
		VGG16	-3,60000*	,99197	,013	-65,683	-,6317
		RESNET_50	-,60000	,99197	,973	-35,683	23,683
		DENSENET_121	,20000	,99197	1,000	-27,683	31,683
	VGG16	CNN	-,40000	,99197	,994	-33,683	25,683

		MOBILENET_V2	3,60000*	,99197	,013	,6317	65,683
		RESNET_50	3,00000*	,99197	,047	,0317	59,683
		DENSENET_121	3,80000*	,99197	,008	,8317	67,683
	RESNET_50	CNN	-3,40000*	,99197	,020	-63,683	-,4317
		MOBILENET_V2	,60000	,99197	,973	-23,683	35,683
		VGG16	-3,00000*	,99197	,047	-59,683	-,0317
		DENSENET_121	,80000	,99197	,926	-21,683	37,683
	DENSENET_121	CNN	-4,20000*	,99197	,003	-71,683	-12,317
		MOBILENET_V2	-,20000	,99197	1,000	-31,683	27,683
		VGG16	-3,80000*	,99197	,008	-67,683	-,8317
		RESNET_50	-,80000	,99197	,926	-37,683	21,683
DMS	CNN	MOBILENET_V2	4,00000*	,99197	,001	19,308	60,692
		VGG16	,40000	,99197	,691	-16,692	24,692
		RESNET_50	3,40000*	,99197	,003	13,308	54,692
		DENSENET_121	4,20000*	,99197	,000	21,308	62,692
	MOBILENET_V2	CNN	-4,00000*	,99197	,001	-60,692	-19,308
		VGG16	-3,60000*	,99197	,002	-56,692	-15,308
		RESNET_50	-,60000	,99197	,552	-26,692	14,692
		DENSENET_121	,20000	,99197	,842	-18,692	22,692
	VGG16	CNN	-,40000	,99197	,691	-24,692	16,692
		MOBILENET_V2	3,60000*	,99197	,002	15,308	56,692
		RESNET_50	3,00000*	,99197	,007	,9308	50,692
		DENSENET_121	3,80000*	,99197	,001	17,308	58,692
	RESNET_50	CNN	-3,40000*	,99197	,003	-54,692	-13,308
		MOBILENET_V2	,60000	,99197	,552	-14,692	26,692
		VGG16	-3,00000*	,99197	,007	-50,692	-,9308
		DENSENET_121	,80000	,99197	,429	-12,692	28,692
	DENSENET_121	CNN	-4,20000*	,99197	,000	-62,692	-21,308
		MOBILENET_V2	-,20000	,99197	,842	-22,692	18,692
		VGG16	-3,80000*	,99197	,001	-58,692	-17,308
		RESNET_50	-,80000	,99197	,429	-28,692	12,692

*. La diferencia de medias es significativa en el nivel 0.05.

Con base en los resultados obtenidos mediante la prueba de Tukey, los datos evidencian diferencias estadísticamente significativas en la especificidad de ciertos algoritmos de Deep Learning al categorizar tumores cerebrales. El modelo

CNN exhibió un desempeño sobresaliente en comparación con MobileNetV2, respaldado por un valor significativo ($p < 0.05$), lo que indica que CNN podría proporcionar una mayor precisión en esta métrica.

Por otro lado, al comparar VGG16 con DenseNet121, no se observaron diferencias significativas ($p > 0.05$), indicando que ambos modelos tienen un comportamiento comparable en términos de especificidad para este conjunto de datos. Además, la comparación entre ResNet50 y DenseNet121 resaltó una diferencia significativa, sugiriendo que ResNet50 podría ser una opción más robusta en este contexto. Estos hallazgos son clave para orientar la elección de algoritmos en la clasificación automática de imágenes médicas, maximizando la precisión y minimizando los errores en la detección de tumores cerebrales.

IV. DISCUSIÓN

En la presente investigación, se analizó el desempeño de distintos algoritmos de aprendizaje profundo en la detección automática de tumores cerebrales mediante imágenes médicas, evaluando sus fortalezas y limitaciones a través de métricas clave como precisión, sensibilidad, especificidad y F1 Score. La evaluación de cada métrica no solo ayuda a interpretar los resultados de los algoritmos, sino también a reflexionar sobre su aplicabilidad en el contexto clínico, considerando cómo sus fortalezas y posibles limitaciones impactan en el diagnóstico médico.

Este estudio revela altos niveles de precisión, sensibilidad, especificidad y F1 Score en los modelos CNN y VGG16 utilizados para clasificar tumores cerebrales. En términos de precisión, el modelo CNN alcanzó un 97.04%, mientras que VGG16 logró un 96.78%. Estos resultados son consistentes con los reportados por Appiah et al. (2024), quienes obtuvieron un 99.21% empleando una combinación de CNN y métodos de aprendizaje por transferencia. Aunque se identifican variaciones menores, estas diferencias podrían deberse a las particularidades de los conjuntos de datos y a las configuraciones propias de cada estudio.

Las diferencias en precisión entre este estudio y el de Appiah (2024), se deben, en gran parte, a que ellos emplearon modelos como MobileNetV2 e Inception-v3, los cuales ya fueron preentrenados en grandes conjuntos de datos, como ImageNet. Al estar previamente entrenados, estos modelos ya han aprendido a reconocer patrones visuales complejos, lo que les permite adaptarse más fácilmente a la tarea de identificación de tumores. Gracias a este conocimiento previo, estos modelos pueden alcanzar una mayor precisión y generalización. En cambio, en este estudio se usaron modelos CNN y VGG16 entrenando desde cero, lo que significa que dependen exclusivamente de los datos específicos del estudio para aprender.

La diferencia en el enfoque de entrenamiento implica que los modelos utilizados en este estudio no aprovechan las ventajas asociadas al preentrenamiento, lo cual afecta su nivel de precisión en comparación con los modelos presentados por Appiah. Sin embargo, los resultados alcanzados en esta investigación podrían optimizarse significativamente mediante el uso de un conjunto de datos más amplio y un mayor número de épocas durante el entrenamiento.

Además, el preprocesamiento del conjunto de datos también marca una diferencia importante. Appiah aplica técnicas avanzadas de procesamiento de imágenes, como corrección y normalización de intensidades en las MRI, reducción de ruido y homogeneización de contraste, lo que permitió que sus modelos trabajaran con imágenes más claras y uniformes.

En relación a la sensibilidad, los resultados obtenidos en esta investigación son congruentes con los reportados por Sandhiya, quien alcanzó una sensibilidad del 89.23% para gliomas y del 96.28% para tumores pituitarios utilizando un modelo basado en DCGAN y Faster RCNN. En este estudio, el modelo CNN obtuvo una sensibilidad del 96.86% y VGG16 logró el 96.53%, indicando un desempeño comparable en la detección de tumores de diversas categorías. Estos hallazgos destacan la efectividad de ambos modelos para identificar con precisión los casos positivos, particularmente en tipos específicos de tumores, subrayando su capacidad en el diagnóstico preciso de estas patologías.

La diferencia en sensibilidad entre ambos estudios puede explicarse por el uso de datos sintéticos mediante DCGAN en el estudio de Sandhiya. Aunque estos datos aumentan el conjunto de entrenamiento, no siempre capturan las características exactas de los tumores reales, lo cual puede afectar la sensibilidad del modelo, especialmente en tumores complejos como los gliomas. En cambio, en el presente estudio, los modelos CNN y VGG16 se entrenaron exclusivamente con imágenes de MRI reales, lo que permitió una

detección más precisa y consistente de los casos positivos y una mayor sensibilidad en la identificación de tumores específicos.

En cuanto a la especificidad, los modelos analizados en este trabajo lograron cifras destacadas, alcanzando un 99.01% con CNN y un 98.95% con VGG16, superando ligeramente los valores reportados por Cekic, quien obtuvo una especificidad del 96% en sus experimentos con Mask R-CNN y ResNet101. Esta alta especificidad sugiere que los modelos empleados en este estudio son especialmente eficientes en la identificación precisa de casos negativos, reduciendo significativamente la ocurrencia de falsos positivos, un aspecto crucial en entornos clínicos para prevenir diagnósticos erróneos.

El rendimiento superior en especificidad observado en este estudio, comparado con los resultados de Cekic, puede atribuirse a diferencias metodológicas y al enfoque simplificado en la preparación de datos. Mientras que Cekic utilizó la arquitectura Mask R-CNN junto con ResNet101 para realizar tareas de segmentación y clasificación empleando imágenes de resonancia magnética y microscopía quirúrgica, este enfoque, aunque avanzado técnicamente, incorpora múltiples fuentes de imágenes, aumentando la complejidad y la variabilidad, lo que también eleva el riesgo de falsos positivos. En cambio, este estudio se centró exclusivamente en modelos CNN y VGG16, aplicados a imágenes de resonancia magnética, optimizando su precisión en la identificación de casos negativos. Este enfoque meticuloso y ajustado a un único tipo de datos resalta la capacidad de estos modelos para lograr un alto desempeño en contextos específicos, asegurando su aplicabilidad y fiabilidad en tareas clínicas.

En lo referente al F1 Score, los resultados alcanzados en esta investigación fueron del 96.90% con CNN y del 96.55% con VGG16. Estos valores superan a los reportados por Zhu et al. (2024), quienes obtuvieron un F1 Score de 76.79% en la clasificación de tumores mamarios benignos y malignos utilizando la arquitectura DBL-Net. A pesar de las diferencias en el tipo de imágenes y la metodología aplicada en cada estudio, los resultados aquí presentados reflejan

un equilibrio sólido entre precisión y sensibilidad, garantizando un desempeño consistente en la clasificación de tumores cerebrales.

La puntuación F1 más alta obtenida en este estudio, en comparación con los hallazgos de Zhu, puede atribuirse a las diferencias en los tipos de imágenes y los enfoques de clasificación utilizados en cada investigación. Mientras Zhu centró su análisis en la clasificación de tumores mamarios a partir de imágenes de ultrasonido, que presentan desafíos significativos debido a características visuales particulares como el ruido y la baja resolución, este estudio se basó en imágenes de resonancia magnética (MRI), que proporcionan un mayor nivel de detalle y claridad en las estructuras internas. Estas ventajas facilitaron que los modelos CNN y VGG16 identificaran patrones específicos asociados con tumores cerebrales con mayor precisión, resultando en una puntuación F1 superior. Este enfoque metodológico subraya la relevancia de utilizar datos de alta calidad y ajustados de manera adecuada al problema de investigación.

V. CONCLUSIONES

En este estudio, se llevó a cabo una comparación exhaustiva de diversos algoritmos de aprendizaje profundo enfocados en la identificación automática de tumores cerebrales a partir de imágenes médicas, analizando el rendimiento de cada modelo con base en las métricas de precisión, sensibilidad, especificidad y F1 Score. Estas métricas facilitaron la evaluación del desempeño de cada algoritmo y su pertinencia en un entorno clínico, donde la confiabilidad y la exactitud son primordiales. A continuación, se exponen las conclusiones obtenidas para cada métrica analizada.

Los hallazgos relacionados con la precisión demuestran que algunos algoritmos de aprendizaje profundo son capaces de identificar tumores cerebrales con gran exactitud, reduciendo la probabilidad de errores en la clasificación. Este resultado resalta la utilidad de estos modelos en situaciones donde la detección correcta de la afección médica es esencial. La notable precisión alcanzada por algunos algoritmos sugiere que el uso de aprendizaje profundo podría desempeñar un papel significativo en la disminución de equivocaciones durante el diagnóstico de tumores, proporcionando una herramienta confiable para los especialistas en salud.

La sensibilidad de los modelos evaluados mostró que ciertos algoritmos tienen la capacidad de identificar con éxito un alto porcentaje de casos positivos, lo cual es crucial en el contexto de detección temprana de tumores. Una sensibilidad elevada permite detectar la mayoría de los casos que requieren intervención, evitando diagnósticos omitidos que podrían retrasar el tratamiento. Esto confirma que, en aplicaciones clínicas, los modelos con alta sensibilidad son preferibles para asegurar una detección exhaustiva y minimizar los riesgos de omisiones.

La especificidad obtenida por algunos algoritmos indicó su efectividad para identificar correctamente los casos negativos, disminuyendo así los falsos positivos y, por ende, las intervenciones innecesarias. Este resultado es de

especial importancia en el ámbito médico, donde se busca reducir el número de diagnósticos incorrectos que pueden generar ansiedad en los pacientes y costos adicionales. Por lo tanto, los modelos con alta especificidad son recomendables en contextos donde la exclusión precisa de tumores es fundamental.

El análisis del F1 Score permitió identificar un equilibrio adecuado entre exactitud y sensibilidad en determinados modelos, lo cual resulta óptimo para un diagnóstico balanceado. Un valor elevado de F1 Score indica que el modelo mantiene un desempeño consistente en la detección de casos positivos sin comprometer la exactitud. Este balance es esencial en el ámbito clínico, ya que garantiza que tanto los verdaderos positivos como los negativos sean considerados con igual relevancia, proporcionando así un diagnóstico confiable y robusto.

VI. RECOMENDACIONES

Ampliar el conjunto de datos de entrenamiento. Aunque el volumen de datos utilizado fue significativo, se sugiere continuar ampliándolo para optimizar la capacidad de generalización del modelo, particularmente en clases con menor representación, como meningioma y glioma. Incorporar imágenes adicionales de estos tipos de tumores podría contribuir a equilibrar el conjunto y mejorar el desempeño del modelo en cada categoría. Asimismo, la inclusión de imágenes con diferentes resoluciones y características, como variaciones en contraste o ruido, puede fortalecer la robustez del modelo.

Optimización del tamaño de los lotes y la cantidad de épocas de entrenamiento. Este es un aspecto fundamental en el desarrollo de modelos. Durante la presente investigación, se utilizó un tamaño de lote de 128 y un rango de 10 a 30 épocas, configuración adecuada para la capacidad de procesamiento disponible. Sin embargo, se recomienda para futuras iteraciones explorar tamaños de lote más pequeños. Esta estrategia, respaldada por evidencia experimental, podría aumentar la precisión del modelo al permitir captar con mayor detalle los patrones en los datos. Por otro lado, incrementar el número de épocas podría favorecer un ajuste más fino del modelo, disminuyendo el riesgo de sobreajuste, especialmente en clases con menor representación dentro del conjunto de datos. Estas modificaciones se consideran fundamentales para maximizar tanto el rendimiento como la solidez del modelo.

La evaluación de distintas configuraciones de imagen representa un factor esencial para maximizar el rendimiento del modelo. En este análisis, el modelo fue entrenado utilizando imágenes de tamaño (224, 224, 3), una configuración convencional ampliamente adoptada en la arquitectura VGG16, la cual ha demostrado gran efectividad en aplicaciones de aprendizaje profundo. Sin embargo, se plantea que explorar configuraciones con resoluciones superiores podría resultar ventajoso, ya que permitiría capturar un mayor nivel de detalle anatómico y potenciar la identificación de características sutiles en los tumores. Además, se recomienda la implementación y el análisis de diversas técnicas de

preprocesamiento de imágenes, como la normalización y el incremento de datos. Estas estrategias tienen el potencial de reforzar la capacidad del modelo para generalizar y, en última instancia, optimizar su desempeño en la detección y clasificación de tumores.

Para optimizar la precisión del modelo en clases con una menor representación de imágenes y enfrentar la variabilidad inherente a las imágenes en entornos clínicos, se sugiere implementar métodos de ampliación de datos (data augmentation). Técnicas como la rotación, ajustes en la escala, traslación y modificaciones en el brillo pueden replicar diversas condiciones en la adquisición de imágenes, lo que resulta en un conjunto de datos más equilibrado y en un modelo más adaptable. Estas estrategias facilitarán que el modelo adquiera una mayor capacidad de generalización y afronte con éxito escenarios clínicos heterogéneos.

A pesar de que los modelos actuales han demostrado resultados satisfactorios, se recomienda explorar arquitecturas más sofisticadas, como EfficientNet o modelos basados en transformers. Estas arquitecturas han evidenciado rendimientos prometedores en la categorización de imágenes médicas debido a su capacidad para manejar interrelaciones espaciales complejas y atributos detallados. Integrar estas arquitecturas no solo podría potenciar la precisión del modelo, sino también adecuarlo de manera más efectiva a los retos específicos de la clasificación de tumores cerebrales.

REFERENCIAS

Aggarwal, C. C. (2019). Neural Networks and Deep Learning: A Textbook. Springer. Disponible en: https://doi.org/10.1007/978-3-030-16743-7

Ahmad, A., et al. (2022). The Role of MRI in Brain Tumor Detection. Frontiers in Medical Imaging, 9, 25-33. Disponible en: https://doi.org/10.3389/fmed.2022.00456

Alghamdi, W.Y. (2023). A novel deep learning method for predicting athletes' health using wearable sensors and recurrent neural networks. Decision Analytics Journal, 7(February), ISSN 27726622. DOI: 10.1016/j.dajour.2023.100213.

Alvarado, J. (2021). Python for Data Science and Machine Learning. Springer. Disponible en: https://doi.org/10.1007/978-3-030-55667-9

Appiah, R., Pulletikurthi, V., Esquivel-Puentes, H.A., Cabrera, C., Hasan, N.I., Dharmarathne, S., Gomez, L.J. y Castillo, L. (2024). Brain tumor detection using proper orthogonal decomposition integrated with deep learning networks. Computer Methods and Programs in Biomedicine, 250(April), ISSN 18727565. DOI: 10.1016/j.cmpb.2024.108167.

Arias-Gómez, J., Villasís-Keever, M.Á. y Miranda-Novales, M.G. (2016). El protocolo de investigación III: la población de estudio. Revista Alergia México, 63(2), ISSN 0002-5151. DOI: 10.29262/ram.v63i2.181.

Asif, S., Yi, W., Ain, Q.U., Hou, J., Yi, T. y Si, J. (2022). Improving effectiveness of different deep transfer learning-based models for detecting brain tumors from MR images. IEEE Access, 10, ISSN 21693536. DOI: 10.1109/ACCESS.2022.3153306.

Brown, T. (2020). Python Programming for Data Science. O'Reilly Media. Disponible en: https://doi.org/10.1007/978-1-4842-6019-4

Carvalho, E.D., Da Silva Neto, O.P. y De Carvalho Filho, A.O. (2024). Deep learning-based tumor segmentation and classification in breast MRI with 3TP method. Biomedical Signal Processing and Control, 93(March), ISSN 17468108. DOI: 10.1016/j.bspc.2024.106199.

Cekic, E., Pinar, E., Pinar, M. y Dagcinar, A. (2024). Deep Learning-Assisted Segmentation and Classification of Brain Tumor Types on Magnetic Resonance and Surgical Microscope Images. World Neurosurgery, 182, ISSN 18788769. DOI: 10.1016/j.wneu.2023.11.073.

Chollet, F. (2022). Deep Learning with Python. 2ª ed. Manning Publications.

Clark, A. (2020). Python for Data Engineering. Apress. Disponible en: https://doi.org/10.1007/978-1-4842-6311-0

Douglas, R. (2022). Mastering Visual Studio Code. Packt Publishing. Disponible en: https://doi.org/10.1007/978-1-4842-6111-5

Feng, L., Chen, C., Wang, L., Zhang, J., Li, Y., Yang, T., Fan, Y., Liu, P., Sun, P. y Huang, F. (2024). ECMS-NET: A multi-task model for early endometrial cancer MRI sequences classification and segmentation of key tumor structures. Biomedical Signal Processing and Control, 93(October 2023), ISSN 17468108. DOI: 10.1016/j.bspc.2024.106223.

Forien, R., Ringbauer, H. y Coop, G. (2024). Demographic inference for spatially heterogeneous populations using long shared haplotypes. Theoretical Population Biology, March, ISSN 10960325. DOI: 10.1016/j.tpb.2024.03.002.

García, L. (2020). Practical Guide to XAMPP and PHP Development. Apress. Disponible en: https://doi.org/10.1007/978-1-4842-4895-6

Gade, V.S.R., Cherian, R.K., Rajarao, B. y Aravind Kumar, M. (2024). BMO based improved Lite Swin transformer for brain tumor detection using MRI images. Biomedical Signal Processing and Control, 92(January), ISSN 17468108. DOI: 10.1016/j.bspc.2024.106091.

Gertsvolf, D., Horvat, M., Aslam, D., Khademi, A. y Berardi, U. (2024). A U-net convolutional neural network deep learning model application for identification of energy loss in infrared thermographic images. Applied Energy, 360(October 2023), ISSN 03062619. DOI: 10.1016/j.apenergy.2024.122696.

Gómez Bastar, S. (2012). Metodología de la investigación. S.l.: s.n. ISBN 9786077331490.

Goodfellow, I., Bengio, Y., & Courville, A. (2021). Deep Learning for Medical Imaging. MIT Press.

González, J., Martínez, P. y Ramírez, M. (2019). Recolección de datos mediante repositorios públicos en investigaciones científicas. Revista de Investigación y Tecnología, 12(3), pp. 40-55.

He, Y., Song, B. y Zhang, D. (2017). Uncertainty-based improved multidisciplinary design optimization methods. Proceedings of 2017 IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference, IAEAC 2017. DOI: 10.1109/IAEAC.2017.8054186.

Hernández, R., Fernández, C. y Baptista, P. (2014). Metodología de la investigación. 6ta ed. México: McGraw-Hill.

Hernández, R. (2023). Data Visualization for Machine Learning. Springer. Disponible en: https://doi.org/10.1007/978-3-030-88745-2

Johnson, T. (2021). Introduction to Deep Learning Metrics. Springer. Disponible en: https://doi.org/10.1007/978-3-030-56789-7

Karim, S., et al. (2024). Brain Tumor Types and Their Clinical Implications. Neuro-Oncology Insights, 7(2), 12-20. Disponible en: https://doi.org/10.1093/neuonc/noy4567

Lin, S., Dong, M., Cao, X., Liang, Z., Guo, H. y Zheng, H. (2024). The pre-trained explainable deep learning model with stacked denoising autoencoders for slope stability analysis. Engineering Analysis with Boundary Elements, 163(March), ISSN 09557997. DOI: 10.1016/j.enganabound.2024.03.019.

Liu, Y. (2022). Efficient Python Programming for Data Science. CRC Press. Disponible en: https://doi.org/10.1201/9781003161069

Martínez, L. (2022). Advanced Deep Learning for Biomedical Applications. Elsevier. Disponible en: https://doi.org/10.1016/B978-0-12-819765-8.00006-3

Morales, P. y Contreras, J. (2019). Aplicaciones del desarrollo modular en proyectos de Deep Learning. Revista Internacional de Inteligencia Artificial, 12(3), pp. 78-85.

Nikfam, F., Marchisio, A., Martina, M. y Shafique, M. (2022). AccelAT: A Framework for Accelerating the Adversarial Training of Deep Neural Networks Through Accuracy Gradient. IEEE Access, 10(August), ISSN 21693536. DOI: 10.1109/ACCESS.2022.3213734.

Noreen, N., Palaniappan, S., Qayyum, A., Ahmad, I., Imran, M. y Shoaib, M. (2020). A Deep Learning Model Based on Concatenation Approach for the Diagnosis of Brain Tumor. IEEE Access, 8, ISSN 21693536. DOI: 10.1109/ACCESS.2020.2978629.

Osorio, A., Ravina, A., Arias, N. y Pérez, M. (2023). Caracterización del consumo de tabaco en población adulta en Colombia: propuestas de mejora. Revista Colombiana de Cancerología, 27(1).

Otzen, T. y Manterola, C. (2017). Técnicas de Muestreo sobre una Población a Estudio. International Journal of Morphology, 35(1), ISSN 07179502. DOI: 10.4067/S0717-95022017000100037.

Patel, M. (2021). Mastering Deep Learning with TensorFlow. Wiley. Disponible en: https://doi.org/10.1002/9781119631380

Pérez, C. (2021). Metrics and Measurements in Deep Learning. Morgan & Claypool. Disponible en: https://doi.org/10.2200/S00985ED1V01Y202109AI020

Pollet, T.V., Bovet, J., Buhaenko, R., Cornelissen, P.L. y Tovée, M.J. (2024). Sample characteristics for quantitative analyses in Body Image: Issues of generalisability. Body Image, 49(May), ISSN 17401445. DOI: 10.1016/j.bodyim.2024.101714.

Rastogi, D., Johri, P., Tiwari, V. y Elngar, A.A. (2024). Multi-class classification of brain tumour magnetic resonance images using multi-branch network with inception block and five-fold cross validation deep learning framework. Biomedical Signal Processing and Control, 88(PA), ISSN 17468108. DOI: 10.1016/j.bspc.2023.105602.

Rogers, P. (2020). Evaluation Metrics in Machine Learning. Wiley. Disponible en: https://doi.org/10.1002/9781119573567

Rozema, J., Graafsma, J., Hoogendoorn, M., Kibbelaars, R., Veeger, N. y Van Roon, E. (2023). Treatment patterns in older patients with myelodysplastic syndromes: A population-based analysis reflecting the real world. Journal of Geriatric Oncology, 14(2), ISSN 18794068. DOI: 10.1016/j.jgo.2022.101418.

Sandhiya, B., Kanaga Suba Raja, S., Shruthi, K. y Praveena Rachel Kamala, S. (2024). Brain tumour segmentation and classification with reconstructed MRI using DCGAN. Biomedical Signal Processing and Control, 92(September 2023), ISSN 17468108. DOI: 10.1016/j.bspc.2024.106005.

Sanju, P. (2023). Enhancing intrusion detection in IoT systems: A hybrid metaheuristics-deep learning approach with ensemble of recurrent neural networks. Journal of Engineering Research (Kuwait), 11(4), ISSN 23071885. DOI: 10.1016/j.jer.2023.100122.

Shaheema, S.B., Suganya Devi, K. y Muppalaneni, N.B. (2024). Explainability based Panoptic brain tumor segmentation using a hybrid PA-NET with GCNN-ResNet50. Biomedical Signal Processing and Control, 94(April), ISSN 17468108. DOI: 10.1016/j.bspc.2024.106334.

Silva, P. (2021). Data Science with Python. Packt Publishing. Disponible en: https://doi.org/10.1007/978-3-030-53097-6

Sinica, A.P., Science, A.L., Stedmon, C.A., Markager, S., Bro, R., Fellman, J.B., Petrone, K.C., Grierson, P.F., D'Orazio, V. y Traversa (2020). Title. Estuarine, Coastal and Shelf Science, 1, ISSN 00243590.

Smith, J. (2021). Data Handling with Pandas in Machine Learning. Elsevier. Disponible en: https://doi.org/10.1016/B978-0-12-819765-8.00006-3

Smith, R. (2023). Deep Learning Techniques for Medical Image Analysis. CRC Press. Disponible en: https://doi.org/10.1201/9781003281460

Tan, P. (2020). Deep Learning: Concepts and Applications. Springer. Disponible en: https://doi.org/10.1007/978-3-030-12345-6

Ureta, Í.G. (2017). La Encuesta: Éxito. DOI: 10.2307/j.ctt1v2xt4b.8.

Vega Falcón, V., Leyva Vázquez, M.Y. y Batista Hernández, N. (2023). Desarrollo y validación de un cuestionario para evaluar el conocimiento en metodología de la investigación. Revista Conrado, 19(2).

Vignesh, T., Kanimozhi, K.V., Sathish, R., Kumar, R.P., Jeyavathana, R.B. y Ezhumalai, P. (2022). Land Use and Land Cover Classification Using Recurrent Neural Networks with Shared Layered Architecture. 2022 International Conference on Computer Communication and Informatics, ICCCI 2022. DOI: 10.1109/ICCCI54379.2022.9740839.

Wael, Y., et al. (2023). Brain Tumor Diagnosis and Survival Rates. Journal of Neuro-Oncology, 14(3), 37-52. Disponible en: https://doi.org/10.1007/s12311-023-4567-8

Wang, Z. (2023). Advanced Data Visualization Techniques with Python. Packt Publishing. Disponible en: https://doi.org/10.1007/978-1-4842-8045-2

Zhu, C., Chai, X., Wang, Z., Xiao, Y., Zhang, R.M., Yang, Z. y Feng, J. (2024). DBL-Net: A dual-branch learning network with information from spatial and frequency domains for tumor segmentation and classification in breast ultrasound image. Biomedical Signal Processing and Control, 93(March), ISSN 17468108. DOI: 10.1016/j.bspc.2024.106221.

Zou, X., Wan, C. y Cao, L. (2024). Evaluation of sampling methods for nuclear data uncertainty quantification. Annals of Nuclear Energy, 200(January), ISSN 18732100. DOI: 10.1016/j.anucene.2024.110380.

ANEXOS

Anexo 1: Tabla de operacionalización de variables o tabla de categorización

Tabla 14: Operacionalización de variables

Variables	Definición conceptual	Definición operacional	Indicadores	Escala de medición
Independiente Modelo de Deep Learning	Según Shukla et al. (2021), los modelos de Deep Learning han transformado el campo de la visión computacional y el análisis de imágenes médicas. Estas arquitecturas comprenden redes neuronales profundas, tales como las redes convolucionales.	El modelo de Deep Learning utilizará una arquitectura de red neuronal convolucional (CNN) entrenada con imágenes médicas cerebrales etiquetadas. Se configurarán hiperparámetros para maximizar su rendimiento en la identificación de tumores cerebrales.		
Dependiente: Identificación automática de tumores cerebrales	Según Tandel et al. (2022), este proceso implica el uso de segmentación semántica en imágenes médicas mediante etiquetas a nivel de píxel. Se evalúa utilizando métricas como precisión, sensibilidad y especificidad, en comparación con las observaciones de expertos.	Se analizará el desempeño del modelo de Deep Learning utilizando un conjunto de imágenes médicas de prueba. Se calcularán métricas de rendimiento, tales como precisión y sensibilidad, para determinar su efectividad en la detección y clasificación de tumores.	-Precisión en la detección. -Sensibilidad en la detección. -Especificidad. -F1 Score.	Razón (porcentajes 0-100%)

Fuente: Elaboración propia

Anexo 2: Instrumento de recolección de datos.

Tabla 15: Tabla de recolección de datos con CNN

Indicador: Precisión, Sensibilidad, Especificidad y F1 Score

FICHA DE REGISTRO			
Investigador	Fredy Alegre Milla	**Tipo de Prueba**	Post Test
Algoritmo	CNN		

MATRIZ DE CONFUSIÓN		
	TUMOR [1]	**NO TUMOR [0]**
Etiqueta Real: Tumor [1]	**TP:** [1,1]	**FN:** [1,0]
Etiqueta Real: No Tumor [0]	**FP:** [0,1]	**TN:** [0,0]

MÉTRICAS POR EVALUAR		
METRICA	**FÓRMULA**	**INDICADOR**
Precisión	TP / (TP + FP)	%
Sensibilidad	TP / (TP + FN)	%
Especificidad	TN / (TN + FP)	%
F1 Score	2*(Precisión*Sensibilidad) / (Precisión + Sensibilidad)	%

TABLA DE RESULTADOS						
EPOCAS	**ALGORITMO**	**PRECISIÓN**	**SENSIBILIDAD**	**ESPECIFICIDAD**	**F1 SCORE**	**TIEMPO (Minutos)**
10	CNN	93.06%	92.74%	97.70%	92.75%	8.37
15	CNN	96.72%	96.47%	98.88%	96.61%	12.37
20	CNN	97.71%	97.54%	99.23%	97.58%	17.17
25	CNN	98.40%	98.28%	99.48%	98.28%	19.97
30	CNN	99.31%	99.26%	99.77%	99.28%	24.36

Firma del Experto.

Dr. Alfredo Daza Vergaray

Firma del Experto.

Mg. Omar, Pérez Huaman

Tabla 16: Tabla de recolección de datos con MobileNetv2

Indicador: Precisión, Sensibilidad, Especificidad y F1 Score

FICHA DE REGISTRO			
Investigador	Fredy Alegre Milla	**Tipo de Prueba**	Post Test
Algoritmo	MobileNetv2		

MATRIZ DE CONFUSIÓN		
	TUMOR [1]	**NO TUMOR [0]**
Etiqueta Real: Tumor [1]	**TP:** [1,1]	**FN:** [1,0]
Etiqueta Real: No Tumor [0]	**FP:** [0,1]	**TN:** [0,0]

MÉTRICAS POR EVALUAR		
METRICA	**FÓRMULA**	**INDICADOR**
Precisión	TP / (TP + FP)	%
Sensibilidad	TP / (TP + FN)	%
Especificidad	TN / (TN + FP)	%
F1 Score	2*(Precisión*Sensibilidad) / (Precisión + Sensibilidad)	%

TABLA DE RESULTADOS						
EPOCAS	**ALGORITMO**	**PRECISIÓN**	**SENSIBILIDAD**	**ESPECIFICIDAD**	**F1 SCORE**	**TIEMPO (Minutos)**
10	MobileNetv2	91.53%	90.98%	97.21%	90.79%	16.82
15	MobileNetv2	92.75%	92.22%	97.61%	92.24%	24.34
20	MobileNetv2	93.21%	92.67%	97.73%	92.74%	33.67
25	MobileNetv2	94.05%	93.60%	98.04%	93.57%	37.74
30	MobileNetv2	95.12%	94.73%	98.40%	94.72%	52.88

Firma del Experto.

Dr. Alfredo Daza Vergaray

Firma del Experto.

Mg. Omar, Pérez Huaman

Tabla 17: Tabla de recolección de datos con VGG16

Indicador: Precisión, Sensibilidad, Especificidad y F1 Score

FICHA DE REGISTRO			
Investigador	Fredy Alegre Milla	**Tipo de Prueba**	Post Test
Algoritmo	VGG16		

MATRIZ DE CONFUSIÓN		
	TUMOR [1]	**NO TUMOR [0]**
Etiqueta Real: Tumor [1]	**TP:** [1,1]	**FN:** [1,0]
Etiqueta Real: No Tumor [0]	**FP:** [0,1]	**TN:** [0,0]

MÉTRICAS POR EVALUAR		
METRICA	**FÓRMULA**	**INDICADOR**
Precisión	TP / (TP + FP)	%
Sensibilidad	TP / (TP + FN)	%
Especificidad	TN / (TN + FP)	%
F1 Score	2*(Precisión*Sensibilidad) / (Precisión + Sensibilidad)	%

TABLA DE RESULTADOS						
EPOCAS	**ALGORITMO**	**PRECISIÓN**	**SENSIBILIDAD**	**ESPECIFICIDAD**	**F1 SCORE**	**TIEMPO (Minutos)**
10	VGG16	95.27%	94.98%	98.45%	94.97%	85
15	VGG16	96.19%	95.87%	98.75%	95.90%	123
20	VGG16	97.03%	96.77%	99.02%	96.80%	153
25	VGG16	97.94%	97.77%	99.32%	97.79%	185
30	VGG16	97.48%	97.28%	99.18%	97.30%	215

Firma del Experto.

Dr. Alfredo Daza Vergaray

Firma del Experto.

Mg. Omar, Pérez Huaman

Tabla 18: Tabla de recolección de datos con RestNet50

Indicador: Precisión, Sensibilidad, Especificidad y F1 Score

FICHA DE REGISTRO			
Investigador	Fredy Alegre Milla	**Tipo de Prueba**	Post Test
Algoritmo	ResNet50		

MATRIZ DE CONFUSIÓN		
	TUMOR [1]	**NO TUMOR [0]**
Etiqueta Real: Tumor [1]	**TP:** [1,1]	**FN:** [1,0]
Etiqueta Real: No Tumor [0]	**FP:** [0,1]	**TN:** [0,0]

MÉTRICAS POR EVALUAR		
METRICA	**FÓRMULA**	**INDICADOR**
Precisión	TP / (TP + FP)	%
Sensibilidad	TP / (TP + FN)	%
Especificidad	TN / (TN + FP)	%
F1 Score	2*(Precisión*Sensibilidad) / (Precisión + Sensibilidad)	%

TABLA DE RESULTADOS						
EPOCAS	**ALGORITMO**	**PRECISIÓN**	**SENSIBILIDAD**	**ESPECIFICIDAD**	**F1 SCORE**	**TIEMPO (Minutos)**
10	ResNet50	92.98%	92.47%	97.67%	92.53%	32
15	ResNet50	93.52%	93.06%	97.85%	93.10%	53
20	ResNet50	93.97%	93.55%	98.03%	93.51%	67
25	ResNet50	94.43%	93.99%	98.14%	94.06%	83
30	ResNet50	94.81%	94.46%	98.30%	94.47%	116

Firma del Experto.

Dr. Alfredo Daza Vergaray

Firma del Experto.

Mg. Omar, Pérez Huaman

Tabla 19: Tabla de recolección de datos con DenseNet121

Indicador: Precisión, Sensibilidad, Especificidad y F1 Score

FICHA DE REGISTRO			
Investigador	Fredy Alegre Milla	**Tipo de Prueba**	Post Test
Algoritmo	DenseNet121		

MATRIZ DE CONFUSIÓN		
	TUMOR [1]	**NO TUMOR [0]**
Etiqueta Real: Tumor [1]	**TP:** [1,1]	**FN:** [1,0]
Etiqueta Real: No Tumor [0]	**FP:** [0,1]	**TN:** [0,0]

MÉTRICAS POR EVALUAR		
METRICA	**FÓRMULA**	**INDICADOR**
Precisión	TP / (TP + FP)	%
Sensibilidad	TP / (TP + FN)	%
Especificidad	TN / (TN + FP)	%
F1 Score	2*(Precisión*Sensibilidad) / (Precisión + Sensibilidad)	%

TABLA DE RESULTADOS						
EPOCAS	**ALGORITMO**	**PRECISIÓN**	**SENSIBILIDAD**	**ESPECIFICIDAD**	**F1 SCORE**	**TIEMPO (Minutos)**
10	DenseNet121	91.00%	90.57%	97.05%	90.38%	39
15	DenseNet121	93.82%	93.43%	97.97%	93.44%	57
20	DenseNet121	93.52%	93.08%	97.88%	93.05%	77
25	DenseNet121	94.36%	94.00%	98.15%	93.97%	90
30	DenseNet121	94.81%	94.46%	98.30%	94.47%	116

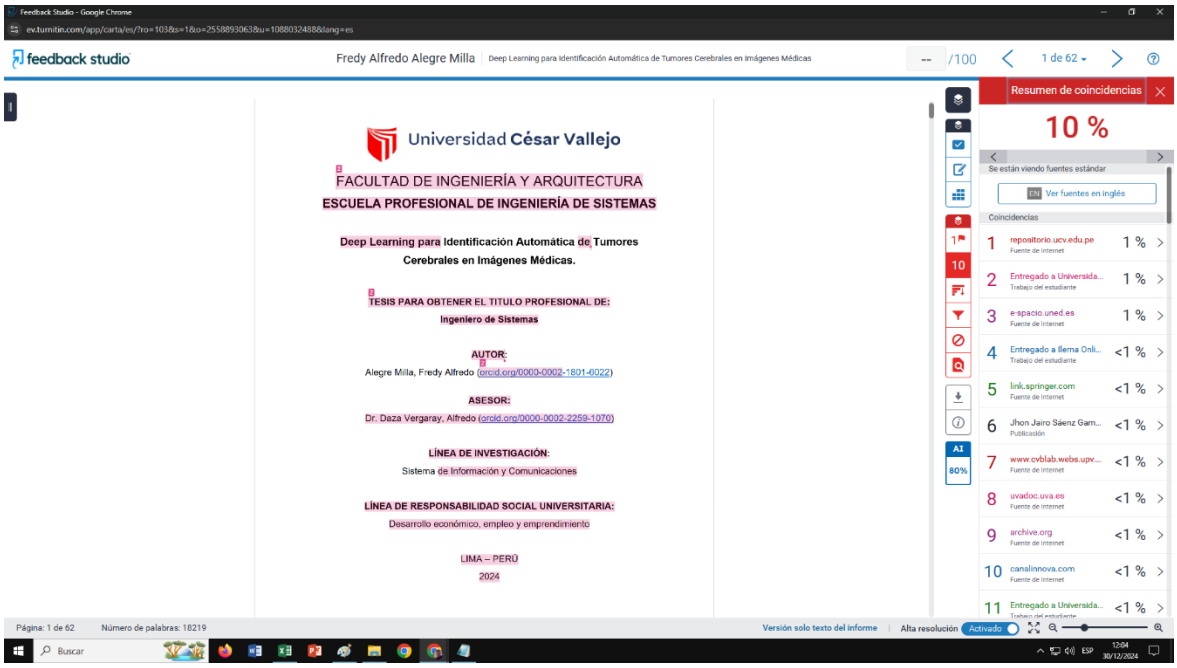
Firma del Experto.

Dr. Alfredo Daza Vergaray

Firma del Experto.

Mg. Omar, Pérez Huaman

Anexo 3. Reporte de similitud en software Turnitin.



Anexo 4. Metodología de desarrollo de software.

Metodología CRISP-DM

1. Comprensión del Negocio

- **Objetivo**: Mejorar la eficiencia y precisión en el diagnóstico de tumores cerebrales mediante el uso de algoritmos de Deep Learning.
- **Problema**: Identificar automáticamente tumores cerebrales (gliomas, meningiomas, pituitarios y ausencia de tumor) en imágenes de resonancia magnética.
- **Resultados esperados**: Un modelo que supere métodos tradicionales en precisión, sensibilidad, especificidad y F1 Score.

2. Comprensión de los Datos

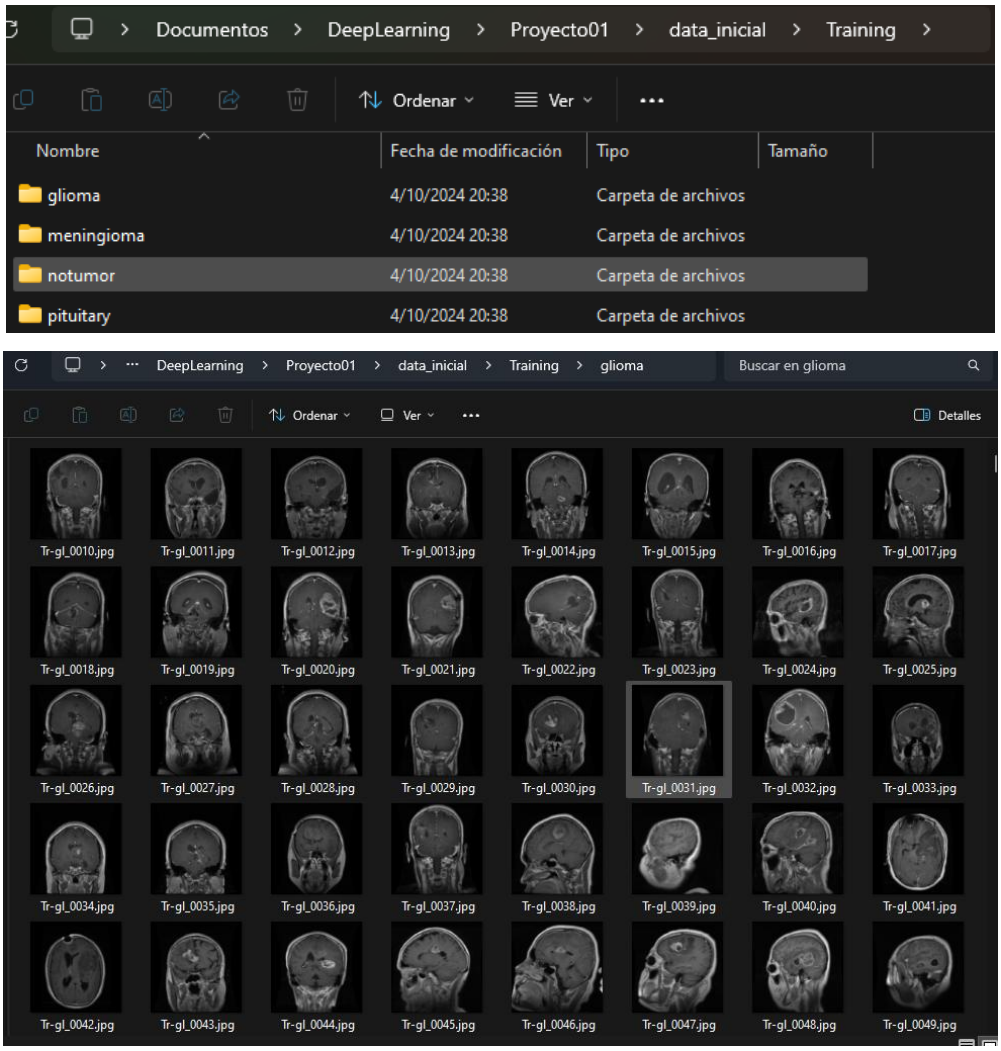


Figura 5: Dataset validación de Imágenes – Tumor Gliomas

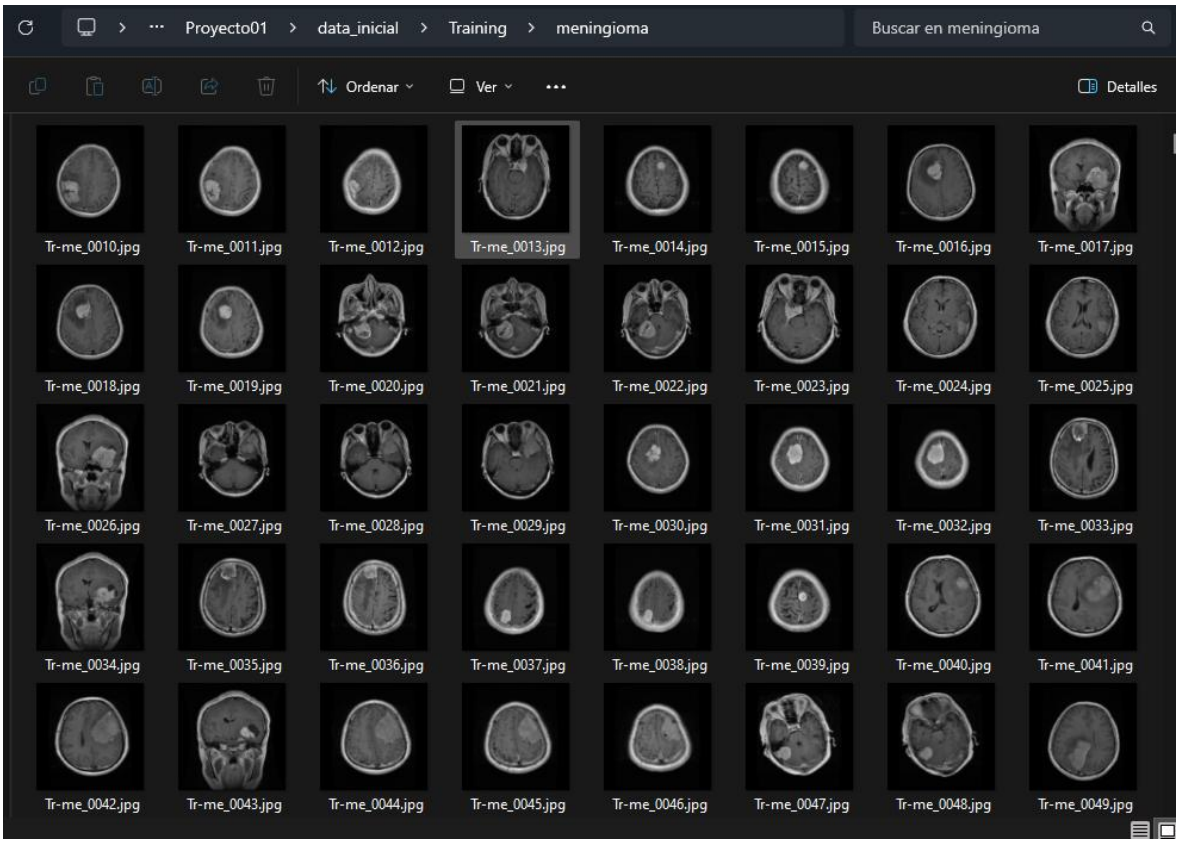


Figura 6: Dataset validación de Imágenes – Tumor Meningiomas

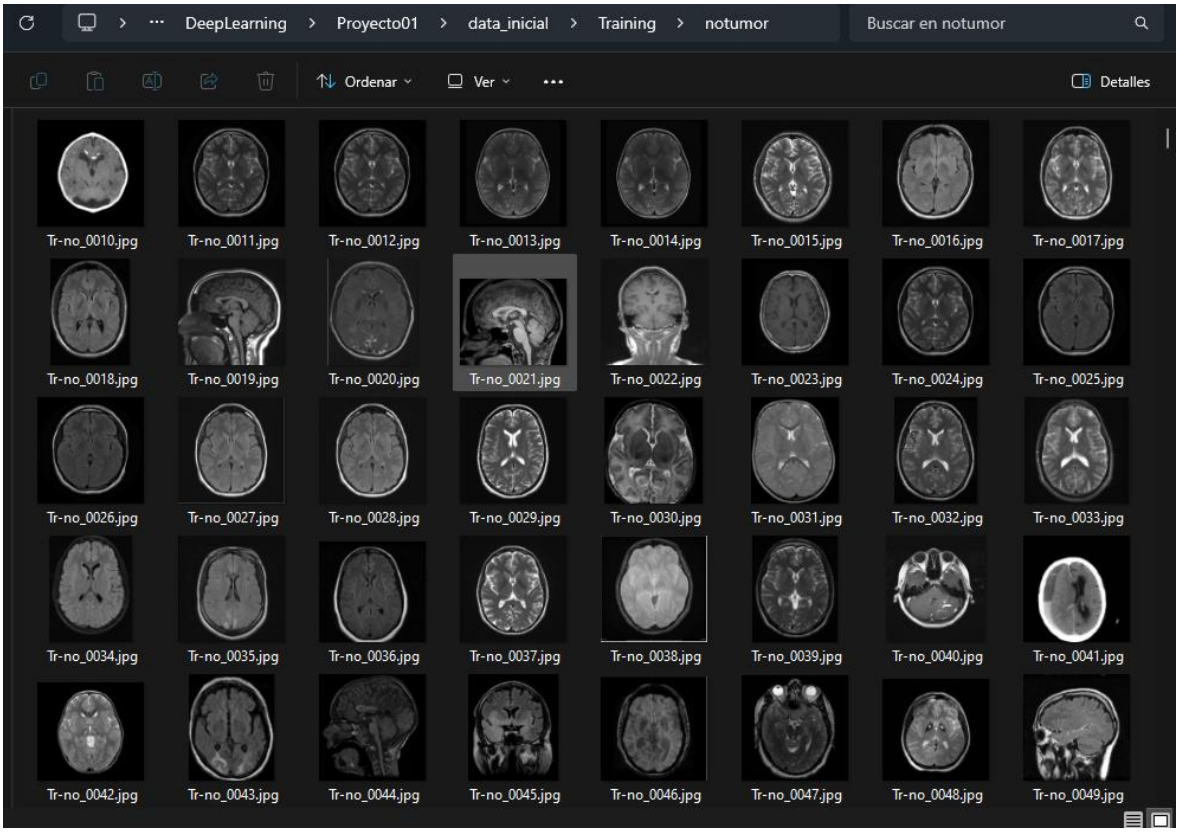


Figura 7: Dataset validación de Imágenes – No Tumor

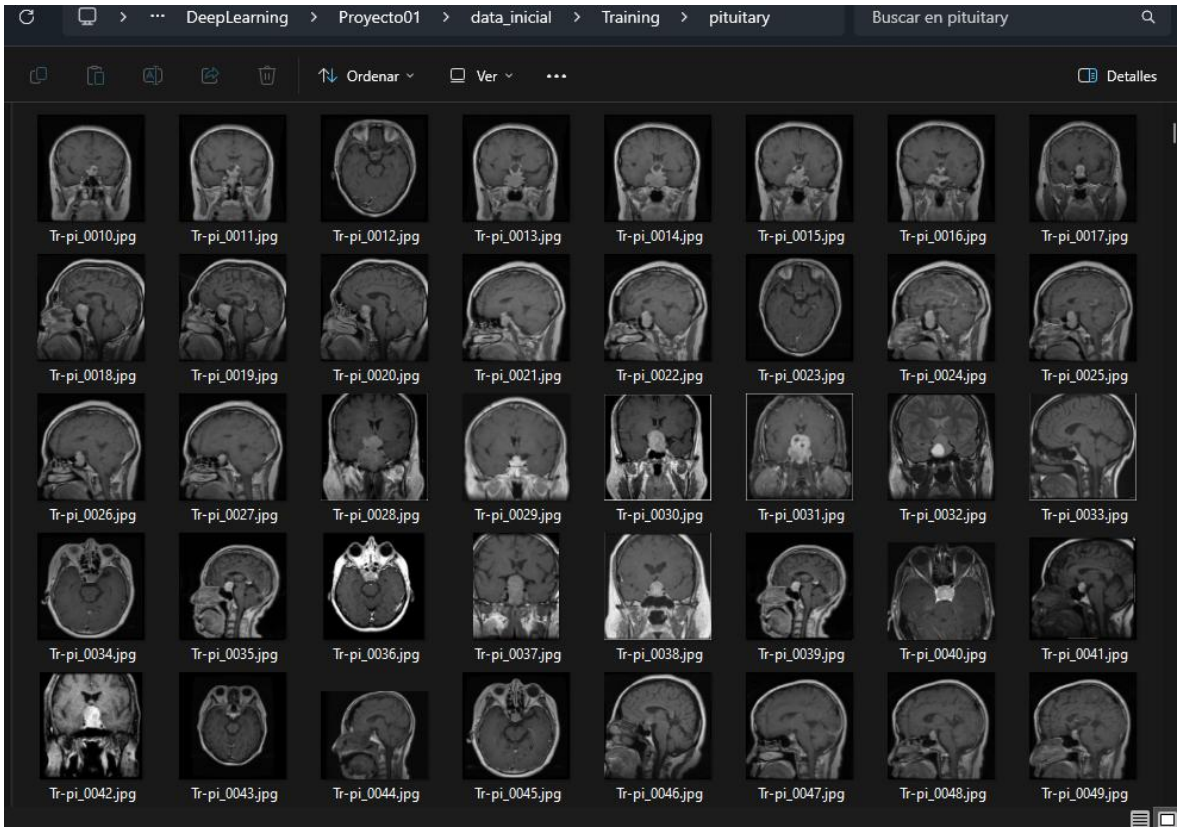


Figura 8: Dataset validación de Imágenes –Pituitary

3. PROCESAMIENTO DE IMÁGENES

Figura 9: UI de predicción del modelo.

El ID 145 es una prueba realiza en el hipotético caso de que el análisis de un médico es de tipo tumor Pitutary pero el modelo determina que es un **no tumor.** En este caso podemos concluir que el modelo entrenado predijo bien con una probabilidad de 100% clasificando como no tumor.

Predicciones recientes

ID	Archivo	Algoritmo	Etiqueta Real	Predicción	Probabilidad	Exactitud	Tiempo	Estado
152	Te-pi_0270.jpg	VGG16	pituitary	pituitary	1.00	1.00	0.311173	Success
150	Te-pi_0105.jpg	ResNet	pituitary	pituitary	1.00	1.00	1.255282	Success
149	Te-pi_0025.jpg	MobileNetV2	pituitary	pituitary	1.00	1.00	0.752991	Success
148	Te-pi_0030.jpg	DenseNet	pituitary	pituitary	1.00	1.00	2.344386	Success
147	Te-pi_0027.jpg	DenseNet	pituitary	pituitary	1.00	1.00	2.599803	Success
146	Te-no_0011.jpg	MobileNetV2	notumor	notumor	1.00	1.00	0.751106	Success
145	Te-no_0011.jpg	VGG16	pituitary	notumor	1.00	0.00	0.27296	Failed

Figura 10: Almacenamiento de predicciones en MySQL.

4. Modelado, selección de algoritmos y entrenamiento.

Preparar el entorno para CNN

Para implementar CNN en la identificación automática de tumores cerebrales, es esencial que los datos se estructuren y preparen adecuadamente, garantizando que el modelo pueda aprender y generalizar de manera efectiva a partir de las imágenes. La preparación incluye:

1. Formato de las imágenes:
    - Las imágenes médicas deben ser transformadas al tamaño y formato que requiere la red neuronal convolucional (por ejemplo, 256x256 píxeles en escala de grises o RGB).
    - Normalización de los valores de los píxeles para que estén entre 0 y 1, lo cual optimiza el proceso de aprendizaje del modelo.
2. Etiquetas y clasificación:
    - Cada imagen debe estar asociada a su categoría (glioma, meningioma, pituitario o sin tumor), siguiendo un formato estructurado compatible con frameworks como TensorFlow o PyTorch.
3. División del conjunto de datos:

- Separación en entrenamiento, validación y prueba (por ejemplo, 70%, 20% y 10%, respectivamente) para evaluar la capacidad de generalización del modelo.

Desarrollo del algoritmo con CNN

Archivo de datos (clasificación_cnn.py):

- Define la ruta a tu conjunto de datos y las clases

```
clasificacion_cnn.py 6 ×
trained_model > clasificacion_cnn.py > ...
N_TYPES = len(CLASS_TYPES)

# Rutas de los directorios de entrenamiento y prueba
train_dir = r"C:\Users\Alegre\Documents\DeepLearning\Proyecto01\data\Training"
test_dir = r"C:\Users\Alegre\Documents\DeepLearning\Proyecto01\data\Testing"

# Ruta base para guardar los resultados
resultados_base_dir = r"C:\Users\Alegre\Documents\DeepLearning\Proyecto01\Resultados\Clasificacion"
```

Definición del modelo CNN para clasificación

```
clasificacion_cnn.py 6 ×
trained_model > clasificacion_cnn.py > ...

# 08 Definición del modelo CNN para clasificación
model_cnn = models.Sequential([
    Conv2D(32, (4, 4), activation="relu", input_shape=image_shape),
    MaxPooling2D(pool_size=(3, 3)),
    Conv2D(64, (4, 4), activation="relu"),
    MaxPooling2D(pool_size=(3, 3)),
    Conv2D(128, (4, 4), activation="relu"),
    MaxPooling2D(pool_size=(3, 3)),
    Conv2D(128, (4, 4), activation="relu"),
    Flatten(),
    Dense(512, activation="relu"),
    Dropout(0.5, seed=SEED),
    Dense(N_TYPES, activation="softmax")
])
```

ENTRENAR EL MODELO

```
PROBLEMS 6   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

PS C:\Users\Alegre\Documents\DeepLearning\Proyecto01\trained_model> python clasificacion_cnn.py
2024-11-20 21:12:59.040131: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly d
n them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2024-11-20 21:12:59.735180: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly d
n them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
Training
Number of Paths: 5712
Number of Labels: 5712

Testing
Number of Paths: 1311
Number of Labels: 1311
Training Counts
{'pituitary': 1457, 'notumor': 1595, 'meningioma': 1339, 'glioma': 1321}

Testing Counts
{'pituitary': 300, 'notumor': 405, 'meningioma': 306, 'glioma': 300}
Found 5712 images belonging to 4 classes.
Found 1311 images belonging to 4 classes.
Ingrese el número de épocas para el entrenamiento de clasificación: 40
Image shape: (150, 150, 3)
Epochs: 40
Batch size: 32
Steps Per Epoch: 179
Validation steps: 41
```

```
PROBLEMS 6   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

 Trainable params: 495,972 (1.89 MB)
 Non-trainable params: 0 (0.00 B)
C:\Users\Alegre\Documents\DeepLearning\Proyecto01\venv\lib\site-packages\keras\src\trainers\data_adapters\py_dataset_adapter.py:121: UserWarning: Your
or. `**kwargs` can include `workers`, `use_multiprocessing`, `max_queue_size`. Do not pass these arguments to `fit()`, as they will be ignored.
  self._warn_if_super_not_called()
Epoch 1/40
179/179 ━━━━━━━━━━━━━━━━━━━━ 56s 304ms/step - accuracy: 0.4926 - loss: 1.0319 - val_accuracy: 0.7376 - val_loss: 0.6151 - learning_rate: 0.0010
Epoch 2/40
179/179 ━━━━━━━━━━━━━━━━━━━━ 50s 277ms/step - accuracy: 0.7870 - loss: 0.5300 - val_accuracy: 0.7368 - val_loss: 0.7491 - learning_rate: 0.0010
Epoch 3/40
 38/179 ━━━━━━━━━━━━━━━━━━━━ 37s 263ms/step - accuracy: 0.8363 - loss: 0.3872
```

INTEGRAR EL MODELO

```
clasificacion_cnn.py 6 ×
trained_model > clasificacion_cnn.py > ...
1   # python clasificacion_cnn.py
2
3   # 01 Importaciones necesarias
4   import os
5   import random
6   import numpy as np
7   import matplotlib.pyplot as plt
8   import tensorflow as tf
9   import datetime
10  import time
11  import pandas as pd
12  import seaborn as sns
13
14  from tensorflow.keras import models
15  from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
16  from tensorflow.keras.optimizers import Adam
17  from tensorflow.keras.preprocessing.image import ImageDataGenerator
18  from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau, ModelCheckpoint
19  from tensorflow.keras.utils import plot_model
20  from sklearn.metrics import confusion_matrix
```

```python
# 02 Configuraciones Generales
# Asegurarse de que Graphviz está en el PATH del sistema
os.environ["PATH"] += os.pathsep + r'C:\Program Files\Graphviz\bin'

# Configuraciones generales
SAVE = True
SEED = 111

# Establecer semillas para reproducibilidad
tf.keras.utils.set_random_seed(SEED)
np.random.seed(SEED)

# Configuración de visualización de gráficos
plt.rcParams["figure.figsize"] = (16, 10)
plt.rcParams.update({'font.size': 14})

# 03 Definición de clases y Rutas
CLASS_TYPES = ['pituitary', 'notumor', 'meningioma', 'glioma']
N_TYPES = len(CLASS_TYPES)

# Rutas de los directorios de entrenamiento y prueba
train_dir = r"C:\Users\Alegre\Documents\DeepLearning\Proyecto01\data\Training"
test_dir = r"C:\Users\Alegre\Documents\DeepLearning\Proyecto01\data\Testing"

# Ruta base para guardar los resultados
resultados_base_dir = r"C:\Users\Alegre\Documents\DeepLearning\Proyecto01\Resultados\Clasificacion"

# 04 Creación del directorio para resultados
# Obtener la fecha y hora actual en formato YYYY-MM-DD_HH-MM-SS
fecha_actual = datetime.datetime.now().strftime("%Y-%m-%d_%H-%M-%S")

# Crear la ruta completa
resultados_dir = os.path.join(resultados_base_dir, 'CNN', fecha_actual)

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

```python
# 05 Cargar Datos y Etiquetas
# Función para obtener rutas y etiquetas de las imágenes
def get_data_labels(directory, shuffle_data=True, random_state=0):
    data_paths = []
    data_labels = []

    for label in CLASS_TYPES:
        label_dir = os.path.join(directory, label)
        if not os.path.isdir(label_dir):
            continue
        for image_name in os.listdir(label_dir):
            image_path = os.path.join(label_dir, image_name)
            data_paths.append(image_path)
            data_labels.append(label)
    if shuffle_data:
        temp = list(zip(data_paths, data_labels))
        random.seed(random_state)
        random.shuffle(temp)
        data_paths, data_labels = zip(*temp)
    return list(data_paths), list(data_labels)
```

```python
# Obtener datos de entrenamiento y prueba
train_paths, train_labels = get_data_labels(train_dir)
test_paths, test_labels = get_data_labels(test_dir)

# Mostrar cantidad de datos
print('Training')
print(f'Number of Paths: {len(train_paths)}')
print(f'Number of Labels: {len(train_labels)}')
print('\nTesting')
print(f'Number of Paths: {len(test_paths)}')
print(f'Number of Labels: {len(test_labels)}')

# 06 Visualización de distribución de clases
def plot_class_distribution(train_labels, test_labels, save=False, save_dir=None):
    _, ax = plt.subplots(ncols=3, figsize=(20, 14))

    # Conteo de clases en entrenamiento
    class_counts_train = [train_labels.count(label) for label in CLASS_TYPES]
    print('Training Counts')
    print(dict(zip(CLASS_TYPES, class_counts_train)))

    ax[0].set_title('Training Data')
    ax[0].pie(
        class_counts_train,
        labels=[label.title() for label in CLASS_TYPES],
        colors=['#FAC500','#0BFA00', '#0066FA','#FA0000'],
        autopct=lambda p: '{:.2f}%\n{:,.0f}'.format(p, p * sum(class_counts_train) / 100),
        explode=tuple(0.01 for i in range(N_TYPES)),
        textprops={'fontsize': 14}
    )

    # Distribución de entrenamiento y prueba
    ax[1].set_title('Train Test Split')
    ax[1].pie(
        [len(train_labels), len(test_labels)],
        labels=['Train','Test'],
        colors=['darkcyan', 'orange'],
        autopct=lambda p: '{:.2f}%\n{:,.0f}'.format(p, p * sum([len(train_labels), len(test_labels)]) / 100),
        explode=(0.1, 0),
        startangle=85,
        textprops={'fontsize': 14}
    )

    # Conteo de clases en prueba
    class_counts_test = [test_labels.count(label) for label in CLASS_TYPES]
    print('\nTesting Counts')
    print(dict(zip(CLASS_TYPES, class_counts_test)))
```

```python
    ax[2].set_title('Testing Data')
    ax[2].pie(
        class_counts_test,
        labels=[label.title() for label in CLASS_TYPES],
        colors=['#FAC500', '#0BFA00', '#0066FA', '#FA0000'],
        autopct=lambda p: '{:.2f}%\n{:,.0f}'.format(p, p * sum(class_counts_test) / 100),
        explode=tuple(0.01 for i in range(N_TYPES)),
        textprops={'fontsize': 14}
    )

    plt.tight_layout()

    if save and save_dir is not None:
        distribution_filename = os.path.join(save_dir, f"distribution_{fecha_actual}.png")
        plt.savefig(distribution_filename)
        plt.close()

    plt.show()

# Llamar a la función para visualizar la distribución de clases
plot_class_distribution(train_labels, test_labels, save=True, save_dir=resultados_dir)
```

```python
# 07 Preprocesamiento y aumento de datos para clasificación
image_size = (150, 150)
batch_size = 32

train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=10,
    brightness_range=(0.85, 1.15),
    width_shift_range=0.002,
    height_shift_range=0.002,
    shear_range=12.5,
    horizontal_flip=True,
    fill_mode="nearest"
)

train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=image_size,
    batch_size=batch_size,
    classes=CLASS_TYPES,
    class_mode="categorical",
    seed=SEED
)

test_datagen = ImageDataGenerator(rescale=1./255)

test_generator = test_datagen.flow_from_directory(
    test_dir,
    target_size=image_size,
    batch_size=batch_size,
    classes=CLASS_TYPES,
    class_mode="categorical",
    shuffle=False,
    seed=SEED
)

# Configuración para entrenamiento de clasificación

# Solicitar al usuario el número de épocas
try:
    epochs = int(input("Ingrese el número de épocas para el entrenamiento de clasificación: "))
except ValueError:
    print("Entrada inválida. Se usará el valor predeterminado de 40 épocas.")
    epochs = 40

image_shape = (image_size[0], image_size[1], 3)
```

```python
# Calcular steps_per_epoch y validation_steps usando np.ceil
steps_per_epoch = int(np.ceil(train_generator.samples / batch_size))
validation_steps = int(np.ceil(test_generator.samples / batch_size))

print(f'Image shape: {image_shape}')
print(f'Epochs: {epochs}')
print(f'Batch size: {batch_size}')
print(f'Steps Per Epoch: {steps_per_epoch}')
print(f'Validation steps: {validation_steps}')
```

```python
# 08 Definición del modelo CNN para clasificación
model_cnn = models.Sequential([
    Conv2D(32, (4, 4), activation="relu", input_shape=image_shape),
    MaxPooling2D(pool_size=(3, 3)),
    Conv2D(64, (4, 4), activation="relu"),
    MaxPooling2D(pool_size=(3, 3)),
    Conv2D(128, (4, 4), activation="relu"),
    MaxPooling2D(pool_size=(3, 3)),
    Conv2D(128, (4, 4), activation="relu"),
    Flatten(),
    Dense(512, activation="relu"),
    Dropout(0.5, seed=SEED),
    Dense(N_TYPES, activation="softmax")
])

# Compilar el modelo de clasificación
optimizer = Adam(learning_rate=0.001, beta_1=0.869, beta_2=0.995)
model_cnn.compile(optimizer=optimizer, loss='categorical_crossentropy', metrics=['accuracy'])

# Mostrar resumen del modelo de clasificación
model_cnn.summary()

# Visualización del modelo de clasificación con explicación
def save_model_architecture_with_explanation(model, filename, explanation):
    # Guardar la arquitectura del modelo en un archivo temporal
    temp_filename = 'temp_model_plot.png'
    plot_model(model, to_file=temp_filename, show_shapes=True, show_layer_names=True)
    # Leer la imagen de la arquitectura
    architecture_img = plt.imread(temp_filename)
    # Crear una figura para combinar la imagen y la explicación
    fig, ax = plt.subplots(figsize=(20, 20))
    ax.imshow(architecture_img)
    ax.axis('off')
    # Añadir la explicación como texto en la figura
    plt.figtext(0.5, 0.01, explanation, wrap=True, horizontalalignment='center', fontsize=12)
    # Guardar la figura combinada
    plt.savefig(filename)
    plt.close()
    # Eliminar el archivo temporal
    os.remove(temp_filename)

# Explicación del modelo
explanation = """
Modelo CNN para Clasificación de Tumores Cerebrales:
```

```python
- **Conv2D + MaxPooling2D**: Se aplican capas convolucionales con activación ReLU y capas de pooling para reducir dimensionalidad y extraer características espaciales.
- **Flatten**: Se aplana la salida de las capas convolucionales para preparar los datos para las capas densas.
- **Dense Layers**: Capas densas para clasificar las características extraídas.
- **Dropout**: Se utiliza para prevenir el sobreajuste, desactivando aleatoriamente neuronas durante el entrenamiento.
- **Output Layer**: Capa de salida con activación softmax para clasificación multiclase, que proporciona probabilidades para cada clase.
"""

# Guardar la arquitectura del modelo con explicación
model_architecture_filename = os.path.join(resultados_dir, f"model_cnn_architecture_{fecha_actual}.png")
save_model_architecture_with_explanation(model_cnn, model_architecture_filename, explanation)

# Callbacks para clasificación
early_stopping = EarlyStopping(monitor='val_loss', patience=8, verbose=1, restore_best_weights=True)
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.3, patience=5, verbose=1)
checkpoint_filename = os.path.join(resultados_dir, f"best_model_cnn_{fecha_actual}.keras")
checkpoint = ModelCheckpoint(checkpoint_filename, monitor='val_loss', save_best_only=True)

# Entrenamiento del modelo de clasificación
start_time_cnn = time.time()
start_datetime_cnn = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
history_cnn = model_cnn.fit(
    train_generator,
    epochs=epochs,
    validation_data=test_generator,
    callbacks=[early_stopping, reduce_lr, checkpoint]
)
end_time_cnn = time.time()
end_datetime_cnn = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
training_time_cnn = end_time_cnn - start_time_cnn
```

```python
# Evaluación del modelo de clasificación
loss, accuracy = model_cnn.evaluate(test_generator)
print(f"Test Loss: {loss:0.5f}")
print(f"Test Accuracy: {accuracy:0.5f}")

# Guardar los resultados de clasificación en un archivo de texto
evaluation_results = f"Test Loss: {loss:0.5f}\nTest Accuracy: {accuracy:0.5f}\n"
evaluation_results += f"Inicio del entrenamiento: {start_datetime_cnn}\n"
evaluation_results += f"Fin del entrenamiento: {end_datetime_cnn}\n"
evaluation_results += f"Tiempo de entrenamiento: {training_time_cnn/60:.2f} minutos\n"
evaluation_results += f"Número de épocas: {len(history_cnn.history['loss'])}\n"
evaluation_filename = os.path.join(resultados_dir, f"evaluation_cnn_{fecha_actual}.txt")
with open(evaluation_filename, 'w') as f:
    f.write(evaluation_results)

# Visualización de precisión y pérdida de clasificación
_, ax = plt.subplots(ncols=2, figsize=(15, 6))

ax[0].plot(history_cnn.history['accuracy'])
ax[0].plot(history_cnn.history['val_accuracy'])
ax[0].set_title('Model Accuracy')
ax[0].set_xlabel('Epoch')
ax[0].set_ylabel('Accuracy')
ax[0].legend(['Train', 'Validation'])
ax[0].grid(alpha=0.2)

ax[1].plot(history_cnn.history['loss'])
ax[1].plot(history_cnn.history['val_loss'])
ax[1].set_title('Model Loss')
ax[1].set_xlabel('Epoch')
ax[1].set_ylabel('Loss')
ax[1].legend(['Train', 'Validation'])
ax[1].grid(alpha=0.2)

# Guardar la figura de clasificación
accuracy_loss_filename = os.path.join(resultados_dir, f"accuracy_loss_cnn_{fecha_actual}.png")
plt.savefig(accuracy_loss_filename)
plt.close()

# Matriz de confusión para clasificación
class_indices_train = train_generator.class_indices
class_indices_train_list = list(class_indices_train.keys())
```

```python
def CM(model, test_generator):
    predictions = model.predict(test_generator)
    predicted_categories = np.argmax(predictions, axis=1)
    true_categories = test_generator.classes
    confusion_matrix_array = confusion_matrix(true_categories, predicted_categories)
    return confusion_matrix_array

confusion_mtx = CM(model_cnn, test_generator)

plt.figure(figsize=(8,8))
sns.heatmap(confusion_mtx, annot=True, fmt="d", cmap="Blues", cbar=False)
plt.title("Matriz de Confusión")
plt.xlabel("Predicción")
plt.ylabel("Real")
plt.xticks(ticks=np.arange(N_TYPES) + 0.5,
           labels=[name.title() for name in class_indices_train_list], ha='center')
plt.yticks(ticks=np.arange(N_TYPES) + 0.5,
           labels=[name.title() for name in class_indices_train_list], va='center')

# Guardar la figura de la matriz de confusión
confusion_matrix_filename = os.path.join(resultados_dir, f"confusion_matrix_cnn_{fecha_actual}.png")
plt.savefig(confusion_matrix_filename)
plt.close()
```

```python
# Cálculo de métricas de clasificación y guardar en archivo
def calculate_metrics(confusion_matrix, categories, start_datetime, end_datetime, training_time, epochs, save=False, save_dir=None):
    # Inicializar variables
    precision = []
    recall = []
    specificity = []
    f1_score = []
    accuracy = np.trace(confusion_matrix) / np.sum(confusion_matrix)

    # Calcular métricas para cada clase
    for i in range(len(categories)):
        TP = confusion_matrix[i, i]
        FP = np.sum(confusion_matrix[:, i]) - TP
        FN = np.sum(confusion_matrix[i, :]) - TP
        TN = np.sum(confusion_matrix) - (TP + FP + FN)

        # Evitar división por cero
        precision_i = TP / (TP + FP) if (TP + FP) != 0 else 0
        recall_i = TP / (TP + FN) if (TP + FN) != 0 else 0  # Sensibilidad
        specificity_i = TN / (TN + FP) if (TN + FP) != 0 else 0
        f1_score_i = 2 * (precision_i * recall_i) / (precision_i + recall_i) if (precision_i + recall_i) != 0 else 0

        precision.append(precision_i)
        recall.append(recall_i)
        specificity.append(specificity_i)
        f1_score.append(f1_score_i)

    # Repetir los valores generales para cada clase
    epochs_list = [epochs] * len(categories)
    start_datetime_list = [start_datetime] * len(categories)
    end_datetime_list = [end_datetime] * len(categories)
    training_time_minutes = round(training_time / 60, 2)
    training_time_list = [training_time_minutes] * len(categories)

    # Crear un DataFrame con las métricas
    metrics_df = pd.DataFrame({
        'Tipo': [category.title() for category in categories],
        'Precisión': precision,
        'Sensibilidad (Recall)': recall,
        'Especificidad': specificity,
        'F1 Score': f1_score,
        'Épocas': epochs_list,
        'Fecha_Hora_inicio': start_datetime_list,
        'Fecha_Hora_finalizado': end_datetime_list,
        'Tiempo(minutos)': training_time_list
```

```python
    })

    # Añadir exactitud general al final de la tabla
    metrics_df.loc[len(metrics_df)] = ['Exactitud', accuracy, '', '', '', '', '', '', '']

    # Imprimir resultados
    print(metrics_df.to_string(index=False))

    # Guardar resultados en un archivo de texto
    if save and save_dir is not None:
        metrics_filename = os.path.join(save_dir, f"metrics_cnn_{fecha_actual}.txt")
        with open(metrics_filename, 'w') as f:
            f.write(metrics_df.to_string(index=False))

    # Guardar resultados en una imagen
    if save and save_dir is not None:
        # Formatear los valores numéricos
        metrics_df_image = metrics_df.copy()
        metrics_df_image['Precisión'] = metrics_df_image['Precisión'].apply(lambda x: f"{x:.3f}" if x != '' else x)
        metrics_df_image['Sensibilidad (Recall)'].apply(lambda x: f"{x:.3f}" if x != '' else x)
        metrics_df_image['Especificidad (variable) metrics_df_image: DataFrame pply(lambda x: f"{x:.3f}" if x != '' else x)
        metrics_df_image['F1 Score'] = metrics_df_image['F1 Score'].apply(lambda x: f"{x:.3f}" if x != '' else x)

        fig, ax = plt.subplots(figsize=(12, len(metrics_df) * 0.6))
        ax.axis('off')
        tabla = ax.table(cellText=metrics_df_image.values, colLabels=metrics_df_image.columns, loc='center')
        tabla.auto_set_font_size(False)
        tabla.set_fontsize(10)
        tabla.scale(1, 2)
        metrics_table_filename = os.path.join(save_dir, f"metrics_table_cnn_{fecha_actual}.png")
        plt.savefig(metrics_table_filename, bbox_inches='tight')
        plt.close()

calculate_metrics(
    confusion_mtx,
    categories=class_indices_train_list,
    start_datetime=start_datetime_cnn,
    end_datetime=end_datetime_cnn,
    training_time=training_time_cnn,
    epochs=len(history_cnn.history['loss']),
    save=True,
    save_dir=resultados_dir
)
```

```python
# Función para mostrar predicciones
def show_predictions(model, generator, class_indices, num_images=5, save=False, save_dir=None):
    # Obtener un batch de imágenes y etiquetas
    generator.reset()
    images, labels = next(generator)
    predictions = model.predict(images)
    predicted_classes = np.argmax(predictions, axis=1)
    true_classes = np.argmax(labels, axis=1)
    class_labels = list(class_indices.keys())
    # Plot images with predictions
    plt.figure(figsize=(15, 15))
    for i in range(num_images):
        ax = plt.subplot(1, num_images, i + 1)
        plt.imshow(images[i])
        plt.title(f"Pred: {class_labels[predicted_classes[i]]}\nTrue: {class_labels[true_classes[i]]}")
        plt.axis("off")
    if save and save_dir is not None:
        predictions_filename = os.path.join(save_dir, f"predictions_{fecha_actual}.png")
        plt.savefig(predictions_filename)
        plt.close()
    plt.show()

# Mostrar y guardar predicciones
show_predictions(model_cnn, test_generator, class_indices_train, num_images=5, save=True, save_dir=resultados_dir)
```

ANALISIS EN TIEMPO REAL

Caja Delimitadora: Generación de Boundingbox para clasificar imágenes de MRI.

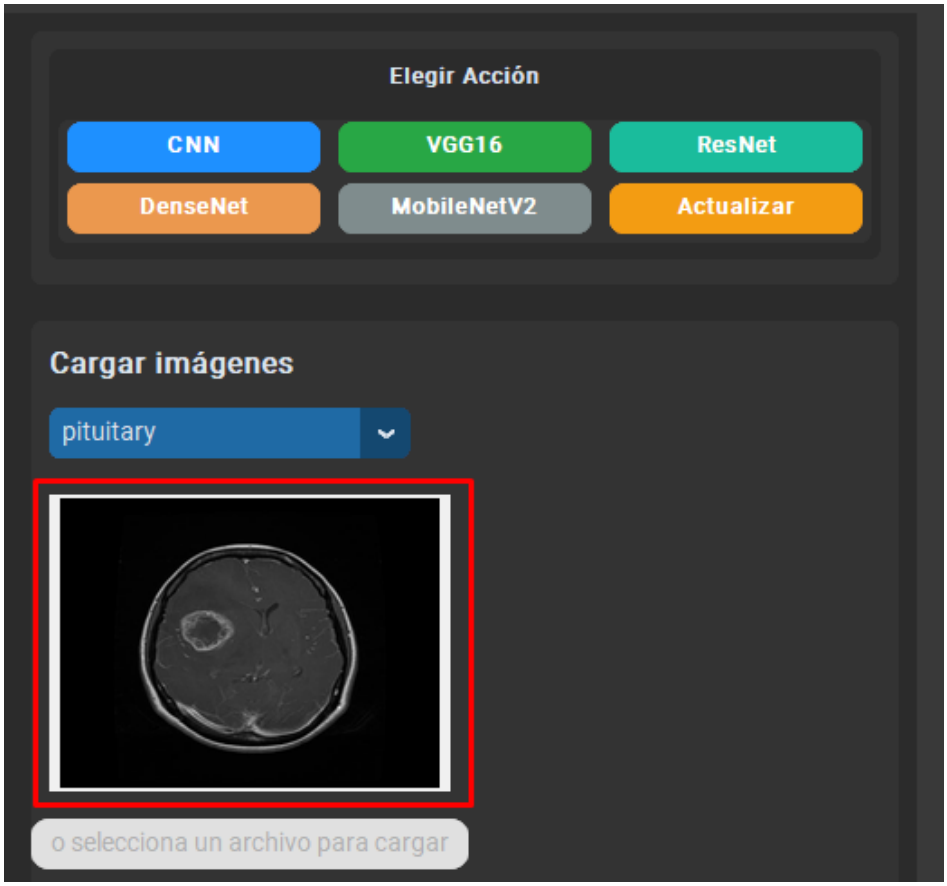


Figura 11: Generación de Boundingbox

1. Selección de una MRI
2. Seleccionar la etiqueta real
3. Seleccionar acción modelo
4. Resultado, exactitud fallida, porque la etiqueta real es errónea no coincide con la predicción del modelo por lo que en este caso la exactitud vendría a ser 0. Esto teniendo en cuenta que la predicción es correcta.

Figura 12: Clasificación del tumor y exactitud erróneo

1. Selección de una MRI
2. Seleccionar la etiqueta real
3. Seleccionar acción modelo
4. Resultado, exactitud correcta, porque la etiqueta real es correcta coincide con la predicción del modelo por lo que en este caso la exactitud vendría a ser 1. Esto teniendo en cuenta que la predicción es correcta.

Figura 13: Clasificación del tumor y exactitud correcto

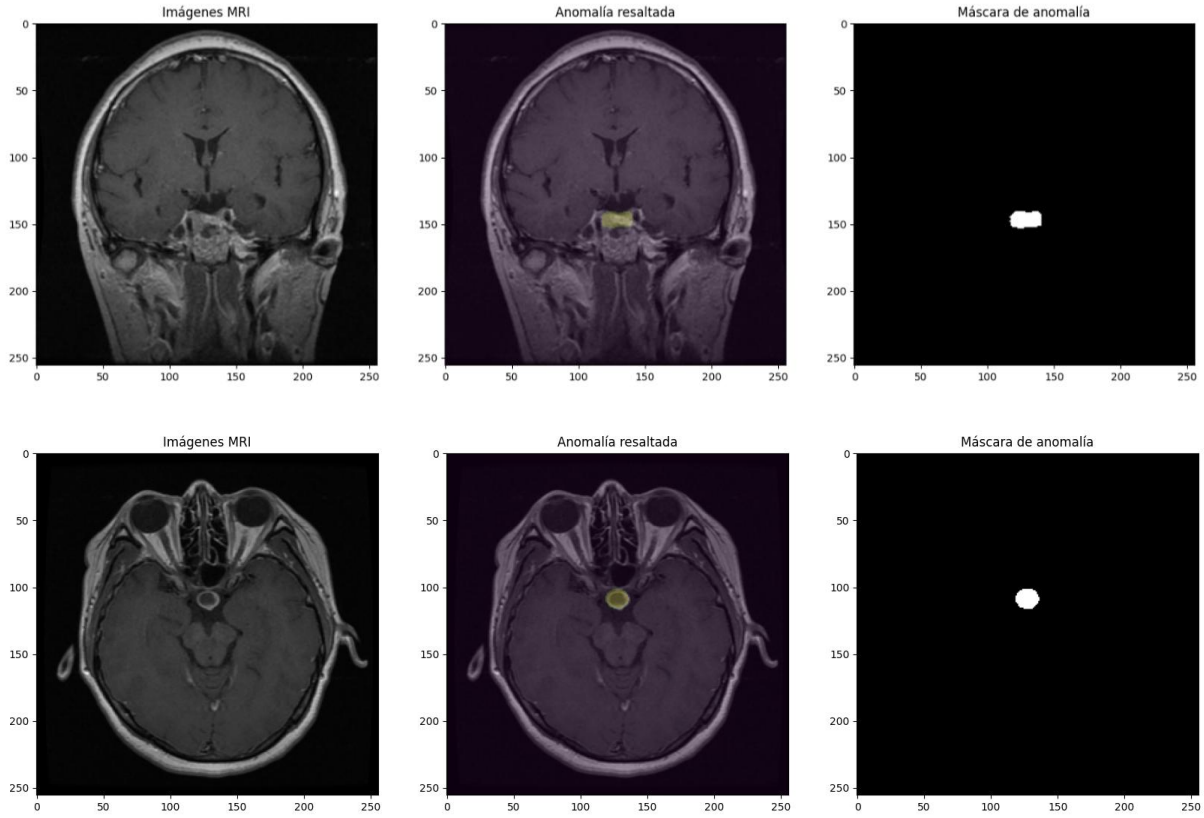
5. Aporte del Trabajo

Innovación en la segmentación:

Este trabajo propone un modelo basado en Deep Learning (U-Net) para segmentar con precisión tumores cerebrales en imágenes de resonancia magnética. Esto permite delimitar los bordes del tumor con alta exactitud, facilitando un diagnóstico visual claro para los especialistas médicos.

Localización precisa del tumor:

Además de segmentar el tumor, se presenta una solución para identificar en qué parte del cerebro se encuentra, lo que ayuda a los médicos a planificar tratamientos personalizados, como cirugías o radioterapias, de manera más efectiva.



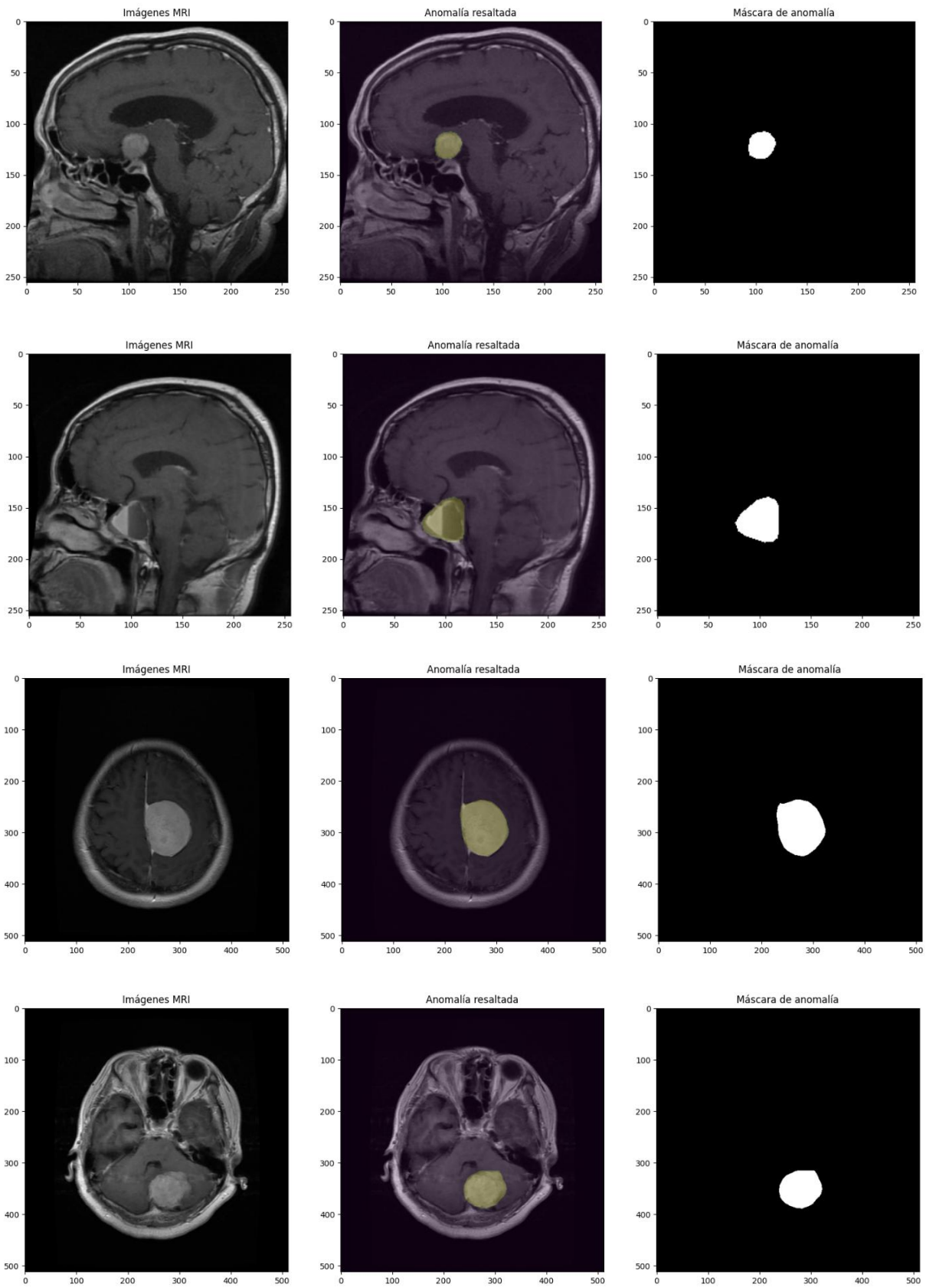


Figura 14: Detección de Anomalía con U-Net

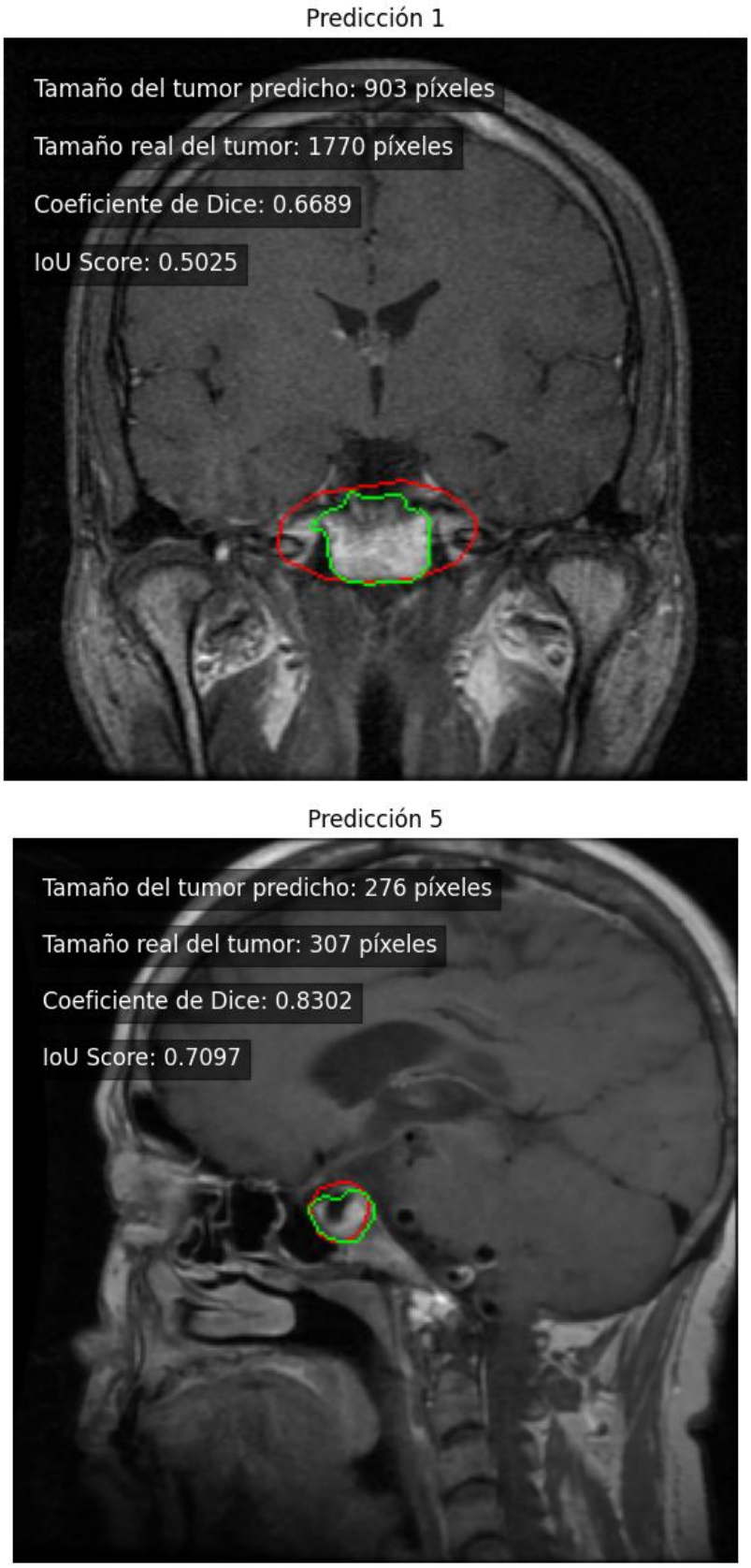


Figura 15: Predicción del tamaño del tumor

Desarrollo del algoritmo con U-Net

Archivo de datos (clasificación_cnn.py):

- Define la ruta a tu conjunto de datos y las clases

```
segmentacion_unet2.py ×
trained_model > segmentacion_unet2.py > ...

# Rutas de los directorios (ajusta las rutas según tu estructura de archivos)
#train_dir = r"C:\Users\Alegre\Documents\DeepLearning\Proyecto01\data3\training\glioma"
#train_dir = r"C:\Users\Alegre\Documents\DeepLearning\Proyecto01\data3\training\meningioma"
train_dir = r"C:\Users\Alegre\Documents\DeepLearning\Proyecto01\data3\training\pituitary"
masks_dir = r"C:\Users\Alegre\Documents\DeepLearning\Proyecto01\data3\masks"
resultados_base_dir = r"C:\Users\Alegre\Documents\DeepLearning\Proyecto01\Resultados\Segmentacion"
fecha_actual = datetime.datetime.now().strftime("%Y-%m-%d_%H-%M-%S")
resultados_dir = os.path.join(resultados_base_dir, 'U-Net', fecha_actual)
if not os.path.exists(resultados_dir):
    os.makedirs(resultados_dir)
```

ENTRENAR EL MODELO

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

PS C:\Users\Alegre\Documents\DeepLearning\Proyecto01\trained_model> python segmentacion_unet2.py
C:\Users\Alegre\Documents\DeepLearning\Proyecto01\venv\lib\site-packages\albumentations\__init__.py:13: UserWarning: A new version of Albumentations is available: 1.4.21 (you have 1.4.18). Upgrade using: pip install -U albumen
tations. To disable automatic update checks, set the environment variable NO_ALBUMENTATIONS_UPDATE to 1.
C:\Users\Alegre\Documents\DeepLearning\Proyecto01\venv\lib\site-packages\albumentations\__init__.py:13: UserWarning: A new version of Albumentations is available: 1.4.21 (you have 1.4.18). Upgrade using: pip install -U albumen
tations. To disable automatic update checks, set the environment variable NO_ALBUMENTATIONS_UPDATE to 1.
tations. To disable automatic update checks, set the environment variable NO_ALBUMENTATIONS_UPDATE to 1.
  check_for_updates()
Número total de imágenes con máscaras: 376
Número de imágenes en el conjunto de entrenamiento: 263
Número de imágenes en el conjunto de validación: 56
Número de imágenes en el conjunto de prueba: 57
Número de lotes en el DataLoader de entrenamiento: 33
Número de lotes en el DataLoader de validación: 7
Número de lotes en el DataLoader de prueba: 57
Ingrese el número de épocas para el entrenamiento: 40
Épocas:   0%|                                                                                                  | 0/40 [00:00<?, ?it/s]E
poch: 1/40 | Training loss: 0.1705 | Validation loss: 0.0870 | Validation Mean IoU: 0.2505 | Validation Dice coefficient: 0.3699
Épocas:   2%|                                                                                                  | 1/40 [01:03<41:35, 64.00s/it]E
poch: 2/40 | Training loss: 0.0460 | Validation loss: 0.0349 | Validation Mean IoU: 0.3725 | Validation Dice coefficient: 0.4746
Épocas:   5%|                                                                                                  | 2/40 [02:03<38:58, 61.54s/it]
```

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

Ingrese el número de épocas para el entrenamiento: 40
Ingrese el número de épocas para el entrenamiento: 40
Épocas:   0%|                                                                                                  | 0/40 [00:
00<?, ?it/s]Epoch: 1/40 | Training loss: 0.1705 | Validation loss: 0.0870 | Validation Mean IoU: 0.2505 | Validation Dice coefficient: 0.3699
poch: 1/40 | Training loss: 0.1705 | Validation loss: 0.0870 | Validation Mean IoU: 0.2505 | Validation Dice coefficient: 0.3699
Épocas:   2%|                                                                                                  | 1/40 [01:03<41:35
, 64.00s/it]Epoch: 2/40 | Training loss: 0.0460 | Validation loss: 0.0349 | Validation Mean IoU: 0.3725 | Validation Dice coefficient: 0.4746
Épocas:   5%|                                                                                                  | 2/40 [02:03<38:58
, 61.54s/it]Epoch: 3/40 | Training loss: 0.0245 | Validation loss: 0.0244 | Validation Mean IoU: 0.4707 | Validation Dice coefficient: 0.5743
Épocas:   8%|                                                                                                  | 3/40 [03:04<37:45
, 61.24s/it]Epoch: 4/40 | Training loss: 0.0162 | Validation loss: 0.0214 | Validation Mean IoU: 0.5585 | Validation Dice coefficient: 0.7103
Épocas:  10%|                                                                                                  | 4/40 [04:02<35:59
, 59.99s/it]Epoch: 5/40 | Training loss: 0.0136 | Validation loss: 0.0134 | Validation Mean IoU: 0.6894 | Validation Dice coefficient: 0.7957
Épocas:  12%|                                                                                                  | 5/40 [05:00<34:25
, 59.00s/it]Epoch: 6/40 | Training loss: 0.0107 | Validation loss: 0.0124 | Validation Mean IoU: 0.6873 | Validation Dice coefficient: 0.8011
Épocas:  15%|                                                                                                  | 6/40 [05:56<33:00
, 58.25s/it]Epoch: 7/40 | Training loss: 0.0087 | Validation loss: 0.0118 | Validation Mean IoU: 0.7036 | Validation Dice coefficient: 0.8184
Épocas:  18%|                                                                                                  | 7/40 [06:53<31:46
, 57.76s/it]Epoch: 8/40 | Training loss: 0.0077 | Validation loss: 0.0123 | Validation Mean IoU: 0.7030 | Validation Dice coefficient: 0.8068
Épocas:  20%|                                                                                                  | 8/40 [07:51<30:47
, 57.75s/it]Epoch: 9/40 | Training loss: 0.0062 | Validation loss: 0.0126 | Validation Mean IoU: 0.6877 | Validation Dice coefficient: 0.7881
Épocas:  22%|                                                                                                  | 9/40 [08:49<29:58
, 58.00s/it]Epoch: 10/40 | Training loss: 0.0058 | Validation loss: 0.0119 | Validation Mean IoU: 0.6996 | Validation Dice coefficient: 0.7996
Épocas:  25%|                                                                                                  | 10/40 [09:46<28:50
, 57.69s/it]Epoch: 11/40 | Training loss: 0.0050 | Validation loss: 0.0109 | Validation Mean IoU: 0.7244 | Validation Dice coefficient: 0.8222
Épocas:  28%|                                                                                                  | 11/40 [10:43<27:44
, 57.39s/it]Epoch: 12/40 | Training loss: 0.0048 | Validation loss: 0.0110 | Validation Mean IoU: 0.7221 | Validation Dice coefficient: 0.8203
Épocas:  30%|                                                                                                  | 12/40 [11:39<26:38
  57.08s/it]
```

INTEGRAR EL MODELO U-NET

segmentacion_unet2.py ×

trained_model > segmentacion_unet2.py > ...

```python
# python segmentacion_unet2.py
import os
import time
import datetime
from glob import glob
from tqdm import tqdm, trange

import cv2
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import albumentations as A
from scipy.ndimage import binary_dilation  # Importación actualizada
import segmentation_models_pytorch as smp
from sklearn.model_selection import train_test_split

import torch
from torch.optim import Adam
from torch.optim.lr_scheduler import ReduceLROnPlateau
from torch.utils.data import Dataset, DataLoader
from torchvision import transforms as T

# Configuración del dispositivo (CPU o GPU)
device = torch.device("cpu")  # Forzar el uso de la CPU

# Rutas de los directorios (ajusta las rutas según tu estructura de archivos)
#train_dir = r"C:\Users\Alegre\Documents\DeepLearning\Proyecto01\data3\training\glioma"
#train_dir = r"C:\Users\Alegre\Documents\DeepLearning\Proyecto01\data3\training\meningioma"
train_dir = r"C:\Users\Alegre\Documents\DeepLearning\Proyecto01\data3\training\pituitary"
masks_dir = r"C:\Users\Alegre\Documents\DeepLearning\Proyecto01\data3\masks"
resultados_base_dir = r"C:\Users\Alegre\Documents\DeepLearning\Proyecto01\Resultados\Segmentacion"
fecha_actual = datetime.datetime.now().strftime("%Y-%m-%d_%H-%M-%S")
resultados_dir = os.path.join(resultados_base_dir, 'U-Net', fecha_actual)
if not os.path.exists(resultados_dir):
    os.makedirs(resultados_dir)

# Crear un DataFrame que mapea imágenes con sus máscaras correspondientes
image_filenames = os.listdir(train_dir)
image_filenames = [f for f in image_filenames if os.path.isfile(os.path.join(train_dir, f))]

data = []

for image_filename in image_filenames:
    image_path = os.path.join(train_dir, image_filename)
    # Asumimos que las máscaras tienen el mismo nombre que las imágenes
    mask_path = os.path.join(masks_dir, image_filename)
    if not os.path.exists(mask_path):
        # Si no existe, intentamos con '_mask' añadido al nombre
        name, ext = os.path.splitext(image_filename)
        mask_filename = f"{name}_mask{ext}"
        mask_path = os.path.join(masks_dir, mask_filename)
        if not os.path.exists(mask_path):
            # Si la máscara no se encuentra, omitimos esta imagen
            print(f"Máscara no encontrada para la imagen {image_filename}")
            continue
    data.append({'image_filename': image_path, 'mask_filename': mask_path})

df = pd.DataFrame(data)

# Verificar el número de imágenes
print(f"Número total de imágenes con máscaras: {len(df)}")
```

```python

# Dividir el DataFrame en conjuntos de entrenamiento, validación y prueba
train_df, temp_df = train_test_split(df, test_size=0.3, random_state=42)
valid_df, test_df = train_test_split(temp_df, test_size=0.5, random_state=42)

print(f"Número de imágenes en el conjunto de entrenamiento: {len(train_df)}")
print(f"Número de imágenes en el conjunto de validación: {len(valid_df)}")
print(f"Número de imágenes en el conjunto de prueba: {len(test_df)}")

# Definir la clase Dataset
class MriDataset(Dataset):
    def __init__(self, df, transform=None, mean=0.5, std=0.25, image_size=(256, 256)):
        super(MriDataset, self).__init__()
        self.df = df.reset_index(drop=True)
        self.transform = transform
        self.mean = mean
        self.std = std
        self.image_size = image_size  # Nuevo: Tamaño al que redimensionaremos las imágenes

    def __len__(self):
        return len(self.df)

    def __getitem__(self, idx, raw=False):
        row = self.df.iloc[idx]
        img = cv2.imread(row['image_filename'], cv2.IMREAD_UNCHANGED)
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        mask = cv2.imread(row['mask_filename'], cv2.IMREAD_GRAYSCALE)

        # Redimensionar imágenes y máscaras al tamaño especificado
        img = cv2.resize(img, self.image_size, interpolation=cv2.INTER_AREA)
        mask = cv2.resize(mask, self.image_size, interpolation=cv2.INTER_NEAREST)

        if raw:
            return img, mask

        if self.transform:
            augmented = self.transform(image=img, mask=mask)
            img, mask = augmented['image'], augmented['mask']

        img = T.functional.to_tensor(img)
        mask = mask / 255.0  # Asegurarse de que la máscara está en el rango [0, 1]
        mask = torch.tensor(mask, dtype=torch.float32)
        return img, mask
```

```python


# Definir las transformaciones de datos
transform = A.Compose([
    A.ChannelDropout(p=0.3),
    A.RandomBrightnessContrast(p=0.3),
    A.ColorJitter(p=0.3),
])

# Crear instancias de los datasets
train_dataset = MriDataset(train_df, transform)
valid_dataset = MriDataset(valid_df)
test_dataset = MriDataset(test_df)

batch_size = 8  # Reducir el tamaño del lote
```

```python
# Crear los DataLoaders
train_loader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True)
valid_loader = DataLoader(valid_dataset, batch_size=batch_size, shuffle=False)
test_loader = DataLoader(test_dataset, batch_size=1, shuffle=False)

# Verificar el número de lotes
print(f"Número de lotes en el DataLoader de entrenamiento: {len(train_loader)}")
print(f"Número de lotes en el DataLoader de validación: {len(valid_loader)}")
print(f"Número de lotes en el DataLoader de prueba: {len(test_loader)}")

# Visualizar algunos ejemplos del dataset y guardarlos
n_examples = 4

for idx in range(n_examples):
    image, mask = train_dataset.__getitem__(idx, raw=True)
    if np.any(mask):
        fig, axs = plt.subplots(1, 3, figsize=(20, 7))
        axs[0].set_title("Imágenes MRI")
        axs[0].imshow(image)
        axs[1].set_title("Anomalía resaltada")
        axs[1].imshow(image)
        axs[1].imshow(mask, alpha=0.2)
        axs[2].imshow(mask, cmap='gray')
        axs[2].set_title("Máscara de anomalía")
        # Guardar la figura
        figura_path = os.path.join(resultados_dir, f'ejemplo_{idx}.png')
        plt.savefig(figura_path)
        plt.close(fig)  # Cerrar la figura para liberar memoria

# Definir funciones auxiliares para entrenamiento y evaluación
class EarlyStopping():
    """
    Detiene el entrenamiento cuando la pérdida deja de disminuir.
    """
    def __init__(self, patience: int = 7, min_delta: float = 0, weights_path: str = 'weights.pt'):
        self.patience = patience
        self.min_delta = min_delta
        self.counter = 0
        self.best_loss = float('inf')
        self.weights_path = weights_path

    def __call__(self, val_loss: float, model: torch.nn.Module):
```

```python
        if self.best_loss - val_loss > self.min_delta:
            self.best_loss = val_loss
            torch.save(model.state_dict(), self.weights_path)
            self.counter = 0
        else:
            self.counter += 1
            if self.counter >= self.patience:
                return True
        return False

    def load_weights(self, model: torch.nn.Module):
        model.load_state_dict(torch.load(self.weights_path))

def iou_pytorch(predictions: torch.Tensor, labels: torch.Tensor, e: float = 1e-7):
    predictions = torch.where(predictions > 0.5, 1, 0)
    labels = labels.byte()
    intersection = (predictions & labels).float().sum((1, 2))
    union = (predictions | labels).float().sum((1, 2))
    iou = (intersection + e) / (union + e)
    return iou
```

```python

def dice_pytorch(predictions: torch.Tensor, labels: torch.Tensor, e: float = 1e-7):
    predictions = torch.where(predictions > 0.5, 1, 0)
    labels = labels.byte()
    intersection = (predictions & labels).float().sum((1, 2))
    return ((2 * intersection) + e) / (predictions.float().sum((1, 2)) + labels.float().sum((1, 2)) + e)

def BCE_dice(output, target, alpha=0.01):
    bce = torch.nn.functional.binary_cross_entropy(output, target)
    soft_dice = 1 - dice_pytorch(output, target).mean()
    return bce + alpha * soft_dice

# Definir el modelo
model = smp.Unet(
    encoder_name="resnet18",  # Modelo más ligero para reducir consumo de recursos
    encoder_weights="imagenet",
    in_channels=3,
    classes=1,
    activation='sigmoid',
)
model.to(device)

# Definir el ciclo de entrenamiento
def training_loop(epochs, model, train_loader, valid_loader, optimizer, loss_fn, lr_scheduler):
    history = {'train_loss': [], 'val_loss': [], 'val_IoU': [], 'val_dice': []}
    early_stopping = EarlyStopping(patience=7)

    # Registrar la hora de inicio del entrenamiento
    training_start_time = datetime.datetime.now()

    for epoch in trange(1, epochs + 1, desc='Épocas'):
        start_time = time.time()

        running_loss = 0
        model.train()
        for data in train_loader:
            img, mask = data
            img, mask = img.to(device), mask.to(device)
            optimizer.zero_grad()
            predictions = model(img)
            predictions = predictions.squeeze(1)
            loss = loss_fn(predictions, mask)
            running_loss += loss.item() * img.size(0)
```

```python
            loss.backward()
            optimizer.step()

        model.eval()
        with torch.no_grad():
            running_IoU = 0
            running_dice = 0
            running_valid_loss = 0
            for data in valid_loader:
                img, mask = data
                img, mask = img.to(device), mask.to(device)
                predictions = model(img)
                predictions = predictions.squeeze(1)
                running_dice += dice_pytorch(predictions, mask).sum().item()
                running_IoU += iou_pytorch(predictions, mask).sum().item()
                loss = loss_fn(predictions, mask)
                running_valid_loss += loss.item() * img.size(0)

        train_loss = running_loss / len(train_loader.dataset)
        val_loss = running_valid_loss / len(valid_loader.dataset)
        val_dice = running_dice / len(valid_loader.dataset)
        val_IoU = running_IoU / len(valid_loader.dataset)

        history['train_loss'].append(train_loss)
        history['val_loss'].append(val_loss)
        history['val_IoU'].append(val_IoU)
        history['val_dice'].append(val_dice)
        print(f'Epoch: {epoch}/{epochs} | Training loss: {train_loss:.4f} | Validation loss: {val_loss:.4f} | Validation Mean IoU: {val_IoU:.4f} '
              f'| Validation Dice coefficient: {val_dice:.4f}')

        lr_scheduler.step(val_loss)
        if early_stopping(val_loss, model):
            early_stopping.load_weights(model)
            break
```

```python
    # Registrar la hora de finalización del entrenamiento
    training_end_time = datetime.datetime.now()
    training_duration = training_end_time - training_start_time
    training_time_minutes = training_duration.total_seconds() / 60  # Convertir a minutos

    model.eval()
    return history, training_start_time, training_end_time, training_time_minutes, epoch

# Configurar el entrenamiento
loss_fn = BCE_dice
optimizer = Adam(model.parameters(), lr=0.001)

# Solicitar al usuario que ingrese el número de épocas con validación
while True:
    try:
        epochs = int(input("Ingrese el número de épocas para el entrenamiento: "))
        if epochs <= 0:
            print("Por favor, ingrese un número entero positivo.")
            continue
        break
    except ValueError:
        print("Entrada inválida. Por favor, ingrese un número entero.")

lr_scheduler = ReduceLROnPlateau(optimizer=optimizer, patience=2, factor=0.2)

# Entrenar el modelo
history, training_start_time, training_end_time, training_time_minutes, total_epochs = training_loop(
    epochs, model, train_loader, valid_loader, optimizer, loss_fn, lr_scheduler
)

# Graficar la pérdida durante el entrenamiento y guardarla
plt.figure(figsize=(7, 7))
plt.plot(history['train_loss'], label='Pérdida de entrenamiento')
plt.plot(history['val_loss'], label='Pérdida de validación')
plt.ylim(0, 1)
plt.legend()
loss_plot_path = os.path.join(resultados_dir, 'pérdida_entrenamiento.png')
plt.savefig(loss_plot_path)
plt.close()
```

```python

# Graficar métricas de validación y guardarlas
plt.figure(figsize=(7, 7))
plt.plot(history['val_IoU'], label='Índice de Jaccard (IoU) de validación')
plt.plot(history['val_dice'], label='Coeficiente de Dice de validación')
plt.legend()
metrics_plot_path = os.path.join(resultados_dir, 'métricas_validación.png')
plt.savefig(metrics_plot_path)
plt.close()

# Evaluar el modelo en el conjunto de prueba
with torch.no_grad():
    running_IoU = 0
    running_dice = 0
    running_loss = 0
    for data in test_loader:
        img, mask = data
        img, mask = img.to(device), mask.to(device)
        predictions = model(img)
        predictions = predictions.squeeze(1)
        running_dice += dice_pytorch(predictions, mask).sum().item()
        running_IoU += iou_pytorch(predictions, mask).sum().item()
        loss = loss_fn(predictions, mask)
        running_loss += loss.item() * img.size(0)
    loss = running_loss / len(test_dataset)
    dice = running_dice / len(test_dataset)
    IoU = running_IoU / len(test_dataset)

    print(f'Pruebas: pérdida: {loss:.4f} | Mean IoU: {IoU:.4f} | Coeficiente de Dice: {dice:.4f}')
```

```python

# Guardar las métricas en un archivo de texto
with open(os.path.join(resultados_dir, 'resultados.txt'), 'w') as f:
    f.write(f'Pruebas: pérdida: {loss:.4f} | Mean IoU: {IoU:.4f} | Coeficiente de Dice: {dice:.4f}\n')

# Crear una tabla con la información del entrenamiento y guardarla como imagen
training_info = pd.DataFrame({
    'Inicio': [training_start_time.strftime("%Y-%m-%d %H:%M:%S")],
    'Fin': [training_end_time.strftime("%Y-%m-%d %H:%M:%S")],
    'Tiempo (minutos)': [round(training_time_minutes, 2)],
    'Épocas': [total_epochs]
})

training_table_filename = os.path.join(resultados_dir, f"training_info_{fecha_actual}.png")
fig, ax = plt.subplots(figsize=(10, 2))
ax.axis('off')
tabla = ax.table(cellText=training_info.values, colLabels=training_info.columns, loc='center')
tabla.scale(1, 2)
plt.savefig(training_table_filename, bbox_inches='tight')
plt.close()

# Visualizar algunas predicciones en el conjunto de prueba y guardarlas
n_examples = 15  # Número total de ejemplos a guardar

for i, data in enumerate(test_loader):
    if i >= n_examples:
        break
    image, mask = data
    mask = mask[0]
    if not mask.byte().any():
        continue
    image = image.to(device)
    prediction = model(image).cpu()[0][0]
    prediction_thresholded = torch.where(prediction > 0.5, 1, 0)
    prediction_edges = prediction_thresholded - binary_dilation(prediction_thresholded)
    ground_truth = mask - binary_dilation(mask)
    image = image.cpu()[0]
    # Crear una copia de la imagen para visualizar
    image_vis = image.clone()
    # Marcar ground truth en rojo
    image_vis[0, ground_truth.bool()] = 1  # Canal R
    image_vis[1, ground_truth.bool()] = 0  # Canal G
    image_vis[2, ground_truth.bool()] = 0  # Canal B
```

```python
    # Marcar predicción en verde
    image_vis[0, prediction_edges.bool()] = 0  # Canal R
    image_vis[1, prediction_edges.bool()] = 1  # Canal G
    image_vis[2, prediction_edges.bool()] = 0  # Canal B

    # Calcular el tamaño del tumor predicho y las métricas de precisión
    predicted_tumor_size = prediction_thresholded.sum().item()
    actual_tumor_size = mask.sum().item()
    dice_coef = dice_pytorch(prediction_thresholded.unsqueeze(0), mask.unsqueeze(0)).item()
    iou_score = iou_pytorch(prediction_thresholded.unsqueeze(0), mask.unsqueeze(0)).item()

    plt.figure(figsize=(7, 7))
    plt.imshow(image_vis.permute(1, 2, 0))
    plt.title(f'Predicción {i+1}')
    plt.axis('off')

    # Añadir texto con el tamaño del tumor y las métricas de precisión
    plt.text(10, 20, f'Tamaño del tumor predicho: {predicted_tumor_size:.0f} píxeles', color='white', fontsize=12, bbox=dict(facecolor='black', alpha=0.5))
    plt.text(10, 40, f'Tamaño real del tumor: {actual_tumor_size:.0f} píxeles', color='white', fontsize=12, bbox=dict(facecolor='black', alpha=0.5))
    plt.text(10, 60, f'Coeficiente de Dice: {dice_coef:.4f}', color='white', fontsize=12, bbox=dict(facecolor='black', alpha=0.5))
    plt.text(10, 80, f'IoU Score: {iou_score:.4f}', color='white', fontsize=12, bbox=dict(facecolor='black', alpha=0.5))

    # Guardar la figura
    pred_path = os.path.join(resultados_dir, f'predicción_{i+1}.png')
    plt.savefig(pred_path)
    plt.close()

```