UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Diseño e implementación de un clasificador del estado de metilación de MGMT en tumores cerebrales a partir de imágenes de resonancia magnética morfológica y funcional.

Trabajo Fin de Grado

Grado en Ciencia de Datos

AUTOR/A: Muñoz Estornell, Alexandre

Tutor/a: Fuster i Garcia, Elies

CURSO ACADÉMICO: 2022/2023

Resumen

La metilación del gen MGMT (O6-metilguanina-ADN metiltransferasa) es un biomarcador clave en la predicción del pronóstico y la respuesta al tratamiento en pacientes con glioblastoma, el tumor cerebral primario más común y agresivo. Las técnicas habituales para detectar la metilación del gen incluyen PCR en tiempo real o secuenciación de bisulfito, métodos invasivos que requieren tejido tumoral. El estudio propone un enfoque innovador para el análisis de imágenes médicas que combina técnicas de procesamiento de imágenes, aprendizaje profundo y transferencia de aprendizaje para desarrollar un modelo de clasificación capaz de detectar y evaluar el estado de metilación de MGMT en pacientes con glioblastomas. La solución propuesta se basa en la extracción de características morfológicas y funcionales de las imágenes de RM, que son relevantes para la identificación del estado de metilación del gen MGMT.

Se llevó a cabo una revisión exhaustiva de la literatura para identificar y seleccionar características relevantes y métodos de clasificación previamente utilizados en el campo de la oncología tumoral. Para validar la efectividad y precisión del modelo se utilizó un conjunto de imágenes de RM de pacientes con glioblastomas, incluyendo información sobre el estado de metilación del promotor del gen MGMT obtenida a través de técnicas de biología molecular. Se evaluó la solución utilizando métricas estándar como la precisión y el área bajo la curva ROC (AUC).

Los resultados obtenidos indican que el clasificador desarrollado es capaz de predecir con cierto éxito el estado de metilación del gen MGMT en tumores cerebrales a partir de imágenes de RM morfológica, ofreciendo un enfoque no invasivo y potencialmente más accesible para la evaluación de este importante biomarcador en pacientes con glioblastoma. Además, se discuten las limitaciones del estudio y se sugieren futuras investigaciones para mejorar la precisión y la aplicabilidad clínica del clasificador propuesto.

Palabras clave: MGMT (O6-metilguanina-ADN metiltransferasa), Imágenes de resonancia magnética (IRM), Redes neuronales convolucionales (CNN), biomarcador, aumento de datos, transferencia de aprendizaje

Abstract

The methylation of the MGMT gene (O6-methylguanine-DNA methyltransferase) is a key biomarker in predicting prognosis and treatment response in patients with glioblastoma, the most common and aggressive primary brain tumor. Standard techniques for detecting gene methylation include real-time PCR or bisulfite sequencing, invasive methods that require tumor tissue. This study proposes an innovative approach to medical image analysis that combines image processing techniques, deep learning and transfer learning to develop a classification model capable of detecting and evaluating the methylation status of MGMT in patients with glioblastomas. The proposed solution is based on the extraction of morphological and functional features from MRI images, which are relevant for the identification of the MGMT gene methylation status.

An extensive literature review was conducted to identify and select relevant features and classification methods previously used in the field of tumor oncology. To validate the effectiveness and accuracy of the model, a set of MRI images from the patients with glioblastoma was used, including information about images from patients with glioblastomas was used, including information on the methylation status of the MGMT gene promoter obtained through molecular biology techniques. The solution was evaluated using standard metrics such as accuracy or area under the ROC curve (AUC).

The results obtained indicate that the developed classifier is able to predict the methylation status of the MGMT gene in brain tumors with some success using morphological MRI images, offering a non-invasive and potentially more accessible approach for the assessment of this important biomarker in patients with glioblastoma. In addition, the study's limitations are discussed, and future research is suggested to improve the accuracy and clinical applicability of the proposed classifier.

Keywords: MGMT (O6-methylguanine-DNA methyltransferase), magnetic resonance images (MRI), convolutional neural networks (CNN), biomarker, radiogenomics, transfer learning

Tabla de contenidos

Índice de Figuras

Índice de Tablas

Abreviaturas

MGMT: O6-metilguanina-ADN metiltransferasa.
GBM: Glioblastoma multiforme
RSNA: Radiological Society of North America (Sociedad Radiológica de América del Norte).
MICCAI: Medical Image Computing and Computer Assisted Intervention (Computación de Imágenes Médicas e Intervención Asistida por Computadora).
TCGA: The Cancer Genome Atlas (El Atlas del Genoma del Cáncer).
TCIA: The Cancer Imaging Archive (El Archivo de Imágenes de Cáncer).
OMS: Organización Mundial de la Salud
IA: Inteligencia Artificial.
IRM: Imágenes de Resonancia Magnética.
MRI: Magnetic Resonance Images.
ML: Machine Learning (aprendizaje automático)
DL: Deep Learning (aprendizaje profundo)
GPU: Graphics Processing Unit
PCR: Polymerase Chain Reaction (Reacción en cadena de la polimerasa)
AUC: Area Under the Roc curve (Área bajo la curva ROC)
SNC: Sistema Nervioso Central
TC: Tomografía Computarizada
ADN: Ácido Desoxirribonucleico
TMZ: Temozolomida
T1/T1w: imágenes ponderadas en T1 (Tiempo de relajación longitudinal)
T2/T2w: imágenes ponderadas en T2 (Tiempo de relajación transversal)
FLAIR:Imagen con recuperación de la inversión atenuada por fluidos
T1wCE: Imagen ponderada en T1 después del contraste

1. Introducción

1.1 Motivación

El uso de la Inteligencia Artificial es indudable en diversos ámbitos de la sociedad. Cada vez más sectores se benefician de las ventajas que ofrecen las tecnologías que provee, y la medicina no iba a ser diferente al resto. Estamos hablando de un sector en constante cambio y evolución que necesita de los últimos avances para proporcionar los mejores servicios a sus pacientes. Es por eso que hay muchos ejemplos de implantación de IA en el campo de la Medicina. La inteligencia artificial (IA) es una tecnología que trata de combinar algoritmos planteados con el propósito de dotar a las máquinas de la capacidad de imitar el razonamiento humano. Las herramientas habilitadas para IA pueden identificar relaciones significativas en datos brutos y son potencialmente aplicables a casi todos los campos de la medicina, incluyendo el desarrollo de medicamentos, las decisiones sobre tratamientos, la atención al paciente y las decisiones financieras y operativas. La gama de aplicaciones y utilidades que arroja esta tecnología en el ámbito de la salud es cada vez más grande: soporte al proceso de toma de decisiones, agilización de procesos fundamentales (tanto en temas de atención al paciente como en la gestión sanitaria), proporcionar un diagnóstico no invasivo de ciertos problemas de salud (sobre todo en patologías oncológicas), facilita la obtención de biomarcadores o ahorrar costes entre muchos más.

En la actualidad, los casos de éxito de aplicación de la IA en el ámbito de la medicina son amplias. En el caso de la medicina oncológica, la detección del cáncer en edades tempranas es una prioridad. El desarrollo de algoritmos inteligentes que detecten casos positivos a través de datos visuales (imagen médica) es una de las principales vías de estudio. Investigadores de la Universidad de Valencia han obtenido resultados asombrosos en la detección del cáncer de mama a través del análisis de mamografías con técnicas de IA, sumado al conocimiento de profesionales de la radiología [1]. Por otro lado, el proyecto *iFIND*, llevado a cabo por el *BioMedIA* del Imperial College de Londres utiliza algoritmos de DL para el análisis de imágenes fetales. De esta forma, es posible estudiar cada parte del feto para detectar y atender posibles anomalías durante todo su desarrollo. En este proceso intervienen sistemas de resonancia magnética en 3D del cerebro fetal y una evaluación automatizada y precisa de los ultrasonidos realizados [2].

Este trabajo de final de grado es totalmente de mi interés debido a que se tratará de aplicar algoritmos de DL a distintos conjuntos de datos de imagen médica. Esto permitirá aplicar mis conocimientos sobre la materia y ampliarlos con creces, ya que el análisis de imágenes médicas no es un campo donde haya profundizado hasta la fecha y creo realmente que es muy interesante e innovador.

1.2 Objetivos

Este trabajo de fin de grado tendrá como objetivo general la creación de un modelo de clasificación basado en el aprendizaje profundo que trate de predecir el estado de metilación del gen MGMT en tumores cerebrales a partir de imágenes de resonancia magnética morfológicas y funcionales. Este objetivo general se desglosa en los siguientes objetivos específicos:

1. Realizar una revisión bibliográfica exhaustiva sobre la metilación de MGMT en tumores cerebrales y la utilización de imágenes de IRM morfológica y funcional para la predecir su estado de metilación.
2. Llevar a cabo un análisis exhaustivo de los proyectos previos que hagan uso de la inteligencia artificial para predecir la metilación del gen MGMT. Seguidamente, utilizar toda la información para establecer un estado del arte sólido del que partir a la hora de diseñar una solución propia.
3. Analizar diferentes técnicas de procesamiento de imágenes para la extracción de características relevantes para que ayuden al modelo de clasificación a aprender a distinguir con más facilidad el estado de metilación del promotor del gen.
4. Diseñar y desarrollar un modelo de clasificación binario basado en redes neuronales convolucionales con la meta de predecir el estado de metilación del gen MGMT utilizando como datos de entrada imágenes de resonancia magnética de tipo estructural y funcional.
5. Evaluar el rendimiento de la solución diseñada mediante pruebas con conjuntos de datos de IRM morfológica y funcional de glioblastomas previamente clasificados por personal experto.

En resumen, el objetivo principal será diseñar e implementar un modelo de clasificación que sea capaz de discriminar si la región tumoral perteneciente al gen MGMT se encuentra en estado de metilación o no, a partir de imágenes de IRM morfológica y funcional, mediante la aplicación de técnicas de procesamiento de imágenes y redes neuronales convolucionales. Posteriormente, se llevará a cabo el proceso de evaluación mediante pruebas con conjuntos de datos con etiquetas reales proporcionadas por personal experto.

1.3 Impacto esperado

En primer lugar cabe resaltar que el mundo de la visión por computador siempre ha sido de gran interés para mi persona desde que supe de su existencia. Por lo tanto tenía claro que quería desarrollar mi TFG en esta línea y surgió la oportunidad de aplicarlo al ámbito de la medicina. La aplicación de técnicas de DL también es a su vez un potente atractivo debido a que quedé completamente fascinado y comprendí el gran potencial que tenían cuando fue introducida en el último curso del grado. La variedad de problemáticas a la que es capaz de dar solución; los casos de éxito reales que están proporcionando auténticas mejoras en sectores como la medicina, el automovilístico o el entretenimiento; y sobre todo mi interés desmesurado en este campo en el cual espero pueda desarrollar mi carrera laboral en futuro.

Así pues, como alumno del grado en Ciencia de Datos espero plasmar mis conocimientos adquiridos hasta la fecha en materia de análisis de datos, así como reforzar otros aspectos no tan técnicos pero a la vez igual de importantes. Para llevar a cabo un proyecto del calibre de este trabajo de fin de grado es necesaria una buena organización y estructuración de todo el contenido, con la intención final de obtener resultados de calidad. Se pretende mostrar las aptitudes adquiridas en lo referente a procesamiento de imágenes, aplicación de modelos de redes convolucionales y demás procedimientos que se describirán posteriormente.

En cuanto a tecnología utilizada, el proyecto ha sido desarrollado en su totalidad utilizando el lenguaje de programación Python. Durante el grado ha sido la herramienta más utilizada ya que

gracias a su estabilidad, flexibilidad, facilidad de comprensión y variedad de recursos disponibles permite llevar a cabo infinidad de tareas de manera relativamente simple. Desarrollar algoritmos de DL puede ser costoso y requerir gran cantidad de tiempo. Python ofrece infinidad de librerías y marcos de trabajo enfocados al desarrollo de código para implementar tareas relacionadas con el aprendizaje profundo. Mi objetivo será pues desarrollar un marco de trabajo basado en DL y desarrollado en Python; que sea capaz de llevar a cabo el objetivo general del proyecto. Con ello espero poder aumentar mis conocimientos en el uso de librerías relacionadas con el aprendizaje profundo como puede ser *pytorch*, *numpy*, *pandas* o *seaborn* entre muchas otras.

Por último, a lo largo del grado se han impartido varias asignaturas que acarreaban la elaboración de proyectos grupales con alta complejidad y duración. Llevar a cabo dichos proyectos nos ha permitido mejorar nuestras habilidades de búsqueda de información, organización, habla en público, documentación o reacción frente a situaciones complejas. Son habilidades no tan vistosas, pero igual o más importantes que las técnicas. Espero firmemente cumplir con creces en todos estos apartados.

En cuanto a las ventajas que supondría para la comunidad médica y los pacientes de GBM el desarrollo de un modelo de DL que fuese capaz de predecir el estado de metilación del gen MGMT de manera óptima, se podrían encontrar:

1. Personalización del tratamiento: La capacidad de predecir el estado de metilación del gen MGMT permitiría una personalización más precisa de los tratamientos para el cáncer. Por ejemplo, en casos donde se detecte dicho estado, podría optarse por la administración de quimioterapia con agentes alquilantes, mientras que en el caso contrario, se optaría por otros métodos. Esto se explica con detenimiento en el punto X.
2. Pronóstico del paciente: Se observa que aquellos pacientes con metilación del gen MGMT suelen tener mayor esperanza de supervivencia global que aquellos sin metilación. Un modelo de DL otorgaría una herramienta útil para pronosticar la evolución probable de la enfermedad y orientar las decisiones de tratamiento.
3. Investigación y desarrollo: El modelo podría ayudar a identificar nuevos objetivos para medicamentos que pueden alterar el estado de metilación del gen MGMT.
4. Eficiencia y escalabilidad: Los métodos actuales para determinar el estado de metilación del gen MGMT pueden ser costosos y consumir mucho tiempo. El uso de un modelo de DL podría representar una forma más eficiente y escalable de obtener esa información.

1.4 Metodología

La metodología planteada para llevar a cabo este proyecto empieza en primer lugar con una amplia búsqueda de artículos en los que se haya aplicado de soluciones basadas en IA, que utilicen imágenes de IRM como fuente de datos, para resolver problemas de predicción de biomarcadores en enfermedades (a poder ser relacionados con tumores cerebrales). Seguidamente, también se han analizado tanto las soluciones propuestas por los participantes que obtuvieron los mejores resultados en la competición de *Kaggle*, así como repositorios de *Github* externos que han resultado de gran ayuda a la hora de diseñar y desarrollar la propuesta presentada.

Tras ello se ha elaborado un marco de trabajo propio donde se han ido añadiendo las distintas partes del código:

- Scripts relacionados con lectura y preprocesamiento de los datos.
- Scripts relacionados con el clasificador basado en redes neuronales convolucionales.
- Scripts relacionados con el entrenamiento y evaluación de los modelos planteados.
- Scripts de comprobación de independencia del conjunto de test.

A continuación se ha llevado a cabo la primera experimentación realizada sobre los datos de naturaleza morfológica proporcionados en la competición *RSNA-MICCAI Brain Tumor Radiogenomic Classificatio*n de *Kaggle* [3]. Se ha efectuado la interpretación de los resultados y se ha establecido las conclusiones del primer análisis.

Posteriormente se ha llevado a término una segunda experimentación, esta vez con datos privados de carácter funcional aplicando el conocimiento obtenido por la mejor solución en el contexto anterior. Se ha realizado a su vez el pertinente análisis de los resultados y evaluación de la segunda propuesta

Finalmente se ha realizado una dilatada reflexión sobre las conclusiones obtenidas y sobre los conocimientos adquiridos durante el proyecto; y se han expuesto propuestas de mejora de cara a futuras problemáticas similares que puedan llegar a surgir.

1.5 Estructura

La primera parte del proyecto tratará de aproximar al lector/a a la situación actual del problema tratado. Se explicará la definición de la patología analizada (el glioblastoma) y sus características; la importancia del estado del promotor de la proteína MGMT en el diagnóstico de la enfermedad y las técnicas que se están llevando a cabo para detectar la situación del promotor de la proteína.

A continuación se tratará de exponer las razones por las que la ciencia de datos puede ser útil en la detección del promotor del gen MGMT. A su vez se hará una resumen del sondeo realizado sobre soluciones propuestas por entidades e investigaciones en las que utilicen técnicas de aprendizaje automático para tratar de aportar soluciones alternativas a los métodos clásicos.

Seguidamente se pasará a exponer detalladamente la solución propuesta incidiendo en cada una de las diferentes fases en las que se ha dividido el proyecto y comentando cada una de las partes resaltables del código desarrollado. Finalmente, se analizarán las diferentes métricas seleccionadas para medir la calidad de la propuesta en relación con las presentadas en el estado del arte. Paralelamente se realizará una pequeña reflexión sobre posibles riesgos que puedan aparecer.

2. Estado del Arte

2.1 Introducción

Como se explicará con más exactitud posteriormente este se nutre de la competición de la plataforma Kaggle RSNA-MICCAI Brain Tumor Radiogenomic Classification [3]. Esta se centra en a partir del conjunto de datos proporcionado, tratar de proponer una solución basada en un modelo de aprendizaje (automático o profundo) que sea capaz de predecir el estado de metilación del promotor del gen O-6-methylguanine-DNA methyltransferase (MGMT) en pacientes con glioblastoma multiforme (GBM). La metilación del gen MGMT se ha utilizado como un indicador pronóstico en pacientes con GBM, por lo que la predicción precisa del biomarcador es importante para la planificación del tratamiento y evaluación del pronóstico del paciente.

Los datos proporcionados a los participantes constan de imágenes MRI funcionales de 585 pacientes en el conjunto de datos etiquetado (Público) y 87 pacientes en la base de datos de test sin etiquetas (Privado).

2.2 Metodologías Utilizadas

Tanto los participantes de la competición como estudios externos utilizaban diferentes enfoques para tratar de predecir la metilación del gen MGMT en pacientes con GBM [4]. Los siguientes han sido algunos de los métodos más utilizados:

1. Redes neuronales convolucionales (CNN) 3D: Los participantes utilizaron CNN para procesar los datos de IRM morfológica y predecir la metilación del gen MGMT. La mayoría utilizaban como dato de entrada volúmenes en tres dimensiones, transformando las series DICOM a formatos como NIFTI. Entre las arquitecturas de CNN más destacadas encontramos *ResNet*, *DenseNet*, *EfficientNet* o *Xception*.
2. Transferencia de aprendizaje (*Transfer Learning*): Es un conjunto de métodos que permite transferir conocimientos adquiridos gracias a la resolución de problemas para resolver otros problemas [4]. En el contexto de aprendizaje automático, el aprendizaje por transferencia es un enfoque en el cual un modelo desarrollado para una tarea específica se puede usar como punto de partida para el desarrollo de otro modelo destinado a otra tarea diferente de la primera. Se utiliza comúnmente porque ofrece ventajas como:
    a. Entrenar modelos con menos datos etiquetados al reutilizar modelos previamente entrenados con conjuntos de datos de gran tamaño [4].
    b. Reducir el tiempo de entrenamiento y los recursos computacionales necesarios [5].

    Este método ha tenido gran éxito especialmente con el crecimiento del DL. En el caso de las redes neuronales profundas, el aprendizaje por transferencia permite aprovechar las características de bajo nivel aprendidas por un modelo previamente entrenado en un

conjunto de datos grande y aplicarlas a un nuevo problema, ajustando solo las últimas capas de la red para adaptarla a la tarea específica en cuestión.

3. Ensamblaje de modelos: El aprendizaje en conjunto es una técnica que combina múltiples modelos de aprendizaje automático para obtener una mejor precisión y rendimiento en la tarea de predicción. La idea básica es que la combinación de modelos, cada uno con sus fortalezas y debilidades, puede dar lugar a un modelo más robusto y preciso que cada modelo individual por separado [6]. Por lo tanto, se centra en compensar los errores de predicción cometidos por un modelo mediante los aciertos del otro. Existen varias técnicas de que han sido utilizadas por los participantes:
    a. *Bagging*: La idea es generar múltiples modelos a partir de diferentes subconjuntos de datos de entrenamiento y se combinan mediante votación o promedio
    b. *Boosting*: Se entrenan secuencialmente modelos que se centran en los errores cometidos por los modelos anteriores, y se combinan mediante ponderación
    c. *Stacking*: Se utiliza un modelo maestro que combina las predicciones de varios modelos secundarios.
4. Aumento de datos: Se ha utilizado por la gran mayoría de participantes y la razón principal es el condicionante del tamaño del conjunto de datos de entrenamiento. Es una técnica utilizada en ML o DL y el procesamiento de imágenes para aumentar el tamaño de un conjunto de datos de entrenamiento mediante la creación de nuevas muestras a partir de las muestras existentes. Éste se utiliza para mejorar el rendimiento de los modelos y reducir el sobreajuste a través de la generación de nuevas muestras de datos que varían en ciertas características como: rotación, traslación, recorte, transformación geométrica y adición de ruido entre muchas otras [7]. De esta manera se aumenta la diversidad de los datos de entrenamiento y se ayuda al modelo a generalizar mejor en los datos nuevos y desconocidos.

2.3 Resultados obtenidos

Como se ha explicado previamente, la competición contaba con dos conjuntos de datos (público y privado). Los resultados obtenidos en el primero, que contaba con etiquetas de los datos, ha sido en gran medida esperanzador debido a la gran multitud de soluciones presentadas que obtenían valores por encima del 0.95 de AUC, principal métrica utilizada para evaluar el rendimiento de los modelos. Sin embargo estos resultados distaban mucho a la realidad debido a que como se argumentaba en el foro de discusión de la competición, gran parte de los participantes habían basado sobreentrenado sus modelos para que se ajustaran casi a la perfección con los datos de entrenamiento. Este hecho se evidenciaba en la tabla de clasificación del conjunto de datos privado donde gran parte de las soluciones casi perfectas obtenían un rendimiento muy por debajo de lo esperado. Todos los participantes con soluciones ganadoras (10 mejores clasificados) coinciden en que la clave era tratar de minimizar el sobreajuste en la base de datos pública, aunque debido al tamaño del ésta era difícil evitar que los modelos fueran capaces de obviar el ruido y las características no relacionadas con la región metilada o no del gen MGMT.

2.4 Crítica al estado del arte

Aunque los resultados obtenidos por algunas de las soluciones presentadas en la clasificación de imágenes médicas son prometedores, aún existen limitaciones y desafíos que deben abordarse. La falta de datos etiquetados de alta calidad es un problema común en la clasificación de imágenes médicas. En general la conclusión que acuerdan es que los resultados no son clínicamente útiles, pero el conocimiento obtenido por la utilización de modelos 3D sobre imágenes MRI para tareas de clasificación binaria es realmente útil para futuros problemas.

Así que la crítica a lo anteriormente explicado viene a ser más una reflexión sobre la importancia de diversos aspectos en una competición de este calibre, si el objetivo es obtener soluciones clínicamente útiles. Claramente la intencionalidad detrás de este tipo de desafíos de ciencia de datos es abordar problemáticas novedosas, hacer comunidad con el resto de participantes y mostrar el talento individual. El primer aspecto es la cantidad de datos disponibles, es entendible que las imágenes médicas son archivos de gran tamaño por la cantidad de información que contiene, pero con un número tan limitado es difícil que las soluciones aprendan las características de las imágenes de entrada y sean capaces de generalizar en datos nuevos. Una opción sería incluir otros tipos de datos clínicos y moleculares que podrían proporcionar información adicional para mejorar la precisión de los modelos de clasificación. Otro aspecto sería el hecho de compartir desarrollos y resultados: Salvo algunas excepciones, la mayoría de participantes no comparten el desarrollo del código. Sería útil proporcionar una plataforma para que los participantes compartan sus avances y desafíos, fomentando la colaboración entre participantes e incluso promover la posibilidad de colaborar con expertos en áreas propias como la biología del cáncer, la genómica y la radiología.

En cuanto a la discusión y soluciones compartidas de la competición, ha sido muy enriquecedor y de alta inspiración para realizar la solución propuesta en este proyecto.

2.5 Propuesta

En este proyecto se diseñará, implementará y evaluará modelos basados en aprendizaje profundo para la predicción del estado de metilación del MGMT de un tumor a partir de IRM morfológica. Además se utilizará información adicional proveniente de secuencia de IRM de perfusión (o funcional) que podrían mejorar de manera significativa la precisión predictiva del modelo a la hora de determinar el objetivo.

3. Análisis del problema

3.1 Aspectos epidemiológicos e importancia de la metilación del gen MGMT como biomarcador pronóstico.

El glioblastoma multiforme (GBM) es el tumor más común de los cánceres primarios del SNC (sistema nervioso central); es causante del 12-15% de todos los tumores intracraneales y al 50-60% de todos los gliomas [8]. La tendencia creciente de su incidencia es debida en gran parte, a una mayor esperanza de vida en la población. En EEUU por ejemplo, la incidencia anual es de 3,21 por cada 100.000 habitantes [9].

Son tumores altamente agresivos y se desarrollan rápidamente, invadiendo y destruyendo el tejido cerebral circundante. La causa exacta de la aparición de esta patología no se conoce con exactitud. Sin embargo, existen evidencias de que ciertos factores pueden aumentar el riesgo de desarrollar dichos tumores cerebrales. Algunos de estos incluyen: la exposición a radiación, ciertos trastornos genéticos (inferior al 5% total de casos diagnosticados) y antecedentes familiares de tumores cerebrales [10]. En relación a la edad, se trata de un cáncer epidemiológicamente más acusado en pacientes de entre 45 y 70 años, independientemente del género (no existen diferencias significativas, en la incidencia entre ambos sexos) [11] .

La clasificación y gradación histológica de los tumores primarios del sistema nervioso es importante para determinar su comportamiento biológico y pronóstico, así como para establecer la indicación terapéutica adecuada. Desde 1979, la Organización Mundial de la Salud ha incluido un sistema de gradación histológica que es ampliamente utilizado en la actualidad. Este sistema divide los gliomas en cuatro grados, siendo el glioblastoma multiforme (GBM) un tumor de grado IV debido a su alta malignidad (tabla 1).

La última clasificación actualizada de la OMS, publicada en 2021, incorpora criterios moleculares además de los criterios histológicos para clasificar los gliomas [12]. Esto pretende reflejar mejor la biología tumoral y su evolución clínica, supervivencia y respuesta al tratamiento. Es importante tener en cuenta estos criterios moleculares para una clasificación precisa y una selección adecuada del tratamiento.

CNS WHO Grades of Selected Types	
Astrocytoma, IDH-mutant	2, 3, 4
Oligodendroglioma, IDH-mutant, and 1p/19q-codeleted	2, 3
Glioblastoma, IDH-wildtype	4
Diffuse astrocytoma, *MYB*- or *MYBL1*-altered	1
Polymorphous low-grade neuroepithelial tumor of the young	1
Diffuse hemispheric glioma, H3 G34-mutant	4
Pleomorphic xanthoastrocytoma	2, 3
Multinodular and vacuolating neuronal tumor	1
Supratentorial ependymoma[a]	2, 3
Posterior fossa ependymoma[a]	2, 3
Myxopapillary ependymoma	2
Meningioma	1, 2, 3
Solitary fibrous tumor	1, 2, 3

Tabla 1. Clasificación WHO de gliomas en función del grado

El diagnóstico del glioblastoma multiforme se realiza a través de una combinación de técnicas de imagen, análisis patológico y evaluación clínica [13]. A continuación, se describen los pasos que se suelen seguir para diagnosticar esta enfermedad:

1. Historial médico y examen físico: El primer paso en el diagnóstico del glioblastoma multiforme es el examen clínico, en el cual el médico puede realizar preguntas sobre los síntomas y la historia médica del paciente.
2. Imágenes cerebrales: La resonancia magnética (RM) es la técnica de imagen más utilizada para diagnosticar el glioblastoma multiforme. La RM permite obtener imágenes detalladas del cerebro, lo que permite a los médicos ver cualquier anomalía en la estructura del cerebro, como la presencia de tumores. También se puede realizar una tomografía computarizada (TC) en algunos casos.
3. Biopsia: Para confirmar el diagnóstico de glioblastoma multiforme, se realiza una biopsia, que consiste en la extracción de una muestra del tejido tumoral. Esto se hace mediante la inserción de una aguja fina en el cerebro bajo la guía de una imagen de RM. La muestra se analiza en un laboratorio para detectar la presencia de células tumorales.
4. Análisis patológico: Una vez obtenida la muestra de tejido, se realiza un análisis patológico para determinar el grado y el tipo de cáncer. El glioblastoma multiforme es un tipo de tumor cerebral de grado IV, lo que significa que es un tumor maligno y agresivo.
5. Evaluación de la extensión del tumor: Después del diagnóstico, se realiza una evaluación de la extensión del tumor para determinar si se ha extendido a otras áreas del cerebro o del cuerpo. Esto se hace mediante técnicas de imagen adicionales, como una RM de cuerpo entero o una TC.

En resumen, el diagnóstico del glioblastoma multiforme requiere una combinación de técnicas de imagen, análisis patológico y evaluación clínica para confirmar la presencia del tumor y determinar su grado y extensión.

El pronóstico de los tumores mencionados es desfavorable y, si no se tratan, puede llevar a la muerte en un corto periodo de tiempo. Aunque la supervivencia media ha mejorado de 12 a 15

meses, muchos pacientes experimentan una progresión de la enfermedad en los primeros 6-9 meses. La tasa de supervivencia a los dos años no alcanza el 30% y menos del 10% de los pacientes sobreviven a los cinco años [14]. Además, el padecimiento de esta enfermedad puede resultar en una gran incapacidad tanto física como cognitiva en muchos casos, lo que puede tener graves consecuencias tanto para el paciente como para su entorno familiar y social. Es importante destacar que se requiere un tratamiento temprano y efectivo para mejorar las posibilidades de supervivencia y calidad de vida.

En lo relativo al tratamiento aplicado, la combinación de radioterapia y temozolomida es el más comúnmente utilizado para pacientes con glioblastoma multiforme. Este tratamiento se conoce como terapia de radiación y quimioterapia concurrente.

La temozolomida es un medicamento de quimioterapia que se utiliza para tratar ciertos tipos de cáncer cerebral, incluyendo el glioblastoma multiforme. Es un agente alquilante, lo que significa que trabaja al agregar grupos químicos al ADN de las células cancerosas. Estos grupos químicos se adhieren a las bases nitrogenadas del ADN, interrumpiendo la replicación y transcripción del ADN y provocando daño en la cadena de ADN [15]. La temozolomida se descompone en el cuerpo en una forma activa llamada metil-triazeno-imidazol-carboxamida (MTIC), que es capaz de atravesar la membrana celular y entrar en el núcleo de las células cancerosas. En el núcleo, MTIC se convierte en una forma altamente reactiva que se adhiere a las bases nitrogenadas del ADN, lo que provoca daño en la cadena de ADN [15].

La célula cancerosa intenta reparar el daño en el ADN, pero en algunos casos, las células no pueden repararlo de manera efectiva, lo que lleva a su muerte [15]. El daño al ADN también puede activar una respuesta del sistema inmunológico que ataca las células cancerosas.

La temozolomida tiene un efecto selectivo sobre las células cancerosas porque las células cancerosas se dividen más rápidamente que las células normales y tienen una mayor tasa de división celular [16]. Las células normales, que se dividen más lentamente, tienen una mayor capacidad para reparar el daño en el ADN causado por la temozolomida. Sin embargo, algunas células cancerosas tienen la capacidad de reparar el daño en el ADN causado por la temozolomida a través de una enzima llamada metil-guanina metil transferasa (MGMT).

El gen MGMT codifica una proteína llamada O6-metilguanina-ADN metiltransferasa, que tiene la capacidad de reparar el ADN dañado por la temozolomida [17]. En otras palabras, si el gen MGMT está activo, las células cancerosas pueden reparar el daño causado por la temozolomida y, por lo tanto, ser resistentes al tratamiento.

Por lo tanto, la expresión del gen MGMT en las células cancerosas puede ser un factor importante en la respuesta de los pacientes al tratamiento con temozolomida. En la práctica clínica, se puede realizar una prueba de expresión del gen MGMT para determinar si un paciente es un buen candidato para el tratamiento con temozolomida, o si se deben considerar otras opciones de tratamiento.

La identificación temprana de glioblastomas es crucial para mejorar la supervivencia de los pacientes. Para ello es muy necesario contar con los llamados “biomarcadores”. Son sustancias medibles en el cuerpo, de tipo clínico o molecular, que se asocian con una enfermedad específica. Los biomarcadores son muy adecuados para estimar con precisión el pronóstico y seleccionar el tratamiento óptimo. Además, también pueden utilizarse para determinar la eficacia del tratamiento y para detectar la reaparición del tumor después de la cirugía y otros tratamientos.

El biomarcador que se va a llevar a análisis en este proyecto es el mencionado anteriormente, el estado de metilación del promotor del gen MGMT. Como se ha explicado anteriormente, la inactivación a través de la metilación del promotor del gen O^6-metilguanina-ADN-metiltransferasa (MGMT), que perjudica la capacidad de reparación del ADN dañado inducida por agentes alquilantes como la TMZ [17], se ha descrito como un biomarcador relevante para la toma de decisiones clínicas en el tratamiento del glioblastoma.

En un análisis post-hoc de un ensayo de fase III, la metilación del promotor de MGMT se asoció con un aumento de la supervivencia a los dos años en los pacientes con glioblastoma tratados con TMZ del 14% al 46% [17]. En este estudio, se consiguió determinar el estado de metilación del gen MGMT en el 67% de los pacientes mediante la realización de una prueba llamada metilación específica del genoma por PCR (MSP-PCR). Tras ello se diseñó un experimento en el cual se asignaba (de manera aleatoria) al 50% de los pacientes un tratamiento exclusivo de radioterapia y la otra mitad radioterapia + temozolomida.

Como se puede observar en la figura 1, la probabilidad de supervivencia de los pacientes con el gen MGMT en estado de metilación es considerablemente mayor a partir del primer año. A los 18 meses por ejemplo los pacientes sin metilación tienen una esperanza de vida del 15% frente al prácticamente 50% del otro grupo.

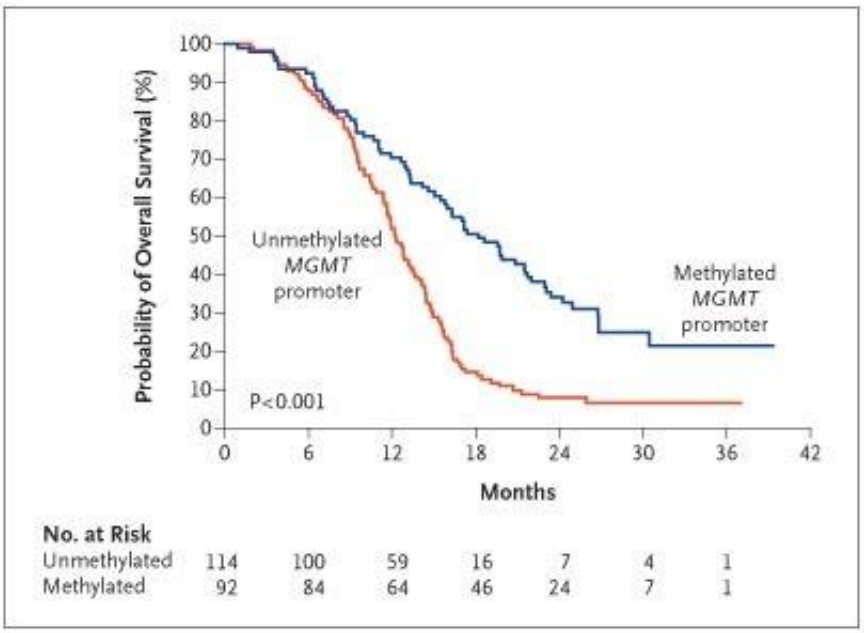

Figura 1. Kaplan-Meier Estimación de supervivencia promedio respecto al estado de metilación del gen MGMT. Fuente [17]

Si además a del estado de metilación se añade a la comparativa en el tratamiento aplicado en los pacientes, se puede observar que aquellos pacientes con MGMT metilado tratados con temozolomida y radioterapia cuentan con un 65% de probabilidad de supervivencia a los 18 meses; aquellos con MGMT metilado pero tratados solo con radioterapia se reduce en un 25% [16]; el grupo con ausencia de metilación en el gen presenta peores resultados en ambos tipos de tratamientos aplicados (figura 2).

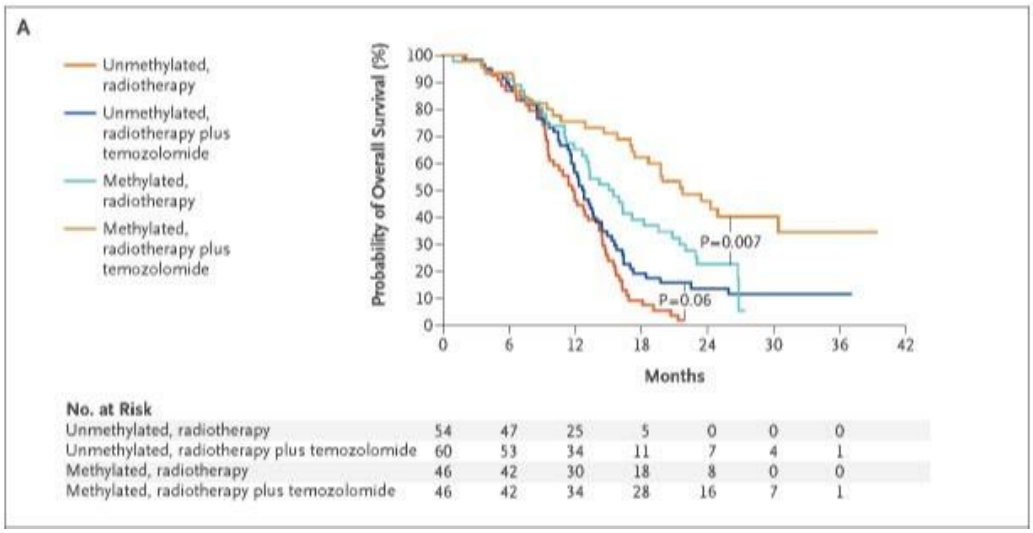

Figura 2. Kaplan-Meier Estimación de supervivencia promedio respecto al estado de metilación del gen MGMT y asignación aleatoria a tratamiento con radioterapia y temozolomida o solo radioterapia. Fuente: [17]

La importancia de contar con información sobre el estado de metilación del gen es realmente crucial a la hora de escoger el tratamiento óptimo para cada paciente. Con esto se tratará de maximizar la esperanza de vida pero siempre atendiendo a los contra efectos ocasionados por la aplicación de agentes alquilantes como la temozolomida.

A pesar del impacto bien documentado de la metilación de MGMT en el pronóstico de los pacientes con glioblastoma tratados con TMZ, la supervivencia de estos pacientes no se explica únicamente por este factor. Algunos estudios han demostrado que la vascularidad del tumor es una variable fuertemente asociada a la malignidad del tumor y la supervivencia del paciente (un valor anormal se correlaciona con baja supervivencia y mayor grado de malignidad) [18]. Además esta característica también se vincula con una alta sensibilidad a la radioterapia así como, a la eficacia de la administración de nutrientes y quimioterapia por vía sanguínea [19].

Para poder caracterizar los glioblastomas y obtener información valiosa acerca de la vascularidad del tumor y las propiedades de los vasos sanguíneos se utilizan técnicas avanzadas de resonancia magnética (RM). Entre ellas, destaca la perfusión que permite evaluar distintas características fisiopatológicas de los glioblastomas. Los parámetros de perfusión, como el volumen sanguíneo cerebral relativo (VSCr), se correlacionan con la vascularidad tumoral y las propiedades de los vasos, que, a su vez, están fuertemente asociados con la progresión del grado del glioma y una peor supervivencia [20]. Por lo tanto, la perfusión se considera una poderosa herramienta pronóstica para estratificar los gliomas según su agresividad, predecir la supervivencia de los pacientes y, finalmente, ayudar en la elección del tratamiento

Otro aspecto a tener en cuenta es la manera en la que se determina la metilación o ausencia de ésta en el gen. Se busca siempre procedimientos fiables, de calidad, no invasivos y con un tiempo de respuesta temprano [21]. Actualmente, los métodos más comunes para establecer el estado de metilación del promotor del gen MGMT son:

1. Secuenciación bisulfita: La secuenciación bisulfita es una técnica que permite determinar el estado de metilación de una región específica del ADN. En este procedimiento, el ADN

se trata con bisulfito de sodio, que convierte los residuos de citosina no metilados en uracilo, mientras que los residuos de citosina metilados permanecen intactos [21]. Luego, se amplifica la región del promotor del gen MGMT mediante PCR y se secuencia el producto amplificado. La secuencia obtenida se compara con la secuencia original para determinar la presencia o ausencia de metilación en la región.

2. PCR en tiempo real (qPCR): La qPCR es una técnica que se utiliza para medir la cantidad de ADN en una muestra. En el caso del análisis del estado de metilación del promotor del gen MGMT, se utilizan sondas fluorescentes específicas para la región metilada y no metilada del promotor [21]. Luego, se amplifica la región mediante PCR en tiempo real y se mide la cantidad de fluorescencia emitida por cada sonda. La relación entre las señales emitidas por las sondas metilada y no metilada indica el estado de metilación del promotor del gen MGMT.
3. Ensayo de metilación específica de PCR (MSP): El ensayo de MSP es una técnica que permite determinar la presencia o ausencia de metilación en una región específica del ADN. En este procedimiento, se amplifican dos productos diferentes mediante PCR, uno para la región metilada y otro para la región no metilada del promotor del gen MGMT [21]. Luego, se analizan los productos amplificados mediante electroforesis en gel de agarosa. La presencia de una banda en el gel indica la presencia de metilación, mientras que la ausencia de una banda indica la ausencia de metilación.

En general, los métodos para determinar el estado de metilación del promotor del gen MGMT no son invasivos, ya que no requieren la obtención de tejido o muestra directamente del tumor o del sitio de la lesión. El método de secuenciación bisulfita implica el aislamiento del ADN a partir de muestras de sangre, saliva o tejido tumoral, y su posterior tratamiento con bisulfito de sodio antes de la amplificación y secuenciación de la región del promotor del gen MGMT. El ensayo de MSP y la qPCR también utilizan ADN aislado de muestras de sangre, saliva o tejido tumoral, y no son invasivos en sí mismos.

Es importante tener en cuenta que, en algunos casos, puede ser necesario obtener una muestra de tejido tumoral mediante biopsia para realizar un análisis más detallado del estado de metilación del promotor del gen MGMT u otros marcadores moleculares. En estos casos, la biopsia sí se consideraría un procedimiento invasivo [21].

3.2 Resonancia Magnética: Formación de la imagen MRI

La generación de imágenes por RM se basa en un principio fundamental de la física, que establece que cualquier átomo con un número impar de cargas positivas (protones) posee la propiedad de rotar sobre sí mismo, conocida como “spin”. Esta rotación produce un momento magnético que sigue trayectorias aleatorias hasta que es influenciado por un campo magnético externo [22].

Desde una perspectiva médica, el núcleo atómico de hidrógeno es el más utilizado, ya que es el elemento más abundante en el cuerpo humano, constituyendo aproximadamente el 70% del mismo en forma de agua (H20). El agua tiene la propiedad de poseer un número impar de protones en su núcleo, lo que le permite producir protones móviles. La cantidad de protones móviles de hidrógeno en un tejido específico, en relación con el agua, se conoce como densidad de spin o densidad de protones. Las imágenes por IRM se obtienen midiendo la concentración y el tiempo

de relajación del núcleo atómico de hidrógeno en las moléculas de agua, que son excitados por un campo magnético fijo y un campo de radiofrecuencias [22].

La resonancia magnética cerebral es una prueba de imágenes no invasiva y segura que utiliza un campo magnético y ondas de radiofrecuencia para producir imágenes detalladas del cerebro y del tronco cerebral. La resonancia magnética estructural convencional se aprovecha de que los diferentes tipos de tejidos en el cerebro contienen diferentes proporciones de agua, lo que influye en la formación de la imagen. Debido a que la resonancia magnética es particularmente útil para la obtención de imágenes de tejidos blandos, se pueden obtener imágenes de alta calidad del cerebro con buenos detalles anatómicos, ofreciendo alta sensibilidad y especificidad en comparación con otras modalidades de imágenes para muchos tipos de afecciones neurológicas. La resonancia magnética también permite una mejor visualización con el uso de agentes de contraste y combinaciones de diferentes tipos de secuencias.

Existen varias modalidades de imagen por resonancia magnética, cada una con sus propias características y aplicaciones. Algunas de las principales modalidades incluyen:

1. RM estructural o morfológica: Es la forma más común de RM, que se utiliza para visualizar la anatomía de la zona corporal y evaluar enfermedades o lesiones en órganos, tejidos y estructuras internas [23]. La RM estructural incluye secuencias ponderadas en T1, T1 con contraste, las imágenes ponderadas en T2 y las imágenes recuperadas por inversión atenuada en llamadas FLAIR. En particular, estas últimas son especialmente útiles para la evaluación de lesiones en el tejido cerebral y las meninges [23].
2. RM de perfusión por contraste dinámico (DSC): Esta técnica utiliza un agente de contraste paramagnético (gadolinio usualmente), que se inyecta en el torrente sanguíneo del paciente. El contraste pasa a través de los vasos sanguíneos del tejido que se está examinando y la máquina de RM toma una serie de imágenes rápidas que muestran cómo este contraste cambia la señal a medida que pasa el tiempo. Estas variaciones se utilizan para calcular mapas de parámetros hemodinámicos, que pueden incluir el volumen sanguíneo cerebral (CBV), el tiempo de tránsito medio (MTT), la velocidad de paso máximo (TTP) y la tasa de flujo sanguíneo cerebral (CBF) [23]. Es particularmente útil para evaluar la perfusión del cerebro; así como para el diagnóstico y seguimiento de accidentes cerebrovasculares y tumores cerebrales
3. RM de difusión (DWI): Esta técnica se basa en el movimiento aleatorio de las moléculas de agua en los tejidos y permite evaluar la integridad de las estructuras microscópicas, como las fibras nerviosas. La DWI se emplea frecuentemente para identificar accidentes cerebrovasculares, lesiones cerebrales traumáticas y tumores [23].

Por otro lado, las imágenes de resonancia magnética se pueden representar en diversos planos anatómicos, que son cortes bidimensionales del cuerpo. Los tres planos principales en los que se pueden representar las imágenes de resonancia magnética son:

1. Plano axial (o transversal): Este plano se orienta horizontalmente y divide el cerebro en partes superiores o inferiores. Proporciona una vista transversal
2. Plano sagital: Es un corte vertical que divide el cerebro en lados izquierdo y derecho. Proporciona una vista lateral.

3. Plano coronal (o frontal): Este plano también es un corte vertical, pero divide el cerebro en partes anterior (frontal) y posterior (trasera). Proporciona una vista frontal.

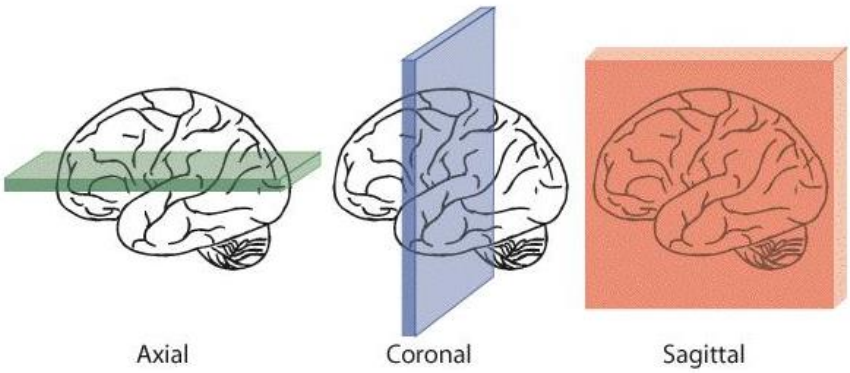

Figura 3: Descripción gráfica de la orientación y división según planos de representación de imágenes de resonancia magnética. Fuente: https://www.researchgate.net/figure/Imagenes-en-los-planos-sagital-coronal-y-axial_fig1

Las imágenes obtenidas mediante escáneres resonancia magnética presentan una gran cantidad de información sobre características multimodalidad que puedan ser aprovechadas mediante técnicas de DL para predecir características genéticas de los glioblastomas (véase, el estado de metilación del promotor del gen MGMT). La gran resolución, contraste y clara separación del tejido sensible permite a los modelos de aprendizaje profundo.

3.3 Relación entre imagen médica y redes neuronales convolucionales

Las redes neuronales convolucionales son una técnica de DL que se ha utilizado con gran éxito en diversas tareas en el ámbito de la imagen médica (clasificación, detección, análisis de supervivencia, segmentación, etc.). Una CNN es una red neuronal que consta de múltiples capas de convolución, capas de agrupación y capas completamente conectadas, que se utilizan para obtener características visuales a partir de las imágenes. Las capas de convolución son capaces de aprender características locales, mientras que las capas completamente conectadas pueden aprender características más globales y abstractas [24]. El funcionamiento de las CNN se puede resumir en los siguientes pasos y se puede observar en la figura 4:

1. Convolución: La convolución es la primera capa de la CNN, que se encarga de aplicar filtros a la imagen para extraer características significativas. Los filtros se aplican en pequeñas áreas de la imagen, llamadas ventanas, y se desplazan por toda la imagen para cubrir cada píxel. Esto genera una serie de mapas de características, que contienen información sobre los patrones de la imagen [25].
2. Submuestreo: Después de la capa de convolución, se aplica una capa de submuestreo para reducir el tamaño de los mapas de características. Esto se hace mediante la selección de la característica más significativa en cada ventana y eliminando las demás. Esto reduce el número de parámetros de la red, lo que la hace más eficiente [25].
3. Capa completamente conectada: Después de la capa de submuestreo, se aplica una capa completamente conectada para clasificar las imágenes. Esta capa utiliza los datos de los

mapas de características para identificar patrones en la imagen y determinar la clase a la que pertenece [25].

4. Retro propagación: Es un algoritmo de aprendizaje que se utiliza para ajustar los pesos de la CNN. La red se entrena con un conjunto de imágenes etiquetadas, y la retropropagación se utiliza para ajustar los pesos de la red para minimizar el error entre la salida de la CNN y la etiqueta real de la imagen [25].

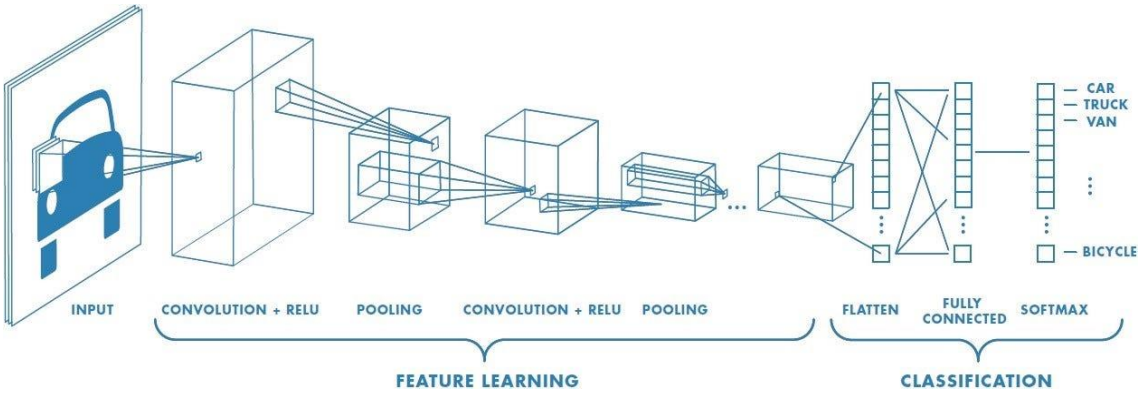

Figura 4. Guía de una red neuronal convolucional. Fuente: https://saturncloud.io/blog/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way/

El uso de las CNN en el análisis de imágenes médicas es relativamente reciente, pero ha crecido rápidamente en popularidad los últimos años. Las CNN fueron desarrolladas originalmente en la década de 1980, pero su uso en análisis de imágenes médicas no se hizo viral hasta la década de 2010, cuando la tecnología de aprendizaje profundo comenzó a ser más accesible [26]. Desde entonces, se ha evidenciado con numerosos estudios las posibilidades que brinda una herramienta como las CNN en el ámbito de la imagen médica. Algunos de los ejemplos más relevantes son:

1. Diagnóstico de cáncer de pulmón: Un estudio publicado en *Nature Medicine* en el 2019 demostró que una CNN entrenada en imágenes de tomografía computarizada (TC) de tórax era capaz de detectar el cáncer de pulmón con una precisión similar a la del personal experto en radiología [27].
2. Detección de enfermedades oculares: Una solución basada en CNN entrenada en imágenes de retina de pacientes con diabetes ha demostrado ser capaz de predecir la progresión de la retinopatía diabética, una complicación ocular común en este tipo de pacientes [28].
3. Diagnóstico de enfermedades de piel: Un estudio publicado en *Annals of Oncology* en 2018 demostró que una CNN entrenada en imágenes de melanoma era capaz de diagnosticar el cáncer de piel con una precisión similar a profesionales expertos de la dermatología [29].
4. Seguimiento de la progresión de la enfermedad del Alzheimer: Un modelo de CNN entrenado con IRM ha demostrado ser capaz de predecir la progresión del Alzheimer en pacientes en una etapa temprana[30].

Estas soluciones pueden ayudar al personal médico a establecer riesgos de desarrollar patologías, detectar enfermedades en etapas tempranas, mejorar las tasas de supervivencia, planificar tratamientos o evaluar la efectividad de los mismos en la progresión de la enfermedad.

En cuanto a eficiencia, las CNN son una solución óptima para el análisis de imagen médica por varias razones:

1. Capacidad para extraer características: Las CNN pueden aprender a extraer características significativas de las imágenes médicas de manera automatizada. Esto significa que no se necesitan características manuales que podrían ser difíciles de definir y que podrían ser diferentes de una imagen a otra. Las CNN pueden aprender las características directamente de las imágenes sin necesidad de una intervención manual [31].
2. Aprendizaje profundo: Se basan en la técnica de aprendizaje profundo, lo que les permite analizar imágenes médicas con gran precisión. Los modelos de CNN pueden aprender a reconocer patrones sutiles en las imágenes, lo que puede ser difícil para los métodos tradicionales de procesamiento de imágenes [31].
3. Escalabilidad: Las CNN se pueden utilizar para analizar grandes conjuntos de datos de imágenes médicas con relativa facilidad. Dado que éstas son capaces de aprender automáticamente, pueden manejar grandes cantidades de datos sin necesidad de intervención humana [31].
4. Velocidad: Son capaces de analizar las imágenes médicas rápidamente, lo que las hace ideales para su uso en la práctica clínica. Esto significa que los resultados pueden obtenerse rápidamente, lo que puede ayudar a mejorar la eficiencia del diagnóstico y tratamiento de enfermedades [31].

Y en cuanto a la aplicabilidad, estas son algunas de las tareas en las que se aplican las CNN en análisis de imagen médica:

1. Detección de enfermedades: Pueden utilizarse para detectar enfermedades como cáncer de mama, cáncer de pulmón, enfermedades cardíacas, enfermedades neurológicas, entre otras [31].
2. Segmentación de imágenes: Las CNN pueden utilizarse para segmentar imágenes médicas, lo que significa dividir una imagen en regiones específicas, como tumores, tejido sano y órganos [31].
3. Registro de imágenes: Otro de sus usos es alinear imágenes tomadas en diferentes momentos o desde diferentes perspectivas. Es importante para el seguimiento de la progresión de enfermedades, evaluación de la respuesta al tratamiento y la planificación quirúrgica [31].
4. Clasificación de imágenes: Las CNN son capaces de clasificar imágenes médicas en diferentes categorías, como diferentes etapas de la enfermedad o diferentes tipos de lesiones [31].

En este caso, se han aplicado modelos de CNN en una tarea de clasificación binaria para que sea capaz de predecir el estado de metilación del promotor del gen MGMT.

3.4 Análisis del marco legal y ético

Las imágenes IRM utilizadas en este proyecto son proporcionadas por la plataforma *Kaggle* en su competición *RSNA-MICCAI Brain Tumor Classification*. Para asegurar la anonimización de las imágenes y salvaguardar la privacidad de los pacientes, la organización diseñó un preprocesamiento donde el primer paso era convertir los escáneres del formato original DICOM

a NIFTI. Esta conversión elimina los metadatos asociados con las series DICOM incluyendo toda la información sensible o PHI (Información Sanitaria Protegida) [32]. Además la técnica de *skull-stripping* aplicada posteriormente reduce el alcance de la reconstrucción facial y a su vez minimiza la posibilidad de reconocimiento facial del paciente. Finalmente se aplicó, tras la reconversión a formato DICOM, un proceso de desidentificación de datos que consiste en el en la herramienta de anonimizado RSNA-CTP (Procesador de Ensayos Clínicos) [33]. Este elimina todas las etiquetas innecesarias, asegurando la eliminación de todas las entradas de PHI de los encabezados DICOM. Gracias a lo anteriormente explicado será posible cumplir con las leyes y regulaciones locales e internacionales, como el Reglamento GDPR [34] de la Unión Europea o la HIPAA de los Estados Unidos [35].

Respecto a los artículos de investigación consultados, webs y discusiones de foros acerca de la competición de *Kaggle* abordada; se han utilizado como herramienta orientativa a la hora de plantear el contenido y estructura del proyecto. Se ha utilizado también para intentar crear un hilo conductor que guíe a los lectores o lectoras a lo largo de la memoria de este trabajo de fin de grado.

En lo referente al marco de trabajo creado para proporcionar la solución, los scripts de Python son en mayor parte de elaboración propia. Se ha consultado algunas de las soluciones planteadas por los participantes que obtuvieron los mejores resultados así como estudios externos a la competición para diseñar la solución planteada; pero todo ello con la finalidad de mera inspiración. Se han utilizado ciertas funciones de código abierto proporcionadas por las propias librerías de Python o repositorios públicos de *GitHub*, que forman parte del conocido como *software libre* [36].

3.5 Análisis de riesgos

A continuación se expone un análisis de riesgos centrado en aspectos clave del proyecto:

1. Riesgo de incumplimiento de la protección de datos y privacidad: Si no se garantiza la adecuada anonimización de las imágenes IRM, existe el riesgo de exponer datos personales y sensibles de los pacientes. Esto podría resultar en violaciones de leyes de protección de datos, multas y daños a la reputación del proyecto.
2. Riesgo de sesgo y discriminación: Si el modelo de clasificación perpetúa o introduce sesgos injustos, existe el riesgo de tomar decisiones clínicas inapropiadas o discriminatorias basadas en los resultados del modelo. Esto podría afectar negativamente la calidad de la atención médica y conducir a posibles litigios.
3. Riesgo de mala interpretación o adopción inadecuada del modelo: Si el modelo no es transparente, explicativo o validado adecuadamente, existe el riesgo de que los profesionales médicos lo interpreten incorrectamente o lo utilicen de manera inapropiada. Esto podría llevar a decisiones clínicas incorrectas y posibles daños a pacientes.

Podrían recalcarse también el riesgo de infracción de la propiedad intelectual, pero la mayoría del código es de desarrollo propio y los datos son de carácter abierto por lo que no conlleva ninguna implicación para el proyecto.

3.6 Identificación y análisis de soluciones posibles

En el presente TFG se analizarán múltiples soluciones para abordar la predicción del estatus de metilación del gen MGMT. Dado que no existe una única solución, se explorarán diversas alternativas, se evaluarán individualmente mediante diversas métricas y más tarde comparará el rendimiento con el resto, determinando sus ventajas y desventajas. Para el diseño de las soluciones se implementarán modelos de clasificación basados en arquitecturas de redes convolucionales 3D pre entrenadas, en las que se rediseñará la capa de clasificación para adaptarla a las necesidades del problema.

Se establecerá un criterio de selección para identificar la solución más adecuada a desarrollar en el proyecto. Las métricas de evaluación, junto con el coste computacional o el progreso de la función de pérdida serán aspectos clave a la hora de elegir la mejor alternativa. Es fundamental transmitir que este trabajo se aborda desde una perspectiva científica, donde se evaluarán y analizarán distintas soluciones antes de seleccionar la más idónea.

3.7 Solución propuesta

En esta sección se va a presentar la solución propuesta al problema planteado. Para lograrlo, se seguirá un proceso estructurado que incluye la definición del problema, la preparación de los datos, la construcción del modelo, la experimentación y la evaluación. En la figura 5 se presenta un cronograma que resume el proceso a seguir.

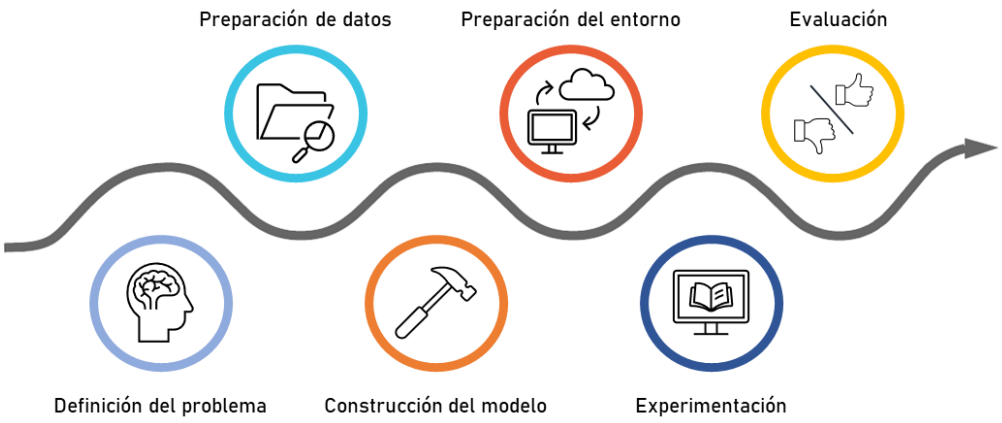

Figura 5: Cronograma de la solución propuesta

Fase 1: Definición del problema

Para comenzar a plantear la solución al problema presentado los primeros pasos serán

1. Establecer el objetivo: Como se ha comentado en anteriores apartados, el objetivo principal es clasificar las imágenes de IRM de pacientes con glioblastoma en aquellos que tengan el estado del promotor del gen MGMT metilado (clase positiva o 1) y aquellos en los que no está metilado (clase negativa o 0). Por lo tanto el problema de clasificación es de naturaleza binaria debido a que son dos las posibles clases que se pueden predecir.

2. Selección de métricas de evaluación relevantes: En este caso la métrica principal que medirá el rendimiento será el AUC ya que en la competición RSNA-MICCAI *Tumor Radiogenomic Classification* de la que se obtienen los datos, éste es el indicador que se utiliza para evaluar el desempeño de las soluciones propuestas por los participantes. También se utilizará la precisión para dar soporte a las conclusiones aportadas por los valores de la métrica principal.

Fase 2: Preparación de datos

Esta fase se ha explicado en anteriores apartados, pero aquí se detallarán cada una de las etapas:

1. Recopilación del conjunto de datos: Como se ha mencionado con anterioridad el conjunto de datos utilizado ha sido el proporcionado por la competición de *Kaggle* RSNA-MICCAI *Tumor Radiogenomic Classification*. El conjunto de datos cuenta con dos bases de datos (público y privado), donde el primero cuenta con imágenes IRM en formato DICOM y etiquetas de cada caso indicando si el gen MGMT se encuentra en estado de metilación o no. El segundo no cuenta con etiquetas debido a que está pensado para que se trate de predecir el estado de metilación de estos casos y se envían las predicciones a la competición y así se evalúe el rendimiento de la solución presentada. En el proyecto solo se hizo uso del conjunto de datos "público" debido a que el conjunto de prueba es el formado por los datos obtenidos de las instituciones TCGA y TCIA, que sí cuentan con etiquetas reales.
2. División del conjunto de datos: Se han probado distintas estrategias de particionamiento de datos con el objetivo de comprobar que técnica de división en subconjuntos se adapta mejor a la naturaleza del problema y trata de evitar mejor el sobreajuste.
3. Preprocesamiento: Se ha diseñado una estrategia para transformar tanto el conjunto de entrenamiento como el de test para asegurar que los datos estén en la mejor forma posible para maximizar la calidad y eficiencia del análisis posterior.

Fase 3: Construcción del modelo

Considerando un aspectos cruciales como el tamaño del conjunto de datos, la naturaleza del problema o las características de los datos de entrada, el plan para construir los modelos de clasificación binaria ha sido:

1. Utilización de arquitecturas de CNN 3D: Las CNN son un tipo de modelo de aprendizaje profundo que ha demostrado ser muy efectivo en tareas de imagen médica. Son especialmente útiles debido a su capacidad de procesar y aprender de los datos a nivel de píxeles. Pueden detectar patrones y características de las imágenes lo que puede ser útil para identificar enfermedades o anomalías. Un ejemplo de arquitectura 3D se puede observar en la figura 6.

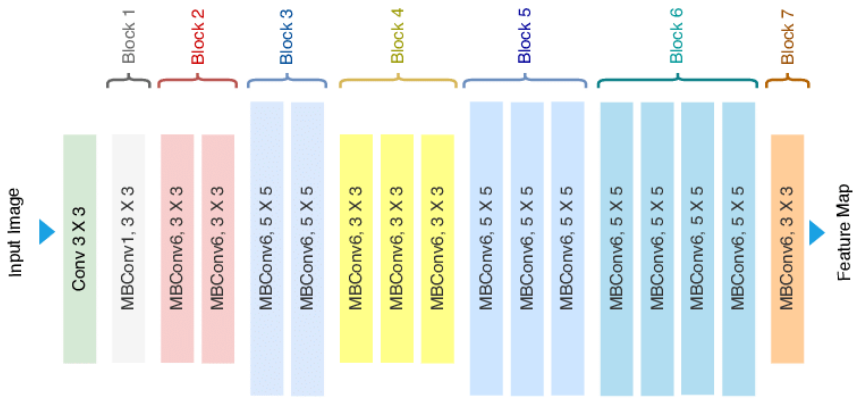

Figura 6. Arquitectura de una EfficientNet-B0. Fuente: https://www.researchgate.net/figure/Architecture-of-EfficientNet-B0-with-MBConv-as-Basic-building-bloc

2. *Transfer Learning*: Se ha explicado en anteriores capítulos las ventajas de aplicar la transferencia de aprendizaje cuando el tamaño del conjunto de datos de entrada no cuenta con el tamaño suficiente.

3. Selección de tasa de aprendizaje, tamaño del lote, función de pérdida y optimizador: Es fundamental seleccionar adecuadamente los hiper parámetros para mejorar el rendimiento y la eficiencia del entrenamiento en un modelo de aprendizaje automático.

Fase 4: Preparación del entorno de trabajo

Los recursos necesarios para realizar el entrenamiento del modelo han supuesto requerir del uso de GPUs de los servidores del BDS Lab (Biomedical Data Science Lab) - Itaca UPV. Ha sido necesario pues adaptar el marco de trabajo a los requisitos del entorno de ejecución, mediante la utilización de la herramienta Docker.

Fase 5: Validación del modelo

La fase de validación implica evaluar el rendimiento del modelo en un conjunto de datos de validación, monitorear las métricas de rendimiento relevantes y realizar ajustes en la arquitectura de la red, hiper parámetros y preprocesamiento. Es crucial para garantizar que el modelo tenga un buen rendimiento en datos nuevos y desconocidos y cumpla con los objetivos del proyecto antes de pasar a la evaluación final en el conjunto de datos de test:

1. Evaluación del modelo en el conjunto de validación: Durante esta etapa se evalúa el rendimiento del modelo entrenado utilizando el conjunto de datos de validación, que es un conjunto de datos no utilizado en el proceso de entrenamiento. La validación permite obtener una estimación preliminar del rendimiento del modelo en datos nuevos y desconocidos antes de realizar la evaluación final en el conjunto de test. También permite identificar posibles problemas de sobreajuste o subajuste.
2. Monitoreo de las métricas de rendimiento relevantes: Es necesario medir el desempeño del modelo en la fase de validación mediante las métricas seleccionadas para la tarea de clasificación binaria.

3. Ajuste de la arquitectura de red, hiper parámetros y preprocesamiento: Si los resultados de validación no cumplen con los objetivos o las expectativas, es posible que sea necesario ajustar cualquiera de los aspectos mencionados. Estos ajustes deben de realizarse de manera iterativa y sistemática, y se debe monitorear el impacto de cada cambio en las métricas de rendimiento para identificar qué ajustes son más efectivos.

Fase 6: Evaluación del modelo

Esta fase implica medir el rendimiento del modelo en un conjunto de datos independiente al de entrenamiento y validación. Para ello se monitorizan las métricas escogidas para medir la eficiencia de la solución:

1. Evaluación del modelo en el conjunto de datos: En esta etapa, se evalúa el rendimiento del modelo entrenado utilizando el conjunto de datos de prueba, que es un conjunto de imágenes que no se ha utilizado durante el proceso de entrenamiento y validación. Esto permite obtener una estimación más precisa del rendimiento del modelo en datos nuevos y desconocidos, lo cual es crucial para entender cómo se comportaría el modelo en el mundo real.
2. Medición de métricas de rendimiento relevantes: Estos indicadores pueden proporcionar información valiosa acerca de la capacidad de la solución para generalizar su desempeño en la clasificación del gen MGMT, en datos independientes a los utilizados en las fases anteriores.

4. Preparación y comprensión de los datos

4. 1 Descripción de los datos

La base de datos principal se corresponde con la incorporada en la competición de *Kaggle RSNA-MICCAI Brain Tumor Radiogenomic Classification 2021*. Éste es un desafío organizado conjuntamente por la RSNA y MICCAI con el objetivo de desarrollar modelos de IA que puedan predecir de manera efectiva el estado de metilación del promotor del gen MGMT utilizando IRM. En pacientes con casos de glioblastoma, la metilación del promotor MGMT es un biomarcador importante ya que está relacionado con una mejor respuesta al tratamiento con agentes alquilantes como la TMZ.

La base de datos proporcionada para esta competición consiste en 585 casos de pacientes con el tumor previamente mencionado. La colección de datos incluye para cada paciente 4 series DICOM. Es un estándar ampliamente conocido en la industria médica para almacenar y compartir imágenes médicas, como IRM, TC y radiografías. El formato DICOM no solo almacena la información de la imagen, sino también metadatos relevantes, como detalles del paciente, parámetros de adquisición y detalles del equipo de escaneo. Estos metadatos pueden ser útiles

para el preprocesamiento y análisis de las imágenes en aplicaciones de aprendizaje automático y procesamiento de imágenes médicas.

Por otro lado se ha incorporado otro conjunto de datos complementario gracias a TCGA y TCIA, tras contactar y obtener el consentimiento para la utilización en el ámbito académico de la base de datos TCGA-GBM [37]. Este conjunto contiene 262 casos de pacientes de GBM que incluyen las 4 modalidades (FLAIR, T1w. T1wCE y T2w) de IRM. Además cuentan también con casos de perfusión DSC, pero de los 262 pacientes proporcionados solo se cuenta con 80 que contengan todas las modalidades completas en su conjunto de imágenes. Complementariamente aporta información relativa al estado de metilación del promotor del gen MGMT, así como de otros biomarcadores que no serán objeto de estudio en este proyecto.

En definitiva, el conjunto de datos contiene:

1. Imágenes IRM y perfusión DSC: Las imágenes de resonancia magnética se proporcionaron en cuatro tipos de secuencias diferentes: FLAIR, T1-weighted, T1-weighted con contraste (realzado gadolinio) y T2-weighted. Estas secuencias ayudan a los médicos a caracterizar los tumores y a evaluar su extensión y gravedad. En el caso de TCGA incluye una modalidad extra que es perfusión DSC.
2. ID del paciente: Cada paciente en la base de datos tenía un identificador único para mantener un seguimiento adecuado de sus datos
3. Etiquetas de metilación del promotor de MGMT: La presencia o ausencia de metilación del promotor de MGMT en el tumor se indicó con etiquetas binarias (1 para metilado y 0 para no metilado). Esta información es relevante porque la metilación del promotor de MGMT se ha asociado con una mejor respuesta al tratamiento y un mejor pronóstico en pacientes con glioblastoma.
4. Datos de entrenamiento y test: La base de datos se dividió en conjuntos de training y test. El conjunto de entrenamiento incluía imágenes de IRM y etiquetas de metilación del promotor del gen MGMT para cada paciente del conjunto BRATS-MICCAI, mientras que el conjunto de prueba sólo contenía imágenes de IRM del conjunto TCGA-GBM.

4.2 Análisis Exploratorio

Haciendo uso de la librería pandas de python se ha realizado un histograma para analizar las distribuciones de del estado de metilación del promotor del gen MGMT en el conjunto de datos proporcionado. Para ello se ha hecho uso del archivo “train-labels.csv” el cual incluye los identificadores de las imágenes y su valor de metilación (0: no metilado, 1:metilado). Como se puede observar en la figura 7, se cuenta con un dataset relativamente balanceado. Sin embargo en la distribución de los valores del estado de metilación en el conjunto de test sí que se puede apreciar un ligero desbalance favorable a los casos no metilados.

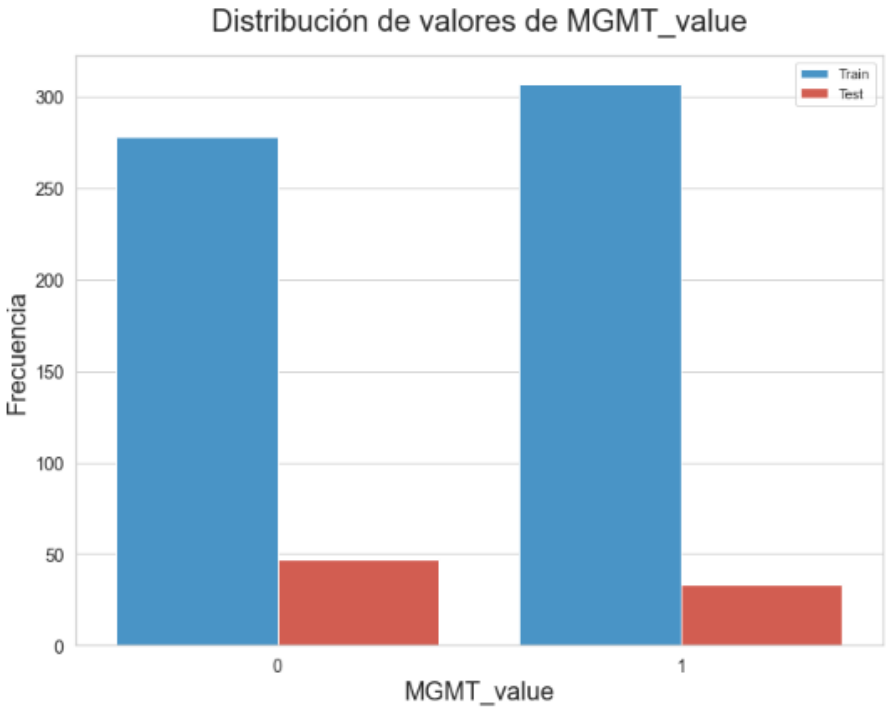

Figura 7: Distribución de metilación (derecha) y no metilación (izquierda) en los datos de entrenamiento y prueba.

Para entender mejor los datos de partida, se han representado diversos casos de cada una de las modalidades complementando con información respecto al estado de metilación del promotor del gen MGMT. Véase figuras 8 y 9.

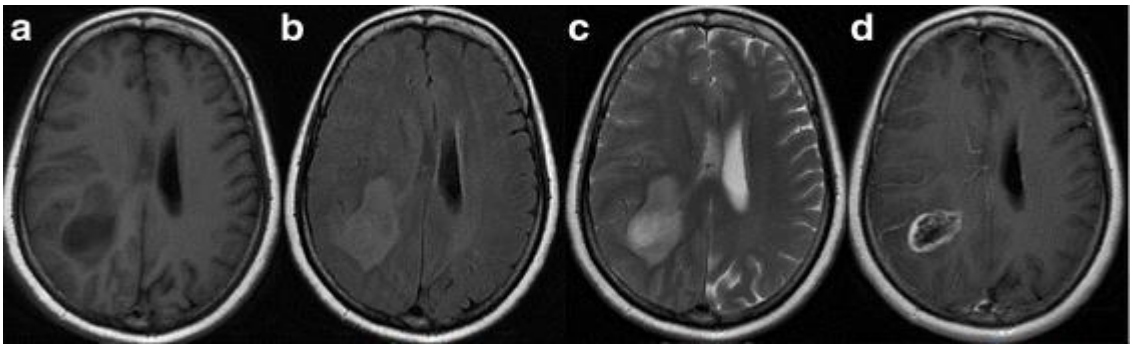

Figura 8: Estructurales (a, b, c y d) obtenidas de un hombre de 55 años que muestra glioblastoma multiforme (GBM) en los lóbulos temporal y parietal derecho sin metilación del promotor MGMT. a) T1w b) FLAIR c) T2w d) Postcontraste T1w. Fuente: [38]

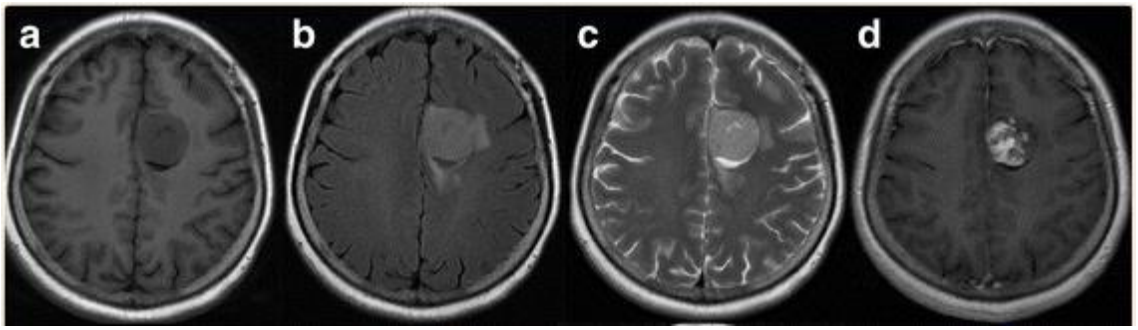

Figura 9: Estructurales (a, b, c y d) obtenidas de un hombre de 55 años que muestra glioblastoma multiforme (GBM) en los lóbulos temporal y parietal derecho con metilación del promotor MGMT. a) T1w b) FLAIR c) T2w d) Postcontraste T1w. Fuente: [38]

4. 3 Preprocesamiento

El preprocesamiento estandarizado con el que se presentaban los datos incluía varias técnicas, entre ellas:

1. Conversión de formato de DICOM a formato NIFTI: La conversión a NIFTI elimina los metadatos adjuntos de las imágenes DICOM y, esencialmente, elimina toda la información de Salud Protegida (PHI) de los encabezados DICOM.
2. Co-registro al modelo SRI 24: Este proceso se basa en alinear y registrar imágenes de resonancia magnética (IRM) de diferentes pacientes a un modelo anatómico de referencia común, en este caso, el atlas SRI 24. El objetivo de este proceso es asegurar que todas las imágenes estén en el mismo espacio anatómico y tengan las mismas dimensiones, lo que facilita la comparación y el análisis de las imágenes en investigaciones y aplicaciones de aprendizaje automático.
3. Remuestreo a una resolución isotrópica uniforme ($1mm^3$): Consiste en ajustar la resolución espacial de las imágenes de resonancia magnética (IRM) para que tengan el mismo tamaño de vóxel (unidad volumétrica) en todas la direcciones (x, y, z). En este caso, la resolución isotrópica deseada es 1 mm^3, lo que significa que cada vóxel tiene dimensiones de 1 mm x 1 mm x 1 mm. Al tener una resolución isotrópica uniforme en todas las imágenes, se garantiza que las características y estructuras en las imágenes sean comparables y estén en la misma escala.
4. *Skull-stripping*: Esta técnica de preprocesamiento de imágenes consiste en separar y eliminar las estructuras óseas del cráneo y otros tejidos no cerebrales de las imágenes de resonancia magnética (IRM). El objetivo principal de esta técnica es centrarse en el tejido cerebral y mejorar la precisión y eficiencia en el análisis de las imágenes. Además es particularmente útil en:
    a. Reducir el ruido y las interferencias en las imágenes
    b. Facilitar la comparación entre sujetos al enfocarse en el tejido cerebral.
    c. Reducir los requisitos de almacenamiento y procesamiento.

Las imágenes pertenecientes a la segunda base de datos no contaban con el procesamiento previamente explicado. Se han aplicado las técnicas requeridas al subconjunto TCGA-GBM utilizando la librería de Python *CBrainMRIPrePro* [39].

Tras el preprocesamiento inicial aplicado a los datos proporcionados por la competición RSNA-MICCAI Tumor Radiogenomic Classification de Kaggle, se han aplicado diversas técnicas para mejorar la calidad de los datos de entrada. Cabe indicar que los datos se facilitaban en formato DICOM (tras el procesamiento se reconvirtieron al formato original). Al basar la solución presentada en redes neuronales convolucionales en tres dimensiones se debe, tras la etapa de preprocesamiento propia, convertir la series DICOM en volúmenes 3D. Las características de las imágenes son las siguientes:

- Dimensiones: 256 x 256 x 64 (altitud, amplitud, profundidad)
- Utilizadas las 4 modalidades suministradas (FLAIR, T1w, T1wCE, T2w)

4.3.1 Carga de datos

Para efectuar la preparación de los datos se ha creado una clase en Python a modo de cargador de datos. Es una utilidad que suele gestionar varias tareas como: la lectura de datos desde bases de datos, la aplicación de transformaciones y el cambio de formato. En este caso esta utilidad devuelve el tensor 3D creado a partir de las imágenes originales en formato DICOM. Entre las transformaciones que se aplicarán a los datos antes de ser procesados destacan:

1. Eliminación de bordes oscuros: El método de eliminar bordes oscuros en el procesamiento de imágenes, también conocido como *crop* o recorte, es útil para eliminar las áreas no relevantes o que no contienen información de interés en una imagen. En muchas ocasiones, los bordes oscuros pueden ser el resultado de la adquisición de imágenes, artefactos o simplemente áreas vacías fuera del objeto o área de interés. Uno de los puntos a destacar es la optimización de los recursos computacionales. Al recortar áreas no relevantes se disminuye el tamaño de la imagen, lo que a su vez reduce tiempo y recursos necesarios para procesar y analizar las imágenes [40].
2. Normalización de la intensidad: Este proceso consiste en la búsqueda de la homogeneización de las intensidades de los píxeles en una imagen o conjunto de imágenes. Es especialmente útil en el análisis de imágenes médicas, donde las variaciones en variaciones en las intensidades pueden ser el resultado de diferencias en la adquisición de imágenes, calibración del equipo o características intrínsecas del tejido [41].

Otro de los aspectos que se implementa en la fase de la carga de datos es la estrategia de división del conjunto de datos. Se han probado diversos métodos entre los que se destaca los basados en validación cruzada y el mantenimiento de la proporción de las clases objetivo. Tras realizar varias experimentaciones, la estrategia escogida para la división de los conjuntos de entrenamiento y validación fue el *Stratified Split*.

Es un método de particionamiento de los datos que se utiliza para mantener la proporción de las clases objetivo en cada partición de los datos [42]. Por ejemplo, se va a suponer que se cuenta con un conjunto de datos de clasificación binaria con un 80% de las clases pertenecientes a la clase 0 y un 20% a la clase 1. En un escenario ideal, se querría que esa proporción (80/20) se mantenga tanto en datos de entrenamiento como en los de validación. Sin embargo, si se realiza una división aleatoria de los datos, es posible que se termine con una proporción diferente en ambos conjuntos. Aquí es donde entra en juego el *Stratified Split* ya que asegura que la proporción de las clases objetivo se mantenga en cada partición de los datos. Así el *Stratified Split* garantizará que el

conjunto de entrenamiento y el conjunto de validación mantengan esa proporción 80/20 de las clases 0 y 1.

Esto es especialmente útil cuando se trabaja en conjunto de datos desequilibrados, donde una clase tiene muchas menos muestras que la otra. Sin un *Stratified Split*, se podría terminar con un conjunto de entrenamiento que no contiene ninguna muestra de la clase minoritaria, lo que haría que el modelo fuera incapaz de predecir esa clase

4.3.2 Generador de datos

Un generador de datos en aprendizaje automático es una función o un método que produce datos de manera continua y controlada. Esto puede ser particularmente útil cuando se trabaja con grandes cantidades de datos que no se pueden cargar en la memoria a la vez, o cuando se desea aumentar la cantidad de datos disponibles para el entrenamiento. Entre sus utilidades se encuentran:

1. Carga de los datos en lotes (*load batching*): Durante el entrenamiento de un modelo, los datos suelen ser procesados en lotes más pequeños para mejorar la eficiencia computacional y la convergencia del modelo. El *data generator* puede dividir los datos en lotes de un tamaño específico y garantizar que se entreguen de manera uniforme al modelo durante el entrenamiento.
2. Asignación aleatoria (*Shuffling*): Es el proceso de reordenar aleatoriamente las observaciones en un conjunto de datos. El objetivo principal de esta técnica es asegurar que el modelo de DL no aprenda patrones incorrectos o no deseados debido a algún tipo de orden o secuencia. Por ejemplo, si en un problema de clasificación binaria como el actual, todas las observaciones positivas estuvieran al principio del conjunto de datos y todas las negativas al final, el modelo podría empezar a "aprender" que las primeras observaciones son positivas y las últimas siempre negativas.
3. Reescalado: Este método procesa la imágenes para equilibrar la resolución de todos los datos de entrada al modelo. Las razones son simples, en términos computacionales un tamaño óptimo de las imágenes reduce la cantidad de datos que el modelo debe procesar, lo que acelera el tiempo de entrenamiento. Se debe tener en cuenta también que el reescalado puede resultar en una pérdida de detalles en las imágenes, por lo tanto es imprescindible armonizar la necesidad de reducir el tamaño con mantener suficientes detalles en las imágenes para que el modelo pueda aprender de manera efectiva.
4. Paralelización: Algunos generadores de datos también pueden cargar y procesar datos en paralelo utilizando múltiples núcleos de CPU o dispositivo de GPU. Esto puede mejorar significativamente la velocidad y la eficiencia del proceso de entrenamiento, especialmente cuando se trabaja con grandes conjuntos de datos o transformaciones intensivas en el tiempo de cálculo.

4.3.2 Aumento de datos

Dentro del proceso de generación de datos, uno de los métodos empleados más importantes ha sido el aumento de datos. Es una técnica empleada en el campo de las soluciones basadas en DL, como las CNN, para el reconocimiento o clasificación de imágenes.

La idea principal detrás del aumento de datos es generar un conjunto de datos más grande y diverso a partir de los datos existentes, lo que puede mejorar el rendimiento y la precisión del modelo [43]. Esto se logra aplicando diversas transformaciones a las muestras de datos originales, lo que crea nuevas versiones de estos datos. Algunas de las transformaciones comunes incluyen:

1. Rotación: Girar la imagen en diferentes imágenes.
2. Escala: Aumentar o disminuir el tamaño de la imagen.
3. Traslación: Desplazar la imagen en diferentes direcciones.
4. Reflexión: Crear reflejos de la imagen en el eje horizontal o vertical.
5. Cambio de brillo: Modificar la intensidad de los colores en la imagen.
6. Ruido: Añadir ruido aleatorio a la imagen.
7. Recorte: Seleccionar una parte de la imagen y usarla como una nueva muestra.

Estas transformaciones pueden ayudar al modelo a aprender características más generales y ser más robusto ante variaciones en los datos de entrada, lo que reduce el riesgo de sobreajuste y mejora la capacidad del modelo de generalizar a partir de los datos de entrenamiento. La figura 10 muestra su aplicación:

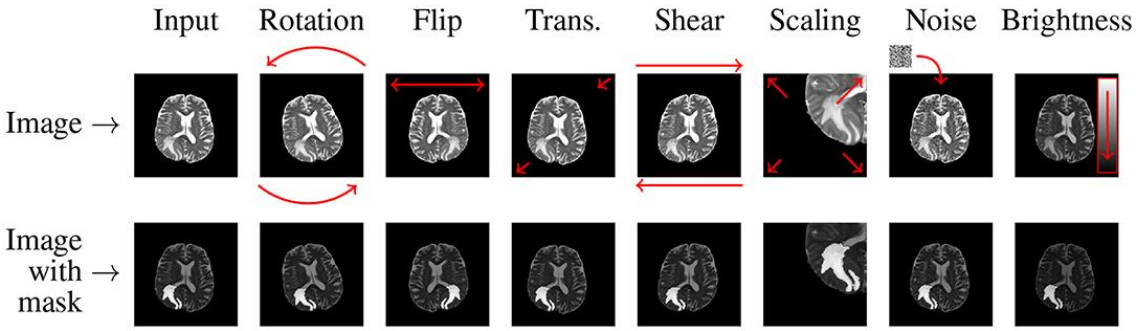

Figura 10. Aplicaciones del data augmentation sobre una imágen de IRM. Fuente: [43]

En este proyecto se ha aplicado el método de aumento de datos para enriquecer el conjunto de datos de entrenamiento y tratar que los modelos planteados sean capaces de generalizar y no sobre aprender de los datos, además de adaptarse a cambios en los datos de entrada. Se ha creado una función llamada *keras_augment* basada en la librería *Keras*. Ésta cuenta con funciones de aumento de datos integradas, pueden ser usadas tanto en el proceso de generación de datos o se pueden incorporar en la definición del modelo de entrenamiento para aprovechar el potencial de las GPUs. Se ha optado finalmente por incluirlas en el *data generator* y las capas de aumento que se han utilizado han sido: rotación, cambio de brillo, recorte y traslación. Además se han creado algunas funciones que implementan capas de aumento personalizadas. Estas se pueden consultar en el Anexo II.

5. Conocimiento extraído y evaluación de modelos

5.1 Funciones aplicadas

En el Anexo II está disponible el código Python de cada una de las funciones o procesos explicados seguidamente.

5.1.1 BrainTSGenerator()

Esta clase es una implementación personalizada de un generador de datos para entrenar modelos de deep learning en *Tensorflow*/*Keras*. Está diseñada específicamente para trabajar con IRM cerebrales almacenadas en formato DICOM. La clase hereda de t*ensorflow.keras.utils.Sequence* lo que permite una carga y procesamiento eficiente de los datos en tiempo real durante el entrenamiento.

La función '__init__' inicializa el generador con la ruta de imágenes DICOM y los datos asociados (etiquetas, identificadores de pacientes y estado de metilación MGMT).

La función '__len__' devuelve la cantidad de pacientes en el conjunto de datos.

La función '__getitem__' se llama cuando se solicita un entrenamiento específico del conjunto de datos. Esta función es responsable de:

1. Cargar las imágenes de las modalidades específicas (FLAIR, T1w, T1wCE Y T2w) para un paciente en particular.
2. Realizar un preprocesamiento básico en las imágenes, como eliminar bordes negros y normalizar las imágenes.
3. Crear un tensor 3D para cada modalidad con las imágenes DICOM preprocesadas.
4. Devolver el tensor 3D y la etiqueta MGMT asociada al paciente.

5.1.2 Keras_Augment()

Esta clase define varias funciones y clases para realizar aumento de datos en MRI durante el entrenamiento de modelos de DL. Las siguientes funciones de aumento de datos se definen utilizando el módulo de preprocesado de *Tensorflow*:

1. 'rand_flip': Aplica un volteo aleatorio horizontal y vertical a la imagen.
2. 'rand_tran': Aplica una traslación aleatoria a la imagen.
3. 'rand_rote': Aplica una rotación aleatoria a la imagen.
4. 'rand_cntr': Aplica un contraste aleatorio a la imagen.
5. 'rand_crop': Aplica un recorte aleatorio a la imagen y luego redimensiona a la altura y ancho de entrada.

La función ‘keras_augment’ toma una imagen y una etiqueta como entrada y aplica las funciones de aumento de datos definidas anteriormente a la imagen. Luego devuelve la imagen aumentada y la etiqueta sin cambios.

Además se definen dos capas personalizadas de Keras para realizar aumentos adicionales:

1. ‘RandomEqualize’: Aplica una ecualización aleatoria del histograma a la imagen con una probabilidad dada.
2. ‘RandomCutOut’: Aplica un recorte aleatorio en la imagen, reemplazando una región rectangular por un valor constante, con una probabilidad dada.

5.1.3 TFDataGenerator()

Esta clase es un generador de datos que se encarga de preparar y cargar los datos en mini lotes para el entrenamiento, validación e inferencia de modelos de aprendizaje profundo. La clase acepta varios argumentos que permiten configurar aspectos como la aplicación de aumento de datos, el tamaño de lote y la normalización de las imágenes. Los argumentos son:

1. ‘data’: Datos de entrada (imágenes y etiquetas).
2. ‘shuffle’: Indica si se deben mezclar los datos (sólo aplicado durante el entrenamiento).
3. ‘augment’: Indica si se debe aplicar aumento de datos a las imágenes.
4. ‘batch_size’: Tamaño del lote de imágenes para cargar en cada iteración.
5. ‘rescale’: Indica si se deben reescalar las imágenes (normalizar los valores de los píxeles).

La función ‘get_3D_data’ de la clase ‘TFDataGenerator’ se encarga de aplicar el aumento de datos, agrupar las imágenes en lotes y, si se solicita, reescalarlas. Finalmente utiliza la función ‘prefetch’ para optimizar la carga de datos durante el entrenamiento.

5.1.4 Get_3D_Model()

La función ‘get_model’ crea y retorna un modelo de clasificación basado en IRM en 3D. Los argumentos de la función definen las dimensiones de las imágenes de entrada. El flujo de la función es:

1. Se crea un tensor de entrada con las dimensiones especificadas (altura, anchura, profundidad)
2. Se carga el modelo base (ej. EfficientNetV2B0), utilizando la clase *Classifier* de la librería classification_models_3D, con los pesos pre entrenados de *ImageNet*, excluyendo la capa superior [52].
3. A continuación, se aplica una serie de capas adicionales para adaptar el modelo a la tarea específica de clasificación binaria:
    - GlobalAveragePooling3D: Para reducir la dimensionalidad de las capas extraídas.
    - BatchNormalization: Para normalizar las activaciones y mejorar la velocidad de entrenamiento.
    - Dropout: Para regularizar el modelo y reducir el sobreajuste.
    - Dense: Para añadir capas ‘fully connected’ con funciones de activación *ReLU*.

- La última capa Dense tiene una activación sigmoide para realizar clasificación binaria (MGMT metilado o no metilado).

4. Finalmente, se crea y retorna un modelo *Keras* que toma el tensor de entrada y produce el tensor de salida.

5.1.5 BrainTumorModel3D()

En el entrenamiento de modelos 3D, se necesita utilizar tamaños pequeños de 'batch size' debido al gran coste computacional, aspecto que se quiere evitar. La clase 'BrainTumorModel3D'es una implementación personalizada de un modelo *Tensorflow* que hereda de la clase *tf.keras.Model*. Tiene las siguientes características:

1. '__init__': El constructor de la clase acepta un modelo base y un número de gradientes 'n_gradients' para realizar la acumulación de éstos. La acumulación de gradientes es una técnica para dividir las actualizaciones de los parámetros del modelo en varios pasos de entrenamiento, lo que puede ser útil cuando se trabaja con GPUs con memoria limitada.
2. 'train_step'/'test_step': Este método calcula las predicciones y en la pérdida utilizando el modelo base y el conjunto de datos de entrada en el paso de entrenamiento y test. En el primero calcula y acumula los gradientes antes de aplicarlos para actualizar los parámetros del modelo cuando se alcanza el número de gradientes especificado. En el test, no se actualizan los parámetros.
3. 'call': Éste método permite que la clase actúe como función, llamando al modelo base y a los datos de entrada proporcionados.
4. 'reg_l2_loss': Este método calcula la pérdida de regularización L2 para los parámetros entrenables del modelo base. La regularización L2 es una técnica que se utiliza para evitar el sobreajuste al agregar una penalización a los parámetros del modelo.
5. 'apply_acu_gradients': Este método aplica los gradientes acumulados a los parámetros entrenables del modelo base y luego reinicia la acumulación de gradientes.

5.2 Explicación del proceso de entrenamiento

El conjunto de datos está formado por una serie de pacientes de los que se proporcionan 4 secuencias distintas generadas a partir de imágenes IRM (FLAIR, T1w, T1wCE Y T2w) tanto en el conjunto de TCGA-GBM como en el de la competición de Kaggle. Este conjunto de datos ha sido preprocesado con anterioridad como se explicó en el apartado 4. Los datos de BRATS se utilizarán en la fase de entrenamiento y validación, mientras que el conjunto de TCGA servirá para evaluar el desempeño del modelo en la generalización a datos nuevos. Además, TCGA cuenta con una secuencia extra, la perfusión, que aportará imágenes con características diferentes a las morfológicas que ayudarán a la solución a "aprender" patrones distintos a la hora de discriminar entre casos de MGMT metilado y no metilado.

El primer paso consiste en leer las columnas 'BraTS21ID' o 'TCGAID' (identificadores) y el 'MGMT_value' de los archivos con extensión CSV (Valores Separados por Comas) de entrenamiento y test respectivamente. A continuación crea una serie de listas para cada una de las modalidades donde se guardarán las rutas a cada una de las imágenes proporcionadas para cada

paciente. Esto servirá como punto de partida para generar los conjuntos de entrenamiento, validación y test.

Para empezar la fase de entrenamiento, el siguiente paso sería la carga de los datos. Para ello se utiliza el *dataloader* que carga los datos de entrada para cada paciente y cada secuencia; realiza un pequeño preprocesamiento (bordes negros y normalización); y genera el volumen 3D de los datos. Para cada secuencia se crea una lista que almacena los tensores generados para cada paciente. Tras ello, se aplica la técnica de *Stratified Split* para contar con la misma proporción de las clases en los conjuntos de datos de entrenamiento y validación, las observaciones de test no se particionan. Se genera entonces un conjunto de tensores para cada una de las secuencias disponibles en los conjuntos de entrenamiento, validación y test.

Finalmente se recoge la salida del cargador de datos en un objeto tipo *tf.data.Dataset.from_generator*. Es una función de la biblioteca *TensorFlow* que toma como entrada la salida del generador y las etiquetas asociadas a cada paciente. Tras ello, devuelve un objeto tf.data.Dataset que es la interfaz estándar para representar un flujo de datos en *Tensorflow*. Este objeto ha sido de alta utilidad ya que facilita trabajar con grandes cantidades de datos que no se pueden cargar en memoria a la vez; así como producir datos de manera dinámica.

La siguiente fase corresponde al generador de datos. Como el conjunto de datos de entrada cuenta con un tamaño reducido, se ha aplicado la técnica de aumento de datos para contar con más variabilidad en las observaciones de entrada del modelo. Algunas de las técnicas empleadas han sido rotación aleatoria, volteo horizontal, volteo vertical y recorte aleatorio, entre otras. Se aplican mediante la función creada llamada *keras_augment* explicada en el apartado 5.1.3 . El generador de datos realiza otras funciones además del *data augmentation*:

1. Mezcla los datos si se corresponden al conjunto de entrenamiento,
2. Reescalado a la resolución establecida en los parámetros de configuración
3. Generación del lote de imágenes para cargar en cada iteración según el tamaño especificado.

A continuación se crea la estructura del modelo mediante la función “build_model”, donde se selecciona la arquitectura 3D que se desee utilizar y se confecciona la capa de clasificación, para adaptarla a las características del problema. Se ha experimentado con varias arquitecturas prediseñadas cómo *EfficientNet*, *Xception* o *DenseNet* entre otras; incluso con alguna arquitectura personalizada. En apartados posteriores se analizará cuál de los modelos propuestos ha sido capaz de aprender más del conjunto de datos y generalizar con nuevos datos. Para ser capaz de adaptarse a los datos de entrada se han utilizado versiones 3D de estas arquitecturas mencionadas gracias a la librería de python *classification_models_3D* [44].

En este caso se ha empleado la técnica de transferencia de aprendizaje debido en gran medida a las reducidas observaciones disponibles. Para ello, se han utilizado las características de bajo nivel aprendidas por modelos que utilizan las mismas arquitecturas empleadas en este proyecto y obtenidos del entrenamiento en el conjunto de datos *ImageNet*. Los pesos son variables ajustables que influyen en la relación entre las entradas y las salidas de la red [45]. En otras palabras, determinan la importancia relativa de cada entrada en el cálculo de la salida de la red.

La estrategia, como se ha comentado, es utilizar como punto de partida un modelo previamente entrenado en *ImageNet* para resolver el problema que se plantea. Éste es un conjunto de datos extenso y desafiante, y las redes convolucionales entrenadas sobre él suelen aprender

características generales y útiles en el proceso [46]. Cuando se utiliza *transfer learning*, los pesos de las capas inferiores de la red pre entrenada se mantienen, ya que estas capas suelen aprender características de bajo nivel que pueden ser aplicables a una amplia variedad de tareas. Luego se pueden ajustar o reemplazar las capas superiores para adaptarse al problema específico que se desea resolver.

Surgía una complicación y es que el conjunto de datos de *ImageNet* está formado por imágenes en dos dimensiones, por lo que el conocimiento adquirido no se podía aplicar en los modelos 3D. Para solventarlo se ha utilizado la función "convert_weights", que convierte los pesos de un modelo de red neuronal convolucional 2D en un modelo de red neuronal convolucional 3D. La función itera a través de todas las capas en ambos modelos (2D y 3D), si la capa es de tipo o 'Conv2D' o 'DepthwiseConv2D', ajusta los pesos de la capa 2D y los convierte a pesos 3D. También ajusta el sesgo si es necesario. Si la capa es de tipo 'Sequential' y contiene 'convnext' en su nombre, realiza ajustes similares a los pesos y sesgos. Esta función también estaba incluida en el módulo *classification_models_3D* [44].

Tras ello se ha aplicado una función llamada BrainTumorModel3D que se encargará de realizar las iteraciones de entrenamiento y evaluación. Se ha creado con el objetivo de reducir el coste computacional debido a la gran cantidad de recursos que requieren la actualización de parámetros en modelos entrenados sobre tensores 3D. Para ello utiliza la acumulación de gradientes, que es una técnica para dividir las actualizaciones de los parámetros del modelo en varios pasos de entrenamiento, lo que puede ser útil cuando se trabaja con GPUs con memoria limitada. La función acepta un modelo base y un número de gradientes, realiza las predicciones, calcula las métricas y el valor de la función de pérdida. Acumula los gradientes y cuando se alcanza el número de gradientes especificado, se actualizan los parámetros de la capas de la red.

Como se ha explicado anteriormente, dos posibilidades se plantearon a la hora de diseñar la fase de entrenamiento: entrenar el modelo con todas las secuencias disponible a la vez o hacerlo de manera individual con cada una. Para realizar la primera experimentación era necesario implementar una capa convolucional antes de introducir los datos de entrada al modelo para adaptarlos a las requerimientos de dimensiones de la capa de entrada del modelo. Este proceso no fue necesario aplicarlo para el segundo diseño ya que los lotes cumplían con las dimensiones establecidas. En la sección 6 se debatirá cuál de los dos ha llevado a un mejor desempeño de las soluciones.

5.2 Adaptación del marco de trabajo al entorno de ejecución

El entorno para ejecutar trabajos en dichos servidores requiere de una "dockerización" del código del proyecto. El término dockerización viene de la herramienta Docker, que es una plataforma que permite automatizar el proceso de desarrollo, implementación y ejecución de aplicaciones en contenedores. Los contenedores son unidades de software independientes que incluyen todo lo necesario para ejecutar una aplicación, como el código, las bibliotecas, las dependencias y los archivos de configuración. Al encapsular las aplicaciones en contenedores, Docker facilita la portabilidad, la escalabilidad y la gestión de las aplicaciones [47].

Docker utiliza la tecnologías de contenedores, que es una forma de virtualización a nivel de sistema operativo. A diferencia de la virtualización tradicional, que emula hardware completo y

ejecuta múltiples sistemas operativos independientes en una única máquina física, los contenedores comparten el mismo núcleo del sistema operativo y se ejecutan de manera aislada en el espacio de usuario. Esto hace que los contenedores sean más ligeros y rápidos que las máquinas virtuales [47].

El primer paso para "dockerizar" la solución es la creación del *Dockerfile*. Es un archivo de texto que contiene instrucciones para construir una imagen Docker. La imagen Docker es una plantilla inmutable que contiene todos los componentes necesarios para ejecutar una aplicación o servicio en un contenedor Docker. El *Dockerfile* especifica las dependencias, el entorno, los archivos y los comandos necesarios para crear y configurar la imagen [48].

Se compone de una serie de instrucciones, cada una de ellas representa una capa en la imagen final. Algunas instrucciones comunes en los *Dockerfile* incluyen:

- 'FROM': Esta instrucción indica la imagen base desde la cual se construirá la nueva imagen.
- 'RUN': Esta instrucción ejecuta comandos en una nueva capa de la imagen y guarda los resultados.
- 'COPY': Esta instrucción copia archivos y directorios desde el sistema local al sistema de archivos de la imagen.
- 'WORKDIR': Esta instrucción establece el directorio de trabajo para las instrucciones 'RUN', 'CMD', 'ENTRYPOINT', 'COPY' o 'ADD'.
- 'CMD': Esta instrucción proporciona los argumentos por defecto para la entrada del contenedor. Pueden ser sobrescritos por los argumentos proporcionados en la línea de comandos al ejecutar el contenedor.

La imagen base sobre la que se ha construido la imagen Docker es *nvcr.io/nvidia/tensorflow:23.04-tfx-py3*, que son imágenes diseñadas especialmente para ser ejecutadas sobre sistemas DGX. Los servidores del BDS Lab son de este tipo, una familia de sistemas de hardware de alto rendimiento desarrollados por *NVIDIA* diseñados específicamente para acelerar la computación en la IA y DL. Estos sistemas vienen equipados con las últimas GPUs de *NVIDIA*, así como con hardware, software y herramientas optimizadas para facilitar y agilizar el desarrollo y la implementación de soluciones de IA [49].

El archivo que se generó en primer lugar fueron las dependencias del proyecto. En el equipo local se creó un entorno virtual donde instalar las bibliotecas necesarias para ejecutar la solución. Mediante el comando *pip freeze* de python se generó una lista de todas las bibliotecas y paquetes instalados en el entorno virtual actual, junto con sus respectivas versiones. Esta lista se suele guardar en archivo llamado "requirements.txt". Este archivo junto con los "scripts" Python de código, el *Dockerfile* y el conjunto de datos fueron cargados en la DGX mediante el comando COPY. Con la instrucción CMD se proporcionaron los argumentos para ejecutar el archivo Python encargado del entrenamiento del modelo.

A continuación, para construir la imagen se ha utilizado el comando "docker build". Por último, se construyó un archivo *Shell* (extensión .sh) que sería el encargado de ejecutar los comandos que "levantaran" el contenedor de Docker a partir de la imagen construida y pudiesen generar los volúmenes que almacenan los datos de entrada del modelo. Para más detalle, se puede consultar el anexo donde se incluye el *Dockerfile* y los demás archivos.

Finalmente ya se puede realizar la tarea de entrenamiento, donde se itera cada uno de los conjuntos de datos de las secuencias disponibles, se generan los datos de entrada con el aumento de datos y se carga el modelo con la acumulación de gradientes. Tras ello comienza el entrenamiento de las diversas soluciones propuestas.

El "batch size" es el número de ejemplos de entrenamiento que se utilizan en una actualización de los pesos del modelo. Algunas consideraciones para seleccionar el tamaño del lota incluyen:

a. Recursos disponibles: memoria GPU o CPU [50]
b. Tamaño del dataset: Los conjunto de datos pequeños funcionan mejor con "batch size" pequeños ya que las actualizaciones de los pesos serán más ruidosas (habrá más variabilidad en las actualizaciones) y ayudará a evitar el sobreajuste [50].
c. Estabilidad del entrenamiento: Tamaños de lotes más pequeños[50].

En este caso el tamaño de lote que mejor rendimiento ha proporcionado a las soluciones ha sido 8.

La tasa de aprendizaje determina qué tan rápido se actualizan los pesos del modelo durante el entrenamiento. Un "learning rate" adecuado es crucial para garantizar una convergencia rápida y estable. En este caso se ha experimentado con varias tasas distintas y la que mejor resultado ha proporcionado es 0.0001.

Los optimizadores son algoritmos que ajustan los pesos del modelo en función de los gradientes de la función de pérdida. La elección del optimizador puede afectar a la velocidad de convergencia y a la calidad de la solución final. En las soluciones presentadas el optimizadores escogido para ser utilizados en la experimentación ha sido Adam. Se ha tenido en consideración aspectos como la naturaleza del modelo (éstos actúan mejor en redes neuronales profundas [51]) o la velocidad de convergencia.

Finalmente, la función de pérdida mide la discrepancia entre las predicciones del modelo y los valores reales. La elección de la función de pérdida adecuada depende en gran medida del tipo de problema y las métricas de rendimiento que se quieran optimizar. En este caso ya que el problema que se quiere abordar es de clasificación y de naturaleza binaria, se ha seleccionado la entropía cruzada binaria (BCE).

La configuración del entrenamiento ha se puede observar en la tabla 2:

Batch_size	8
Epochs	10
Resolución de las IRM	(256,256,64)
Reescalado	Si
Optimizador	Adam
Learning rate	0.0001
Función de pérdida	BCE

Tabla 2: Configuración del proceso de entrenamiento

Cabe destacar también que se han aplicado ciertas funciones para registrar información sobre el progreso del rendimiento del modelo a lo largo del entrenamiento, guardar el modelo, condiciones de parada temprana o reducción de tasa de aprendizaje:

- **CSVLogger**: Es una clase *callback* en *Keras* que proporciona en extensión .csv un registro de la evolución del rendimiento de la solución guardando información como métricas o valor de la función de pérdida. En este caso se recogen las métricas a evaluar y el valor de la función de pérdida al final de cada época en las fases de entrenamiento y validación
- **ModelCheckpoint**: Se utiliza para guardar el modelo o sus pesos en un archivo durante el entrenamiento en intervalos o cuando se logra un cierto rendimiento. En este caso minimizar la función de pérdida en la etapa de validación.
- **Early Stopping**: Se usa para detener el entrenamiento del modelo si no se observa una mejora en la métrica de interés durante un número determinado de épocas. En este caso se ha optado por parar el entrenamiento si la pérdida de validación no mejora en 3 épocas consecutivas.
- **ReduceLROnPlateau**: Esta clase *callback* de *Keras* reduce la tasa de aprendizaje del optimizador si este no mejora durante un número determinado de épocas. Esto puede ser útil para mejorar la convergencia del modelo y evitar estancamientos en el entrenamiento.

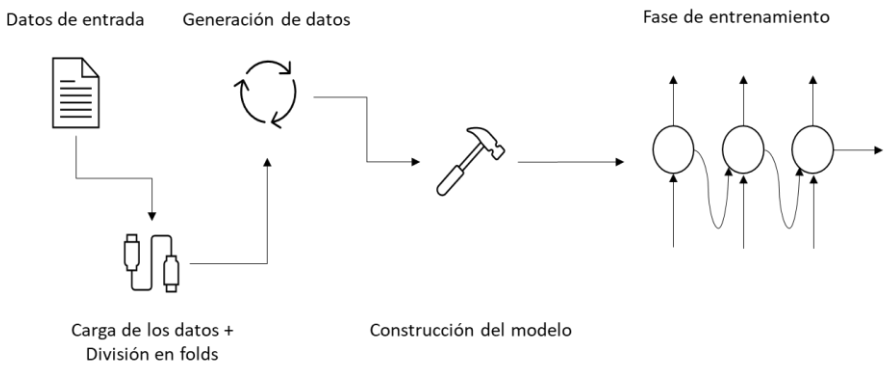

Figura 11: Flujo del proceso de entrenamiento.

5.4 Evaluación del Modelo

Durante el entrenamiento, el objetivo principal es ajustar los parámetros del modelo para minimizar el error en los datos de entrenamiento. Sin embargo esto puede llevar a un modelo que se ajusta demasiado a estos datos, lo que se conoce como sobreajuste. Un modelo sobre ajustado puede tener un rendimiento deficiente en datos nuevos y desconocidos.

Para abordar este problema y evaluar la capacidad del modelo para generalizar a nuevos datos, se utiliza un conjunto de datos de validación. Este conjunto está separado de los datos de entrenamiento y no se utiliza durante el ajuste de los parámetros del modelo. En lugar de eso, se utiliza para evaluar periódicamente el rendimiento durante la fase de *training*.

Como se ha comentado anteriormente, dos posibles experimentaciones fueron planteadas a la hora de diseñar el proceso de entrenamiento:

a) Utilizar como datos de entrada todas las secuencias a las vez, crear una capa convolucional para adaptar los datos de entrada a las dimensiones requeridas por el modelo y entrenar un único modelo.
b) Utilizar como datos de entrada una sola secuencia y entrenar un modelo por cada conjunto de datos de entrada.

Tras comprobar el desempeño de las soluciones en ambas experimentaciones se observó que la segunda variante era la óptima ya que las soluciones presentaban menor sobreajuste y mayor desempeño a la hora de discriminar las imágenes y clasificarlas en la clase objetivo.

En la tabla 3 se presenta la evolución del rendimiento promedio de los modelos para cada una de las secuencias, según la arquitectura utilizada, en la etapa de validación al terminar el entrenamiento con 10 épocas. Para comparar la capacidad de los modelos para aprender de las características de los datos de entrenamiento, se deberá examinar las métricas utilizadas para evaluar la eficiencia de los modelos. En este caso se utiliza la precisión y el AUC como métricas para comparar el desempeño entre las diferentes arquitecturas sobre los conjuntos de datos de secuencias de RM.

FLAIR		
Arquitectura	% Precisión	AUC
EfficientNetB3	56.0894	0.5955
EfficientNetB1	57.6748	0.5817
Xception	52.8528	0.5507
Densenet121	54.8648	0.5992

T1wCE		
Arquitectura	% Precisión	AUC
EfficientNetB3	54.7999	0.57846
EfficientNetB1	58.5414	0.6064
Xception	55.5924	0.4756
Densenet121	40.5684	0.4347

T1w		
Arquitectura	% Precisión	AUC
EfficientNetB3	47.7925	0.5651
EfficientNetB1	49.8714	0.4917
Xception	51.5852	0.5062
Densenet121	57.1658	0.5636

T2w		
Arquitectura	% Precisión	AUC
EfficientNetB3	56.0894	0.5955
EfficientNetB1	57.9615	0.6104
Xception	52.8528	0.5507
Densenet121	53.7307	0.5914

Tabla 3: Rendimiento promedio de los modelos candidatos a posible solución final en la etapa de validación.

Si se analizan los resultados, los modelos basados en *EfficientNetB1* y *DenseNet121* parecen ser los más efectivos en términos de precisión y AUC. respectivamente, para la mayoría de secuencias de RM. *EfficientNetB1* muestra la mejor precisión en 3 de las 4 secuencias (FLAIR, T1wCE y T2w) y también tiene el mejor AUC en T1wCE y T2w. *Xception* generalmente tiene una productividad inferior en comparación con las otras arquitecturas. *DenseNet121* muestra un rendimiento sólido en las secuencias FLAIR y T1w en términos de AUC, lo que indica que también puede ser una arquitectura viable en ciertos casos.

Esto podría sugerir que *EfficientNetB1* debería ser considerado como el modelo principal para este problema. Sin embargo antes de tomar la decisión final debería de tenerse en cuenta aspectos como la evolución de la función de pérdida y evaluar cómo las diferencias de AUC y precisión afectan a la aplicación específica del modelo en un conjunto independiente, para comprobar la robustez y capacidad de generalización del modelo en datos nuevos.

Otro aspecto a tener en cuenta para medir el desempeño de las soluciones es comprobar la evolución del valor de la función de pérdida tras cada iteración para observar problemas potenciales como el sobreajuste, el subajuste o la convergencia lenta. Como se puede observar en la Figura 12, la tendencia en la evolución para cada una de las soluciones es realmente similar, la pérdida de entrenamiento disminuye hasta la tercera época, a partir de este punto se estanca y converge en un intervalo entre 0,65 y 0,70. Sin embargo en la evolución de la validación se estanca prácticamente desde la primera época, lo que podría interpretarse como un sobreajuste del modelo. Aunque se han aplicado técnicas de regularización como la penalización de pesos, las soluciones no pueden evitar el sobreajuste de los datos. En las conclusiones se reflexionará en detalle sobre este aspecto.

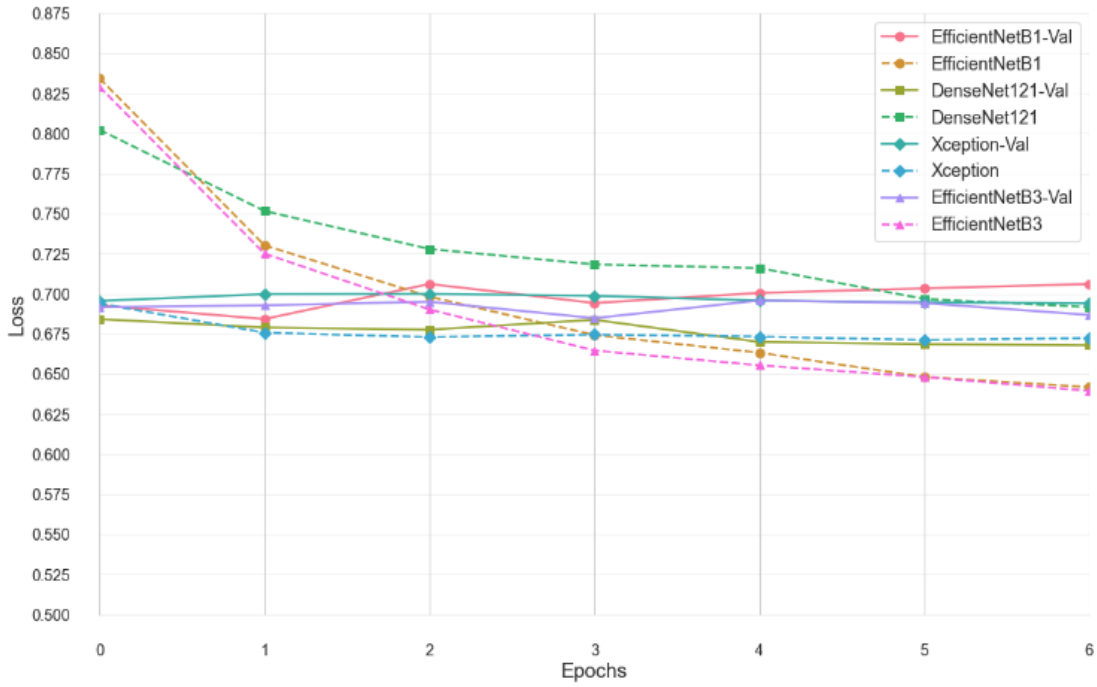

Figura 12. Evolución de la función de pérdida Entrenamiento/Validación para todas las soluciones posibles para la modalidad FLAIR.

El caso anterior se analizaba el caso del conjunto de FLAIR, pero las conclusiones se pueden extender al entrenamiento del resto de modalidades, como se puede ver en la figura 13:

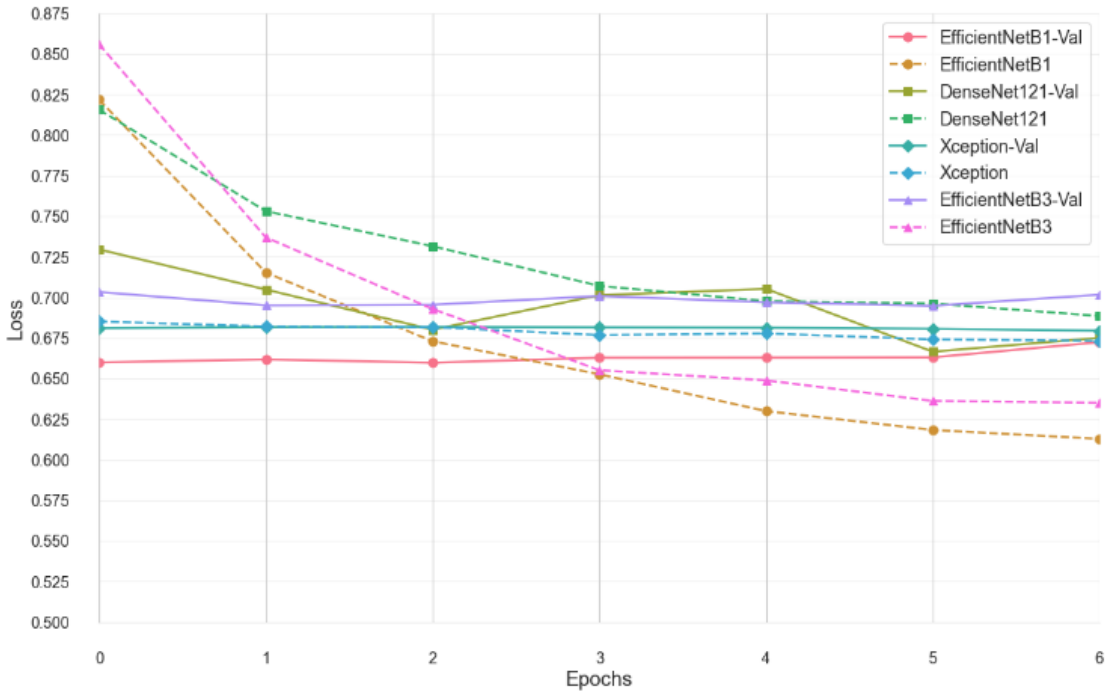

Figura 13. Evolución de la función de pérdida Entrenamiento/Validación para todas las soluciones posibles para la modalidad T2w.

6. Validación y despliegue

Finalmente para escoger cual es la solución que mejor se adapta a los requisitos del problema, se ha utilizado el conocimiento adquirido por cada modelo para cada modalidad y se ha evaluado su desempeño sobre el conjunto de test de TCGA-GBM. Esta fase de test es crucial en el proceso de desarrollo y evaluación de los modelos debido a que en ella se mide la capacidad de generalización en nuevos datos, lo cual es esencial para determinar qué tan eficiente serán las soluciones en situaciones del mundo real.

Debido a que la base de datos de entrenamiento consistente en la proporcionada en la competición de *Kaggle RSNA-MICCAI Brain Tumor Radiogenomic Classification* era multicéntrica, existía la posibilidad de que algunas observaciones de este conjunto formaran también parte de la base de datos de test obtenida de TCGA y TCIA. Por lo tanto era necesario asegurarse que no existía superposición entre ambos *datasets*. Para comprobar la independencia total de los datos de test respecto a los de entrenamiento, se ha medida la similitud de cada una de las imágenes de cada secuencia de los datos de prueba con sus homólogas de entrenamiento y validación. Para ello se ha hecho uso de dos métricas:

1. Correlación cruzada: Es una medida estadística que compara las intensidades de los píxeles en posiciones correspondientes en las dos imágenes. Cuando se aplica a dos imágenes, se calcula la correlación cruzada desplazando una imagen con respecto a la

otra y calculando la suma de los productos de las intensidades de los píxeles en las posiciones correspondientes. El rango de valores es entre -1 y 1, siendo 1 la similitud total entre las dos imágenes [52].

2. Información mutua: La información mutua de dos imágenes X e Y se puede definir como:

$$MI(X, Y) = H(X) + H(Y) - H(X, Y)$$

donde:

- H(X) y H(Y) son las entropías de X e Y, respectivamente que miden la incertidumbre o la cantidad de información en cada imagen individualmente.
- H(X,Y) es la entropía conjunta de X e Y, que mide la incertidumbre o la cantidad de información en las dos imágenes juntas.

La información mutua tiene un rango de 0 a 1, y es cero si y sólo si las dos variables son independientes [53].

Se ha realizado un función para calcular esta métricas en entre cada una de las imágenes de test y entrenamiento de cada una de las secuencias. Esta implementación genera un reporte en el que se anotan aquellas imágenes de test que tienen una alta correlación e información mutua con alguna imagen de entrenamiento (valor en ambas métricas mayor a 0.8). El análisis ha determinado que las imágenes son totalmente independientes ya que no se ha reportado ningún caso de correlación cruzada e información mutua elevada. La función se puede encontrar en el Anexo II.

Los datos mostrados en la tabla 4 muestran el valor de la pérdida, como la precisión y el AUC para el conjunto de datos de prueba:

Arquitectura	Valor de Pérdida	%Precisión	AUC
DenseNet121	0.5848	91.92%	0.5208
Xception	0.5797	99.91%	0.6757
EfficientNetB3	0.6579	56,27%	0.6588
EfficientNetB1	0.5686	72.42%	0.7857

Tabla 4: Rendimiento promedio de las soluciones propuestas para todas las secuencias en los datos test TCGA-GBM.

Para evaluar y comparar el rendimiento de los diferentes modelos, es útil considerar todos estos valores en conjunto:

1. *DenseNet121*: La precisión del modelo es del 91,92% pero tiene un AUC de 0.5208, lo que es un valor bastante bajo y cercano a 0.5, que es el valor que se obtendría de un clasificador aleatorio. Esto sugiere que, aunque el modelo pueda clasificar correctamente un alto porcentaje de muestras, su capacidad de distinguir entre clases positivas y negativas es limitada.
2. *Xception*: Este modelo tiene una precisión muy alta (99,91%) y un AUC razonablemente bueno (0.6757). Aunque el AUC no es aceptable, es significativamente mejor que el del modelo anterior. La pérdida también es menor que *DenseNet121*.

3. *EfficientNetB3*: Tiene una menor precisión que el resto de soluciones (56,27%) y una pérdida bastante alta (0.6579). Sin embargo su AUC es más alto que *DenseNet121*, pero aun así menor que *Xception*. Por lo tanto este modelo no parece ser el más óptimo.
4. *EfficientNetB1*: La precisión de esta última propuesta es del 72,42% y su AUC es el más alto entre todos los modelos (0.7857). Aunque su precisión es menor que la de *Xception* y *DenseNet121*, su mayor AUC sugiere que es mejor para distinguir el estado de metilación del gen MGMT. Además, tiene la menor pérdida de todos los modelos comparados.

En conclusión, si se buscara un modelo con más precisión *Xception* sería la opción escogida. Sin embargo, el AUC es una métrica más importante para medir el rendimiento sobre el problema abordado. La comparación entre estas métricas puede generar controversia a la hora de escoger cuál es la mejor solución. La precisión mide la proporción de verdaderos positivos con respecto al total de casos, mientras que el AUC es una medida de la capacidad del modelo para distinguir entre clases positivas y negativas en general. Un modelo puede tener una alta precisión pero un AUC bajo si es muy conservador a la hora de hacer predicciones positivas, lo que resulta en pocos falsos positivos pero también muchos falsos negativos [54].

Por lo tanto, y teniendo en cuenta todo lo anterior, la solución propuesta que mejor se adapta a la naturaleza del problema sería la basada en una arquitectura *EfficientNetB1*, utilizando transferencia de aprendizaje del conjunto de datos *ImageNet* y confeccionando la capa de clasificación.

7. Conclusiones

En este apartado se intentarán transmitir las reflexiones finales sobre el proyecto, su relación con los estudios cursados así como los principales inconvenientes que han aparecido durante su realización y cómo se han abordado.

7.1 Conclusiones del trabajo realizado

El modelo seleccionado es relativamente capaz de detectar el estado de metilación en distintas secuencias de IRM (FLAIR, T2w, T1w, T1wCE). Todo ello mediante redes neuronales convolucionales, técnicas de aumento de datos y transferencia de aprendizaje. Tal y como se ha comprobado, el conjunto de prueba era totalmente independiente de los datos de entrenamiento y validación, aspecto que refuerza la reflexión sobre los resultados obtenidos.

De cara a una posible futura aplicabilidad clínica, se debería de desarrollar un modelo más robusto entrenado con un conjunto de datos de mayor tamaño. Debido a esta limitación no se han podido realizar aspectos como la optimización de hiper parámetros, clave a la hora de mejorar el rendimiento del modelo. El sobreajuste presente en todas las soluciones posibles es otro de los

riesgos causado en mayor medida por el tamaño del conjunto de datos. Este hecho puede hacer que no se capture adecuadamente la variabilidad real presente en la población objetivo. Esto puede hacer que el modelo aprenda patrones que son específicos del conjunto de datos de entrenamiento, pero que no sea generalizable a otro. Además, el conjunto de datos puede contener ruido y fluctuaciones aleatorias que no son representativas de la población general. El modelo puede capturar este ruido en lugares de características subyacentes relevantes. Por lo tanto, un punto a mejorar sería realizar estudios de validación externa para determinar la generalización del modelo en diferentes poblaciones y entornos clínicos.

7.2 Legado

Este Trabajo de Fin de Grado contribuye al campo de la ciencia de datos y la medicina, con un enfoque particular en la detección del estado de metilación a través de secuencias de IRM morfológica mediante el uso de redes neuronales convolucionales. Los beneficiarios de este proyecto, principalmente profesionales en ciencias de la salud y ciencia de datos, que podrían aprovechar la investigación y los resultados presentados para desarrollar modelos más robustos y efectivos en la práctica clínica.

El TFG proporciona acceso a datos y código a través de repositorios abiertos, permitiendo a terceros reproducir el análisis y obtener resultados similares, siempre que se pueda acceder a los datos y repetir los procedimientos. Respecto a los elementos persistentes, el código y la documentación complementaria, incluyendo el diseño del modelo y las técnicas de preprocesamiento y aumento de datos, formarán una base sólida para futuros trabajos de investigación. A corto plazo, estos recursos permitirán la optimización de hiper parámetros y la validación externa del modelo. A medio y largo plazo, podrían guiar el desarrollo de soluciones más robustas y generalizables para la detección del estado de metilación del gen MGMT.

Personalmente este proyecto me ha servido para consolidar mis conocimientos en ciertas áreas de la ciencia de datos, pero sobre todo me quedo con los aspectos aprendidos. El DL es un campo en el que no profundicé tanto en mis estudios y ha supuesto un reto constante aprender y adaptar mis ideas iniciales a la naturaleza y demandas del proyecto. La cantidad de estrategias empleadas para enfrentarse a la cantidad tan reducida de datos ha sido un auténtico desafío ya que el código del proyecto cambiaba día a día a medida que investigaba y descubría nuevas técnicas que emplear y evaluar. El aumento de datos, la transferencia de aprendizaje o las redes convolucionales 3D eran métodos totalmente desconocidos para mí, y entender el proceso, realizar el diseño y conseguir procesar el resultado final ha sido de los más fructífero.

Aunque me he apoyado en foros, notebooks de *Kaggle*, repositorios de *GitHub*, etc. He sido totalmente autodidacta en ciertas partes como en el preprocesamiento, el aumento de datos o la construcción del modelo ya que muchas de estas técnicas no las había explorado durante el grado. Me ha impresionado la cantidad infinita de aplicaciones que tiene la ciencia de datos para el análisis de imágenes en el ámbito médico, ya que los proyectos consultados similares a este TFG y sus aplicaciones en la vida real son realmente interesantes y útiles.

En cuanto al legado para la sociedad, este TFG contribuye en la investigación de aplicación de técnicas de DL en problemas de imagen médica, con el potencial de mejorar la detección y el tratamiento de pacientes con GBM en el futuro.

7.3 Relación del trabajo desarrollado con los estudios cursados

La realización de este trabajo de fin de grado ha sido una oportunidad para aplicar los conocimientos adquiridos en el grado en ciencia de datos a un problema de ámbito médico. Durante mis estudios, he aprendido técnicas para procesar grandes conjuntos de datos, realizar análisis estadísticos y desarrollar modelos predictivos. Estos conocimientos me han permitido entender y abordar el problema de clasificación del estado de metilación del gen MGMT usando IRM morfológica y funcional de pacientes con glioblastoma.

El trabajar con imágenes médicas ha sido un auténtico reto: cambios de formato, preprocesamiento, generación de volúmenes 3D, anonimización, etc. Diseñar el proceso a seguir para preparar los datos de entrada al modelo utilizando herramientas que nunca había utilizado como *data loaders*, generadores de datos o técnicas de aumento de datos; ha sido todo un descubrimiento y enriquecimiento de mis conocimientos. En cuanto al modelo de clasificación, utilizar CNN con transferencia de aprendizaje y adaptar la red a los volúmenes 3D de entrada también ha sido un proceso gratificante ya que nunca me había enfrentado a una situación similar. El proceso de validación y evaluación también ha servido para afianzar mis habilidades a la hora de desarrollar un ciclo completo de extremo a extremo, desde la entrada de datos hasta la salida de resultados.

En resumen, el trabajo que he desarrollado en este proyecto está estrechamente relacionado con mi formación. He aplicado técnicas de procesamiento de datos, análisis estadístico y modelado predictivo para resolver un problema concreto en el ámbito médico. El uso de las redes convolucionales y la validación de modelos han sido particularmente importantes en este proyecto, y he aplicado los conocimientos que he adquirido durante mis estudios para desarrollar soluciones efectivas.

En relación con las habilidades transversales, este TFG ha involucrado entre muchas otras: el análisis y solución de problemas, desglosando el proyecto desde el principio y examinándolo en detalle; innovación creatividad y emprendimiento, así como conocimiento de cuestiones contemporáneas, ya que se aborda un problemas actual y se aplican técnicas de ciencia de datos en este campo; y aprendizaje continuo, ya que las técnicas de ciencia de datos evolucionan constantemente y siempre se adquieren nuevos conocimientos, como ha sucedido en este proyecto.

8. Trabajos futuros

Uno de los objetivos iniciales era comprobar la eficiencia de la solución sobre IRM de perfusión DSC. Debido a la limitación temporal para la realización del TFG no ha sido posible verificar de manera convincente el rendimiento de la solución planteada en observaciones de perfusión. Por ende, la evaluación integral del desempeño de la solución planteada ha quedado fuera del alcance de este proyecto. No obstante, se ha dejado preparado el pipeline de la solución y se espera que futuras investigaciones puedan abordar esta tarea pendiente con el tiempo y los recursos necesarios.

Por otro lado en el estado actual de la solución se necesitaría contar con un conjunto de datos más extenso y diverso para que el modelo pudiese mejorar la precisión de clasificación y generalizar en nuevos datos. Si no hay bases de datos similares disponibles, se podría intentar integrar otras modalidades de imagen médica a parte de MRI como tomografía por emisión de positrones (PET) o tomografía computarizada (CT).

Por otro lado, se podrían tener en cuenta otros tipos de enfoques para abordar el problema. Una opción podrían ser los modelos híbridos donde se combinan diferentes técnicas, como redes neuronales y modelos basados en reglas, para mejorar el rendimiento de la solución. Esto podría implicar la utilización de un modelo basado en reglas para la selección inicial de características, seguido de la utilización de una CNN para la clasificación final. Otra opción serían los modelos *Siamese*, estos utilizan la comparación de pares de imágenes para detectar si pertenecen al mismo estado de metilación o no. Esto podría ser útil para abordar el problema de la falta de datos y mejorar la capacidad de generalización del modelo.

Como continuación a este proyecto otra de las líneas a seguir, si se contara con una solución clínicamente efectiva, sería desarrollar una interfaz de usuario y crear una aplicación clínica donde se integre. Para que la solución sea accesible y fácil de usar por el personal médico o de investigación, se elaboraría una interfaz de usuario que permita a los usuarios cargar los datos, ejecutar el modelo y visualizar los resultados. Como se ha dicho antes, si la solución propuesta fuese confiable y funcional se podría considerar el desarrollo de una aplicación clínica para la clasificación automatizada del estado de metilación del gen MGMT. Esto podría tener un gran impacto en la atención médica personalizada y en la toma de decisiones.

9. Referencias

1 - Universitat de València. (2021, Diciembre 14). Desarrollan un nuevo sistema de ayuda al diagnóstico del cáncer de mama basado en inteligencia artificial. UV Noticias. https://www.uv.es/uvweb/uv-noticias/es/noticias/desarrollan-nuevo-sistema-ayuda-al-diagnostico-del-cancer-mama-basado-inteligencia-artificial-1285973304159/Novetat.html?id=1286030737530&plantilla=UV_Noticies/Page/TPGDetaillNews

2 - El Mundo. (s.f.). La inteligencia artificial que cura. Recuperado de https://lab.elmundo.es/inteligencia-artificial/salud.html

3 - MICCAI BRATS - The Multimodal Brain Tumor Segmentation Challenge. http://braintumorsegmentation.org/.

4 - Yogananda CGB, Shah BR, Nalawade SS, Murugesan GK, Yu FF, Pinho MC, Wagner BC, Mickey B, Patel TR, Fei B, Madhuranthakam AJ, Maldjian JA. MRI-Based Deep-Learning Method for Determining Glioma MGMT Promoter Methylation Status. AJNR Am J Neuroradiol. 2021 May;42(5):845-852. doi: 10.3174/ajnr.A7029. Epub 2021 Mar 4. Erratum in: AJNR Am J Neuroradiol. 2023 Jan;44(1):E1. PMID: 33664111; PMCID: PMC8115363.

5 - Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., & He, Q. (2021). A Comprehensive Survey on Transfer Learning. Proceedings of the IEEE, 109(1), 43-76. https://doi.org/10.1109/JPROC.2020.3004555

6 - Dong, X., Yu, Z., Cao, W. et al. A survey on ensemble learning. Front. Comput. Sci. 14, 241–258 (2020). https://doi.org/10.1007/s11704-019-8208-z

7 - Wang, H., & Perez, L. (2017). The effectiveness of data augmentation in image classification using deep learning. Convolutional Neural Networks Vis. Recognit. https://arxiv.org/abs/1712.04621

8 - Alexander, B. M., & Cloughesy, T. F. (2017). Adult Glioblastoma. Journal of Clinical Oncology, 35(21), 2402-2409. https://ascopubs.org/doi/pdf/10.1200/JCO.2017.73.0119?role=tab

9 - Ostrom QT, Gittleman H, Liao P, et al (2017) CBTRUS Statistical Report: Primary brain and other central nervous system tumors diagnosed in the United States in 2010-2014. Neuro-Oncol 19:v1–v88. https://doi.org/10.1093/neuonc/nox158

10 - Wen PY, Kesari S (2008) Malignant gliomas in adults. N Engl J Med 359:492–507. https://doi.org/10.1056/NEJMra0708126

11 - Brodbelt, A., Greenberg, D., Winters, T., Williams, M., Vernon, S., & Collins, V. P. (2015). Glioblastoma in England: 2007–2011. European Journal of Cancer, 51(4), 533-542. https://doi.org/10.1016/j.ejca.2014.12.014

12 - 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

13 - Díaz Ojeda, J., Reyes Tápanes, M., Rodríguez Sánchez, L., & Sierra Benítez, E. (2020). Medios diagnósticos y tratamientos actuales del glioblastoma multiforme. Progaleno, 3(2), 87-102. Recuperado de https://revprogaleno.sld.cu/index.php/progaleno/article/view/165

14 - Parsa, A. T., Wachhorst, S., Lamborn, K. R., Prados, M. D., McDermott, M. W., Berger, M. S., & Chang, S. M. (2005). Prognostic significance of intracranial dissemination of glioblastoma multiforme in adults. Journal of neurosurgery, 102(4), 622–628. https://doi.org/10.3171/jns.2005.102.4.0622

15 - Yung, W. K. A., Albright, R. E., Olson, J., Fredericks, R., Fink, K., Prados, M. D., . . . Chang, S. M. (2000). Temozolomide and treatment of malignant glioma. Clinical Cancer Research, 6(7), 2585-2597. Recuperado de https://aacrjournals.org/clincancerres/article/6/7/2585/288264/Temozolomide-and-Treatment-of-Malignant-Glioma1

16 - Singh N, Miner A, Hennis L, Mittal S. Mechanisms of temozolomide resistance in glioblastoma - a comprehensive review. Cancer Drug Resist 2021;4:17-43. http://dx.doi.org/10.20517/cdr.2020.79

17 - Stupp R, Hegi ME, Mason WP, et al (2009) Effects of radiotherapy with concomitant and adjuvant temozolomide versus radiotherapy alone on survival in glioblastoma in a randomised phase III study: 5-year analysis of the EORTC-NCIC trial. Lancet Oncol 10:459–466. https://doi.org/10.1016/S1470-2045(09)70025-7

18 - Russell SM, Elliott R, Forshaw D, et al (2009) Glioma vascularity correlates with reduced patient survival and increased malignancy. Surg Neurol 72:242–246; discussion 246-247. https://doi.org/10.1016/j.surneu.2008.11.01

19 - Batchelor TT, Gerstner ER, Emblem KE, et al (2013) Improved tumor oxygenation and survival in glioblastoma patients who show increased blood perfusion after cediranib and chemoradiation. Proc Natl Acad Sci 110:19059–19064.https://doi.org/10.1073/pnas.1318022110

20 - Ulyte A, Katsaros VK, Liouta E, et al (2016) Prognostic value of preoperative dynamic contrast-enhanced MRI perfusion parameters for high-grade glioma patients. Neuroradiology 58:1197–1208. https://doi.org/10.1007/s00234-016-1741-7

21 - Estival González, A. (2019). Análisis de la metilación del promotor del gen MGMT, en muestras de tejido y sangre de pacientes con glioblastoma, mediante pirosecuenciación (Tesis doctoral). Balañá Quintero, M. C. (Dir.); Sanz Monte, C. (Dir.); Feliu Frasnedo, E. [Identificador ORCID dir.]. Universitat Autònoma de Barcelona. Departament de Medicina. https://ddd.uab.cat/record/215448

22 - Armony, Jorge L., Trejo-Martínez, David, & Hernández, Dailett. (2012). Resonancia Magnética Funcional (RMf): principios y aplicaciones en Neuropsicología y Neurociencias Cognitivas. Neuropsicologia Latinoamericana, 4(2), 36-50. https://dx.doi.org/10.5579/rnl.2012.010

23 - Ley-Zaporozhan, J., Ley, S. & Kauczor, HU. Morphological and functional imaging in COPD with CT and MRI: present and future. Eur Radiol 18, 510–521 (2008). https://doi.org/10.1007/s00330-007-0772-1

24 - S. Albawi, T. A. Mohammed and S. Al-Zawi, "Understanding of a convolutional neural network," 2017 International Conference on Engineering and Technology (ICET), Antalya, Turkey, 2017, pp. 1-6, doi: 10.1109/ICEngTechnol.2017.8308186.

25 - Z. Li, F. Liu, W. Yang, S. Peng and J. Zhou, "A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects," in IEEE Transactions on Neural Networks and Learning Systems, vol. 33, no. 12, pp. 6999-7019, Dec. 2022, doi: 10.1109/TNNLS.2021.3084827.

26 - The History of Deep Learning. (2019). Tryolabs. Recuperado el 2 de mayo de 2023, de https://tryolabs.com/blog/2019/02/07/history-of-deep-learning/

27 - Ardila, D., Kiraly, A. P., Bharadwaj, S., Choi, B., Reicher, J. J., Peng, L., Tse, D., Etemadi, M., Ye, W., Corrado, G. S., Naidich, D. P., & Shetty, S. (2019). End-to-end lung cancer screening with three-dimensional deep learning on low-dose chest computed tomography. Nature Medicine, 25(6), 954-961. doi: 10.1038/s41591-019-0447-x.

28 - Gulshan, V., Peng, L., Coram, M., Stumpe, M. C., Wu, D., Narayanaswamy, A., … & Webster, D. R. (2016). Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. Jama, 316(22), 2402-2410. doi: 10.1001/jama.2016.17216.

29 - 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. doi: 10.1038/nature21056.

30 - Ding, X., Suk, H. I., & Shen, D. (2018). Deep learning for disease progression prediction in Alzheimer's disease: A critical review. IEEE/ACM Transactions on Computational Biology and Bioinformatics, 15(3), 817-833. doi: 10.1109/TCBB.2017.2769618.

31 - Lundervold, A. S., & Lundervold, A. (2019). An overview of deep learning in medical imaging focusing on MRI. Zeitschrift für Medizinische Physik, 29(2), 102-127. https://doi.org/10.1016/j.zemedi.2018.11.002

32 - Journal of Nuclear Medicine Technology December 2019, 47 (4) 269-272; DOI: https://doi.org/10.2967/jnmt.119.227819

33 - Aryanto, K.Y.E., Oudkerk, M. & van Ooijen, P.M.A. Free DICOM de-identification tools in clinical research: functioning and safety of patient privacy. Eur Radiol 25, 3685–3695 (2015). https://doi.org/10.1007/s00330-015-3794-0

34 - Unión Europea. (2016). Reglamento (UE) 2016/679 del Parlamento Europeo y del Consejo de 27 de abril de 2016 relativo a la protección de las personas físicas en lo que respecta al tratamiento de datos personales y a la libre circulación de estos datos (Reglamento General de Protección de Datos). https://eur-lex.europa.eu/eli/reg/2016/679/oj

35 - Congreso de los Estados Unidos. (1996). Health Insurance Portability and Accountability Act of 1996, Pub. L. No. 104-191, 110 Stat. 1936.

36 - Facultad de Psicología, Universidad de la República. (s. f.). ¿Qué es el software libre? Recuperado el 11 de mayo de 2023, de https://psico.edu.uy/gestion/informatica/software-libre/que-es-el-software-libre

37 - Scarpace, L., Mikkelsen, T., Cha, S., Rao, S., Tekchandani, S., Gutman, D., Saltz, J. H., Erickson, B. J., Pedano, N., Flanders, A. E., Barnholtz-Sloan, J., Ostrom, Q., Barboriak, D., & Pierce, L. J. (2016). The Cancer Genome Atlas Glioblastoma Multiforme Collection (TCGA-GBM) (Version 4) [Data set]. The Cancer Imaging Archive. https://doi.org/10.7937/K9/TCIA.2016.RNYFUYE9

38 - Han, Y., Yan, L. F., Wang, X. B., Sun, Y. Z., Zhang, X., Liu, Z. C., Nan, H. Y., Hu, Y. C., Yang, Y., Zhang, J., Yu, Y., Sun, Q., Tian, Q., Hu, B., Xiao, G., Wang, W., & Cui, G. B. (2018). Structural and advanced imaging in predicting MGMT promoter methylation of primary glioblastoma: a region of interest based analysis. BMC cancer, 18(1), 215. https://doi.org/10.1186/s12885-018-4114-2

39 - Carré, A., Battistella, E., Niyoteka, S. et al. AutoComBat: a generic method for harmonizing MRI-based radiomic features. Sci Rep 12, 12762 (2022). https://doi.org/10.1038/s41598-022-16609-1

40 - Otsu, N. (1979). A threshold selection method from gray-level histograms. IEEE Transactions on Systems, Man, and Cybernetics, 9(1), 62-66. [DOI: 10.1109/TSMC.1979.4310076]

41 - Nyúl, L. G., Udupa, J. K., & Zhang, X. (2000). New variants of a method of MRI scale standardization. IEEE Transactions on Medical Imaging, 19(2), 143-150.: https://ieeexplore.ieee.org/document/836373

42 - Farias, F., Ludermir, T., & Bastos-Filho, C. (n.d.). Similarity Based Stratified Splitting: an approach to train better classifiers. arXiv. https://doi.org/10.48550/arXiv.2010.06099

43 - Nalepa J, Marcinkiewicz M and Kawulok M (2019) Data Augmentation for Brain-Tumor Segmentation: A Review. Front. Comput. Neurosci. 13:83. doi: 10.3389/fncom.2019.00083

44 - Solovyev, R., Kalinin, A. A., & Gabruseva, T. (2022). 3D convolutional neural networks for stalled brain capillary detection. Computers in Biology and Medicine, 141, 105089. https://doi.org/10.1016/j.compbiomed.2021.105089

45 - Han, S., Pool, J., Tran, J., & Dally, W. (2015). Learning both weights and connections for efficient neural networks. En C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, & R. Garnett (Eds.), Advances in Neural Information Processing Systems (Vol. 28). Curran Associates, Inc. Recuperado de https://proceedings.neurips.cc/paper_files/paper/2015/file/ae0eb3eed39d2bcef4622b2499a05fe6-Paper.pdf

46 - Morid, M. A., Borjali, A., & Del Fiol, G. (2021). A scoping review of transfer learning research on medical image analysis using ImageNet. Computers in Biology and Medicine, 128, 104115. https://doi.org/10.1016/j.compbiomed.2020.104115

https://openaccess.thecvf.com/content_CVPR_2019/papers/Zou_A_Sufficient_Condition_for_Convergences_of_Adam_and_RMSProp_CVPR_2019_paper.pdf

47 - Rad, B. B., Bhatti, H. J., & Ahmadi, M. (2017). An Introduction to Docker and Analysis of its Performance. Asia Pacific University of Technology and Innovation. https://www.researchgate.net/profile/Harrison-Bhatti/publication/318816158_An_Introduction_to_Docker_and_Analysis_of_its_Performance/links/61facc0c007fb504472fd6c7/An-Introduction-to-Docker-and-Analysis-of-its-Performance.pdf

48 - Moses Openja, Forough Majidi, Foutse Khomh, Bhagya Chembakottu, and Heng Li. 2022. Studying the Practices of Deploying Machine Learning Projects on Docker. In Proceedings of the International Conference on Evaluation and Assessment in Software Engineering 2022 (EASE '22). Association for Computing Machinery, New York, NY, USA, 190–200. https://doi.org/10.1145/3530019.3530039

49 - Y. Ren, S. Yoo and A. Hoisie, "Performance Analysis of Deep Learning Workloads on Leading-edge Systems," 2019 IEEE/ACM Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS), Denver, CO, USA, 2019, pp. 103-113, doi: 10.1109/PMBS49563.2019.00017.

50 - Masters, D., & Luschi, C. (2018). Revisiting Small Batch Training for Deep Neural Networks. https://arxiv.org/abs/1804.07612

51 - Fangyu Zou, Li Shen, Zequn Jie, Weizhong Zhang, Wei Liu; Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 11127-11135.

52 - F. Maes, D. Vandermeulen and P. Suetens, "Medical image registration using mutual information," in Proceedings of the IEEE, vol. 91, no. 10, pp. 1699-1722, Oct. 2003, doi: 10.1109/JPROC.2003.817864.

53 - J. N. Sarvaiya, S. Patnaik and S. Bombaywala, "Image Registration by Template Matching Using Normalized Cross-Correlation," 2009 International Conference on Advances in Computing, Control, and Telecommunication Technologies, Bangalore, India, 2009, pp. 819-822, doi: 10.1109/ACT.2009.207.

54 - Ling, C.X., Huang, J., Zhang, H. (2003). AUC: A Better Measure than Accuracy in Comparing Learning Algorithms. In: Xiang, Y., Chaib-draa, B. (eds) Advances in Artificial Intelligence. Canadian AI 2003. Lecture Notes in Computer Science, vol 2671. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-44886-1_25

54 - Cajamarca-Barón, J. H. (2014). El cáncer y su impacto en salud pública. MedUNAB, 17(1), 41–45. https://doi.org/10.29375/01237047.1965

55 - Ordoñez Sánchez, J. L., Palacios Albarracín, I. de los Ángeles, Calderón Vallejo, C. E., & Navas Román, J. I. (2019). Las tecnologías sanitarias: Su importancia y evaluación. *RECIAMUC*, *2*(3), 659-680. https://doi.org/10.26820/reciamuc/2.(3).septiembre.2018.659-680

56 - Fernández Pinte, Manuela. (2022). ¿Ciencia abierta para intereses privados? la lógica de la ciencia abierta y la comercialización de la investigación. *Revista de Economía Institucional*, *24*(47), 179-201. Epub January 20, 2023. https://doi.org/10.18601/01245996.v24n47.08

10. Anexos

ANEXO I: Objetivos de Desarrollo Sostenible

Objetivos de Desarrollo Sostenibles	Alto	Medio	Bajo	No Procede
ODS 1. El fin de la pobreza.				x
ODS 2. Hambre cero.				x
ODS 3. Salud y bienestar.	x			
ODS 4. Educación de calidad.				x
ODS 5. Igualdad de género.				x
ODS 6. Agua limpia y saneamiento.			x	
ODS 7. Energía asequible y no contaminante.			x	
ODS 8. Trabajo decente y crecimiento económico.			x	
ODS 9. Industria, innovación e infraestructuras	x			
ODS 10. Reducción de las desigualdades.				x
ODS 11. Ciudades y comunidades sostenibles.			x	
ODS 12. Producción y consumo responsables.			x	
ODS 13. Acción por el clima.				x
ODS 14. Vida submarina.				x
ODS 15. Vida de ecosistemas terrestres.				x
ODS 16. Paz, justicia e instituciones sólidas.				x
ODS 17. Alianzas para lograr objetivos.	x			

Tabla 5: Objetivos de desarrollo sostenible.

Reflexión sobre la relación del TFG con los ODS (Objetivos de desarrollo sostenible) y con los ODS seleccionados

El proyecto propuesto de diseñar e implementar un modelo de clasificación basado en DL para la determinación del estado de metilación del gen MGMT en tumores, a partir de imágenes de

resonancia magnética (IRM), se alinea claramente con varios Objetivos de Desarrollo Sostenible (ODS) establecidos por las Naciones Unidas.

El más evidente es el ODS 3, "Salud y Bienestar", que busca garantizar una vida saludable y promover el bienestar en todas las edades. El proyecto se alinea con este objetivo al abordar directamente una de las principales causas de mortalidad en el mundo: el cáncer. Al desarrollar un modelo de aprendizaje profundo que puede predecir el estado de metilación del gen MGMT, se facilitaría la personalización de los tratamientos para el cáncer, lo que podría mejorar los resultados para los pacientes y potencialmente aumentar su supervivencia [55]. Además, al mejorar la eficiencia y la escalabilidad de este tipo de pruebas, se podría hacer que estos tratamientos personalizados sean más accesibles, ayudando a reducir las desigualdades en la atención sanitaria.

El proyecto también puede estar relacionado con el ODS 9, "Industria, Innovación e Infraestructura". Este objetivo busca construir infraestructuras resilientes, promover la industrialización inclusiva y sostenible, y fomentar la innovación. El proyecto está a la vanguardia de la innovación en el campo de la medicina y la inteligencia artificial, utilizando técnicas de procesamiento de imágenes y redes neuronales convolucionales para abordar un problema médico de gran importancia [56].

Además, si el proyecto tiene como objetivo compartir abiertamente sus métodos y resultados, también puede contribuir al ODS 17, "Alianzas para lograr los objetivos". Este objetivo destaca la importancia de la colaboración y la cooperación para lograr los ODS. Al compartir sus hallazgos con la comunidad médica y científica, el proyecto puede ayudar a promover la cooperación y el intercambio de conocimientos en este campo, lo que puede acelerar el desarrollo de nuevos tratamientos para el cáncer y otras enfermedades [57].

En resumen, este proyecto muestra cómo la innovación y la tecnología pueden ser utilizadas para abordar algunos de los desafíos más urgentes que enfrentamos en el campo de la salud. Al alinearse con los ODS, se reconoce el papel que juegan estos avances en la construcción de un futuro más sostenible y saludable para todos.

ANEXO II: Funciones y Clases creadas

```
class BrainTumorModel3D(tf.keras.Model):
    def __init__(self,
                 model,           # Sequential or Functional or Subclass Model
                 n_gradients=1,   # e.g total_batch_size = batch_size * n_gradients
                 *args, **kwargs):
        super(BrainTumorModel3D, self).__init__(*args, **kwargs)
        self.model = model
        self.n_gradients = tf.constant(n_gradients, dtype=tf.int32)
        self.n_acum_step = tf.Variable(0, dtype=tf.int32, trainable=False)
        self.gradient_accumulation = [tf.Variable(tf.zeros_like(v, dtype=tf.float32),
                                                  trainable=False)
                                      for v in self.model.trainable_variables]

    # The training step, forward and backward propagation
    def train_step(self, data):
        # Adding 1 to num_acum_step till n_gradients and start GA
        self.n_acum_step.assign_add(1)
        # Unpack the data
        images, labels = data

        # Open a GradientTape to record the operations run
        # during the forward pass, which enables auto-differentiation.
        with tf.GradientTape() as tape:
            # Run the forward pass of the layer or model .
            # The operations that the layer applies
            # to its inputs are going to be recorded
            # on the GradientTape.
            predictions = self.model(images, training=True)
            # Compute the loss value for this minibatch.
            loss = self.compiled_loss(labels, predictions)

        # Compute batch gradients
        gradients = tape.gradient(loss, self.model.trainable_variables)

        # Accumulating the batch gradients
        for i in range(len(self.gradient_accumulation)):
            self.gradient_accumulation[i].assign_add(gradients[i])

        # If n_acum_step reach the n_gradients then we apply accumulated gradients -
        # - to update the variables otherwise do nothing
        tf.cond(tf.equal(self.n_acum_step, self.n_gradients),
                self.apply_accu_gradients, lambda: None)
```

```
        # update metrics
        self.compiled_metrics.update_state(labels, predictions)
        return {m.name: m.result() for m in self.metrics}

    # Function for applying Gradient Accum.
    def apply_accu_gradients(self):
        # Apply accumulated gradients
        self.optimizer.apply_gradients(zip(self.gradient_accumulation,
                                           self.model.trainable_variables))

        # Reset
        self.n_acum_step.assign(0)
        for i in range(len(self.gradient_accumulation)):
            self.gradient_accumulation[i].assign(
                tf.zeros_like(self.model.trainable_variables[i],  dtype=tf.float32)
            )

    # The test step for evaluation and inference
    def test_step(self, data):
        # Unpack the data
        images, labels = data

        # Run model on inference mode
        predictions = self.model(images, training=False)

        # Compute the loss value for this minibatch.
        loss = self.compiled_loss(labels, predictions)

        # Update metrics
        self.compiled_metrics.update_state(labels,  predictions)
        return {m.name: m.result() for m in self.metrics}

    # A call funciton needs to be implemented
    def call(self, inputs, *args, **kwargs):
        return self.model(inputs)

    # A custom l2 regularization loss for model to tackle overfit
    def reg_l2_loss(self, weight_decay = 1e-5):
        return weight_decay * tf.add_n([
            tf.nn.l2_loss(v)
            for v in self.model.trainable_variables
        ])
```


```
AUTO = tf.data.AUTOTUNE

class TFDataGenerator:
    def __init__(self,
                 data,
                 modeling_in,
                 shuffle,
                 aug_lib,
                 batch_size,
                 rescale):
        if modeling_in not in ['2D', '3D']:
            raise ValueError('modeling_in is not set either 2D or 3D')
        self.data        = data             # data files
        self.modeling_in = modeling_in      # 2D or 3D
        self.shuffle     = shuffle          # true for training
        self.aug_lib     = aug_lib          # type of augmentation library
        self.batch_size  = batch_size       # batch size number
        self.rescale     = rescale          # normalize or not

    # a convinient function to get 2D data set
    def get_2D_data(self):
        self.data = self.data.shuffle(buffer_size = self.batch_size * 100) \
                                                    if self.shuffle \
                                                    else self.data

        if self.aug_lib == 'tf' and self.shuffle:
            # augmentaiton using tf.image.stateless_random* functions
            # applicable for multiple channel (channel > 3 )
            # same augmentation would be applied to each modalites: flair, t1w, t1wce, t2w
            # check the tf_image_augmentation code for details
            self.data = self.data.map(lambda x, y: (tf_image_augmentation(x), y),
                                      num_parallel_calls=AUTO).batch(self.batch_size,
                                                                     drop_remainder=self.shuffle)
        elif self.aug_lib == 'keras' and self.shuffle:
            # augmentaion using keras image preprocessing layers
            # applicable for multiple channels (channel > 3)
            # [known issue]:  same augmentaion would be applied for all modalites
            # check the tf_image_augmentation code for details
            self.data = self.data.batch(self.batch_size, drop_remainder=self.shuffle)
            self.data = self.data.map(lambda x, y: (keras_augment(x,training=True),
                                                    y),
                                      num_parallel_calls=AUTO)
        elif not self.shuffle:
            # no shuffle generally assuming no augmentaion too, so not training
            # for inference and evaluation
            self.data = self.data.batch(self.batch_size, drop_remainder=self.shuffle)
```

```
            self.data = self.data.map(lambda x, y: (keras_augment(x,training=True),
                                                    y),
                                      num_parallel_calls=AUTO)
        elif not self.shuffle:
            # no shuffle generally assuming no augmentaion too, so not training
            # for inference and evaluation
            self.data = self.data.batch(self.batch_size, drop_remainder=self.shuffle)


        # rescaling the data for faster convergence
        if self.rescale:
            self.data = self.data.map(lambda x, y: (layers.Rescaling(scale=1./255, offset=0.0)(x), y),
                                      num_parallel_calls=AUTO)

        # prefetching the data
        return self.data.prefetch(-1)

    # a convinient function to get 3D data set
    def get_3D_data(self):
        # augmentation on 3D data set
        # volumentation is based on albumentation, followed by tf and keras
        if self.aug_lib == 'volumentations' and self.shuffle:
            self.data = self.data.map(partial(volumentations_aug), num_parallel_calls=AUTO)
            self.data = self.data.batch(batch_size, drop_remainder=self.shuffle)
        elif self.aug_lib == 'tf' and self.shuffle:
            self.data = self.data.map(lambda x, y: (tf_image_augmentation(x), y),num_parallel_calls=AUTO)
            self.data = self.data.batch(self.batch_size,drop_remainder=self.shuffle)
        elif self.aug_lib == 'keras' and self.shuffle:
            self.data = self.data.batch(self.batch_size, drop_remainder=self.shuffle)
            self.data = self.data.map(lambda x, y: keras_augment(x, y), num_parallel_calls=AUTO)
        else:
            # true for evaluation and inference, no augmentation
            self.data = self.data.batch(self.batch_size, drop_remainder=self.shuffle)

        # rescaling the data for faster convergence
        if self.rescale:
            self.data = self.data.map(lambda x, y: (layers.Rescaling(scale=1./255, offset=0.0)(x), y),
                                      num_parallel_calls=AUTO)

        # prefetching the data
        return self.data.prefetch(-1)
```

```
# data loader
class BrainTumorGenerator(tf.keras.utils.Sequence):
    def __init__(self, dicom_path, data, is_train=True):
        self.is_train = is_train # to control training/validation/inference part
        self.data = data
        self.dicom_path = dicom_path
        self.label = self.data['MGMT_value']

    def __len__(self):
        return len(self.data['BraTS21ID'])

    def __getitem__(self, index):
        id = self.data['BraTS21ID'][index]
        if id.startswith("BRATS"):
            prefix, num = id.split('-')
            num_padded = num.zfill(5)
            patient_ids = f"{self.dicom_path}/{prefix}-{num_padded}/"
        else:
            patient_ids = f"{self.dicom_path}/{self.data['BraTS21ID'][index]}/"

        # for 3D modeling
        flair = []
        t1w   = []
        t1wce = []
        t2w   = []

        # Iterating over each modality
        for m, t in enumerate(input_modality):
            t_paths = sorted(
                glob.glob(os.path.join(patient_ids, t, "*")),
                key=lambda x: int(x[:-4].split("-")[-1]),
            )

            # Pick input_depth times slices -
            # - from middle range possible
            strt_idx = (len(t_paths) // 2) - (input_depth // 2)
            end_idx = (len(t_paths) // 2) + (input_depth // 2)
            # slicing extracting elements with 1 intervals
            picked_slices = t_paths[strt_idx:end_idx:1]

            # Iterating over the picked slices and do some basic processing,
            # such as removing black borders
            # and lastly, bind them all in the end.
            for i in picked_slices:
                # Reading pixel file from dicom file
                image = self.read_dicom_xray(i)

                # It's possible that among picked_slices, there can be some black image,
                # which is not wanted, so we iterate back to dicom file to get -
                # - any non-black image otherwise move on with black image
                j = 0
                while True:
                    # if it's a black image, try to pick any random slice of non-black
                    # otherwise move on with black image.
                    if image.mean() == 0:
```

```
# It's possible that among picked_slices, there can be some black image,
# which is not wanted, so we iterate back to dicom file to get -
# - any non-black image otherwise move on with black image
j = 0
while True:
    # if it's a black image, try to pick any random slice of non-black
    # otherwise move on with black image.
    if image.mean() == 0:
        # do something
        image = self.read_dicom_xray(random.choice(t_paths))
        j += 1
        if j == 100:
            break
    else:
        break

# Now, we remove black areas; remove black borders from brain image
rows = np.where(np.max(image, 0) > 0)[0]
cols = np.where(np.max(image, 1) > 0)[0]
if rows.size:
    image = image[cols[0]: cols[-1] + 1, rows[0]: rows[-1] + 1]
else:
    image = image[:1, :1]

# In 3D modeling, we now add frames / slices of individual modalities
if modeling_in == '3D':
    if m == 0:
        # Adding flair
        flair.append(cv2.resize(image, (input_height, input_width)))
    elif m == 1:
        # Adding t1w
        t1w.append(cv2.resize(image, (input_height, input_width)))
    elif m == 2:
        # Adding t1wce
        t1wce.append(cv2.resize(image, (input_height, input_width)))
    elif m == 3:
        # Adding t2w
        t2w.append(cv2.resize(image, (input_height, input_width)))
elif modeling_in == '2D':
    # Adding all frames at a time
    channel.append(cv2.resize(image, (input_height, input_width)))
```


```
set_seed = None
rand_flip = layers.RandomFlip("horizontal_and_vertical", seed=set_seed)
rand_tran = layers.RandomTranslation(height_factor=0.1, width_factor=0.2, seed=set_seed)
rand_rote = layers.RandomRotation(factor=0.01, seed=set_seed)
rand_cntr = layers.RandomContrast(factor=0.6, seed=set_seed)
rand_crop = layers.RandomCrop(int(input_height*0.97), int(input_width*0.97), seed=set_seed)
rand_eqlz = RandomEqualize()

def keras_augment(image, label):
    all_modality = tf.reshape(image, [-1, input_height, input_width, input_depth*input_channel])

    def apply_augment(x):
        x = rand_eqlz(x)
        x = rand_flip(x)
        x = rand_tran(x)
        x = rand_rote(x)
        x = rand_cntr(x)
        x = rand_crop(x)
        x = layers.Resizing(input_height, input_width)(x)
        return x

    aug_images = apply_augment(all_modality)
    image = tf.reshape(aug_images,
                       [-1, input_height, input_width, input_depth, input_channel])

    return image, label
```

```
    if modeling_in == '3D':
        # [ATTENTION!!!]
        # input_shape: (None, h, w, depth, channel)
        # It's possible that with current data loader set up,
        # All modalities MAY NOT HAVE SAME number of slices.
        # In that case, we adopt some workaround.
        # Right now, we re-append the existing slice with small color variation.
        # Just to avoid this issue.

        # for flair
        while True:
            if len(flair) < input_depth and flair:
                flair.append(cv2.convertScaleAbs(random.choice(flair), alpha=1.2, beta=0))
            else:
                break

        # for t1w
        while True:
            if len(t1w) < input_depth and t1w:
                t1w.append(cv2.convertScaleAbs(random.choice(t1w), alpha=1.1, beta=0))
            else:
                break

        # for t1wce
        while True:
            if len(t1wce) < input_depth and t1wce:
                t1wce.append(cv2.convertScaleAbs(random.choice(t1wce), alpha=1.2, beta=0))
            else:
                break

        # for t2w
        while True:
            if len(t2w) < input_depth and t2w:
                t2w.append(cv2.convertScaleAbs(random.choice(t2w), alpha=1.1, beta=0))
            else:
                break

        return np.array((flair, t1w, t1wce, t2w),
                        dtype="object").T, self.label.iloc[index,]


    elif modeling_in == '2D':
        # (None, h, w, channel == depth)
        return np.array(channel).T, self.label.iloc[index,]

# Function to read dicom file
def read_dicom_xray(self, path):
    data = pydicom.read_file(path).pixel_array
    if data.mean() == 0:
        # If all black, return data and find non-black if possible.
        return data
    data = data - np.min(data)
    data = data / np.max(data)
    data = (data * 255).astype(np.uint8)
    return data
```

```
from tensorflow.keras import backend as K
from tensorflow.keras import layers
from tensorflow import keras
import efficientnet_3D.tfkeras as efn
from classification_models_3D.tfkeras import Classifiers
from tensorflow.keras import regularizers

def unfreeze_model(model):
    # We unfreeze the top 10 layers while leaving BatchNorm layers frozen
    for layer in model.layers[-10:]:
        if not isinstance(layer, layers.BatchNormalization):
            layer.trainable = True
    return model


def get_model(width=128, height=128, depth=64, channel=1,trainable_layers=15):
    input_tensor = keras.Input((height, width, depth, channel), name='input3D')
    mapping3feat = keras.layers.Conv3D(3, (3,3,3),
                                   strides=(1, 1, 1),
                                   padding='same',
                                   use_bias=True,
                                   name = 'my_conv3d_1')(input_tensor)

    efficientnetv2, _ = Classifiers.get('efficientnetv2-b3')
    feat_ext = efficientnetv2(input_shape=(height, width, depth, 3),
                      include_top=False, weights='imagenet')

    feat_ext.trainable = False


    start_trainable_index = len(feat_ext.layers) - trainable_layers

    # Descongela las últimas 'trainable_layers' capas
    for index, layer in enumerate(feat_ext.layers):
        layer.trainable = index >= start_trainable_index


    output = feat_ext(mapping3feat)
    output = keras.layers.GlobalAveragePooling3D(keepdims=False, name='my_gb_avg_3d_pool')(output)
    output = keras.layers.BatchNormalization()(output)
    output = keras.layers.Dropout(0.4, name='top_dropout_1')(output)
    output = keras.layers.Dense(32, activation='relu')(output)
    output = keras.layers.BatchNormalization()(output)
    output = keras.layers.Dropout(0.4, name='top_dropout_2')(output)
    output = keras.layers.Dense(1, activation='sigmoid', name='my_output_1')(output)
    model  = keras.Model(input_tensor, output)
    return model
```

```
import os

import nibabel as nib

from scipy.signal import correlate

from sklearn.metrics import mutual_info_score

from scipy.ndimage import zoom

import numpy as np



def calculate_metrics(nifti1, nifti2):

   img1 = nib.load(nifti1).get_fdata()
```

```
    img2 = nib.load(nifti2).get_fdata()


    # Si las imágenes no tienen la misma forma, redimensiona la segunda para que coincida con la primera
    if img1.shape != img2.shape:
        factors = [i_f / i_t for i_f, i_t in zip(img1.shape, img2.shape)]
        img2 = zoom(img2, factors)


    img1 = img1.flatten()
    img2 = img2.flatten()


    # Normaliza las imágenes para que tengan media 0 y varianza 1
    img1 = (img1 - img1.mean()) / img1.std()
    img2 = (img2 - img2.mean()) / img2.std()


    # Calcula la correlación cruzada
    cross_correlation = correlate(img1, img2)


    # Normaliza la correlación cruzada
    cross_correlation /= np.sqrt(np.sum(img1 ** 2) * np.sum(img2 ** 2))


    mutual_information = mutual_info_score(img1, img2)
    return cross_correlation.max(), mutual_information



def compare_images(root_dir1, patient_dir1, root_dir2):
    # Abre un archivo de informe en modo de escritura
```

```
  with open('report.txt', 'w') as report:
    patient_dir1_full = os.path.join(root_dir1, patient_dir1)
    for patient_dir in os.listdir(root_dir2):
      patient_dir2 = os.path.join(root_dir2, patient_dir)
      for modality_dir in os.listdir(patient_dir1_full):
        modality_dir1 = os.path.join(patient_dir1_full, modality_dir)
        modality_dir2 = os.path.join(patient_dir2, modality_dir)
        if not os.path.exists(modality_dir2):
          report.write(f"No se encontró el directorio para la modalidad {modality_dir} en {patient_dir2}\n")
          continue
        image_files1 = [f for f in os.listdir(modality_dir1) if f.endswith('.nii.gz')]
        image_files2 = [f for f in os.listdir(modality_dir2) if f.endswith('.nii.gz')]
        if image_files1 and image_files2:
          image_path1 = os.path.join(modality_dir1, image_files1[0])
          image_path2 = os.path.join(modality_dir2, image_files2[0])
          cross_correlation, mutual_information = calculate_metrics(image_path1, image_path2)
          print(f"Imagenes {image_path1} y {image_path2} tienen una correlación cruzada de {cross_correlation} y una información mutua de {mutual_information}\n")
          if cross_correlation > 0.9 or mutal_information > 0.9:
            print(f"Imagenes {image_path1} y {image_path2} tienen una alta correlación cruzada de {cross_correlation} y una información mutua de {mutual_information}\n")
            report.write(f"Imagenes {image_path1} y {image_path2} tienen una alta correlación cruzada de {cross_correlation} y una información mutua de {mutual_information}\n")


root_dir1 = "/content/TCGA-NIFTI-Preprocessed"
root_dir2 = "/content/drive/MyDrive/BRATS-NIFTI"
```

```
# Uso de la función:

for patient_dir1 in os.listdir(root_dir1):

  compare_images(root_dir1, patient_dir1, root_dir2)
```