UNIVERSIDAD INTERNACIONAL SEK

FACULTAD BUSINESS & DIGITAL SCHOOL

Trabajo de Titulación:

"DISEÑO E IMPLEMENTACIÓN DE UNA APLICACIÓN WEB CON MACHINE LEARNING PARA LA DETECCIÓN TEMPRANA DE TUMORES CEREBRALES MEDIANTE RESONANCIAS MAGNÉTICAS"

Realizado por:

MARIO ESTHEFANO GALLO JURADO

Director del proyecto:

PHD. JOE CARRIÓN

Como requisito para la obtención del título de:

INGENIERO EN SISTEMAS DE INFORMACIÓN CON ÉNFASIS EN CONTENIDOS INTERACTIVOS

Quito, Julio 2025

DECLARACIÓN JURAMENTADA

Yo, Mario Esthefano Gallo Jurado con cédula de identidad No.1722464656, declaro bajo juramento que el trabajo aquí desarrollado es de mi autoría, que no ha sido previamente presentado para ningún grado a calificación profesional; y que ha consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración, cedo mis derechos de propiedad intelectual correspondiente a este trabajo, a la UNIVERSIDAD INTERNACIONAL SEK, según lo establecido por la Ley de Propiedad Intelectual, por su reglamento y por la normativa institucional vigente.

Mario Esthefano Gallo Jurado

C.C: 1722464656

DECLARATORIA

El presente Trabajo de investigación titulado:

“DISEÑO E IMPLEMENTACIÓN DE UNA APLICACIÓN WEB CON MACHINE LEARNING PARA LA DETECCIÓN TEMPRANA DE TUMORES CEREBRALES MEDIANTE RESONANCIAS MAGNÉTICAS”

Realizado por:
MARIO ESTHEFANO GALLO JURADO

Como requisito para la obtención del título de:
INGENIERO EN SISTEMAS

Ha sido dirigido por el profesor:
PhD**. JOE CARRIÓN JUMBO**
Quien considera que constituye un trabajo original de su autor

PHD. JOE CARRIÓN JUMBO
DIRECTOR

LOS PROFESORES INFORMANTES:

PhD. VIVIANA ELIZABETH CAJAS CAJAS

MSc. VERÓNICA ELIZABETH RODRIGUEZ ARBOLEDA

Después de revisar el trabajo presentado lo han calificado como apto para su defensa oral ante el tribunal examinador.

PhD. Viviana Cajas

Ing. Verónica Rodríguez , MBA

DEDICATORIA

Dedico este trabajo, en primer lugar, a mis padres, quienes han sido mi pilar fundamental a lo largo de mi vida. A mi madre, por enseñarme que la constancia y la humildad son la base de todo logro, y por estar siempre dispuesta a brindarme una palabra de aliento en los momentos más difíciles. A mi padre, por su ejemplo de responsabilidad, disciplina y trabajo duro, que me ha inspirado a dar lo mejor de mí en cada etapa de este camino académico.

A mi familia en general, por creer en mis capacidades incluso cuando yo mismo dudaba, y por acompañarme con paciencia, comprensión y apoyo incondicional. Su confianza ha sido la fuerza que me ha impulsado a superar los obstáculos y a mantenerme firme en mis objetivos. A mis amigos y compañeros, quienes compartieron conmigo largas horas de estudio, retos y risas. Gracias por hacer de esta etapa no solo un desafío intelectual, sino también un tiempo lleno de recuerdos y aprendizajes compartidos.

Dedico también este logro a todas las personas que, directa o indirectamente, han aportado su conocimiento, tiempo o palabras de motivación para que este proyecto se hiciera realidad. A mis docentes, por transmitirme no solo conocimientos técnicos, sino también la pasión por la investigación, la innovación y el compromiso social.

AGRADECIMIENTO

Deseo expresar mi más sincero agradecimiento a mi supervisor, Joe Carrión, por su guía y acompañamiento durante todo este proceso, lo que me permitió culminar este proyecto de la mejor manera.

Asimismo, extiendo mi gratitud a mi madre, Jenny Jurado, por su constante apoyo a lo largo de mi trayectoria académica y por estar presente en mis momentos más difíciles.
A mi padre, Bladimir Gallo, le agradezco profundamente por su respaldo y ayuda a lo largo de mi formación.

También reconozco con gratitud a todos mis profesores, quienes contribuyeron significativamente en mi desarrollo como estudiante.

Finalmente, agradezco a la Universidad Internacional SEK, institución que me brindó los conocimientos necesarios y me formó como persona durante estos años de preparación académica.

RESUMEN

El presente trabajo de titulación desarrolla una aplicación web orientada a la detección temprana de tumores cerebrales mediante el uso de técnicas de inteligencia artificial y visión por computadora. El sistema se basa en un modelo de redes neuronales convolucionales (CNN) entrenado con imágenes de resonancia magnética (MRI), utilizando la arquitectura VGG16 para la clasificación en cuatro categorías: glioma, meningioma, tumor pituitario y ausencia de tumor. El desarrollo técnico comprende un backend implementado con Flask en Python, un frontend construido con HTML, CSS y Bootstrap, y un módulo de análisis ejecutado en Jupyter Notebook. Además de ofrecer un diagnóstico preliminar, la interfaz integra recursos educativos, como un modelo 3D interactivo del cerebro y recomendaciones personalizadas según el resultado. Los resultados obtenidos evidencian un rendimiento satisfactorio del modelo, destacando su potencial para servir como herramienta de apoyo en contextos con limitado acceso a especialistas y equipos de diagnóstico avanzado.

Palabras clave: inteligencia artificial, tumores cerebrales, redes neuronales convolucionales, resonancia magnética, Flask.

ABSTRACT

This research project presents the development of a web application for the early detection of brain tumors using artificial intelligence and computer vision techniques. The system is based on a convolutional neural network (CNN) model trained with magnetic resonance imaging (MRI) scans, applying the VGG16 architecture to classify images into four categories: glioma, meningioma, pituitary tumor, and no tumor. The technical implementation includes a backend developed with Flask in Python, a frontend designed with HTML, CSS, and Bootstrap, and an analysis module built in Jupyter Notebook. In addition to providing a preliminary diagnosis, the interface incorporates educational resources, such as an interactive 3D brain model and personalized recommendations based on the output. The results demonstrate satisfactory model performance, highlighting its potential as a support tool in scenarios with limited access to specialists and advanced diagnostic equipment.

Keywords: artificial intelligence, brain tumors, convolutional neural networks, magnetic resonance imaging, Flask.

CAPÍTULO 1

INTRODUCCIÓN

1.1 PLANTEAMIENTO DEL PROBLEMA

En el contexto del sistema de salud en el Ecuador, se evidencia una creciente problemática relacionada con la saturación de servicios médicos especializados, particularmente en el área de neurología y diagnóstico por imágenes. Esta situación genera largos tiempos de espera para la obtención de citas, retrasando diagnósticos cruciales y, en consecuencia, comprometiendo la atención oportuna de enfermedades graves como los tumores cerebrales (Colegio de Médicos de Pichincha, 2025).

Además, muchas personas no cuentan con los recursos económicos suficientes para acceder a servicios privados de imagenología y diagnóstico especializado, lo que limita aún más su acceso a una atención médica adecuada y temprana. Esta combinación de colapso del sistema público y barrera económica en el sistema privado genera un entorno de vulnerabilidad para miles de ecuatorianos, especialmente en zonas rurales y de escasos recursos.

Por otro lado, el diagnóstico de tumores cerebrales requiere no solo equipamiento técnico, sino también personal altamente capacitado, cuya disponibilidad es limitada y centralizada en las grandes ciudades. Esto agrava la brecha entre quienes pueden acceder a un diagnóstico temprano y quienes no.

En este contexto, el avance de las tecnologías de inteligencia artificial, y en particular del machine learning (ML) aplicado a imágenes médicas, presenta una oportunidad concreta y viable para mitigar esta problemática. Sin embargo, actualmente en el país no existe una solución accesible y automatizada que permita a los usuarios subir sus propias tomografías y recibir una predicción preliminar sobre la presencia de tumores cerebrales.

Por tanto, se plantea como problema central la falta de una herramienta tecnológica accesible y automatizada que permita realizar una evaluación temprana de tomografías cerebrales, reduciendo la dependencia exclusiva del diagnóstico tradicional y permitiendo una atención más ágil, especialmente para sectores desatendidos del país.

1.2 JUSTIFICACIÓN

La detección temprana de enfermedades neurológicas, como los tumores cerebrales, representa un factor determinante en la tasa de supervivencia y en la calidad de vida de los pacientes. En contextos donde el acceso a servicios de salud especializados es limitado o insuficiente, como ocurre en diversas regiones de Ecuador, la implementación de herramientas tecnológicas basadas en inteligencia artificial puede marcar una diferencia significativa en la atención médica. Este proyecto propone el diseño e implementación de una aplicación web que, mediante algoritmos de *ML*, analice imágenes de tomografía cerebral para predecir la posible presencia de tumores cerebrales. Esta propuesta responde a una problemática concreta del sistema de salud pública del país: la saturación del sistema de salud, la falta de especialistas en neurología y el acceso limitado a diagnósticos oportunos.

Según datos presentados por Moreno-Zambrano et al. (2023) en la Revista Ecuatoriana de Neurología, hasta el año de publicación del estudio, solo existían 94 neurólogos en todo el sistema público de salud, de los cuales el 39,4 % se encuentran en la provincia de Pichincha y el 25,5 % en Guayas. Esto implica que más del 85 % de los especialistas están concentrados únicamente en dos provincias, dejando a otras ocho sin ningún neurólogo en el sistema público. Esta distribución desigual constituye una barrera crítica para el diagnóstico oportuno de patologías complejas como los tumores cerebrales, especialmente en poblaciones rurales y marginadas del país.

La OMS (2022) ha destacado que el cáncer, en general, es una de las principales causas de muerte a nivel mundial. Dentro de las neoplasias, los tumores cerebrales, aunque menos comunes que otros tipos de cáncer, presentan una alta tasa de mortalidad y morbilidad debido a su ubicación y complejidad para ser tratados. En América Latina, se estima que los tumores del sistema nervioso central representan aproximadamente el 2 % de todos los cánceres diagnosticados anualmente, con un comportamiento clínico agresivo y consecuencias funcionales significativas. En Ecuador, las estadísticas disponibles son limitadas, pero se estima que la incidencia de tumores cerebrales puede oscilar entre 5 y 10 casos por cada 100.000 habitantes (Gamma Knife Ecuador, 2022).

Ante esta realidad, se hace evidente la necesidad de desarrollar soluciones innovadoras que complementen y fortalezcan la capacidad diagnóstica del sistema sanitario. La inteligencia artificial, particularmente el aprendizaje automático (*machine learning*), ha demostrado en múltiples investigaciones científicas su potencial para detectar patrones en imágenes médicas con alta precisión.

Estudios como el de Siddique et al. (2020) han reportado una precisión del 96 % en la detección de tumores cerebrales mediante redes neuronales convolucionales profundas aplicadas a imágenes de resonancia magnética. Estas tecnologías permiten no solo acelerar el proceso diagnóstico, sino también reducir errores humanos, estandarizar criterios clínicos y facilitar el acceso remoto a análisis preliminares.

Del mismo modo, la noticia de la revista médica ecancer.org (2021) destaca la complejidad que representa la evolución epidemiológica de los tumores cerebrales y de otros tumores del sistema nervioso central (SNC) en los Estados Unidos. Aunque el estudio revela una disminución anual del 0,8% en la incidencia de tumores malignos entre 2008 y 2017 en la población general, también señala un incremento preocupante en la incidencia entre niños y adolescentes, lo que evidencia una problemática diferenciada según la edad. Asimismo, a pesar de los avances en investigación

y de la mejora en las tasas de supervivencia en comparación con décadas pasadas, el pronóstico sigue siendo desfavorable.

Estas cifras reflejan la persistente dificultad en la detección temprana y el limitado impacto de los tratamientos actuales, lo que convierte a los tumores cerebrales en una de las enfermedades oncológicas de mayor letalidad y carga social. Además, el informe resalta disparidades significativas por sexo, raza y edad, lo cual evidencia la necesidad de impulsar estrategias de investigación y políticas de salud más equitativas que permitan comprender mejor la etiología de estos tumores y, en consecuencia, mejorar las intervenciones médicas y preventivas.

El presente proyecto busca aprovechar estas capacidades técnicas para democratizar el acceso a una evaluación inicial confiable a través de una plataforma web, sin necesidad de contar con equipos especializados de alto costo o la presencia física de un neurólogo. Mediante la carga de imágenes de tomografías por parte del usuario, el sistema podrá predecir la probabilidad de presencia de un tumor cerebral con base en modelos previamente entrenados y validados. Aunque esta herramienta no reemplazará el diagnóstico médico definitivo, funcionará como un sistema de apoyo a la decisión clínica y como una alerta temprana que podría motivar una consulta médica inmediata.

Una justificación adicional de este proyecto es su alineación con los Objetivos de Desarrollo Sostenible (ODS), específicamente el ODS 3: Salud y Bienestar, que promueve el acceso equitativo a servicios de salud esenciales para toda la población, y el ODS 9: Industria, Innovación e Infraestructura, al fomentar el uso de tecnologías emergentes para resolver problemas estructurales. El desarrollo de esta aplicación web representa un ejemplo claro de cómo la innovación tecnológica puede ser aplicada en beneficio del bien público, generando impactos positivos en la salud, la equidad social y la eficiencia institucional.

Desde un enfoque metodológico, la aplicación se apoyará en bibliotecas de ML como TensorFlow y Scikit-learn, que han sido ampliamente validadas en entornos científicos. Las imágenes de entrenamiento serán obtenidas de bases de datos médicas de acceso abierto como BraTS (Brain Tumor Segmentation), lo que asegura la calidad y diversidad de los casos utilizados para entrenar el modelo. La plataforma será accesible desde cualquier navegador, facilitando su implementación en zonas con conectividad básica.

Cabe mencionar que, a nivel regional, existen aplicaciones y sistemas similares en países con mayores niveles de inversión en salud digital, como México, Colombia o Brasil. Sin embargo, la mayoría de estas soluciones están desarrolladas para contextos clínicos cerrados o integradas en sistemas hospitalarios privados. En Ecuador no existe, hasta la fecha, una herramienta pública o de libre acceso que ofrezca este tipo de funcionalidad basada en IA para la detección de tumores cerebrales. Esta carencia representa tanto un vacío como una oportunidad para implementar una solución local que responda a las necesidades y limitaciones del sistema ecuatoriano.

En definitiva, la propuesta de desarrollar una aplicación web para la detección temprana de tumores cerebrales mediante inteligencia artificial se justifica en la urgencia sanitaria que enfrenta el país, en la evidencia científica que respalda la eficacia de estas tecnologías, en la brecha existente en el acceso a servicios neurológicos especializados y en la viabilidad técnica de implementar un sistema funcional, accesible y confiable. Este trabajo de titulación no solo responde a una problemática real, sino que aspira a aportar una herramienta concreta que, con la debida validación clínica futura, podría integrarse en procesos de atención primaria y prevención médica en todo el territorio nacional.

1.3 OBJETIVOS DE LA IMPLEMENTACIÓN

Objetivo General

Construir una aplicación web basada en modelos de aprendizaje automático para la detección temprana de tumores cerebrales a partir de imágenes de tomografía, con el fin de proporcionar una herramienta tecnológica funcional, precisa y accesible, como soporte tecnológico del sistema público de salud y facilite el acceso temprano al diagnóstico en poblaciones vulnerables.

Objetivos Específicos

1. Seleccionar y entrenar un modelo de aprendizaje profundo capaz de clasificar imágenes de tomografías cerebrales con alta precisión, mediante la elección de una arquitectura eficiente, el ajuste con técnicas de transferencia de aprendizaje y el entrenamiento con conjuntos de datos médicos validados, a fin de garantizar una alta capacidad de generalización, sensibilidad y especificidad diagnóstica.

2. Diseñar e implementar una plataforma web intuitiva que permita a los usuarios cargar tomografías y visualizar resultados de forma rápida y clara, mediante el desarrollo de una interfaz centrada en la experiencia del usuario, de manera que cualquier persona, independientemente de su nivel técnico, pueda interactuar con la plataforma y comprender los resultados generados por el modelo.

3. Evaluar el desempeño funcional, técnico y clínico del sistema mediante pruebas de precisión, validación cruzada, pruebas piloto, simulaciones controladas, análisis del tiempo de respuesta y verificación de la interfaz web, con el fin de comprobar su robustez, utilidad clínica y aplicabilidad en entornos hospitalarios o rurales.

1.4 HIPÓTESIS

La implementación de una aplicación web desarrollada en Flask, basada en el modelo de ML denominado VGG16, entrenado con imágenes de resonancias magnéticas cerebrales, permite pronosticar tempranamente tumores cerebrales con alta precisión, de una manera sencilla e interactiva para el usuario final.

1.5 METODOLOGÍA

I. Investigación documental

La primera fase del proyecto consistirá en una investigación documental exhaustiva sobre los fundamentos teóricos y técnicos relacionados con el ML, las técnicas de clasificación de imágenes médicas, así como las aplicaciones actuales de inteligencia artificial en el campo de la salud. Se consulta artículos científicos indexados en bases de datos como IEEE Xplore, Scopus, ScienceDirect y PubMed. Además, se estudian antecedentes de proyectos similares, normativa legal sobre privacidad de datos médicos en Ecuador, y lineamientos éticos del uso de IA en medicina.

Esta fase permitirá consolidar los conocimientos necesarios para la selección adecuada del modelo de aprendizaje automático, así como comprender los desafíos técnicos y clínicos asociados a la detección de tumores cerebrales.

II. Identificación de herramientas

Una vez establecidos los fundamentos teóricos, se procederá a la selección de herramientas tecnológicas necesarias para el desarrollo de la aplicación. Las herramientas se muestran a continuación en la Tabla 1.

Tabla 1: Recursos del proyecto

Recursos	Descripción
Lenguajes de programación	Python (para modelado de IA) y JavaScript (para la interfaz web).
Librerías y frameworks	TensorFlow, Keras, Scikit-learn (para ML); Flask (backend); ReactJS (frontend).
Bases de datos médicas	Datasets públicos como **BraTS** (Brain Tumor Segmentation Challenge), que contienen imágenes de resonancia magnética etiquetadas por expertos.
Herramientas de control de versiones	Git y GitHub.
Entornos de desarrollo y pruebas	Jupyter Notebooks, VS Code, Google Colab, y servidores locales o en la nube para pruebas iniciales.

Esta fase permitirá configurar el entorno de desarrollo, asegurar la compatibilidad entre los componentes del sistema, y preparar el flujo de trabajo para el desarrollo del modelo y la interfaz web.

III. Implementación y pruebas

Durante esta etapa se procederá con el entrenamiento, validación e integración del modelo de ML con la interfaz web. Las actividades específicas incluyen:

Tabla 2: Actividades de la aplicación

Etapa	Descripción
Preprocesamiento de datos	Conversión, normalización y segmentación de imágenes médicas.
Entrenamiento del modelo	Se entrenan diferentes arquitecturas de redes neuronales convolucionales (CNN) para identificar patrones asociados a tumores cerebrales.
Evaluación del modelo	Se aplicarán métricas como precisión, sensibilidad, especificidad y curva ROC-AUC.

Diseño de la interfaz web	El usuario podrá subir imágenes y obtener un resultado diagnóstico preliminar.
Integración del modelo con la interfaz	A través de una API RESTful que permitirá conectar el backend de IA con el frontend web.
Pruebas funcionales	Validación del correcto funcionamiento del sistema, detección de errores y mejoras iterativas.

1. **Preprocesamiento de datos:** Conversión, normalización y segmentación de imágenes médicas.
2. **Entrenamiento del modelo:** Se entrenan diferentes arquitecturas de redes neuronales convolucionales (CNN) para identificar patrones asociados a tumores cerebrales.
3. **Evaluación del modelo:** Se aplicarán métricas como precisión, sensibilidad, especificidad, y curva ROC-AUC.
4. **Diseño de la interfaz web:** El usuario podrá subir imágenes y obtener un resultado diagnóstico preliminar.
5. **Integración del modelo con la interfaz:** A través de una API RESTful que permitirá conectar el backend de IA con el frontend web.
6. **Pruebas funcionales:** Validación del correcto funcionamiento del sistema, detección de errores y mejoras iterativas.

Esta fase asegurará que el sistema sea funcional, accesible, y capaz de brindar resultados confiables en tiempo real.

IV. Metodología de desarrollo de software

Para el desarrollo de la aplicación web se adoptará la metodología **Scrum**, un marco ágil que permite la gestión iterativa e incremental de proyectos de software. Esta elección responde a la necesidad de flexibilidad, retroalimentación constante y entregas parciales que garanticen la calidad del producto.

El desarrollo se organizará en sprints de dos semanas, en los cuales se completarán incrementos funcionales del sistema. Los roles principales serán:

- **Product Owner:** responsable de definir los requerimientos funcionales y priorizar el backlog.
- **Scrum Master:** encargado de facilitar el proceso ágil, remover impedimentos y garantizar la adherencia a Scrum.
- **Equipo de desarrollo:** conformado por quienes implementan el modelo de IA, el backend, el frontend y las pruebas.

Cada sprint incluirá las siguientes ceremonias:

- **Sprint planning:** definición de objetivos y tareas a desarrollar.
- **Daily meeting:** reuniones breves para revisar avances y obstáculos.
- **Sprint review:** demostración del incremento desarrollado.
- **Sprint retrospective:** análisis de fortalezas, debilidades y oportunidades de mejora.

Esta metodología permitirá mantener una alta productividad, priorizar funcionalidades críticas, y ajustar rápidamente el desarrollo a los resultados de pruebas y retroalimentación de usuarios.

V. Evaluación y recomendaciones

En la última fase se realizará una evaluación del desempeño general del sistema, considerando aspectos técnicos, éticos y funcionales. Se documentan los resultados de las pruebas, así como las limitaciones encontradas en el modelo y en la plataforma web.

Posteriormente, se propondrán recomendaciones para su escalamiento, validación clínica y posible integración con entidades del sistema de salud o unidades de medicina preventiva. También se abordan consideraciones de mejora futura, como incorporar nuevas modalidades de imagen (como resonancia magnética funcional), mejorar la interfaz de usuario, o añadir módulos de historial clínico.

CAPÍTULO 2

2.1 Introducción a la Inteligencia Artificial

Dentro de la informática contemporánea, la inteligencia artificial (IA) es uno de los campos más innovadores y dinámicos. Se enfoca en desarrollar sistemas capaces de realizar funciones que, bajo circunstancias normales, necesitan de la inteligencia humana. Estas funciones abarcan desde el razonamiento lógico hasta la interpretación de información visual y el procesamiento del lenguaje natural. Desde sus primeros desarrollos conceptuales en la década de 1950, cuando Alan Turing propuso la idea de que una máquina podía pensar, hasta la formalización del término por John McCarthy en 1956, la IA ha pasado por diversas fases de evolución tecnológica (Russell & Norvig, 2021).

Con el avance de los microprocesadores, la expansión del acceso a internet y la acumulación masiva de datos, la IA ha evolucionado desde ser una propuesta teórica a convertirse en una tecnología práctica y de amplio uso. Actualmente, es aplicada en áreas tan variadas como la medicina, la industria automotriz, la agricultura inteligente, la ciberseguridad, los servicios financieros y la educación. Su desarrollo ha sido posible gracias a factores como el crecimiento exponencial del poder computacional, la reducción del costo de almacenamiento de datos y la evolución de algoritmos de aprendizaje automático.

La IA moderna se apoya en diversas disciplinas como la estadística, la lógica matemática, las ciencias cognitivas y las ciencias de la computación. Este enfoque multidisciplinario ha permitido el surgimiento de aplicaciones concretas, como los asistentes virtuales, los

sistemas de reconocimiento de voz y rostro, los algoritmos de recomendación y los sistemas de diagnóstico asistido por computadora, entre muchos otros.

2.2 Tipos de Inteligencia Artificial

La clasificación de la IA se puede realizar desde diferentes perspectivas. Una de las más comunes distingue entre los tipos de IA de acuerdo con su grado de complejidad, capacidad cognitiva y generalización del conocimiento.

2.2.1 Inteligencia Artificial Débil (IA Estrecha)

La IA débil se refiere a sistemas diseñados para llevar a cabo tareas específicas. Aunque pueden superar a los humanos en ciertas actividades concretas, carecen de conciencia, entendimiento y adaptabilidad general. Ejemplos de este tipo de IA incluyen asistentes personales como Alexa o Google Assistant, algoritmos de recomendación como los de Netflix o Amazon, y sistemas de navegación por GPS. Estas herramientas son muy eficientes, pero su inteligencia está limitada a dominios estrechos.

2.2.2 Inteligencia Artificial Fuerte (IA General)

La IA general, aún en etapa teórica, hace referencia a sistemas con capacidades cognitivas similares a las humanas. Estos sistemas serían capaces de aprender de la experiencia, razonar lógicamente, y tomar decisiones en múltiples contextos sin depender de una programación específica. Aún no existen ejemplos reales de este tipo de IA, pero sigue siendo un objetivo de largo plazo para la comunidad científica.

2.2.3 Superinteligencia Artificial

La superinteligencia artificial es un concepto especulativo que alude a una inteligencia que superaría a la humana en todos los aspectos: creatividad, toma de decisiones, análisis emocional y habilidades sociales. Esta idea ha sido objeto de numerosos debates éticos y filosóficos sobre el impacto que podría tener en la humanidad (Brundage, 2014).

2.3 Aprendizaje Automático y Redes Neuronales

Una de las evoluciones más relevantes en el ámbito de la inteligencia artificial es el aprendizaje automático. Se basa en el concepto de que los sistemas pueden aprender patrones a partir de datos sin necesidad de programar explícitamente cada una de las tareas. El aprendizaje automático se divide en tres categorías: el de refuerzo, el supervisado y el no supervisado.

En una investigación reciente realizada por Ali (2024) se probaron distintos algoritmos de aprendizaje automático para identificar estos tumores, entre ellos Random Forest y Support Vector Classifier (SVC). Aunque Random Forest alcanzó la mayor precisión (98,27%), el SVC se destacó por su desempeño consistente (97,74%), evidenciando su potencial como herramienta confiable para el diagnóstico automatizado y oportuno de esta enfermedad, con el fin de mejorar los tratamientos y la calidad de vida de los pacientes.

El aprendizaje profundo, también conocido como deep learning, es un progreso reciente y poderoso del aprendizaje automático que utiliza redes neuronales artificiales para representar vínculos complejos en volúmenes considerables de datos. Estas redes neuronales se fundamentan en la anatomía biológica del cerebro humano y están constituidas por capas de neuronas artificiales que transforman los datos iniciales en representaciones más abstractas y útiles.

Entre las redes más destacadas se encuentran las redes neuronales convolucionales (CNN, por sus siglas en inglés), especialmente eficaces en tareas relacionadas con imágenes, como la clasificación, segmentación y detección de objetos.

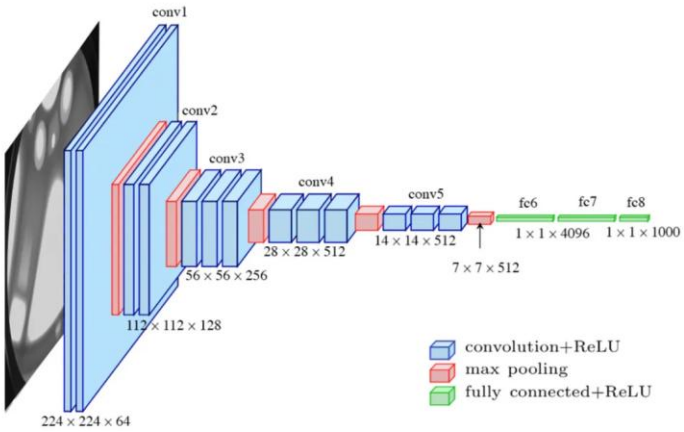
Estas redes son el núcleo de muchas aplicaciones modernas en visión por computadora, reconocimiento facial, análisis médico y vehículos autónomos (Simonyan & Zisserman, 2015).

2.4 El Modelo VGG16

VGG16 es un modelo de red neuronal convolucional profunda creado por el Grupo de Geometría Visual (VGG) perteneciente a la Universidad de Oxford. Simonyan y Zisserman (2015) lo presentaron como una contribución a la competencia ImageNet Large Scale Visual Recognition Challenge (ILSVRC) de 2014. La exactitud y la sencillez arquitectónica fueron las cualidades que hicieron que este modelo se destacara.

La arquitectura secuencial que emplea filtros de convolución de 3x3, a la que se le añaden capas de agrupamiento (pooling) y capas totalmente conectadas al final del modelo, es lo que define a VGG16. En total, tiene 16 capas con parámetros que se pueden entrenar: tres son totalmente conectadas y trece son convolucionales. Su diseño modular facilita su implementación en comparación con otras arquitecturas más complicadas.

Imagen 1. Arquitectura VGG16



Fuente: Tomado de *viso.ai* [Fotografía], por Gaudenz Boesch, 2021, viso.ai (https://viso.ai/deep-learning/vgg-very-deep-convolutional-networks/)

El modelo ha sido empleado en gran medida para clasificar imágenes, gracias a su habilidad para extraer características significativas de manera jerárquica. En métodos de aprendizaje por transferencia (transfer learning), donde se aplican estos conocimientos a nuevos conjuntos de datos con menos ejemplos, tener acceso a pesos preentrenados en conjuntos de datos extensos como ImageNet hace más fácil su utilización.

Tabla 3: Ventajas y desventajas del model VGG16

Ventajas de VGG16	**Limitaciones de VGG16**
Arquitectura uniforme y clara, lo que facilita su análisis y adaptación.	Elevado uso de recursos informáticos, tanto en términos de memoria como de tiempo dedicado a la capacitación.
Gran eficacia en las labores de categorización de imágenes.	Menor eficiencia comparado con modelos más recientes como ResNet o EfficientNet en tareas complejas.
Amplia disponibilidad de implementaciones y documentación en bibliotecas como TensorFlow y PyTorch.	

A pesar de sus limitaciones, VGG16 sigue siendo una herramienta valiosa para desarrolladores e investigadores que buscan aplicar visión por computadora en proyectos prácticos y académicos.

CAPÍTULO 3: Tumores Cerebrales y el Rol de la Inteligencia Artificial en la Medicina Moderna

3.1 Tumores Cerebrales: Definición, Tipos y Causas

Los tumores cerebrales son masas o crecimientos anómalos de células en el cerebro o en sus estructuras adyacentes, como las meninges o los nervios craneales. Estos pueden clasificarse como benignos (no cancerígenos) o malignos (cancerígenos), y su categorización depende de diversos factores, incluyendo el tipo de célula afectada, la localización dentro del encéfalo, el ritmo de crecimiento y su capacidad para invadir tejidos circundantes. Según la Organización Mundial de la Salud (OMS), los tumores cerebrales se clasifican en grados que van del I al IV, siendo los de grado I los menos agresivos y los de grado IV los más malignos, como el glioblastoma multiforme (Louis et al., 2021).

Entre los tipos más prevalentes de tumores cerebrales se destacan:

- **Gliomas**: Son los tumores más comunes y se originan en las células gliales del sistema nervioso central. Incluyen subtipos como astrocitomas, oligodendrogliomas, ependimomas y glioblastomas. Los glioblastomas, en particular, son de rápido crecimiento y altamente invasivos.
- **Meningiomas**: Tumores derivados de las meninges, las membranas que recubren el cerebro y la médula espinal. Aunque suelen ser benignos, pueden comprimir estructuras cerebrales importantes.
- **Schwannomas**: Tumores benignos de los nervios craneales, especialmente del nervio vestibulococlear.

- **Meduloblastomas**: Tumores embrionarios del cerebelo, de comportamiento agresivo y que afectan mayormente a niños.
- **Tumores metastásicos**: Son tumores secundarios que se originan en otras partes del cuerpo, como el pulmón o la mama, y se diseminan al cerebro a través del torrente sanguíneo.

Las causas de estos tumores no siempre están bien definidas, aunque se han identificado diversos factores de riesgo como predisposición genética (síndromes hereditarios como la neurofibromatosis), exposición prolongada a radiaciones ionizantes, antecedentes familiares, inmunodeficiencias, y en algunos casos, exposición prolongada a productos químicos tóxicos (Ostrom et al., 2021).

En el contexto ecuatoriano, el Instituto Nacional de Estadística y Censos (INEC) no tiene cifras desagregadas anualmente para tumores cerebrales, pero se conoce que las neoplasias ocupan uno de los cinco primeros lugares de causa de muerte en el país. Según SOLCA (2023), los tumores del sistema nervioso central representan un desafío creciente, particularmente en regiones con escaso acceso a servicios de neuroimagen y especialistas en neurología oncológica.

3.2 Imagenología Médica para el Diagnóstico de Tumores Cerebrales

La imagenología médica constituye la piedra angular del diagnóstico de tumores cerebrales. Las tecnologías de diagnóstico por imágenes permiten visualizar estructuras cerebrales internas sin necesidad de intervención quirúrgica, facilitando la localización precisa del tumor, su tamaño, forma y efectos sobre tejidos circundantes.

- **Tomografía Computarizada (CT)**: Método de imagenología por rayos X que produce secciones axiales del cerebro. Es beneficioso para identificar lesiones en los huesos, hemorragias o calcificaciones relacionadas con tumores.

- **Resonancia Magnética (MRI)**: Método de imagen de alta definición que emplea campos magnéticos y ondas de radio para obtener imágenes minuciosas del cerebro. Es la herramienta preferida para la caracterización de tumores cerebrales por su capacidad de diferenciar tejidos blandos y detectar edema peritumoral, necrosis y angiogénesis.
- **Tomografía por Emisión de Positrones (PET)**: Evalúa la actividad metabólica y funcional del tumor, siendo útil para distinguir tejido tumoral activo de necrosis o cicatriz postoperatoria.

La interpretación de estas imágenes requiere experiencia y tiempo por parte de radiólogos expertos. La gran cantidad de información generada representa un reto logístico y cognitivo, donde la IA comienza a desempeñar un rol transformador.

En Ecuador, la disponibilidad de estos métodos diagnósticos es desigual. Mientras hospitales de tercer nivel como el Hospital Carlos Andrade Marín o el Hospital de los Valles cuentan con resonancia magnética, muchos centros de segundo nivel en provincias carecen de esta tecnología. Esto limita significativamente la detección temprana y el seguimiento adecuado de tumores cerebrales.

3.3 Inteligencia Artificial en el Diagnóstico Médico

La IA se ha convertido en un elemento fundamental de la medicina contemporánea, especialmente en el diagnóstico por imágenes. Los sistemas de inteligencia artificial pueden identificar patrones complejos en imágenes médicas con una exactitud similar o

incluso más alta que la de los expertos humanos, empleando modelos de aprendizaje profundo como las redes neuronales convolucionales (CNN).

Es así como, Ting et al. (2018) menciona que el uso de la inteligencia artificial (AI) basada en aprendizaje profundo es capaz de clasificar imágenes de la retina obtenidas mediante tomografía de coherencia óptica, lo que facilita el diagnóstico temprano de enfermedades y ofrece un gran potencial para aplicarse también en otros diagnósticos médicos basados en imágenes.

En el contexto del diagnóstico de tumores cerebrales, las CNN se entrenan con grandes volúmenes de datos etiquetados para aprender a distinguir entre diferentes tipos de tumores, grados de malignidad y características morfológicas específicas. Estas redes pueden automatizar tareas críticas como:

- Segmentación de lesiones tumorales.
- Clasificación de tumores benignos vs. malignos.
- Predicción de la evolución clínica o respuesta a tratamientos.

Estudios como el de Esteva et al. (2017) han demostrado que las redes neuronales pueden lograr niveles de precisión comparables al juicio clínico experto en áreas como dermatología, lo cual se ha extrapolado con éxito a la neuro imagenología.

El aprendizaje transferido, adicionalmente, posibilita el uso de modelos preentrenados en extensos conjuntos de datos y su adaptación a funciones particulares con un número reducido de imágenes médicas; esto es lo que ocurre con el modelo VGG16.

En Ecuador, la adopción de IA en el sistema de salud es aún incipiente. Sin embargo, universidades como la Escuela Politécnica Nacional (EPN) (Laboratorio de Investigación en Inteligencia y Visión Artificial “Alan Turing”, 2025), y la Universidad de las Américas

(UDLA; Bucheli Caballero, 2024) han comenzado a incorporar proyectos de investigación en IA aplicada a la salud, algunos de ellos enfocados en imagenología médica. Estas iniciativas buscan cerrar la brecha tecnológica y mejorar el acceso al diagnóstico oportuno, especialmente en comunidades rurales y de difícil acceso.

3.4 Análisis de Antecedentes y Tecnologías Existentes

La investigación en diagnóstico asistido por IA ha mostrado un crecimiento exponencial en la última década. Algunas de las tendencias tecnológicas más relevantes incluyen:

- **Modelos de redes convolucionales profundas**: Arquitecturas como VGG16, ResNet50 y DenseNet se han aplicado exitosamente a la clasificación de tumores cerebrales en imágenes de resonancia magnética, alcanzando precisiones superiores al 90% (Chakrabarty & Nandhini, 2021).
- **Modelos híbridos (CNN + RNN)**: Combinan capacidades de análisis espacial y secuencial para mejorar la sensibilidad y especificidad en la detección de patrones tumorales.
- **Técnicas de segmentación automática**: Algoritmos como U-Net permiten aislar el área tumoral de forma precisa, facilitando el análisis volumétrico y la planificación quirúrgica.
- **Transfer learning y fine-tuning**: Permiten reutilizar modelos entrenados en bases de datos generales como ImageNet, adaptándose a datos clínicos mediante ajustes específicos.
- **Bases de datos colaborativas**: Iniciativas como el BraTS Challenge proporcionan conjuntos de datos públicos anotados de imágenes de tumores cerebrales, lo que fomenta la innovación y la validación cruzada entre investigaciones.

3.5 Estudios de Caso Relevantes

Caso 1: Clasificación de Tumores con VGG16

Parnian et al. (2018) emplearon la arquitectura VGG16 para clasificar imágenes de resonancia magnética cerebral en tres tipos tumorales: glioma, meningioma y adenoma pituitario. El modelo, entrenado con un conjunto de datos balanceado, alcanzó una precisión diagnóstica del 94%, lo que valida la utilidad de modelos pre entrenados en contextos clínicos.

Caso 2: IA en el Hospital Universitario de Heidelberg

Con el objetivo de respaldar el diagnóstico de tumores cerebrales a través de imágenes por resonancia magnética, el Hospital Universitario de Heidelberg ha puesto en funcionamiento un sistema de inteligencia artificial que se basa en redes neuronales convolucionales. Este sistema, entrenado con más de 10.000 imágenes médicas, fue capaz de diferenciar con alta precisión entre gliomas de bajo y alto grado, alcanzando una exactitud del 91% en pruebas clínicas iniciales. Además, la herramienta ofrecía segmentaciones automáticas del tumor que servían como insumo directo para planificaciones quirúrgicas y terapias personalizadas. El uso del sistema no solo mejoró la precisión diagnóstica, sino que también redujo significativamente el tiempo de análisis de imágenes por parte de los radiólogos, pasando de varias horas a pocos minutos (Kickingereder et al., 2019)

Caso 3: IA en Zonas de Bajo Acceso Médico

En regiones rurales de India, se ha implementado un sistema de diagnóstico preliminar mediante teléfonos inteligentes y algoritmos basados en redes neuronales. Este enfoque

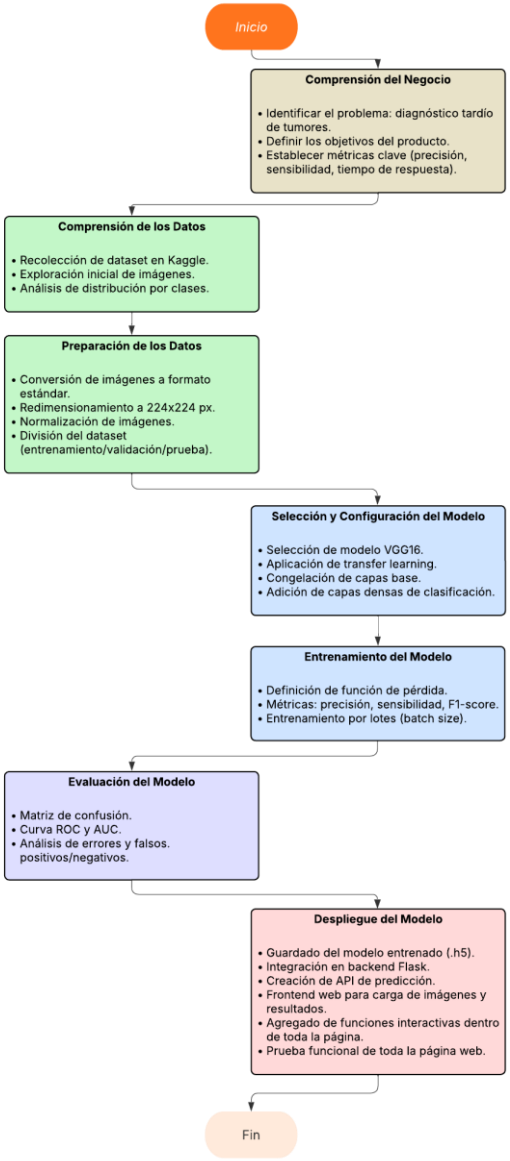
ha permitido reducir los tiempos de atención y mejorar la tasa de diagnósticos tempranos, compensando la falta de radiólogos en zonas remotas (Rajpurkar et al., 2018).

CAPÍTULO 4: Implementación del modelo de machine learning

Antes de la implementación es necesario tener el conocimiento de cómo se planificó la implementación mediante un diagrama de pasos, este es una representación visual que organiza y secuencia las actividades necesarias para alcanzar un objetivo dentro de un proceso determinado. Este tipo de herramienta permite observar de manera clara las fases que deben cumplirse, identificando entradas, transformaciones y resultados en cada etapa. Además, facilita la comprensión global del proyecto, ya que muestra la lógica y la dependencia entre actividades. Según Aguilar (2021), los diagramas de flujo y pasos contribuyen a estructurar procesos complejos, simplificando su análisis y ejecución. De igual manera, permiten identificar áreas de mejora, optimizar recursos y comunicar de forma efectiva las tareas a los involucrados (Díaz & Ramírez, 2020).

Una vez se conoce la definición del diagrama de pasos, se construyó un diagrama con los pasos que se tomaron para la construcción e implementación del modelo de ML.

Imagen 2. Diagrama de pasos



En este caso, el diagrama fue dividido en varias categorías diferenciadas por colores, cada una representa lo siguiente:

1. **Comprensión del problema (gris):** En esta primera etapa se busca comprender el contexto y la necesidad que da origen al proyecto. Se identifica el problema a resolver (en este caso, el diagnóstico tardío de tumores), se establecen los objetivos del producto y se definen métricas clave como la precisión, la sensibilidad y el tiempo de respuesta. Esta fase es fundamental, pues marca la dirección del proyecto y determina los criterios de éxito.

2. **Gestión y preparación de datos (verde):** Una vez comprendido el problema, es necesario trabajar con los datos. Esto incluye la recolección de un dataset adecuado, la exploración inicial de las imágenes y el análisis de la distribución de clases. Posteriormente, se realiza la preparación de los datos: conversión a un formato estándar, redimensionamiento de imágenes, normalización y división en conjuntos de entrenamiento, validación y prueba. Estas acciones garantizan que la información esté lista para ser procesada por el modelo de inteligencia artificial.

3. **Construcción del modelo (azul):** En esta fase se selecciona y configura el modelo de aprendizaje profundo. Para el caso estudiado, se utiliza VGG16 con la técnica de transfer learning, congelando capas base y añadiendo capas densas para la clasificación. Asimismo, se definen la función de pérdida, las métricas de evaluación (precisión, sensibilidad y F1-score) y los parámetros de entrenamiento, como el tamaño de lote (batch size). Este proceso constituye el núcleo del desarrollo del sistema de detección.

4. **Validación del modelo (morado):** Una vez entrenado, el modelo debe ser evaluado para determinar su eficacia. Se utilizan herramientas como la matriz de confusión, la curva ROC y el área bajo la curva (AUC). Además, se realiza un análisis de errores, identificando falsos positivos y falsos negativos. Esta etapa es esencial para conocer las limitaciones del modelo y establecer posibles mejoras antes de su despliegue.

5. **Implementación y uso (rojo):** Se procede al despliegue del modelo en un entorno funcional. Esto incluye el guardado del modelo entrenado en formato .h5, su integración en un backend con Flask y la creación de una API de predicción. También se desarrolla un frontend web para la carga de imágenes y visualización

de resultados, incorporando funciones interactivas que mejoran la experiencia del usuario. Por último, se realizan pruebas funcionales de la aplicación web para asegurar que el sistema cumpla con los objetivos planteados en la fase inicial.

4.1 Diseño experimental del producto

El diseño experimental es una estrategia sistemática que permite planificar, ejecutar y analizar experimentos de manera que se puedan obtener conclusiones válidas sobre relaciones causa-efecto entre variables. Su objetivo es controlar la variabilidad y asegurar la confiabilidad de los resultados.

Según Montgomery (2017), el diseño experimental es "un conjunto de procedimientos estadísticos utilizados para planear experimentos de forma que los datos obtenidos puedan analizarse objetivamente y se puedan extraer conclusiones válidas y eficientes". De manera similar, Hernández Sampieri et al. (2014) lo definen como "el plan o estrategia concebida para responder al problema de investigación y lograr sus objetivos, guiando al investigador en el proceso de recolección, análisis e interpretación de los datos".

Tabla 4: Diseño experimental

Elemento	Descripción
Objetivo	Detectar y clasificar automáticamente tumores cerebrales (glioma, meningioma, pituitario o sin tumor) a partir de imágenes MRI.
Datos de entrada	Imágenes de resonancia magnética (MRI) en formato .jpg ubicadas en carpetas por clase (Training/ y Testing/).
Clases	● Glioma ● Meningioma

	● Pituitary ● Notumor
Tamaño de imagen	128 x 128 píxeles
Preprocesamiento	● Redimensionamiento ● Normalización (valores entre 0 y 1) ● Aumentación de datos (brillo y contraste aleatorios)
Modelo base	VGG16 preentrenado en ImageNet, sin capas superiores (include_top=False)
Transfer learning	Congelación de capas excepto las últimas 3 de VGG16
Arquitectura final	VGG16 + Flatten + Dropout (0.3) + Dense(128, ReLU) + Dropout (0.2) + Dense (n clases, Softmax)
Compilación	● Optimizer: Adam ● Learning rate: 0.0001 ● Loss: sparse_categorical_crossentropy
Parámetros de entrenamiento	● Epochs: 5 ● Batch size: 20 ● Steps per epoch: calculado automáticamente
Evaluación del modelo	● Accuracy y Loss por época (gráfico) ● Reporte de clasificación ● Matriz de confusión ● Curvas ROC y AUC por clase
Predicción individual	Visualización de la imagen con etiqueta predicha y nivel de confianza
Guardado del modelo	Archivo .h5 con arquitectura y pesos del modelo: model.h5

Herramientas utilizadas	• TensorFlow / Keras • PIL- NumPy • Scikit-learn • Matplotlib / Seaborn
Resultado esperado	Un modelo capaz de clasificar imágenes MRI en las 4 categorías con alto nivel de precisión.

4.2 Proceso de entrenamiento del modelo de ML

Una vez se ha comprendido a detalle cada uno de los elementos que conforman el producto a realizar, ya se puede empezar con la implementación y desarrollo de la aplicación web. El primer paso para esto fue la creación y entrenamiento del modelo de ML, para realizar esto se utilizó la herramienta de Jupyter Notebook para de esa manera entrenarlo, probar y exportar el modelo. A continuación, se muestra el proceso paso a paso de cómo se logra esto:

```
import os  # Para operaciones con directorios y archivos
import numpy as np  # Para operaciones numéricas y manejo de arreglos de imágenes
import random  # Para generar valores aleatorios para aumentación de datos
from PIL import Image, ImageEnhance  # Para procesamiento y mejora de imágenes
from tensorflow.keras.preprocessing.image import load_img  # Para cargar imágenes
from tensorflow.keras.models import Sequential  # Para construir el modelo secuencial
from tensorflow.keras.layers import Input, Flatten, Dropout, Dense  # Para las capas del modelo
from tensorflow.keras.optimizers import Adam  # Para el optimizador Adam
from tensorflow.keras.applications import VGG16  # Para utilizar el modelo VGG16 preentrenado
from sklearn.utils import shuffle  # Para mezclar aleatoriamente los datos
```

Código 1: importación de librerías

Este código prepara el entorno para construir un modelo de clasificación de imágenes usando aprendizaje profundo con TensorFlow y Keras. Se importa os para trabajar con archivos y carpetas, y numpy para operaciones numéricas sobre matrices de imágenes. La biblioteca random y PIL se usan para aplicar aumentación de datos, es decir, generar variaciones de las imágenes originales para mejorar el entrenamiento del modelo.

Keras proporciona herramientas clave como load_img para cargar imágenes y Sequential para construir el modelo de forma secuencial. Las capas como Input, Flatten, Dropout y Dense permiten transformar los datos de entrada, prevenir sobreajuste y generar salidas clasificatorias. Además, se utiliza el modelo pre entrenado VGG16, lo que permite aplicar aprendizaje por transferencia, aprovechando conocimiento previo para mejorar la eficiencia del nuevo modelo.

El optimizador Adam facilita el ajuste automático del aprendizaje durante el entrenamiento. Posteriormente, shuffle de sklearn mezcla aleatoriamente los datos, ayudando a que el modelo generalice mejor.

```
# Directorios para los datos de entrenamiento y prueba
train_dir = 'C:/Users/Pc/Downloads/Tumor/Training/'
test_dir = 'C:/Users/Pc/Downloads/Tumor/Testing/'

# Cargar y mezclar los datos de entrenamiento
train_paths = []
train_labels = []
for label in os.listdir(train_dir):
    for image in os.listdir(os.path.join(train_dir, label)):
        train_paths.append(os.path.join(train_dir, label, image))
        train_labels.append(label)

train_paths, train_labels = shuffle(train_paths, train_labels)

# Cargar y mezclar los datos de prueba
test_paths = []
test_labels = []
for label in os.listdir(test_dir):
    for image in os.listdir(os.path.join(test_dir, label)):
        test_paths.append(os.path.join(test_dir, label, image))
        test_labels.append(label)

test_paths, test_labels = shuffle(test_paths, test_labels)
```

Código 2: Introducción de los datos

Este código se encarga de cargar las rutas de las imágenes tanto para el conjunto de entrenamiento como para el de prueba, que están organizados en carpetas según sus clases (por ejemplo, con y sin tumor). A través de os.listdir, recorre cada subcarpeta dentro de los directorios Training y Testing, y guarda la ruta completa de cada imagen junto con su etiqueta correspondiente (el nombre de la subcarpeta).

Después de cargar todas las rutas y etiquetas, usa shuffle para mezclar aleatoriamente los datos, lo cual es esencial para evitar que el modelo aprenda un orden específico de las clases y pueda generalizar mejor durante el entrenamiento y evaluación.

```
# Seleccionar índices aleatorios para 10 imágenes
random_indices = random.sample(range(len(train_paths)), 10)

# Crear una figura para mostrar las imágenes en 2 filas
fig, axes = plt.subplots(2, 5, figsize=(15, 8))
axes = axes.ravel()

for i, idx in enumerate(random_indices):
    # Cargar imagen
    img_path = train_paths[idx]
    img = Image.open(img_path)
    img = img.resize((224, 224))  # Redimensionar a tamaño consistente

    # Mostrar imagen
    axes[i].imshow(img)
    axes[i].axis('off')  # Ocultar los ejes
    # Mostrar la etiqueta de clase
    axes[i].set_title(f"Etiqueta: {train_labels[idx]}", fontsize=10)

plt.tight_layout()
plt.show()
```

Código 3: Normalización de datos

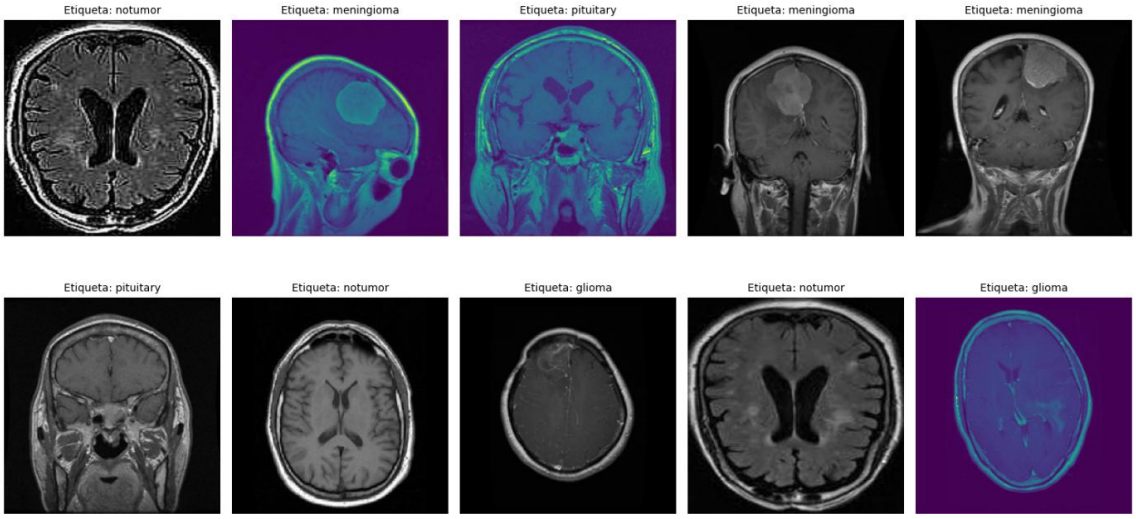
Este código sirve para visualizar aleatoriamente 10 imágenes del conjunto de entrenamiento. Primero, selecciona 10 índices aleatorios de la lista train_paths, que

contiene las rutas a las imágenes de entrenamiento. Luego, configura una figura de Matplotlib con 2 filas y 5 columnas para mostrar esas imágenes de manera organizada.

Para cada imagen seleccionada, se abre el archivo con la biblioteca PIL, se redimensiona a 224x224 píxeles que, cabe recalcar es el tamaño estándar esperado por muchos modelos como el que se utiliza en este proyecto. Además, se ocultan los ejes para una visualización más limpia y se añade el nombre de la clase correspondiente como título.

Esta visualización es útil para verificar que las imágenes estén correctamente cargadas y etiquetadas antes de entrenar el modelo como se puede presenciar en el ejemplo a continuación:

Imagen 3. Resultado de visualización



En cuanto al código 4, que se lo puede encontrar en el Anexo A, define varias funciones clave para preprocesar y alimentar imágenes a un modelo de red neuronal en forma de lotes (batches), una técnica que se la considera muy común en el entrenamiento de redes con grandes volúmenes de datos.

Primero, la función augment_image aplica la aumentación de datos a una imagen individual. Convierte la imagen a formato PIL, luego ajusta aleatoriamente su brillo y

contraste, y posteriormente normaliza sus valores de píxeles dividiéndolos entre 255 para que estén en el rango [0, 1], lo que es esencial para un entrenamiento estable.

La función open_images carga imágenes desde las rutas dadas, redimensionándolas a un tamaño uniforme definido por IMAGE_SIZE, aplica aumentación a cada una y devuelve un arreglo de imágenes listas para el modelo.

Luego, encode_label convierte las etiquetas en valores numéricos, lo que es necesario para que el modelo pueda aprender a clasificarlas.

Por último, la función datagen actúa como generador de datos por lotes, lo cual es útil para entrenar modelos sin cargar todos los datos en memoria al mismo tiempo. Divide los datos en pequeños grupos, carga y procesa esas imágenes y etiquetas, y las entrega (yield) una por una durante cada época de entrenamiento.

El código 5 del Anexo A, define y entrena el modelo de clasificación de imágenes utilizando la arquitectura VGG16 como base pre entrenada. Se establece un tamaño de entrada de 128x128 píxeles y se carga el modelo sin su capa de salida original (include_top=False) y con pesos entrenados previamente en el conjunto de datos ImageNet. Esto permite reutilizar el conocimiento ya aprendido sobre características visuales.

Todas las capas del modelo base se congelan inicialmente para evitar que sus pesos se modifiquen durante el entrenamiento, salvo las últimas tres, que se desbloquean para permitir una ligera adaptación al nuevo conjunto de datos (aprendizaje por transferencia o fine-tuning). Después, se construye una nueva red neuronal secuencial, agregando VGG16 como bloque base, seguido de capas adicionales: aplanamiento, capas densas y

de regularización (Dropout) y una capa de salida con activación softmax para clasificación multiclase.

El modelo se compila con el optimizador Adam, una tasa de aprendizaje baja (0.0001), y se usa la función de pérdida sparse_categorical_crossentropy para etiquetas codificadas como enteros. Por último, se entrena el modelo durante 5 épocas utilizando el generador de datos (datagen) previamente definido, el cual suministra imágenes en lotes con aumentación, asegurando una alimentación eficiente y variada al modelo.

```
plt.figure(figsize=(8,4))
plt.grid(True)
plt.plot(history.history['sparse_categorical_accuracy'], '.g-',
linewidth=2)
plt.plot(history.history['loss'], '.r-', linewidth=2)
plt.title('Model Training History')
plt.xlabel('epoch')
plt.xticks([x for x in range(epochs)])
plt.legend(['Accuracy', 'Loss'], loc='upper left', bbox_to_anchor=(1, 1))
plt.show()
```

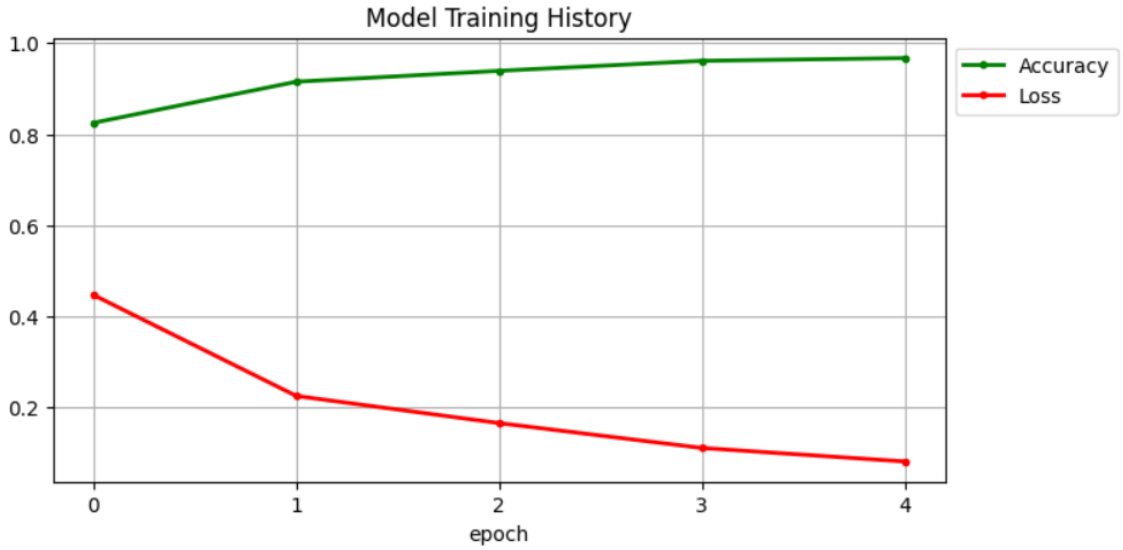
Código 6: Historial de entrenamiento

Este fragmento de código genera una gráfica del historial de entrenamiento del modelo, mostrando cómo evolucionaron la precisión (accuracy) y la pérdida (loss) a lo largo de las épocas.

Se crea una figura, se activa una cuadrícula para mejorar la legibilidad del gráfico, y se trazan dos líneas: una en verde (.g-) que representa la precisión de clasificación (sparse_categorical_accuracy), y otra en rojo (.r-) que representa la función de pérdida (loss) durante el entrenamiento. Estas métricas fueron registradas automáticamente por Keras durante el proceso de entrenamiento y se accede a ellas a través de history.history.

El gráfico incluye un título ("Model Training History"), etiquetas en el eje X que corresponden al número de época, y una leyenda posicionada fuera del área del gráfico para indicar qué línea representa cada métrica. Por último, se muestra el gráfico con plt.show().

Imagen 4. Historial de entrenamiento



Esta visualización es muy útil para evaluar el comportamiento del modelo durante el entrenamiento, permitiendo identificar problemas como sobreajuste (cuando la pérdida disminuye, pero la precisión no mejora o empeora).

```
# 1. Predicción sobre los datos de prueba
test_images = open_images(test_paths)  # Cargar y aumentar imágenes de
prueba
test_labels_encoded = encode_label(test_labels)  # Codificar las etiquetas
de prueba

# Predecir usando el modelo entrenado
test_predictions = model.predict(test_images)

# 2. Reporte de clasificación
print("Reporte de clasificación:")
print(classification_report(test_labels_encoded,
np.argmax(test_predictions, axis=1)))
```

Código 7: Evaluación del modelo

Este fragmento evalúa el rendimiento del modelo utilizando el conjunto de prueba. Primero, se cargan y procesan las imágenes de prueba mediante la función open_images, que aplica aumentación y normalización. Las etiquetas se codifican a valores numéricos usando encode_label, lo cual es necesario para compararlas con las predicciones del modelo.

Luego, se realizan predicciones con el modelo previamente entrenado sobre las imágenes de prueba. Estas predicciones son probabilidades por clase, por lo que se usa np.argmax(..., axis=1) para obtener la clase con mayor probabilidad como resultado final para cada imagen.

Ulteriormente, se imprime un reporte de clasificación usando classification_report de sklearn, el cual muestra métricas como precisión, recall, F1-score y soporte para cada clase.

Imagen 5. Reporte de clasificación

```
41/41 ━━━━━━━━━━━━━━━━━━━━ 32s 783ms/step
Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.95      0.95       300
           1       0.91      0.93      0.92       306
           2       0.98      1.00      0.99       405
           3       1.00      0.95      0.97       300

    accuracy                           0.96      1311
   macro avg       0.96      0.96      0.96      1311
weighted avg       0.96      0.96      0.96      1311
```

Este reporte permite evaluar con más detalle cómo se está desempeñando el modelo en la clasificación de cada categoría, lo cual es crucial para aplicaciones como la detección de tumores, donde los errores pueden tener consecuencias importantes.

```
# 3. Matriz de confusión
conf_matrix = confusion_matrix(test_labels_encoded,
np.argmax(test_predictions, axis=1))
print("Matriz de confusión:")
print(conf_matrix)

# Graficar la matriz de confusión
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues",
xticklabels=os.listdir(train_dir), yticklabels=os.listdir(train_dir))
plt.title("Matriz de Confusión")
plt.xlabel("Etiquetas Predichas")
plt.ylabel("Etiquetas Verdaderas")
plt.show()
```

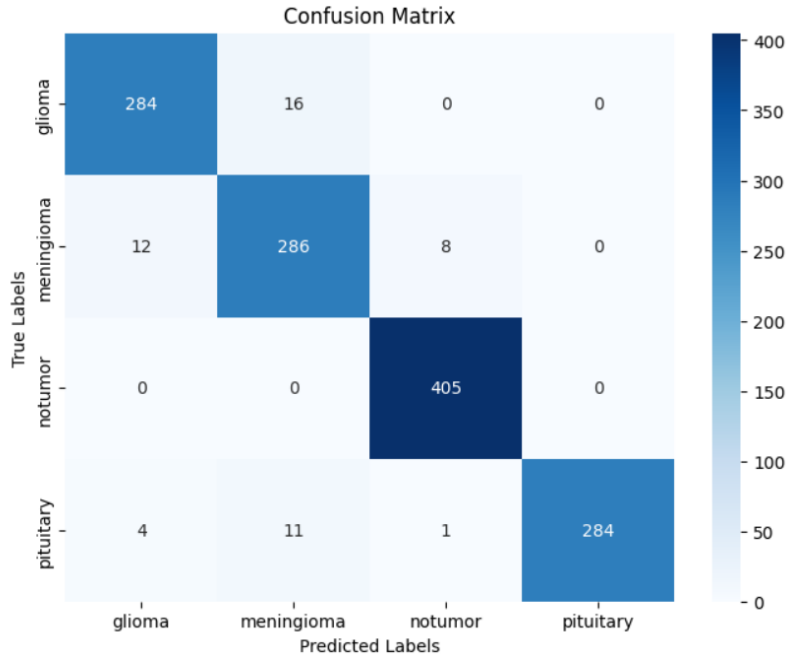
Código 8: Evaluación del modelo

Este bloque de código genera y visualiza la matriz de confusión, una herramienta esencial para evaluar el desempeño de un modelo de clasificación. La matriz compara las etiquetas verdaderas de las imágenes de prueba con las etiquetas predichas por el modelo, mostrando cuántas veces se predijo correctamente cada clase y cuántas veces se confundió con otra.

Primero, se crea la matriz usando confusion_matrix de sklearn, pasando las etiquetas verdaderas codificadas y las predicciones finales del modelo (convertidas desde probabilidades a clases mediante np.argmax). Luego, la matriz se imprime en consola para su revisión.

A continuación, se visualiza la matriz con seaborn.heatmap, donde los valores se representan mediante una escala de color azul. Las filas indican las clases reales y las columnas las clases predichas. El uso de annot=True muestra los valores numéricos en cada celda, y los nombres de clase se extraen directamente de las carpetas de entrenamiento (os.listdir(train_dir)).

Imagen 6. Matriz de confusión



Esta visualización permite identificar de manera clara en qué clases el modelo acierta o se equivoca, lo cual es muy útil, como en este caso, para detectar si hay confusión frecuente entre la clasificación de los diferentes tipos de tumores que van a ser clasificados en la aplicación.

El código 9, encontrado en el Anexo A, genera las curvas ROC (Receiver Operating Characteristic) y calcula el AUC (Área Bajo la Curva) para evaluar el rendimiento del modelo en una clasificación multiclase.

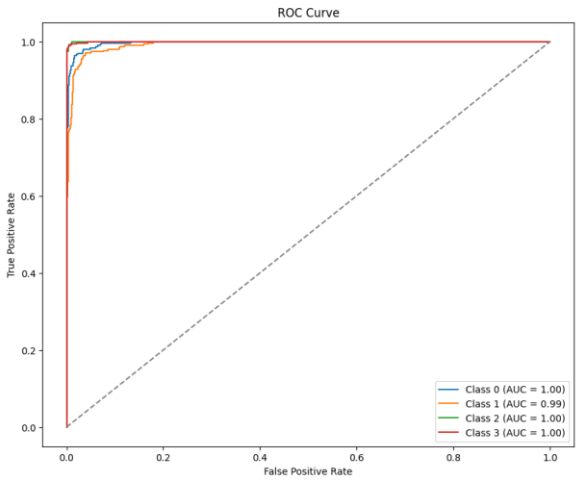
Primero, se binarizan las etiquetas verdaderas usando label_binarize para convertirlas en una matriz donde cada columna representa una clase y contiene valores binarios (0 o 1). Esto es necesario porque la curva ROC requiere comparar una clase contra el resto. Las predicciones (test_predictions) ya contienen probabilidades para cada clase y se usan directamente.

Luego, se calcula la curva ROC y el AUC para cada clase individualmente utilizando roc_curve y auc de sklearn. Esto genera las tasas de verdaderos positivos (TPR) y falsos

positivos (FPR) para cada clase, y permite visualizar cómo de bien el modelo separa las clases.

Así pues, se grafican todas las curvas ROC en una sola figura, con una línea diagonal gris que representa un clasificador aleatorio (AUC = 0.5).

Imagen 7. Curva ROC



Cuanto más alejada esté cada curva de esa diagonal y más cercana al vértice superior izquierdo, mejor es el rendimiento del modelo para esa clase.

```
# Guardar todo el modelo
model.save('model.h5')

from tensorflow.keras.models import load_model
# Cargar el modelo entrenado
model = load_model('model.h5')
```

Código 10: Guardado y carga del modelo

Este fragmento permite guardar y reutilizar un modelo de aprendizaje profundo construido y entrenado previamente con Keras. Al usar model.save('model.h5'), se guarda todo el modelo en un único archivo con extensión .h5, lo que incluye absolutamente todo lo que se ha realizado hasta el momento, cada paso, arquitectura, etc.

Luego, con load_model('model.h5'), se recupera el modelo guardado exactamente como estaba, listo para hacer predicciones o continuar el entrenamiento. Esta funcionalidad es muy útil para evitar volver a entrenar el modelo desde cero, compartir modelos con otros usuarios o integrarlos en aplicaciones de producción.

La función encontrada en el código 11 del Anexo A, permite realizar una predicción individual sobre una imagen de resonancia magnética (MRI) utilizando un modelo previamente entrenado para clasificar tipos de tumores cerebrales. A partir de una ruta de imagen proporcionada, la función carga la imagen, la redimensiona, normaliza sus valores y la transforma en el formato adecuado para el modelo (agregando una dimensión de lote).

Luego, el modelo realiza la predicción y se identifica la clase con mayor probabilidad. Si la clase predicha es 'notumor', se interpreta que no hay tumor en la imagen. En caso contrario, se muestra el tipo específico de tumor (glioma, meningioma o pituitary) junto con el porcentaje de confianza. A continuación, se realizarán pruebas del funcionamiento de este:

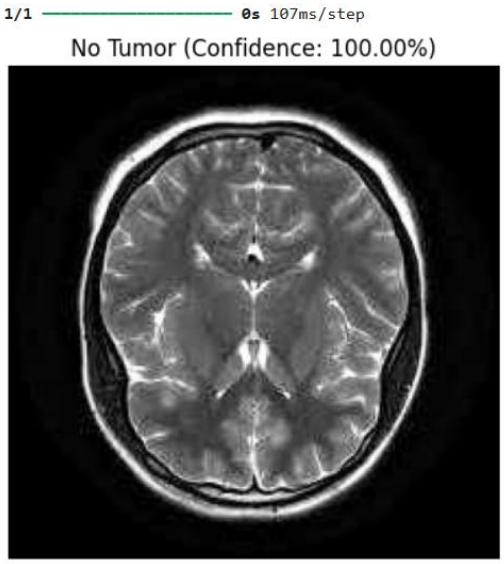
```
# Ejemplo de uso
image_path = 'C:/Users/Pc/Downloads/Tumor/Testing/notumor/Te-no_0374.jpg'
# Proporciona la ruta de la nueva imagen
detect_and_display(image_path, model)
```

Código 12: Prueba del modelo

Este fragmento muestra cómo utilizar la función detect_and_display con una imagen nueva. Se proporciona la ruta de una imagen específica de prueba (en este caso, de la clase 'notumor'), y se llama a la función pasándole tanto la imagen como el modelo entrenado.

El propósito es verificar cómo responde el modelo ante una imagen real, mostrando visualmente el resultado de la predicción junto con el nivel de confianza.

Imagen 8. Resultado de la predicción



Así pues, la imagen 7 muestra el resultado correcto de la predicción de prueba, ya que se cargó una imagen donde la resonancia magnética no contiene ningún tipo de tumor. Una vez se comprobó que el modelo funciona correctamente, se puede crear una interfaz amigable para el usuario. Ese proceso se detalla en el siguiente capítulo.

CAPÍTULO 5: Implementación de la aplicación en Python Flask

5.1 Desarrollo de backend

La base funcional de esta aplicación está construida con el framework Flask, una microestructura web escrita en Python. Este backend permite la conexión entre la interfaz del usuario y el modelo de IA previamente entrenado con redes neuronales convolucionales (CNN) utilizando TensorFlow y Keras.

El servidor se inicia instanciando un objeto Flask:

```
app = Flask __name__
```

Código 13: Inicio de la app

Esto permite manejar peticiones HTTP, renderizar plantillas HTML y gestionar el flujo de información entre el cliente (navegador) y el servidor.

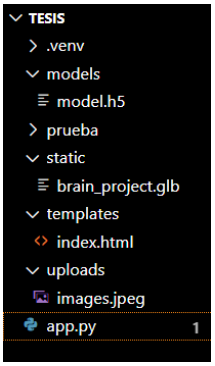
```
# Cargar modelo
model =            'models/model.h5'
class_labels =  'pituitary'  'glioma'  'notumor'  'meningioma'

UPLOAD_FOLDER = '../uploads'
os.makedirs UPLOAD_FOLDER  exist_ok=True
app config 'UPLOAD_FOLDER'  = UPLOAD_FOLDER
```

Código 14: Carga de modelo en flask

Este código carga un modelo entrenado (model.h5) que clasifica imágenes en cuatro categorías de tumores cerebrales. Define las etiquetas de clase (pituitary, glioma, notumor, meningioma) y configura una carpeta llamada uploads donde se guardarán las imágenes que el usuario suba desde la aplicación web creada con Flask. Esto prepara el entorno para recibir imágenes y hacer predicciones con el modelo.

La aplicación maneja la siguiente organización de cada uno de los componentes:

Imagen 9. Organización en VS Code



Después, la función principal que realiza la predicción se llama predict_tumor, y ejecuta los siguientes pasos:

1. **Carga de imagen**: se redimensiona la imagen a 128x128 píxeles.
2. **Preprocesamiento**: se normaliza la imagen para que los valores estén entre 0 y 1.
3. **Predicción**: el modelo genera un array con probabilidades para cada clase.
4. **Resultado**: se retorna la clase con mayor probabilidad y el porcentaje de confianza.
5. **Extra:** Se agrega un sleep(1) para simular un pequeño retraso visual para mostrar una animación de carga en la interfaz.

El código que ejecuta cada uno de estos pasos es el siguiente:

```
# Predicción
def predict_tumor image_path
    IMAGE_SIZE = 128
    img = load_img image_path  target_size= IMAGE_SIZE  IMAGE_SIZE
    img_array = img_to_array img  / 255.0
    img_array = np expand dims img_array  axis=0

    # Simulamos un pequeño retraso para mostrar la animación de carga
    time sleep 1

    predictions = model  img_array
    predicted_class = np argmax predictions
    confidence = np max predictions
    tumor_label = class_labels predicted_class
    return  "No tumor" if tumor_label == 'notumor' else
tumor_label capitalize    confidence
```

Código 15: Predicción del modelo

Después se encuentran las rutas principales, el backend maneja distintas rutas que controlan el flujo de navegación y procesamiento:

- **/:** Ruta principal que permite cargar una imagen y renderiza los resultados. Este código se puede encontrar en el Anexo A como código 16.

- **/uploads/<filename>:** Sirve la imagen subida para ser visualizada en la interfaz. El código de la ruta es el siguiente:

```
# Mostrar imagen
@app.route '/uploads/<filename>'
def get_uploaded_file filename
    return send_from_directory app config 'UPLOAD_FOLDER'
filename
```

Código 17: Función de imágenes

- **/clear:** Limpia los archivos subidos y reinicia el análisis. El código de la ruta es el siguiente:

```
# Ruta de limpieza manual
@app.route('/clear')
def clear():
    clear_uploads()
    return redirect(url_for('index'))
```

Código 18: Limpieza de resultados

La lógica contenida en la función index() se encarga de gestionar la recepción de archivos mediante POST, ejecutar la predicción y renderizar la plantilla HTML con los resultados. En caso de error, se muestran mensajes explicativos para el usuario.

Posteriormente, se puede observar que el código de esta sección incluye verificaciones básicas para asegurar que:

- El archivo tenga una extensión válida.
- No se procese si no se subió ningún archivo.
- Se limpia la carpeta automáticamente para evitar acumulación.

Se emplea secure_filename de werkzeug.utils para garantizar que el nombre del archivo no sea malicioso.

5.2 Desarrollo del frontend

El frontend de la aplicación constituye la capa visual e interactiva con la que el usuario final interactúa directamente. Su diseño e implementación se llevaron a cabo utilizando HTML5 como lenguaje de estructura, CSS3 para los estilos visuales, Bootstrap 5 para el diseño responsive y adaptable, y JavaScript para añadir interactividad dinámica.

Adicionalmente, se integró el sistema de plantillas Jinja2 que provee Flask, lo cual permitió inyectar datos y resultados procesados en el backend directamente dentro del código HTML. Esta integración fue fundamental para lograr que el flujo entre la carga de imágenes, el análisis con inteligencia artificial y la visualización de resultados sea fluido y sin interrupciones para el usuario.

La filosofía de diseño que se adoptó estuvo centrada en la experiencia del usuario (UX), siguiendo criterios como:

- Minimizar la cantidad de pasos para subir y analizar una imagen.
- Proporcionar retroalimentación inmediata sobre el estado del análisis.
- Mantener una estructura visual limpia, con jerarquía clara de información.
- Asegurar compatibilidad con dispositivos móviles, tablets y computadores de escritorio.

5.2.1 Estructura general de la interfaz

La interfaz principal (index.html) se divide en secciones bien diferenciadas para guiar al usuario de manera intuitiva:

1. **Encabezado**: Contiene el nombre de la aplicación, un subtítulo explicativo y el menú de navegación.
2. **Visor 3D del cerebro**: Permite explorar un modelo interactivo en tiempo real mediante <model-viewer>.
3. **Bloque educativo**: Brinda información sobre el cerebro y sobre tumores que afectan al mismo.
4. **Formulario de carga de imágenes**: Con zona de arrastre (drag & drop) y botón de selección de archivo.
5. **Sección de resultados**: Muestra diagnóstico, porcentaje de confianza y la imagen procesada junto con recomendaciones basadas en el diagnóstico.

La estructura base del documento se define de la siguiente manera:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Detección de Tumores Cerebrales</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.c
ss" rel="stylesheet">
    <link rel="stylesheet" href="{{ url_for('static',
filename='css/styles.css') }}">
</head>
<body>
    <!-- Encabezado -->
    <header class="text-center py-4 bg-primary text-white">
        <h1>Brain Tumor Detection</h1>
        <p>Sube una imagen MRI y recibe un diagnóstico en segundos</p>
    </header>

    <!-- Contenido principal -->
    <main class="container mt-4">
        {% block content %}{% endblock %}
    </main>
</body>
</html>
```

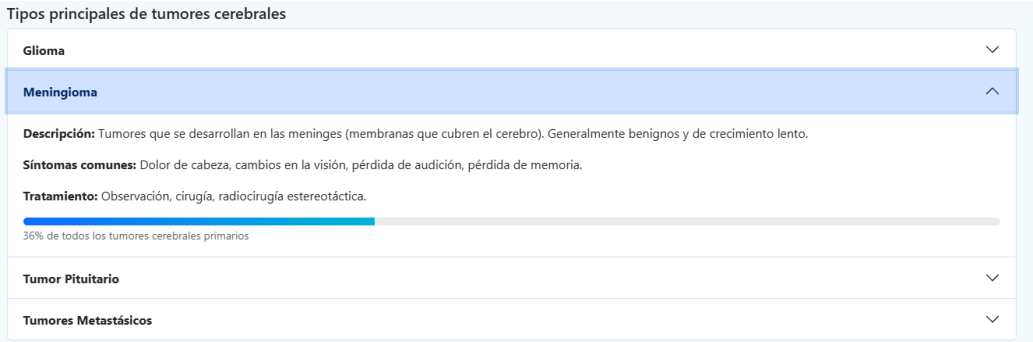
Código 19: Estructura base de la plantilla.

Esta estructura base permite reutilizar elementos comunes en todas las vistas, como el encabezado y los estilos, optimizando la mantenibilidad del código.

5.2.5 Sección educativa

Un elemento diferenciador de este frontend es la inclusión de contenido educativo dinámico, entre algunas cosas que incluye son: un modelo 3D del cerebro humano que está dividido en cada una de sus lóbulos para así tener información de cada uno de estos y que al final tiene un botón que lleva a los usuarios a páginas médicas por si es que desean más información de la estructura, también posee cards que contienen datos interesantes del cerebro, y por último, una sección de desplegables que contienen información valiosa sobre los posibles tumores cerebrales que existen como se puede observar en la siguiente imagen:

Imagen 10. Información de los tumores cerebrales.



Esto no solo brinda información, sino que también agrega valor educativo y de concientización a la aplicación.

5.2.3 Formulario de carga de imágenes

El formulario para cargar imágenes fue diseñado pensando en la simplicidad y accesibilidad. Se incluye un campo de selección de archivo con validación de formato (sólo JPG y PNG) y un botón para enviar la imagen.

```
<form action="/" method="POST" enctype="multipart/form-data" class="p-4
border rounded shadow-sm">
    <div class="mb-3">
        <label for="file" class="form-label">Selecciona una imagen
MRI</label>
        <input type="file" id="file" name="file" class="form-control"
accept=".jpg, .jpeg, .png" required>
    </div>
    <button type="submit" class="btn btn-success w-100">Analizar
Imagen</button>
</form>
```

Código 20: Formulario de carga de imágenes.

Este formulario envía los datos al backend usando POST, donde la función index() de Flask procesa la imagen. La integración con Bootstrap asegura que el formulario sea responsivo y fácil de usar incluso en pantallas pequeñas.

5.2.4 Visualización de resultados dinámicos

Una vez procesada la imagen en el backend, los resultados se devuelven a la misma plantilla usando variables de Jinja2. Esto evita que el usuario tenga que navegar a otra página, manteniendo la interacción continua.

```
{% if result %}
<div class="mt-4 p-3 border rounded shadow-sm bg-light">
    <h3 class="text-primary">Diagnóstico: {{ result }}</h3>
    <p>Confianza: {{ (confidence * 100) | round(2) }}%</p>
    <img src="{{ url_for('get_uploaded_file', filename=filename) }}"
class="img-fluid rounded shadow mt-3">
</div>
{% endif %}
```

Código 21: Bloque de visualización de resultados.

Este enfoque permite mostrar la etiqueta del diagnóstico (result), el porcentaje de certeza (confidence), la imagen analizada (filename) de forma automática sin requerir recarga

manual de la página y, eventualmente un grupo de recomendaciones dependiendo del diagnóstico recibido por parte de la aplicación.

Las recomendaciones aportan un nivel de personalización, ya que cada diagnóstico incluye indicaciones específicas. Algunas son más leves y orientadas al cuidado general, mientras que otras son más estrictas y buscan que el usuario acuda de inmediato a un médico.

5.2.5 Interactividad con JavaScript

Para mejorar la experiencia del usuario, se integraron varias funciones JavaScript que permiten:

- **Animación de carga** mientras se procesa la imagen.
- **Controles de zoom** para ampliar y reducir la imagen del resultado.
- **Botón de reinicio** que llama a la ruta /clear para limpiar archivos temporales.

Ejemplo de funciones de zoom:

```
function zoomIn() {
    let img = document.querySelector('.result-container img');
    img.style.transform = 'scale(1.2)';
}

function zoomOut() {
    let img = document.querySelector('.result-container img');
    img.style.transform = 'scale(0.8)';
}

function resetZoom() {
    let img = document.querySelector('.result-container img');
    img.style.transform = 'scale(1)';
}
```

Código 22. Funciones JavaScript para manipular la escala de la imagen.

Estas funciones se aplican directamente a la etiqueta <img> que muestra la imagen analizada.

CAPÍTULO 6: Resultados

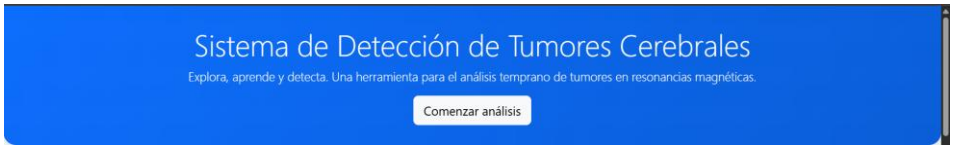
Una vez se ya se conocen los procesos de desarrollo de la aplicación web, es necesario describir los diferentes tipos de resultados obtenidos tras la implementación completa del sistema web para detección de tumores cerebrales mediante inteligencia artificial, integrando tanto el backend en Flask como el frontend en HTML.

6.1 Resultado visual e interactivo

La página principal está compuesta por varias secciones:

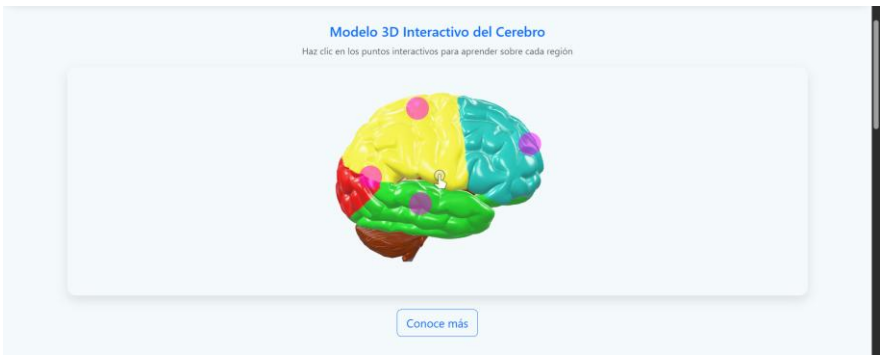
- **Encabezado**: presenta el título del sistema, una breve introducción y un botón que guía al usuario al análisis.

Imagen 11. Encabezado página



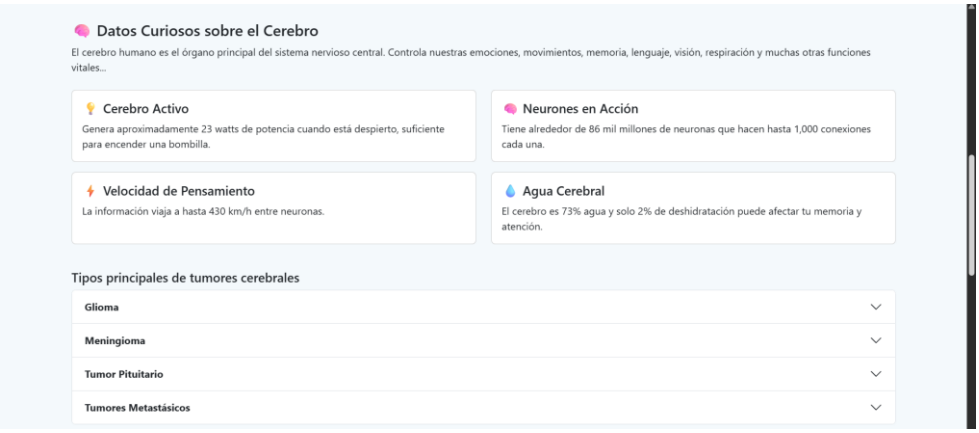
- **Modelo 3D interactivo**: permite al usuario visualizar un cerebro tridimensional y explorar sus lóbulos mediante botones interactivos.

Imagen 12. Cerebro 3D



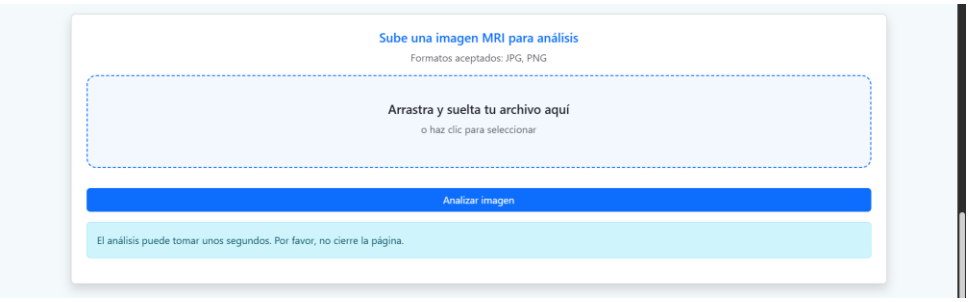
- **Sección educativa**: contiene información útil sobre el cerebro humano y los tipos más comunes de tumores cerebrales.

Imagen 13. Información educacional



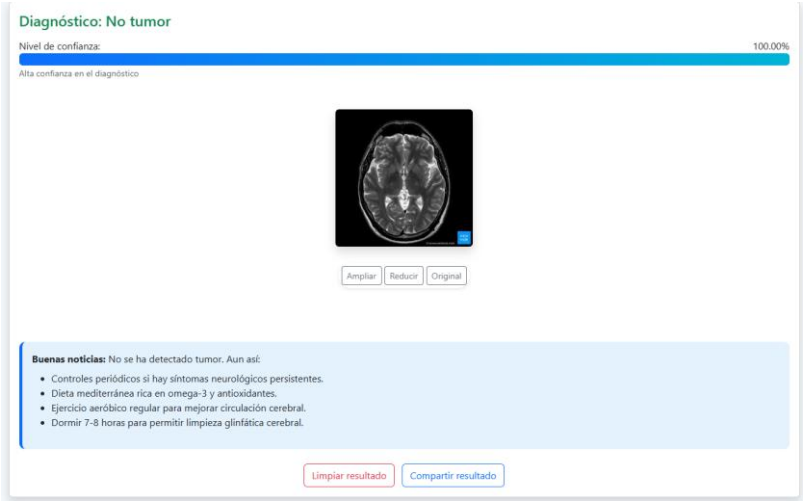
- **Formulario de análisis**: permite al usuario subir una imagen de resonancia magnética (MRI) y recibir una predicción.

Imagen 14. Formulario de predicción



- **Sección de resultados**: muestra el diagnóstico, nivel de confianza, imagen subida y recomendaciones médicas personalizadas.

Imagen 15. Ejemplo de resultado



6.1.1 Interactividad

Para el modelo 3D, se utiliza el componente <model-viewer> para mostrar un modelo .glb del cerebro en 3D. Se utiliza botones interactivos o hotspots que despliegan modales con información relevante sobre los lóbulos:

- Lóbulo frontal
- Lóbulo parietal
- Lóbulo temporal
- Lóbulo occipital

Esto convierte a la herramienta en una plataforma no sólo diagnóstica, sino también educativa, ya que adicionalmente, después del modelo en 3D, también se pueden encontrar elementos como el botón en la parte inferior del modelo, que lleva al usuario a una página con toda la información sobre el cerebro.

También, abajo de este, se encuentran datos curiosos del mismo, en formato de tarjetas y, por último, desplegables que contienen información sobre los tumores cerebrales más comunes que pueden existir.

6.1.2 Subida de imagen y análisis

La zona de análisis permite a los usuarios subir una imagen mediante arrastrar y soltar, o hacer clic en un área estilizada. Se despliega una animación de carga mientras el sistema analiza la imagen y, al final muestra:

- El diagnóstico estimado.
- Nivel de confianza en porcentaje.

- Imagen cargada, con botones para modificar el tamaño de esta.
- Consejos personalizados dependiendo del tipo de tumor.

6.1.3 Consejos personalizados

Una de las fortalezas del sistema es que adapta los mensajes finales a cada diagnóstico, como se puede observar en la tabla 5.

Tabla 5: Recomendaciones personalizadas

Diagnóstico	Recomendaciones principales
Glioma	Consultar a un neuro-oncólogo, seguir una dieta rica en antioxidantes y llevar un registro de síntomas.
Meningioma	Seguimiento periódico, práctica de ejercicio regular y monitoreo de síntomas visuales.
Tumor pituitario	Evaluación endocrina constante y vigilancia de cambios hormonales o visuales.
No tumor	Mantener hábitos saludables de prevención y realizar chequeos médicos regulares.

6.2 Diseño centrado en el usuario

Se utilizaron técnicas de diseño accesible y responsivo para garantizar que el sistema funcione bien en computadoras, tabletas y móviles. Elementos clave:

7. Paleta de colores azul-blanco para transmitir confianza médica.
8. Animaciones suaves para mejorar la experiencia sin distraer.
9. Botones de zoom sobre la imagen para mejor visualización.
10. Preguntas frecuentes para resolver dudas comunes.

11. Modal de privacidad y términos para transparencia y ética.

6.3 Pruebas

Se realizaron pruebas funcionales para garantizar el correcto desempeño del sistema. Esto se encuentra detallado en la Tabla 6.

Tabla 6: Revisión de pruebas

Caso de Prueba	Acción	Resultado Esperado	Resultado Obtenido
Subida de imagen válida (.jpg)	Cargar archivo MRI y presionar "Analizar imagen".	Diagnóstico y porcentaje de confianza mostrados en pantalla.	Cumplido
Subida de imagen con formato inválido	Cargar archivo PDF.	Mensaje de error y rechazo de carga.	Cumplido
Imagen con tumor detectable	Ejecutar predicción con imagen de prueba con glioma.	Diagnóstico: Glioma, consejos específicos y porcentaje de confianza >85%.	Cumplido
Imagen sin tumor	Ejecutar predicción con imagen sin patologías.	Mensaje de “No tumor detectado” con recomendaciones de control preventivo.	Cumplido
Uso de controles de zoom	Ampliar, reducir y restablecer imagen cargada.	La imagen responde correctamente a los botones sin distorsión.	Cumplido

Adicionalmente, se realizaron otro tipo de pruebas detalladas en la Tabla 7 en los Anexos B, en este caso para validar la correcta implementación de la aplicación web desarrollada, para esto, se estableció un plan de pruebas que abarca tanto el frontend como el backend,

así como aspectos de rendimiento, accesibilidad y seguridad. A continuación, se describen las pruebas realizadas y sus resultados en la tabla 7 del Anexo B.

En primer lugar, se verificó el comportamiento de la interfaz principal (Prueba P01). El formulario de carga de imágenes debía adaptarse de manera adecuada a diferentes dispositivos, incluyendo ordenadores de escritorio, tabletas y teléfonos móviles. Se comprobó que, gracias a la utilización de diseño responsive con Bootstrap, la disposición de los elementos se ajustaba correctamente en cada tamaño de pantalla, lo que asegura la accesibilidad del sistema para distintos usuarios.

Posteriormente, se realizó una validación de entradas en el módulo de carga de imágenes (Prueba P02). El sistema debía aceptar únicamente archivos con formato .jpg o .png, evitando la carga de documentos no válidos como .txt o .pdf. Durante la prueba, se comprobó que el sistema desplegaba un mensaje de error en caso de intentos de carga indebida, garantizando que solo se procesaran imágenes compatibles con el modelo de IA.

En cuanto al procesamiento en el servidor, se verificó la correcta recepción de los archivos enviados desde el formulario (Prueba P03). Se cargaron imágenes MRI de prueba y se confirmó que estas eran almacenadas en el directorio temporal del servidor Flask sin pérdida de datos, lo que asegura la integridad de la información enviada por el usuario antes de su procesamiento por el modelo.

Una prueba fundamental fue la validación del modelo de IA (Prueba P04). Para ello, se utilizaron imágenes MRI con diagnósticos previamente conocidos. El modelo fue capaz de identificar con precisión el tipo de tumor cerebral y mostrar el porcentaje de confianza asociado. Esto permitió corroborar que la integración entre Flask y el modelo de IA se encontraba funcionando de manera confiable.

Asimismo, se probó la ruta /predict del backend (Prueba P05). Esta ruta debía devolver los resultados de la predicción en un formato adecuado para ser consumido por el frontend, en particular un objeto JSON con la información del diagnóstico, el nivel de confianza y el nombre del archivo procesado. La prueba demostró que la ruta cumplía con su propósito y proporcionaba datos estructurados de forma coherente.

En el lado del frontend, se realizó una verificación sobre la barra de progreso (Prueba P06). La barra debía representar gráficamente el porcentaje de confianza obtenido por el modelo en tiempo real. Se validó que los datos enviados desde el servidor eran captados correctamente por la interfaz y reflejados en la barra, tanto visualmente como mediante el valor numérico mostrado.

Otra prueba funcional consistió en comprobar la visualización de la imagen cargada (Prueba P07). Una vez que el usuario seleccionaba un archivo válido, este debía mostrarse en el visor dentro de la aplicación. La prueba confirmó que el sistema renderizaba la imagen junto a los resultados, lo que otorga claridad al proceso de diagnóstico para el usuario final.

El rendimiento del sistema también fue evaluado (Prueba P08). Se midió el tiempo total entre la carga de la imagen y la obtención del resultado procesado por el modelo. Con imágenes de aproximadamente 1 MB, el sistema respondió en menos de 10 segundos, lo cual se considera un tiempo adecuado para aplicaciones web de diagnóstico médico asistido por IA.

Posteriormente, se realizó una prueba de integración de extremo a extremo (Prueba P09). Se verificó que todo el flujo, desde la carga de la imagen hasta la visualización de resultados y recomendaciones, se ejecutará de manera continua y sin errores. El sistema funcionó correctamente, integrando todos los módulos de manera armónica.

Después de todo, se probó la compatibilidad de la aplicación en diferentes navegadores (Prueba P10). Se accedió desde Chrome, Firefox y Microsoft Edge, confirmando que en todos ellos la aplicación cargaba y funcionaba de manera correcta, con la misma fluidez en las funcionalidades principales.

CONCLUSIONES

1. El entrenamiento del modelo VGG16, ajustado con técnicas de transferencia de aprendizaje y datasets médicos validados, demostró que es posible alcanzar una precisión diagnóstica elevada en la clasificación de tumores cerebrales. Esto confirma la viabilidad técnica del uso de modelos de aprendizaje profundo para tareas de diagnóstico asistido, garantizando resultados consistentes y confiables en escenarios de prueba controlados.

2. El desarrollo de la aplicación web permitió integrar de manera efectiva el modelo de IA con una interfaz amigable, asegurando que usuarios sin conocimientos técnicos puedan interactuar con la plataforma y obtener resultados claros y comprensibles. La simplicidad en el diseño contribuye a la accesibilidad y fomenta el uso de la herramienta en entornos médicos y no médicos.

3. La evaluación funcional del sistema demostró que la herramienta es capaz de procesar imágenes con eficiencia, manteniendo tiempos de respuesta aceptables y una precisión diagnóstica alta en los escenarios de validación. Estos resultados evidencian que la aplicación no solo es técnicamente robusta, sino también potencialmente útil en contextos clínicos y comunitarios, especialmente en lugares con limitada disponibilidad de especialistas.

4. La propuesta presenta un potencial impacto positivo en el ámbito de la salud pública, especialmente en contextos con limitado acceso a especialistas y equipos de diagnóstico avanzado. Aunque no sustituye la evaluación médica profesional, puede constituir un apoyo relevante para la detección temprana y la referencia oportuna a atención especializada.

5. El desarrollo de la aplicación web para la detección temprana de tumores cerebrales cumplió de manera satisfactoria el objetivo general y los objetivos específicos planteados. Se implementó un modelo de ML basado en la arquitectura VGG16, entrenado y validado con imágenes de resonancia magnética (MRI), alcanzando un rendimiento adecuado para la clasificación preliminar en cuatro categorías: glioma, meningioma, tumor pituitario y ausencia de tumor.

RECOMENDACIONES

1. Se recomienda ampliar el entrenamiento del modelo con datasets más heterogéneos y representativos de distintas poblaciones, con el fin de mejorar la generalización y evitar sesgos diagnósticos. Además, sería pertinente explorar arquitecturas más modernas para comparar y optimizar el rendimiento.

2. Se recomienda implementar pruebas de usabilidad con usuarios reales, para identificar posibles mejoras en la experiencia de uso. Además, incorporar elementos de accesibilidad universal (contrastes de color, soporte para lectores de pantalla) garantizaría una mayor inclusión y ampliaría el rango de usuarios potenciales.

3. Se recomienda realizar pruebas piloto en colaboración con instituciones médicas, de forma que se validen los resultados en condiciones clínicas reales. Asimismo,

establecer protocolos de actualización del modelo y auditoría periódica de su desempeño permitirá mantener la confiabilidad y relevancia de la herramienta a largo plazo.

4. Ampliar y diversificar el conjunto de datos de entrenamiento incorporando imágenes provenientes de centros médicos locales, así como explorar otras modalidades de imagen, como la resonancia magnética funcional y la tomografía computarizada.

5. Mejorar la experiencia del usuario mediante la inclusión de funcionalidades de accesibilidad para personas con discapacidades visuales, soporte multilingüe y traducciones automáticas, ampliando así el alcance potencial de la herramienta.

BIBLIOGRAFÍA

Aguilar, J. (2021). Modelado y análisis de procesos empresariales. *Editorial Alfaomega*.

Ali, U. (2024). Comparative Evaluation Of Machine Learning Classifiers For Brain Tumor Detection. *medRxiv*. https://doi.org/10.1101/2024.07.28.24311114

Brundage, M. (2014). Taking superintelligence seriously. Superintelligence: Paths, dangers, strategies by Nick Bostrom (Oxford University Press, 2014). *Futures*, *72*. https://doi.org/10.1016/j.futures.2015.07.009

Bucheli Caballero, N. E. (2024). *Implementación de tecnologías de inteligencia artificial en el diagnóstico médico: Un enfoque integral para mejorar la eficiencia y precisión en el Hospital Clínica Kennedy Samborondón periodo 2023 – 2024* (Tesis de maestría). Universidad de las Américas, Quito. Recuperado de https://dspace.udla.edu.ec/handle/33000/16029

Chakrabarty, A., & Nandhini, M. S. (2021). Brain Tumor Classification Using VGG16. *International Journal of Engineering and Advanced Technology*, *10*(5), 87–92.

Colegio de Médicos de Pichincha. (2025, 10 de junio). *Estudio revela saturación de médicos en Ecuador y carencia de especialistas clave*. *Expedientes*. Recuperado de https://expedientes.ec/estudio-revela-saturacion-de-medicos-en-ecuador-y-carencia-de-especialistas-clave/

Díaz, P., & Ramírez, L. (2020). Herramientas visuales para la gestión de proyectos tecnológicos. *Revista de Innovación y Tecnología*, *12*(3), 45–57.

ecancer.org. (2021). Un informe muestra que las tasas de incidencia de los tumores cerebrales en adultos están disminuyendo, pero las tasas de supervivencia a cinco años siguen siendo bajas. *ecancer*. https://ecancer.org/es/news/20811-un-informe-muestra-que-las-tasas-de-incidencia-de-los-tumores-cerebrales-en-adultos-estan-disminuyendo-pero-las-tasas-de-supervivencia-a-cinco-aos-siguen-siendo-bajas

Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., & Thrun, S. (2017). Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, *542*(7639), 115–118. https://doi.org/10.1038/nature21056

Gamma Knife Ecuador. (2022). *Casos de tumores cerebrales en aumento: la intervención oportuna es vital para controlar el resultado*. Gamma Knife Center Ecuador. https://gammaknife.com.ec/los-tumores-en-aumento/

Hernández Sampieri, R., Fernández Collado, C., & Baptista Lucio, P. (2014). *Metodología de la investigación* (6.ª ed.). McGraw-Hill.

Kickingereder, P., Isensee, F., Tursunova, I., Petersen, J., Neuberger, U., Bonekamp, D., Brugnara, G., Schell, M., Kessler, T., Foltyn, M., Harting, I., Sahm, F., Prager, M., Nowosielski, M., Wick, A., Nolden, M., Radbruch, A., Debus, J., Schlemmer, H.-P., … Maier-Hein, K. H. (2019). Automated quantitative tumour response assessment of MRI in neuro-oncology with artificial neural networks: a multicentre, retrospective study. *The Lancet Oncology*, *20*(5), 728–740. https://doi.org/10.1016/S1470-2045(19)30098-1

Laboratorio de Investigación en Inteligencia y Visión Artificial “Alan Turing”. (2025). *Laboratorio de Investigación en Inteligencia y Visión Artificial “Alan Turing”*. Escuela Politécnica Nacional. Recuperado de https://laboratorio-ia.epn.edu.ec/es

Louis, D. N., Perry, A., Wesseling, P., Brat, D. J., Cree, I. A., Figarella-Branger, D., Hawkins, C., Ng, H. K., Pfister, S. M., Reifenberger, G., Soffietti, R., von Deimling, A., & Ellison, D. W. (2021). The 2021 WHO Classification of Tumors of the Central Nervous System: a summary. *Neuro-Oncology*, *23*(8), 1231–1251. https://doi.org/10.1093/neuonc/noab106

Montgomery, D. C. (2017). *Design and analysis of experiments* (9th ed.). John Wiley & Sons.

Moreno-Zambrano, D., Wong-Ayoub, J. A., Arevalo-Mora, M., San Andrés-Suárez, I., Santana, D., Meza-Venegas, J., Urquizo-Rodríguez, E., Jimenez-Zambrano, J., & Garcia-Santibanez, R. (2023, noviembre 5). Number of Neurologists and Neurology Training Programs available in the Public Health System of Ecuador: Analysis and Recommendation. *Revista Ecuatoriana de Neurologia*, 55–60. https://doi.org/10.46997/revecuatneurol32200055

OMS. (2022, febrero 3). *Cáncer*. Organización Mundial de la Salud. https://www.who.int/es/news-room/fact-sheets/detail/cancer

Ostrom, Q. T., Cioffi, G., Waite, K., Kruchko, C., & Barnholtz-Sloan, J. S. (2021). CBTRUS Statistical Report: Primary Brain and Other Central Nervous System Tumors Diagnosed in the United States in 2014-2018. *Neuro-Oncology*, *23*. https://doi.org/10.1093/neuonc/noab200

Parnian, A., Arash, M., & Konstantinos, N. P. (2018). Brain Tumor Type Classification via Capsule Networks. *Cornell University*. https://doi.org/https://doi.org/10.48550/arXiv.1802.10200

Rajpurkar, P., Irvin, J., Ball, R. L., Zhu, K., Yang, B., Mehta, H., Duan, T., Ding, D., Bagul, A., Langlotz, C. P., Patel, B. N., Yeom, K. W., Shpanskaya, K., Blankenberg, F. G., Seekins, J., Amrhein, T. J., Mong, D. A., Halabi, S. S., Zucker, E. J., … Lungren, M. P. (2018). Deep learning for chest radiograph diagnosis: A retrospective comparison of the CheXNeXt algorithm to practicing radiologists. *PLoS Medicine*, *15*(11). https://doi.org/10.1371/journal.pmed.1002686

Russell, S., & Norvig, P. (2021). *Artificial Intelligence: A Modern Approach* (4th ed.). Pearson.

Siddique, Md. A. B., Sakib, S., Khan, M. M. R., Tanzeem, A. K., Chowdhury, M., & Yasmin, N. (2020). Deep Convolutional Neural Networks Model-based Brain Tumor Detection in Brain MRI Images. *Cornell University*. https://doi.org/10.1109/I-SMAC49090.2020.9243461

Simonyan, K., & Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. *Cornell University*. https://doi.org/10.48550/arXiv.1409.1556

SOLCA. (2023). *Registro Nacional de Tumores*. SOLCA Núcleo de Quito. https://solcaquito.org.ec/datos-del-rnt-de-solca-quito-valorados-internacionalmente/

Ting, D. S. W., Liu, Y., Burlina, P., Xu, X., Bressler, N. M., & Wong, T. Y. (2018). AI for medical imaging goes deep. *Nature Medicine*, *24*(5), 539–540. https://doi.org/10.1038/s41591-018-0029-3

Imágenes

ANEXO A

```
# Función de aumentación de imágenes
def augment_image(image):
    image = Image.fromarray(np.uint8(image))
    image = ImageEnhance.Brightness(image).enhance(random.uniform(0.8,
1.2))  # Brillo aleatorio
    image = ImageEnhance.Contrast(image).enhance(random.uniform(0.8, 1.2))
# Contraste aleatorio
    image = np.array(image) / 255.0  # Normalizar los valores de píxeles al
rango [0, 1]
    return image

# Cargar imágenes y aplicar aumentación
def open_images(paths):
    images = []
    for path in paths:
        image = load_img(path, target_size=(IMAGE_SIZE, IMAGE_SIZE))
        image = augment_image(image)
        images.append(image)
    return np.array(images)

# Codificación de etiquetas (convertir nombres de etiquetas a enteros)
def encode_label(labels):
    unique_labels = os.listdir(train_dir)  # Asegura obtener etiquetas
únicas
    encoded = [unique_labels.index(label) for label in labels]
    return np.array(encoded)

# Generador de datos por lotes
def datagen(paths, labels, batch_size=12, epochs=1):
    for _ in range(epochs):
        for i in range(0, len(paths), batch_size):
            batch_paths = paths[i:i + batch_size]
            batch_images = open_images(batch_paths)  # Abrir y aumentar
imágenes
            batch_labels = labels[i:i + batch_size]
            batch_labels = encode_label(batch_labels)  # Codificar
etiquetas
            yield batch_images, batch_labels  # Entregar el lote
```

Código 4: Proceso de entrenamiento

```
# Arquitectura del modelo
IMAGE_SIZE = 128  # Tamaño de la imagen
base_model = VGG16(input_shape=(IMAGE_SIZE, IMAGE_SIZE, 3),
include_top=False, weights='imagenet')

# Congelar todas las capas del modelo base VGG16
for layer in base_model.layers:
    layer.trainable = False

# Permitir el entrenamiento de las últimas capas del modelo base VGG16
base_model.layers[-2].trainable = True
base_model.layers[-3].trainable = True
base_model.layers[-4].trainable = True

# Construir el modelo final
model = Sequential()
model.add(Input(shape=(IMAGE_SIZE, IMAGE_SIZE, 3)))  # Capa de entrada
model.add(base_model)  # Agregar el modelo base VGG16
model.add(Flatten())  # Aplanar la salida del modelo base
model.add(Dropout(0.3))  # Capa Dropout para regularización
model.add(Dense(128, activation='relu'))  # Capa densa con activación ReLU
model.add(Dropout(0.2))  # Otra capa Dropout para evitar sobreajuste
model.add(Dense(len(os.listdir(train_dir)), activation='softmax'))  # Capa
de salida con activación softmax

# Compilar el modelo
model.compile(optimizer=Adam(learning_rate=0.0001),
              loss='sparse_categorical_crossentropy',
              metrics=['sparse_categorical_accuracy'])

# Parámetros
batch_size = 20
steps = int(len(train_paths) / batch_size)  # Pasos por época
epochs = 5

# Entrenar el modelo
history = model.fit(datagen(train_paths, train_labels,
batch_size=batch_size, epochs=epochs),
                    epochs=epochs, steps_per_epoch=steps)
```

Código 5: Creación del modelo

```
# 4. Curva ROC y AUC
# Binarizar las etiquetas de prueba y las predicciones para una curva ROC
multiclase
test_labels_bin = label_binarize(test_labels_encoded,
classes=np.arange(len(os.listdir(train_dir))))
test_predictions_bin = test_predictions  # Probabilidades predichas para
cada clase

# Calcular la curva ROC y el AUC para cada clase
fpr, tpr, roc_auc = {}, {}, {}
for i in range(len(os.listdir(train_dir))):
    fpr[i], tpr[i], _ = roc_curve(test_labels_bin[:, i],
test_predictions_bin[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])

# Graficar la curva ROC
plt.figure(figsize=(10, 8))
for i in range(len(os.listdir(train_dir))):
    plt.plot(fpr[i], tpr[i], label=f'Clase {i} (AUC = {roc_auc[i]:.2f})')

plt.plot([0, 1], [0, 1], linestyle='--', color='gray')  # Línea diagonal
plt.title("Curva ROC")
plt.xlabel("Tasa de Falsos Positivos")
plt.ylabel("Tasa de Verdaderos Positivos")
plt.legend(loc="lower right")
plt.show()
```

Código 9: Análisis del modelo

```python
# Etiquetas de clases
class_labels = ['pituitary', 'glioma', 'notumor', 'meningioma']

def detect_and_display(img_path, model, image_size=128):
    """
    Función para detectar tumores y mostrar los resultados.
    Si no se detecta tumor, muestra "No Tumor".
    De lo contrario, muestra el tipo de tumor predicho y el nivel de
confianza.
    """
    try:
        # Cargar y preprocesar la imagen
        img = load_img(img_path, target_size=(image_size, image_size))
        img_array = img_to_array(img) / 255.0  # Normalizar los valores de
píxeles
        img_array = np.expand_dims(img_array, axis=0)  # Añadir dimensión
de lote

        # Realizar la predicción
        predictions = model.predict(img_array)
        predicted_class_index = np.argmax(predictions, axis=1)[0]
        confidence_score = np.max(predictions, axis=1)[0]

        # Determinar la clase
        if class_labels[predicted_class_index] == 'notumor':
            result = "No Tumor"
        else:
            result = f"Tumor: {class_labels[predicted_class_index]}"

        # Mostrar la imagen con la predicción
        plt.imshow(load_img(img_path))
        plt.axis('off')
        plt.title(f"{result} (Confianza: {confidence_score * 100:.2f}%)")
        plt.show()

    except Exception as e:
        print("Error al procesar la imagen:", str(e))
```

Código 11: Creación del modelo de predicción

```
        # Ruta principal
        @app.route '/'  methods= 'GET'  'POST'
        def index
            if request.method == 'POST'
                # Verificar si el archivo está en la solicitud
                if 'file' not in request.files
                    return redirect request url

                file = request.files 'file'

                # Si el usuario no selecciona archivo
                if file.filename == ''
                    return redirect request url

                if file and allowed_file file.filename
                    clear_uploads
                    filename = secure_filename file.filename
                    filepath  =  os.path.join app.config 'UPLOAD_FOLDER'
filename

                    file.save filepath

                    try
                        result  confidence = predict_tumor filepath
                        return render_template 'index.html'
                                               result=result

confidence=f"{confidence*100:.2f}"

file_path=f'/uploads/{filename}'
                    except Exception as e
                        return render_template 'index.html'
                                               error="Error  procesando  la
imagen. Asegúrese de que es una imagen válida."

            clear_uploads    # Limpia al hacer GET
            return render_template 'index.html'  result=None
```

Código 16: Aplicación en flask

ANEXO B

ID de Prueba	Módulo	Descripción de la Prueba	Tipo de Prueba	Entradas	Resultado Esperado	Resultado Obtenido	Estado
P01	Frontend - HTML	Verificar que el formulario de carga de imágenes se visualice correctamente en diferentes dispositivos.	Prueba de interfaz (UI)	Acceso a la página en móvil, tablet y PC.	La interfaz se adapta a cada pantalla usando diseño responsivo de Bootstrap.	Correcto	Aprobado
P02	Frontend - HTML	Validar que el campo de carga solo acepte imágenes en formato .jpg y .png.	Validación de entrada	Subida de imágenes en formatos no permitidos (.txt, .pdf).	El sistema debe mostrar un mensaje de error y no permitir la carga.	Correcto	Aprobado
P03	Backend - Flask	Comprobar que el servidor recibe correctamente la imagen subida desde el formulario.	Prueba funcional	Imagen MRI válida.	La imagen es recibida en el directorio temporal del servidor y lista para procesar.	Correcto	Aprobado
P04	Backend - IA	Validar la predicción del modelo IA con una imagen de prueba conocida.	Prueba de precisión	Imagen MRI con diagnóstico conocido.	El sistema debe predecir correctamente el tipo de tumor y mostrar porcentaje de confianza.	Correcto	Aprobado
P05	Backend - Flask	Probar la ruta /predict para garantizar que devuelve los resultados en formato correcto para el frontend.	Prueba funcional	Imagen válida enviada mediante POST.	Respuesta JSON con diagnóstico, confianza y nombre del archivo procesado.	Correcto	Aprobado
P06	Frontend - JS	Validar que la barra de progreso muestre el porcentaje de confianza del modelo en tiempo real.	Prueba de interfaz (UI)	Datos de confianza enviados desde backend.	La barra se llena correctamente y muestra el valor numérico.	Correcto	Aprobado
P07	Frontend - HTML	Comprobar que la imagen cargada se visualice en la interfaz después de subirla.	Prueba funcional	Imagen válida.	Imagen renderizada en el visor junto al resultado.	Correcto	Aprobado
P08	Backend - Rendimiento	Medir el tiempo de procesamiento desde la carga hasta la entrega del resultado.	Prueba de rendimiento	Imagen MRI de 1 MB.	Tiempo total inferior a 10 segundos.	Correcto	Aprobado
P09	Full Stack	Verificar que el flujo completo (carga de imagen → predicción → visualización de resultados) se ejecute sin errores.	Prueba de integración	Imagen MRI válida.	Resultado final mostrado correctamente con diagnóstico y recomendaciones.	Correcto	Aprobado

P10	Compatibilidad	Verificar el funcionamiento en navegadores diferentes (Chrome, Firefox, Edge).	Prueba de compatibilidad	Acceso a la web en cada navegador.	La aplicación carga y funciona correctamente en todos los navegadores probados.	Correcto	Aprobado

Tabla 7: Diseño de prueba