

“Aplicación basada en redes neuronales convolucionales para predicción de tumores cerebrales por medio de imágenes de resonancia magnética”

Diego L. Carrillo Santamaría

Madrid, octubre 2021

DETECCIÓN DEL PLAGIO

La Universidad utiliza el programa **Turnitin Feedback Studio** para comparar la originalidad del trabajo entregado por cada estudiante con millones de recursos electrónicos y detecta aquellas partes del texto copiadas y pegadas. Copiar o plagiar en un TFM es considerado una **Falta Grave**, y puede conllevar la expulsión definitiva de la Universidad.



[Incluir en el caso del interés de su publicación en el archivo abierto]

Esta obra se encuentra sujeta a la licencia Creative Commons **Reconocimiento – No Comercial – Sin Obra Derivada**

RESUMEN

El presente estudio busca apoyar la investigación en el análisis de los tumores cerebrales y su detección, clasificación y predicción haciendo uso de las herramientas y técnicas disponibles de Inteligencia Artificial (IA) por medio de la implementación de Redes Neuronales Artificiales para clasificación y análisis de imágenes médicas obtenidas por diferentes métodos ya conocidos.

En este trabajo se desarrollará una aplicación que facilite la tarea de diagnóstico al personal médico especializado en la detección de tumores cerebrales en pacientes que sufren de este mal, especialmente de las áreas de neurociencia, radiología y oncología, para ello será necesario utilizar bases de datos obtenidas en centros de salud por medio de la técnica de Resonancia Magnética (*MRi*).

Por otra parte, se expondrán los resultados de la implementación de dichas redes neuronales utilizando software “open source” como lo es Python con las bibliotecas de *Keras* y *TensorFlow* en entornos de desarrollo como *Anaconda* y *Google Colab* los cuales están disponibles para tales efectos de forma libre. De igual manera, se muestran los diferentes sets de datos disponibles para la comunidad de desarrolladores e investigadores, así mismo como su uso y tratamiento en el análisis cualitativo de los resultados.

De igual manera, se expondrán soluciones utilizando diferentes métodos de implementación, así como su rendimiento y precisión en el resultado a fin de proponer el uso de un sistema confiable. También, se pretende desarrollar una aplicación web que facilite la tarea de predicción del tipo de tumor, ya que, como estudiaremos más adelante no es una tarea sencilla por la naturaleza física de estos tumores. Para ello, utilizaremos los recursos que tenemos disponibles para hacer agregar valor de forma visual de una forma fácil y accesible para todos.

Dichos recursos comprenden el tratamiento de imágenes que incluyen modificación de color, mapas de calor, modificación de tamaños de imagen, rotaciones de imagen, entre otros, para poder entrenar nuestro sistema de manera eficaz y eficiente, de manera que podamos comprender mejor los fenómenos que ocurren durante este proceso con el objetivo principal de poder detectar el tipo de tumor cerebral que puede tener potencialmente un paciente.

Como bien sabemos, las exigencias tecnológicas modernas en el campo de la salud demandan un alto grado de fiabilidad, precisión y facilidad de uso, así que en este trabajo nos plantearemos objetivos que reflejen los conocimientos adquiridos en el curso del Máster, que a posteriori puedan contribuir a la sociedad en temas que son de interés de las personas, los gobiernos, la sociedad y en general, de quien requiera sobreponerse de una enfermedad compleja, que tanto daño puede ocasionar en un paciente.

PALABRAS CLAVES

Inteligencia artificial, redes neuronales artificiales, machine learning, deep learning, dataset, tumor cerebral, glioma, meningioma, tumor pituitario, resonancia magnética.

AGRADECIMIENTOS

Aprovecho esta oportunidad para reconocer las muestras de apoyo y soporte de parte de mi familia y amigos más cercanos quienes de una u otra manera me han influenciado de forma positiva para poder llegar donde estoy. A ellos, quienes me han comprendido, aceptado y apoyado en mi decisión de iniciar esta fascinante aventura y cuya experiencia ha marcado mi vida y me ha hecho evolucionar no solo a nivel profesional, sino que a nivel personal.

De igual forma, quiero agradecer a todo el personal académico, profesores y compañeros quienes me han acompañado a lo largo del Máster y quienes me han brindado un tesoro invaluable gracias a su provechoso aporte no solo en lo educativo y académico sino en el aprendizaje humano que han sumado a mi vida. Estoy convencido que, sin todo el aporte de estas personas, el camino hubiese sido mucho más complicado y difícil de manejar.

A TODOS, ¡MUCHAS GRACIAS!

TABLA DE CONTENIDOS

CAPÍTULO I: INTRODUCCIÓN	1
CAPÍTULO II: OBJETIVOS	3
CAPÍTULO III: JUSTIFICACIÓN DEL PROBLEMA	4
CAPÍTULO IV: ESTADO DEL ARTE	5
4.1 MODALIDAD INTERDISCIPLINAR DE LA MEDICINA Y LA INTELIGENCIA ARTIFICIAL	5
4.2 ESTUDIO MÉDICO DE LOS PRINCIPALES TUMORES CEREBRALES.....	7
4.3 USO DE CNN PARA RECONOCIMIENTO DE IMÁGENES MÉDICAS.	8
CAPÍTULO V: HERRAMIENTAS UTILIZADAS EN LA IMPLEMENTACIÓN	10
5.1 GOOGLE COLAB.....	10
5.2 GOOGLE DRIVE.....	12
5.3 JUPYTER NOTEBOOK	12
5.4 TENSORFLOW	13
5.5 KERAS	14
5.6 RED NEURONAL	14
5.7 RED NEURONAL CONVOLUCIONAL (CNN).	16
5.8 DATASETS.....	17
5.9 BIG DATA.....	18
5.10 FLASK.....	19
CAPÍTULO VI: ESTUDIO CAULITATIVO DE LOS PRINCIPALES TIPOS DE TUMOR	21
6.1 MENINGIOMAS	21
6.2 PITUITARIO	23
6.3 GLIOMA.....	24
6.4 IMÁGENES OBTENIDAS POR RESONANCIA MAGNÉTICA (mri).....	25
6.5 NATURALEZA DE UN TUMOR CEREBRAL	28
CAPÍTULO VII: DESARROLLO E IMPLEMENTACIÓN DEL SISTEMA	30
7.1 DATASETS UTILIZADOS EN LA IMPLEMENTACION.....	30
7.1.1 DATASET DE RENDIMIENTO MEJORADO	30
7.1.2 DATASET KAGGLE.....	31
7.1.3 GITHUB DATASETS	31
7.2 PREPARACIÓN DE LOS DATASET	32
7.3 OVERFITTING	33
7.4 USO DE CNN PRE-ENTRENADAS Y TRANSFER LEARNING	34

7.5 LA CONVOLUCIÓN.....	35
7.6 VGG16.....	36
7.7 INTEGRACIÓN DEL ENTORNO DE GOOGLE COLAB	38
CAPÍTULO VIII: RESULTADOS EXPERIMENTALES.....	41
8.1 USO DE UNIDAD DE PROCESAMIENTO GRÁFICO (gpu).....	41
8.2 PARÁMETROS INICIALES.....	43
8.3 USO DE LIBRERÍAS.....	44
8.4 PRUEBAS INICIALES.....	46
8.5 PROCESO MEJORADO DE ENTRENAMIENTO.....	56
8.6 DATA AUGMENTATION.....	65
8.7 USO DE TRANSFER LEARNING.....	67
8.8 ANALISIS COMPARATIVO CON OTROS ESTUDIOS	75
CAPÍTULO IX: DESARROLLO DE APLICACIÓN	78
9.1 RETROALIMENTACIÓN PREOFESIONAL	78
9.2 DESARROLLO DE APLICACIÓN.....	80
9.2.1 DESARROLLO DE APLICACIÓN BACK-END.....	81
9.2.2 DESARROLLO DE APLICACIÓN FRONT-END	82
9.2.3 EJECUCIÓN DE LA APLICACIÓN WEB.	84
9.2.4 EJECUCIÓN DE APLICACIÓN VISUAL	87
CONCLUSIONES.....	92
SIGUIENTES PASOS POR REALIZAR.....	94
BIBLIOGRAFÍA.....	95

ÍNDICE DE FIGURAS

Figura 1: Principales áreas de las ciencias médicas donde se aplica IA.	6
Figura 2: Integración de Google Colab con otros servicios e interfaces de usuario.	11
Figura 3: Jerarquía de kits de herramientas de TensorFlow.	13
Figura 4: Orden de jerarquía de Keras	14
Figura 5: Modelo básico de una red neuronal.	15
Figura 6: Arquitectura fundamental de una NN.	16
Figura 7: Representación general de la arquitectura de una CNN.	17
Figura 8: Las tres "V" del Big Data según Gartner.	18
Figura 9: Ejemplo de integración utilizando el framework de Flask.	19
Figura 10: Gráfico de incidencia por edad y sexo por cada 100.000 habitantes en los Estados Unidos entre 2002 y 2006.	21
Figura 11: Imagen por resonancia magnética (MRI) de un meningioma cerebral.	22
Figura 12: Vista superior y lateral de microadenoma pituitario localizado en la base de la glándula pituitaria.	23
Figura 13: Imágenes obtenidas por MRI de un tumor pituitario A) Vista Superior B) Vista lateral izquierda.	24
Figura 14: Imagen de Tomografía Axial Computarizada de un Glioma localizado en el lóbulo axial.	25
Figura 15: Interpretación física del campo magnético creado por el dispositivo generador durante el proceso de obtención de imagen.	26
Figura 16: Representación gráfica de una máquina para MRI.	27
Figura 17: Similitud anatómica de un tumor Meninglioma y Pituitario.	28
Figura 18: a) Fase temprana y Avanzada de un tumor cerebral Glioma. b) Imagen real del proceso de extirpación de un tumor Glioma avanzado en un paciente.	29
Figura 19: distribución de carpetas en Google Drive.	32
Figura 20: Estructura interna de cada de una de las carpetas para Entrenamiento, Validación y Testeo.	32
Figura 21: Comportamiento de una función con sobreajuste.	33
Figura 22: Convolución aplicando un filtro 3 x 3 píxeles para una imagen de 5 x 5 píxeles.	35
Figura 23: Actuación de una convolución en una imagen.	36
Figura 24: Arquitectura del modelo VGG16 con 2 capas totalmente conectadas.	37
Figura 25: Cuadro resumen de la arquitectura VGG16.	38

Figura 26: Integración de Google Drive con el entorno de Google Colab.	39
Figura 27: Mensaje de integración satisfactoriamente completado.	39
Figura 28: Ejemplo de almacenamiento de datasets para entrenamiento, tipo de tumor Meninglioma.	40
Figura 29: Información del GPU brindada por Google Colab.	41
Figura 30: Especificaciones técnicas de la GPU NVIDIA T4.	42
Figura 31: Muestreo de GPU disponibles para iniciar con la ejecución de código.	42
Figura 32: Asignación de GPU Tesla K80 en nueva sesión de Google Colab. Fuente: Google Colab (Elaboración propia).....	43
Figura 33: Efecto en el rendimiento de aprendizaje utilizando el optimizador ADAM para diferentes batch size.	44
Figura 34: Uso de librerías de TensorFlow y Keras para el desarrollo del sistema.	45
Figura 35: Total de imágenes elegidas para Entrenamiento, Validación y Prueba, para ambas clases.	46
Figura 36: Aplicación de filtro RGB para poder extraer características de la imagen con predicción de clases.	47
Figura 37: Clasificación de imagen según su clase.	47
Figura 38: Resumen del modelo creado en la prueba inicial.	48
Figura 39: Diagrama de bloques de modelo creado.....	49
Figura 40: Función de activación sigmoidea*	51
Figura 41: Función de activación ReLU**	51
Figura 42: Funcion de activacion tanh.....	52
Figura 43: Exactitud en la validación y comportamiento de las pérdidas durante el proceso de entrenamiento.....	53
Figura 44: Gráfica de precisión en proceso de entrenamiento y validación.....	53
Figura 45: Gráfica de pérdidas durante el proceso de entrenamiento y validación.	54
Figura 46: Arreglo de una matriz de confusión	54
Figura 47: Matriz de confusión para las clases de Meninglioma y NO_Tumor.	55
Figura 48: Función de matplotlib para obtener modelo de red neuronal por bloques.	56
Figura 49	57
Figura 50: MRi de Gliomas con distintas resoluciones.	58
Figura 51: Imágenes pre-preprocesadas antes de iniciar el entrenamiento de la red neuronal. A) Cerebro con tumor en vista lateral y frontal. B) Cerebro sin presencia de tumores en vista lateral y frontal.	59
Figura 52: Experimentación con nuevo dataset.	60

Figura 53: Resumen del modelo creado en la segunda prueba con dataset más grande.....	61
Figura 54: Etiquetas de las clases de la red neuronal.....	62
Figura 55: Distribución de imágenes por clase.....	62
Figura 56: Matriz de confusión para las clases de Meninglioma y NO_Tumor con nuevo dataset.	63
Figura 57: Gráfica de precisión de los procesos entrenamiento y validación.....	64
Figura 58: Gráfica de pérdidas de los procesos entrenamiento y validación.....	65
Figura 59: Configuración de la función de data augmentation de nuestra cnn.....	66
Figura 60: Resultado de aplicar el data augmentation sobre una imagen.....	67
Figura 61: Modelo de Transfer Learning aplicado a nuevo modelo.....	68
Figura 62: Dataset de 4 clases para implementación con Transfer Learning.	68
Figura 63: Etiquetado por clases para Transfer Learning.....	69
Figura 64: Modelo VGG utilizado para Transfer Learning.	69
Figura 65: Últimas 3 capas del modelo VGG16.....	70
Figura 66: Últimas 3 capas del modelo VGG16 con ajuste de la capa de salida.....	71
Figura 67: Rendimiento de entrenamiento en exactitud y pérdidas con transfer learning.....	71
Figura 68: a) Matriz de confusión para CNN de 4 clases., b) Evaluación de la exactitud en el proceso de entrenamiento.	72
Figura 69: Gráficas de rendimiento en exactitud y pérdidas para el proceso de entrenamiento y validación de la cnn.	73
Figura 70: Similitud entre imágenes de distintas clases.	74
Figura 71: Vistas en imagenes MRi de diferentes ángulos en datasets.	75
Figura 72: a) Matriz de Confusión obtenida en estudio previo. B) Porcentaje de lecturas verdaderas positivas en las predicciones de nuestro sistema.	76
Figura 73: Modelo de Red Neuronal aplicado en estudio.....	77
Figura 74: Esquema de funcionamiento de la aplicación propuesta.....	80
Figura 75: Aplicación Flask y repositorio de modelo de red neuronal.....	81
Figura 76: Pre-procesamiento de imagen dentro de la aplicación de Flask llamada app_cnn_predict4.py.....	82
Figura 77: Habilidad de entorno Flask.....	83
Figura 78: Acceso a librería jQuery para hacer el llamado a la función "prediction".....	83
Figura 79: Publicación en entorno web en la app escrita en HTML.	84
Figura 80: Repositorio de aplicaciones html dentro nuestro repositorio maestro.	84
Figura 81: Acceso a la aplicación web.	85
Figura 82: Predicciones obtenidas en nuestra aplicación web.....	87

Figura 83: Habilitación de librería para funcionalidad D3js para aplicación gráfica.	88
Figura 84: Aplicación visual para predicción de clases. Ejemplo con una imagen MRi sin tumor.	88
Figura 85: Resultados numéricos de predicciones sobre la misma imagen utilizada en el ejemplo anterior.	89
Figura 86: Aplicación visual para un Tumor Pituitario con semejanza con otras clases.	90
Figura 87: Análisis numérico de predicciones para la misma imagen MRi con Tumor Pituitario.	91

CAPÍTULO I: INTRODUCCIÓN

Durante los últimos años, el auge de la Inteligencia Artificial aplicada en diferentes áreas de la ciencia e industria, han evolucionado a niveles extraordinarios y específicamente el campo de las ciencias médicas no ha sido la excepción. Como es conocido, en particular el tema de procesamiento de imágenes ha venido a ayudar significativamente en la investigación y desarrollo de tecnologías y técnicas que colaboran para que los procesos en el área médica sean no sólo mucho más eficientes, sino que efectivas y confiables también.

Esto genera un impacto positivo en la calidad de vida de las personas, principalmente aquellas que sufren de enfermedades del sistema nervioso central y especialmente hablamos de tumores cerebrales, cuyo impacto en la salud del paciente puede ser letal si no se detecta o se trata a tiempo. Según la Asociación de Afectados por Tumores Cerebrales en España (ASATE), en el mundo la incidencia de tumores cerebrales es de 7,5 por cada 100.000 habitantes y son considerados la segunda causa de muerte en niños menores de 5 años, aunque se reconoce que el 90% de los tumores son benignos, el restante 10% de los casos pueden llegar a ser tumores letales por su naturaleza ya que atacan directamente el tejido de uno de los órganos más delicados y sensibles del ser humano, como lo es el cerebro. (ASATE, 2021).

Una de las particularidades que más destaca de esta enfermedad es que se desconoce abiertamente cuál es la causa que la genera, por ende, su predicción y diagnóstico puede llegar a ser complejo en muchas ocasiones. Por este motivo, nos hemos planteado la idea de explorar este campo, su complejidad lo hace ser abstracto y muchas veces indefinido, no sólo en el cómo sino en el por qué se presenta y de qué forma lo hace, sabiendo que las técnicas para recopilar información son principalmente el uso de imágenes médicas obtenidas por métodos ya bien conocidos como: resonancias magnéticas (*RM*), tomografías axiales computarizadas (*TAC*), Single Photon Emission Computer Tomography (*SPECT*) y Photo Emission Tech (*PET*), cuyos resultado pueden ser utilizados para el análisis computacional haciendo uso de técnicas de IA (García Fenoll, 2010).

Además, este estudio pretende enfocarse en el análisis de los principales tipos de tumores cerebrales que son diagnosticados por parte del personal médico del campo de neurología, radiología y oncología, y quienes al año detectan una cantidad considerable de nuevos casos en España, situación que ha servido de base para investigar y desarrollar herramientas que apoyen el diagnóstico, y que a su vez, ayude en el proceso de recuperación de paciente, gracias al uso de técnicas de IA para procesamiento de imágenes por computador que permita generar modelos predictivos.

Gracias a estos trabajos de investigación que se han desarrollado recientemente en esta área, es que ha sido posible estabilizar y hasta disminuir la mortalidad en la mayoría de países, la detección temprana y el correcto tratamiento de la enfermedad son los pilares fundamentales para combatir dicha mortalidad en personas de todas las edades en las que pueda manifestarse este padecimiento, claramente, aún queda mucho trabajo por hacer ya que todavía se siguen presentando una gran cantidad de casos. Sacar provecho de todas las ventajas tecnológicas que tenemos disponibles hoy en día nos permite profundizar cada vez en el estudio de las enfermedades, y en este caso, en las aplicaciones que apoyen el campo de las ciencias neurológicas y radiológicas.

CAPÍTULO II: OBJETIVOS

El desarrollo de este proyecto tiene la intención de crear una aplicación para la detección y clasificación de tumores cerebrales haciendo uso imágenes médicas de Resonancia Magnética, utilizando Redes Neuronales Artificiales para de clasificación de imágenes.

- Desarrollar una aplicación para la detección multiclase de tumores cerebrales por medio del uso de imágenes médicas y set de datos predefinidos.
- Construir una plataforma de predicciones para los tumores cerebrales: Meningioma, Glioma, Pituitario y No Tumor, cuyos casos de incidencias son los más frecuentes en los centros de salud especializados.
- Implementar el uso de técnicas de clasificación de imágenes por medio de Redes Neuronales con el fin de predecir el tipo de tumor cerebral que presenta la imagen.
- Elaborar una aplicación web que sea capaz de reproducir el modelo de Red Neuronal creado, el cual permita mostrar las predicciones hechas por el modelo de CNN implementado.
- Utilizar la plataforma Google Colab para el desarrollo del algoritmo, así como todas las funciones que ofrece, principalmente el uso de la GPU suministrada de forma gratuita.
- Desarrollar funciones facilitadas por TensorFlow y Keras para la implementación de la red neuronal propuesta.

CAPÍTULO III: JUSTIFICACIÓN DEL PROBLEMA

Como se ha mencionado anteriormente, los tumores cerebrales son enfermedades ampliamente heterogéneas y que pueden ser consideradas desde simples tumores benignos que requieren o no intervención quirúrgica hasta tumores mortales dependiendo de su naturaleza, tamaño, forma, crecimiento y región donde se desarrolla. El simple hecho de desconocerse a ciencia cierta el origen de este tipo de tumores, ha hecho que sea objeto de investigación y estudio desde hace algunos años atrás.

En los tiempos actuales, con el despliegue y desarrollo tecnológico al que tenemos acceso, así como las herramientas que se ofrecen para el análisis y estudio multidisciplinario de esta enfermedad, han hecho que las autoridades sanitarias de todo el mundo y por supuesto, de España, brinden los recursos necesarios para el estudio y análisis de este fenómeno, ya que representa un importante porcentaje de personas que se ven afectadas por este mal y por el cual se debe prestar especial atención, ya que la salud es un derecho fundamental del ser humano que le compete al individuo, a la sociedad y al estado.

Uno de los principales retos que supone la exploración de este campo, es lograr crear un sistema lo suficientemente confiable de predicción el cual manifieste resultados seguros, esto porque los especialistas médicos deben tener extrema precaución a la hora de dar un diagnóstico, ya que de ello depende el tratamiento al paciente posteriormente. Una falla en el diagnóstico puede traer consecuencias muy negativas para el que padece este mal.

Es por ello que deben explorarse diferentes técnicas que nos permitan generar un modelo seguro y confiable de predicciones, hoy en día contamos con una amplia gama de modelos de CNN que se han creado a partir de estudios previos, la idea de este trabajo de investigación no solo se limita a la creación del sistema, sino que también pretende construir una aplicación que permita de manera sencilla y simple poder evaluar imágenes reales y a partir de ello generar modelos predictivos que ayuden al personal médico en la tarea de detección y diagnóstico, teniendo en consideración que estamos ante un escenario complejo y diverso, ya que no existen patrones definidos en cuanto a la forma en que se presentan los tumores.

CAPÍTULO IV: ESTADO DEL ARTE

4.1 MODALIDAD INTERDISCIPLINAR DE LA MEDICINA Y LA INTELIGENCIA ARTIFICIAL

La investigación y desarrollo en el campo de las ciencias médicas haciendo uso de la *IA* para creación de aplicaciones ha venido mostrando un aumento significativo en los años recientes, gracias a la infinidad de recursos que se brindan para el diagnóstico, tratamiento y recuperación de enfermedades. En este trabajo, el enfoque se centra en el área de la neurociencia, específicamente la neurología en combinación con otras disciplinas como la radiología y la oncología.

Como es conocido, el denso mundo del procesamiento de imágenes haciendo uso de técnicas de *IA*, es pilar fundamental para el análisis de imágenes médicas obtenidas por distintos métodos tales como las resonancias magnéticas (*RM*), tomografías axiales computarizadas (*TAC*), Single Photon Emission Computer Tomography (*SPECT*) y Photo Emission Tech (*PET*), ya que gracias a estos procedimientos es posible determinar y diagnosticar el estado en que se encuentra un paciente, y a partir de esta valoración es posible evaluar el tratamiento al que debe ser sometido.

Propiamente, el análisis de imágenes está basado en modelos lógicos y matemáticos y nos permiten comprender diferentes mecanismos neurocognitivos que subyacen no solo en el entendimiento de la actividad biológica, sino que también el uso de tecnologías como por ejemplo la magneto encefalografía (*MEG*) que nos facilitan el análisis de la actividad cerebral prácticamente en tiempo real (Braininvestigators, 2018).

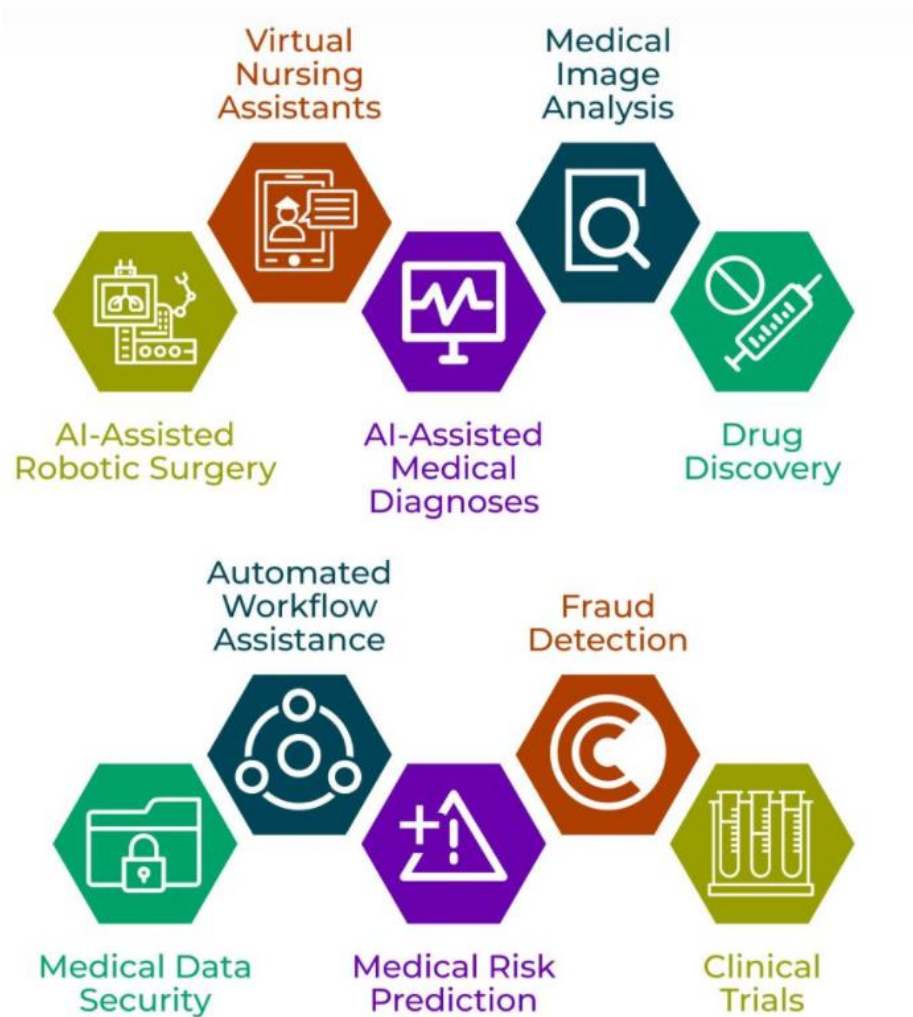


Figura 1: Principales áreas de las ciencias médicas donde se aplica IA.

Fuente: Ignite R&D Outsourcing (Disponible en: <https://igniteoutsourcing.com/healthcare/artificial-intelligence-in-healthcare/>)

Tal y como se muestra en la *Figura 1*, observamos que existen 10 principales áreas donde se aplica la IA en el campo médico, para este desarrollo investigativo, nos enfocaremos en el área de Análisis de Imágenes Médicas y Diagnósticos Médicos Asistidos, y en el que debe emplearse una visión multidisciplinar debido a que los conceptos tecnológicos sirven de base para apoyar la ciencia y el desarrollo de aplicaciones que faciliten al personal médico su labor.

Esto significa un enorme aporte ya que facilitará de gran forma una tarea que no solo es compleja, sino que también suele requerir mucho tiempo, ya que se demanda un riguroso análisis que siempre está propenso al error humano, y en los que se invierte grandes cantidades de recursos económicos y profesionales para poder dar un diagnóstico efectivo.

4.2 ESTUDIO MÉDICO DE LOS PRINCIPALES TUMORES CEREBRALES

Los tumores cerebrales representan un importante porcentaje de causas de muerte en muchos países alrededor del mundo, lo cual muestra un alto impacto social por su mal pronóstico, lo que ha hecho que las autoridades de salud de diferentes países hayan presentado un especial interés en el estudio e investigación de este mal que padecen muchas personas, desde finales de la década de los 80's y hasta la fecha actual se han desarrollado múltiples estudios que han venido acompañados de la revolución tecnológica de las últimas décadas. En 2016 la OMS incorpora una nueva edición de la "Clasificación de tumores primarios del sistema nervioso central" (Organización Mundial de la Salud, 2016), este documento de consulta ha venido no solo a demostrar la importancia que tiene para la OMS el conocimiento público de estos tumores, sino también que sirve de apoyo para la investigación de la comunidad de médicos y tecnólogos de la salud.

En España, estudios revelan que los tumores cerebrales abarcan aproximadamente un 2% del total de personas con cáncer en adultos y 15% en niños menores de 15 años, este índice es relativamente bajo en comparación con otros tipos de cáncer y su mortalidad se ha reducido con los avances tecnológicos y nuevos procedimientos médicos para el tratamiento del mismo (Sociedad Española de Oncología Médica, 2019), sin embargo, la pluralidad y diversidad en la forma en que éstos se manifiestan así como la dificultad para dar un diagnóstico preciso hace que sea un área de estudio de interés por parte de la comunidad médica mundial.

Otra de las singularidades de esta enfermedad es que se desconoce a ciencia cierta cuales factores pueden orinarlos, aunque se cree que podría estar ligado a diversos factores como virus, factores genéticos, radiaciones y consumo de sustancias químicas, entre otros, por otra parte, los trabajos investigativos han demostrado que los hombres, la raza blanca y los niños son quienes presentan el mayor índice de incidencia.

Los desarrollos más importantes de los últimos años se han enfocado en la creación aplicaciones capaces de clasificar y predecir los tumores cerebrales más comunes, apoyándose en bases de datos de imágenes que pueden ser obtenidas por distintos métodos como lo es la resonancia magnética (MRI) y Tomografías Axiales Computarizadas (TAC) las cuales se encuentran disponibles de forma libre y gratuita. Estas bases de datos de datos suelen contener una inmensa cantidad de datos en forma de imágenes ya que trabajan sobre el Big Data, esto es de gran importancia ya que estas CNN requieren una amplia cantidad de imágenes que nos permitan entrenar la red de manera confiable y con resultados muy acertados y veraces.

4.3 USO DE CNN PARA RECONOCIMIENTO DE IMÁGENES MÉDICAS.

El principal reto en el uso de métodos de IA para el análisis y estudio de imágenes es que se suele requerir un alto grado de precisión en los resultados, ya que de ello depende el diagnóstico de la enfermedad del paciente, es por eso que se han creado técnicas para clasificación de imágenes por medio de modelos de CNN, las cuales se han diseñado para dar una alta exactitud en el aprendizaje y así convertirlos en modelos muy precisos y confiables, esta tarea es fundamental ya que se trata de una actividad altamente sensible y que debe brindar confianza no solo al personal médico sino también al paciente.

En el campo de la neurociencia, se debe tener en cuenta que debido a la complejidad en la detección y diagnóstico es que se ha venido trabajando arduamente de forma multidisciplinar para aprovechar al máximo las ventajas tecnológicas que están disponibles recientemente, la visión por computador es, por mucho, una de las aplicaciones más utilizadas que han venido a ayudar esta área, ya que prácticamente es la forma más fiable y menos invasiva de obtener “inputs” que permiten analizar y hacer aproximaciones referentes a enfermedades que afectan al cerebro y al sistema nervioso central.

Estudios recientes sugieren el uso de *Deep Learning* con redes pre-entrenadas para crear un sistema robusto de clasificación de imágenes mucho más eficiente en cuanto al tiempo requerido para el entrenamiento y en la precisión en los resultados, esto nos facilita en gran manera obtener salidas más veraces. Los estudios más recientes proponen la implementación de CNN tales como *VGG16 y 19, Inception V3, ResNet50 e EfficenNet*, siendo estos los más populares, sin embargo, no los únicos que se encuentran disponibles.

Es importante destacar que esto nos permite usar una técnica conocida como aprendizaje por transferencia, la cual facilita de gran manera la tarea de entrenamiento de las CNN, esto se puede interpretar como un punto de partida en el tratamiento de nuestro trabajo, en el que se toma una referencia en la solución de un trabajo previo para dar solución a otro problema similar, evidenciando así su flexibilidad, la cual permite la extracción de características de las imágenes que utilizamos en el entrenamiento.

Para el procesamiento de imágenes por computador se asegura que el aprendizaje por transferencia proporciona una importante ventaja y es la disminución en el tiempo de entrenamiento impactando directamente en una disminución en la generalización del error, ya que se reutilizan los pesos en las capas de nuestro modelo de CNN.

Esto resulta aún mejor en casos donde el modelo pre-entrenado tiene una significativa mayor cantidad de etiquetas y entradas (Goodfellow, Bengio, & Courville, 2016). Los principales usos que se le dan a los modelos pre-entrenados están vinculados a: clasificación de imágenes, extracción de funciones independientes e integradas y como inicialización de pesos.

Un trabajo importante que se ha hecho en materia de clasificación de Tumores Cerebrales en imágenes MRi es “*Classification of Brain Tumors from MRi Images Using a Convolutional Neural Network*” (Milica M. & Barjatarovic, 2020) realizado por Milica M. Badža y Marko Č. Barjaktarovic, en el que se utilizó el *dataset* de Figshare (Cheng, Brain Tumor Dataset Figshare, 2017), en este estudio se trata el tema de la heterogeneidad de las imágenes MRi de tumores cerebrales y como éstas afectan las predicciones de la red neuronal. De igual forma se demuestran resultados en las predicciones, estos resultados obtenidos nos sirven de parámetro para evaluar nuestra propuesta que veremos en capítulos posteriores en detalle.

CAPÍTULO V: HERRAMIENTAS UTILIZADAS EN LA IMPLEMENTACIÓN

Durante el desarrollo del trabajo se utilizaron algunas herramientas de software necesarias para el tratamiento del problema, la mayoría de éstas se encuentran disponibles de forma gratuita y libre en la web y cuya única finalidad es servir de apoyo a la investigación. A continuación, se hace referencia a cada una de ellas.

5.1 GOOGLE COLAB

La evolución de la IA, principalmente en los últimos 10 años en mundo informático y de desarrollo computacional, así como sus aplicaciones en una enorme cantidad de áreas de la ciencia, han influenciado notablemente a la gigante corporación americana con sede central en California para habilitar una poderosa interfaz para el desarrollo de programas y aplicaciones de software que utilizan modelos basados en *Machine Learning* y *Deep learning*.

Esta interfaz se basa en el entorno de *Jupyter Notebook* y se creó con el principal objetivo de entrenar modelos basados en redes neuronales, el sistema ofrece unidades de procesamiento gráfico (GPU por sus siglas en inglés) para procesamiento de imágenes y el servicio está disponible en el “*cloud*” por lo tanto ofrece muchísimas ventajas dentro de la que podemos mencionar:

- **Integración:** Ya que nos permite trabajar con distintos lenguajes, librerías y ejemplos previamente diseñados por otros desarrolladores. De igual manera es amigable con la plataforma GitHub cuya popularidad en los últimos años le ha colocado como una de las principales aplicaciones web gracias a sus funciones colaborativas para el desarrollo de software y apps en general.

De igual manera, al ser un servicio de Google se puede integrar con otras plataformas como lo es Google Drive, en donde se pueden almacenar *datasets*, imágenes, repositorios, archivos ejecutables, entre otros, de forma gratuita. El servicio gratuito ofrece hasta 15 GB libres, y en caso de ser requerido más espacio de almacenamiento se brinda la opción de pago según la necesidad del usuario.

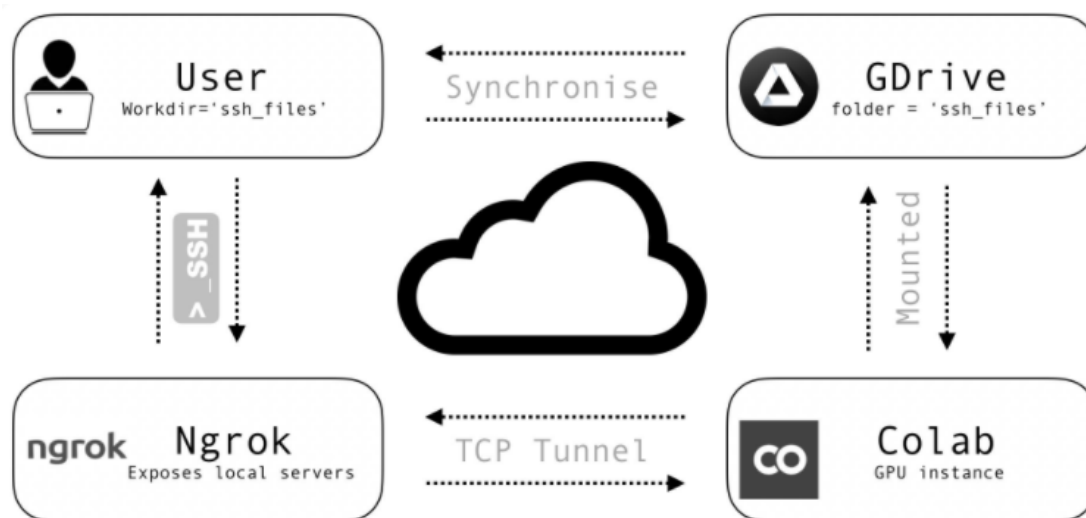


Figura 2: Integración de Google Colab con otros servicios e interfaces de usuario.

Fuente: Data Science blog of Imad El Hanafi (Disponible en: https://imadelhanafi.com/posts/google_colab_server/).

En la *Figura 2* vemos un ejemplo de una integración utilizando los servicios de *Google Colab* con *Google Drive* y otros servicios de terceros, esto supone ser una gran ventaja en la implementación, ya que evita tener que utilizar el espacio de memoria de nuestros ordenadores personales y también evita la pérdida de información ya que el servicio se encuentra disponible en la nube a toda hora y los archivos y repositorios se puede proteger de mejor manera. Por otra parte, es ideal para el almacenamiento y uso de *datasets* para entrenar redes neuronales.

- **Portabilidad:** Nos ofrece la facilidad de poder trabajar desde cualquier estación de trabajo, e incluso desde diferentes ordenadores, debido a que el servicio se ofrece en la nube y a que solo es requerida la conexión a internet para poder acceder a ella.
- **Eficiencia:** Por el hecho que no se necesita instalación ni configuración especial para poder utilizar sus servicios, también, las GPU que se tienen a disposición son muy poderosas lo que lo convierte en una herramienta ideal para entrenar redes neuronales y crear nuevos modelos utilizando los lenguajes más populares.

5.2 GOOGLE DRIVE

Es otra de las herramientas creada por *Google* cuyo principal propósito es servir como memoria de almacenamiento en la nube, para poder utilizar este servicio solo se requiere tener una cuenta de *Google*, que, dicho sea de paso, se puede crear de forma gratuita. El espacio que nos ofrece esta APP es de 15 GB, sin embargo, justo como se mencionó anteriormente, se puede ampliar la capacidad pagando una tarifa mensual o eligiendo alguno de los planes que ofrece *Google*. Es importante mencionar que no está disponible en todos los países, así como también hay que tener en cuenta que dependiendo del país pueden cobrarse cargos e impuestos por el uso del mismo. Actualmente se pueden tener un máximo de espacio de memoria de 2 TB.

La principal ventaja de este servicio es la facilidad de integración con todos los servicios que ofrece *Google*, lo cual lo hace ideal para la implementación del trabajo en cuestión, ya que al utilizarse la plataforma de *Google Colab*, podemos almacenar los *datasets* con los que se va a trabajar. Por otro lado, evita la dependencia de trabajar con un solo dispositivo, permitiendo que podamos trabajar desde cualquier estación con solo tener acceso a internet.

5.3 JUPYTER NOTEBOOK

Otro proyecto creado con el fin de apoyar la investigación y desarrollo entre la comunidad de programadores, desarrolladores e investigadores ofreciendo un entorno completamente gratuito y de libre acceso, amigable con el entorno *Python* y sus aplicaciones. Este *open source* es ideal para crear proyectos o bien compartir trabajos y así poder realizar cambios o mejoras sobre otros desarrollos. Dentro las principales ventajas podemos mencionar que permite ejecutar código línea a línea, es decir, podemos monitorizar la respuesta de cada línea de código que se haya implementado, lo cual nos permite detectar errores o problemas en la ejecución al instante sin tener que realizar procesos de compilación y ejecución de forma global.

Jupyter Notebook al igual que *Google Colab* requiere de acceso a internet si se van a utilizar bibliotecas especiales o *frameworks* previamente creados para integrarlos a nuestro código, esta poderosa herramienta es ampliamente utilizada por desarrolladores debido a su excelente rendimiento y confiabilidad en los resultados, así como la gran facilidad que nos brinda en el uso *apps* que sirven de apoyo para darle sustento teórico y gráfico a los proyectos.

5.4 TENSORFLOW

Según el sitio web oficial de *Tensorflow* se asegura que “es una plataforma de código abierto de extremo a extremo para el aprendizaje automático” (TensorFlow, 2015). Esta plataforma nos permite la integración flexible de una serie bibliotecas para desarrollo de proyectos y tesis de carácter educativo e investigativo. Al igual que muchas de las herramientas que están disponibles libremente para la comunidad de desarrolladores, nos permite entrenar los modelos desde la nube y con una amplia cantidad de lenguajes de programación.

Su arquitectura sencilla y flexible han hecho que hoy en día tenga una enorme popularidad ya que permite ampliar el concepto de integración y complejidad de los modelos que creamos, así mismo como su fácil y rapidez de interactuar con modelos ya existentes a un alto nivel. Esto nos facilita la resolución de problemas aun cuando estos sean muy complicados y con una eficiencia muy alta.

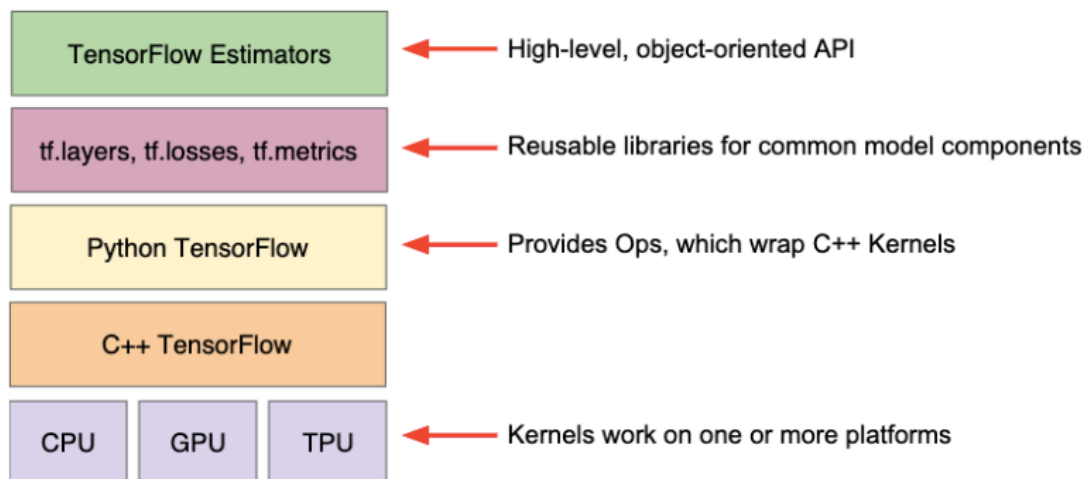


Figura 3: Jerarquía de kits de herramientas de TensorFlow.

Fuente: The Health Big Data Blog (Disponible en: <https://mksaad.wordpress.com/2020/01/14/tensorflow-toolkit-hierarchy/>).

La *Figura 3*, nos muestra los kits de herramientas con los que *Tensorflow* tiene capacidad de trabajar, desde el más alto nivel orientado a objetos hasta el uso de hardware a nivel de CPU, TPU y GPU, es decir, *TensorFlow* nos da la posibilidad de hacer una integración total con diferentes niveles computacionales.

5.5 KERAS

Es una popular biblioteca de Python para procesamiento de redes neuronales que trabaja sobre la plataforma de *Machine Learning*, su uso se ha divulgado ampliamente por las grandes ventajas y facilidades que tiene para integrarse con *TensorFlow* y *Theano* (Keras, 2015), lo que lo convierte en una poderosa herramienta de gran capacidad, rapidez y precisión en los resultados. Su principal ventaja es que se encuentra en libre disposición y tiene una inmensa cantidad de documentación que lo convierte en una herramienta ideal para la investigación y desarrollo de trabajos.

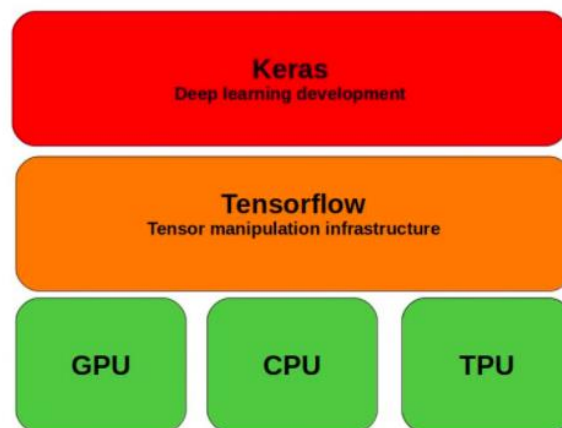


Figura 4: Orden de jerarquía de Keras

Fuente: The Health Big Data Blog (Disponible en: <https://starship-knowledge.com/awesome-python-data-science-libraries>).

La *Figura 4*, nos muestra la jerarquía de *Keras*, su uso es especialmente dedicado para trabajar con modelos de *Deep Learning* enfocado en una experiencia del tipo *Front-End* para ser desarrollado con *API's* y conectar con servicios web, de igual manera, *Keras* permite trabajar tanto con hardware especial como convencional con una amplia cantidad de *datasets*, sus aplicaciones, características y funcionalidades nos permiten hacer un análisis mucho más profundo y complejo de nuestros trabajos, añadiendo un valor agregado importante para dar sustento teórico y práctico a los trabajos de investigación.

5.6 RED NEURONAL

Las redes neuronales o *NN* (Neural Network por sus siglas en inglés) son patrones computacionales inspirados en la estructura de una neurona humana, el cual representa un sistema adaptativo que puede ser ajustado en función del propósito para el cual fue diseñado (García Benítez, López Molina, Ramos Trejo,, Palacios Morales, & Mora Méndez, 2016). Las

NN forman parte de los algoritmos de IA y sus aplicaciones se emplean en un sinnúmero de campos tecnológicos que involucran las ciencias médicas, ingenieriles, aplicaciones domésticas e industriales, seguridad y vigilancia, entre muchas otras más.

Su función se orienta principalmente en el aprendizaje automático de tareas basado en un conjunto de datos de entrada que pueden ser tablas con información tabulada, imágenes, vídeos, audios, letras, números, etc., los cuales tiene como principal función servir como referencia para brindar información al sistema para que pueda aprender a partir de los datos de entrada de forma automática.

La representación básica de una red neuronal la podemos observar en la *Figura 5*, en donde observamos un conjunto de entrada de datos la cual varía dependiendo de la aplicación, así mismo como una serie de interconexiones que tienen implícitos unos pesos que determinan la actuación de la neurona, estos pesos pueden interpretarse como negativos o positivos en función de las entradas. Por otra parte, se tiene una función de activación que suele ser no lineal capaz de limitar la amplitud de la salida.

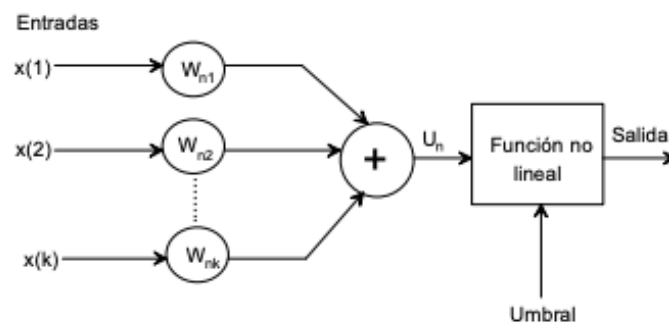


Figura 5: Modelo básico de una red neuronal.

Fuente: Redes Neuronales Artificiales (pág. 15).

La notación matemática que describe el comportamiento de la salida se obtiene por las siguientes ecuaciones:

Ecuación 1	$U_n = \sum_{j=1}^k W_{nj} * x(j)$
------------	------------------------------------

y,

Ecuación 2	$Salida = \rho(U_n - umbral)$
------------	-------------------------------

Donde ρ es la función de activación.

Arquitectura de una red neuronal:

En la Figura 6, se muestra el esquema fundamental de la arquitectura de una red neuronal:

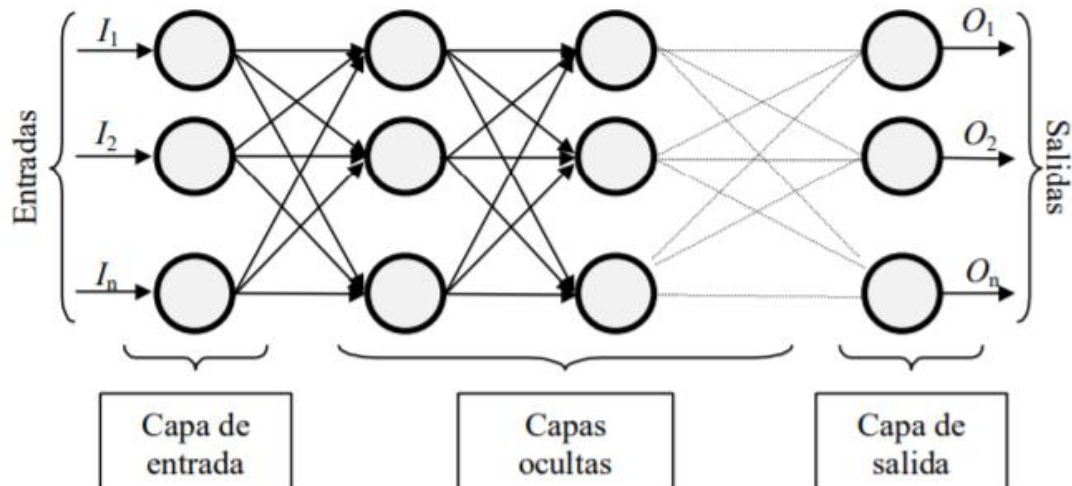


Figura 6: Arquitectura fundamental de una NN.

(Fuente: Redes Neuronales: Conceptos Básicos. pág 12).

Esta estructura muestra un modelo segmentado en capas, de la siguiente forma:

- Capa de entrada: donde se reciben los datos de entrada.
- Capas ocultas: las que interconectan las neuronas de la entrada con las de la salida.
- Capa de salida: diseñada para extraer los datos ya procesados y mostrar una salida con los resultados que se obtuvieron.

5.7 RED NEURONAL CONVOLUCIONAL (CNN).

Las redes neuronales convolucionales (CNN por sus siglas en inglés) son un tipo de red neuronal que tiene pesos y sesgos que le permiten a aprender, esto se hace de manera tal que una neurona reciba una o varias entradas, luego aplica una operación matemática que realiza un producto escalar y luego una función de activación.

Luego se debe tener en cuenta las pérdidas al cual le llamaremos costo que actúa en la última capa la cual suele estar totalmente conectada. Las *CNN* son principalmente utilizadas para el tratamiento de imágenes. La estructura fundamental de las *CNN* se basa en las capas convolucionales, *pooling* o reducción y capa clasificadora (IAARbook, 2018).

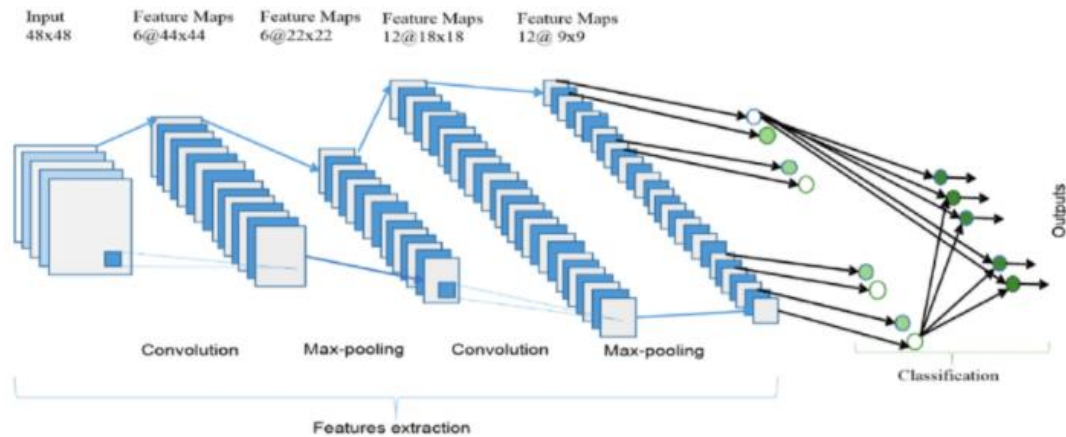


Figura 7: Representación general de la arquitectura de una CNN

Fuente: Researchgate (Disponible en: https://www.researchgate.net/figure/The-overall-architecture-of-the-Convolutional-Neural-Network-CNN-includes-an-input_fig4_331540139)

La *Figura 7* nos muestra la arquitectura básica de una *CNN*, en dónde se tiene una entrada con características bien definidas en cuanto al peso y tamaño de la imagen, y estas se van transfiriendo a las subsecuentes capas ocultas para extracción de características de interés, aplicando filtros especiales hasta llegar a la última capa que suele estar totalmente conectada y esto nos permite hacer una clasificación de imágenes a partir una entrada.

5.8 DATASETS

Los *datasets* o conjunto de datos contienen las referencias necesarias para poder entrenar y procesar una red neuronal y normalmente estas colecciones suelen presentarse en una configuración tabulada de datos que pueden ser estructuras numerales, textos, letras, sonidos, imágenes, vídeos o cualquier referencia que vaya a utilizarse para hacer que la red neuronal aprenda automáticamente algún patrón para obtener un resultado a la salida.

Si bien es cierto no existe una definición oficial, se puede asegurar que el conjunto de datos es una representación de datos alojados en una memoria el cual responde a un modelo de programación coherente e independiente, sin importar el origen de dichos datos (Ozdemir & Susarla, Feature Engineering Made Easy, 2018).

5.9 BIG DATA

El big data hace referencia a conjuntos de datos a gran escala y con una mayor complejidad estructural derivados de nuevas fuentes, y que en muchos casos se requiere hardware con características especiales para poder gestionar el peso computacional para su procesamiento. Tal y como lo asegura el gigante informático *ORACLE*, se puede definir el *Big Data* con la regla de las “tres V”, haciendo referencia al Volumen, Velocidad y Variedad que gestiona el *Big Data* en el procesamiento y valor que les da a modelos de redes neuronales para poder entrenar las redes neuronales. (Oracle Inc., 2016)

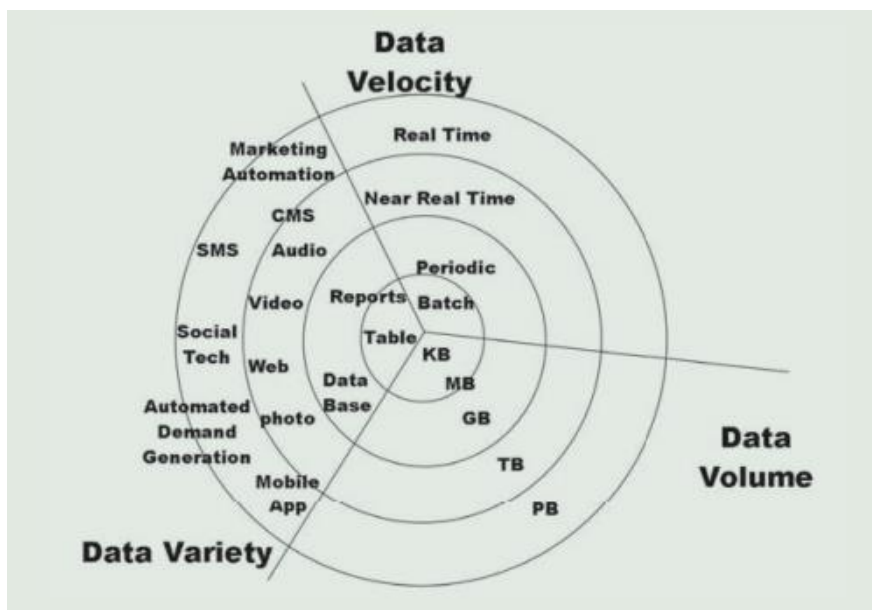


Figura 8: Las tres "V" del Big Data según Gartner.

Fuente: Big Bunker Data. (Disponible en: <https://thebigbunkerdata.wordpress.com/2015/04/01/las-3vs-del-big-data/>).

Tal y como lo apreciamos en la *Figura 8*, se muestra la definición dada; la variedad viene dada por la extensa cantidad de datos e información en diferentes campos tecnológicos y científicos, la velocidad de procesamiento de estos también es una variable fundamental, ya que las exigencias tecnológicas modernas demandan rapidez en el procesamiento de la información y, por último; el volumen o capacidad de almacenamiento y procesamiento de la información con la que se está trabajando.

5.10 FLASK

Flask es un *framework* que se ha popularizado en los últimos años debido a su propiedad minimalista cuya raíz está escrita en *Python*, ésta no requiere de otras herramientas o librerías en particular, pero sí permite trabajar con extensiones y aplicaciones para integraciones orientadas a objetos.

Flask nos permite crear aplicaciones web e integraciones haciendo uso de su arquitectura simple, motivo por el cual se le conoce como *microframework*, aunque esto no quiere decir que sea una plataforma limitada por su sencillez. Una de sus principales ventajas es que nos permite el uso de ambientes virtuales, ideales para prevenir conflictos entre versiones de intérprete y desorden de paquetes y repositorios, facultando la creación de ambientes exclusivos para cada aplicación sin interferir en otras, mientras que el intérprete global se mantiene limpio y con opción a crear más ambientes virtuales para nuevas aplicaciones en caso de ser requerido (Grinberg, 2014).

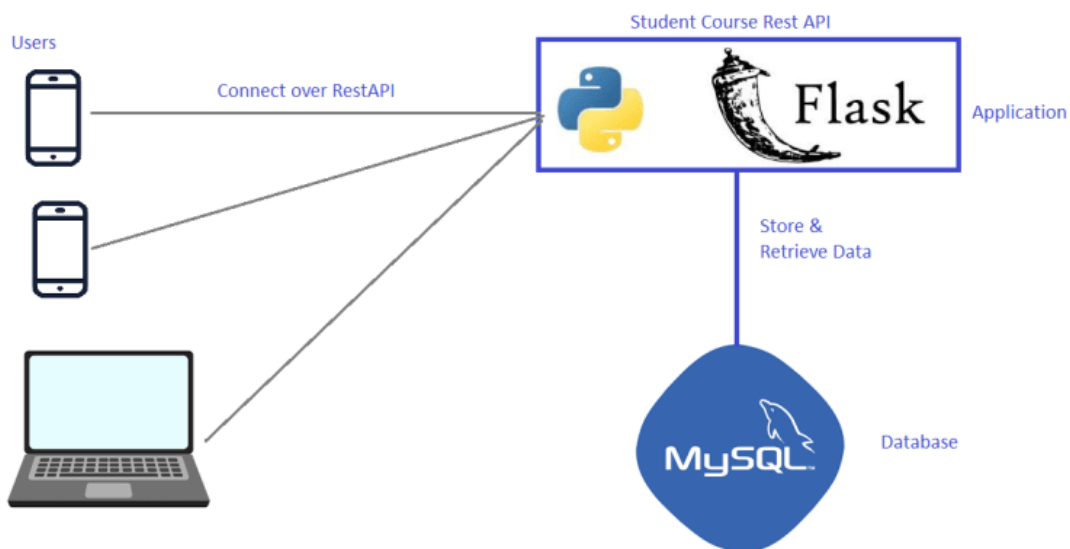


Figura 9: Ejemplo de integración utilizando el framework de Flask.

Fuente: Direct DevOps.blog (Disponible en <https://directdevops.blog/2019/11/02/deploying-the-docker-application-and-mysql-with-volume-support-into-kubernetes-from-code-to-docker-registries-like-acr-ecr-and-then-to-eks-aks/>)

La Figura 9, nos muestra un ejemplo de una integración utilizando *Flask* + *Python*, dentro de un entorno global, en donde los usuarios finales tienen acceso a una aplicación que llama a servicios de terceros como en este caso *MySQL*, siendo *Flask* el *framework* que administra una

aplicación web dentro de un entorno virtual, de esta manera la base de datos de MySQL nunca se verá modificada por la influencia de un tercero, y la aplicación de Flask funciona transparentemente ante el usuario.

CAPÍTULO VI: ESTUDIO CAULITATIVO DE LOS PRINCIPALES TIPOS DE TUMOR

Este estudio se orienta a los tipos de tumor cerebral más frecuentes según la OMS los cuales son Meningioma, Gliomas y Pituitarios.

6.1 MENINGIOMAS

Este tipo de tumores suelen ser benignos y en la mayoría de los casos se presentan con una anatomía encapsulada, dependiendo de su ubicación intracraneal pueden llegar a ser muy dañinos y en algunas ocasiones, letales. El interés de su estudio radica en el hecho de ser el tumor cerebral con mayor incidencia en los Estados Unidos entre 2002 y 2006 (Wiemels, Wrensch, & Claus, 2010). Su origen puede deberse a múltiples causas, sin embargo, se asegura que puede deberse a antecedentes familiares, por mutaciones en el gen de la neurofibromatosis (NF2), o bien por la exposición a altas de radiaciones ionizantes.

En comparación con otros tumores cerebrales, los meningioma están relativamente menos estudiados con respecto a los factores de riesgo etiológicos, lo cual hace que se requieran estudios más profundos ya que muchos se han descubierto mediante observación y no por medio de extirpación quirúrgica, al día de hoy solo pocos estudios epidemiológicos han tenido un peso estadístico importante para estudiar por separado los factores de riesgo que impactan directamente en la formación de este tipo de tumos.

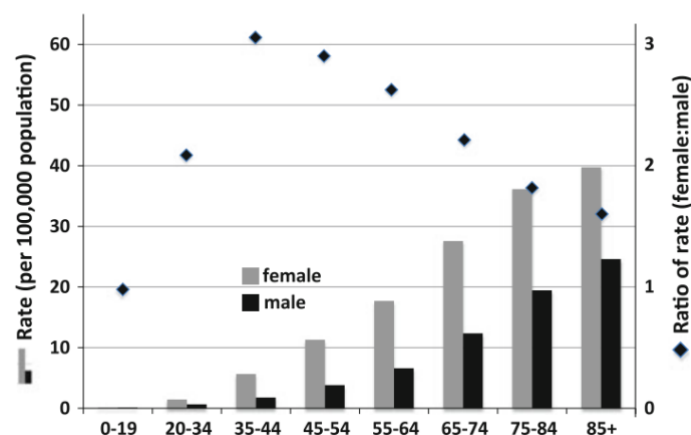


Figura 10: Gráfico de incidencia por edad y sexo por cada 100.000 habitantes en los Estados Unidos entre 2002 y 2006.

Fuente: Epidemiology and Etiology of Meningioma (Disponible en: <https://link.springer.com/article/10.1007/s11060-010-0386-3>)

En la *Figura 10*, podemos observar como la incidencia de Meningioma en la población de mayor edad es mucho más frecuente y principalmente en mujeres, mientras que en el caso de los más jóvenes la incidencia es considerablemente menor, sin embargo, siguen siendo las mujeres quienes están más propensas a sufrir de esta enfermedad. Por otra parte, de la gráfica se puede inferir que la población menor de 20 años tiene una muy baja incidencia o casi nula, no obstante, en los casos que se manifiesta la enfermedad en niños hay un alto porcentaje de posibilidad que esta llegue a ser letal (USA National Cancer Institute , 2020).

Con respecto a la anatomía de los Meningioma, se clasifican según sus características, estructura molecular y localización del tejido, divididos en tres grados, siendo el numero 1 el más lento en crecimiento, y el 3 el Meningioma anaplástico maligno cuyo crecimiento es mucho mayor.

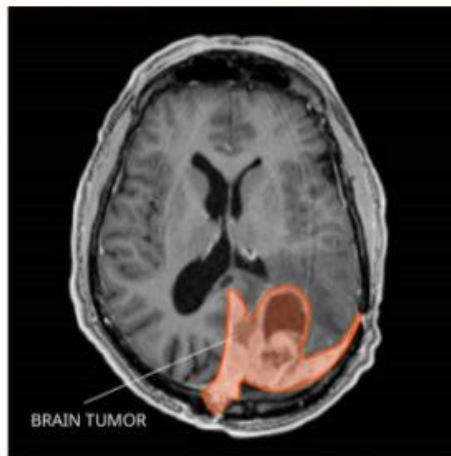


Figura 11: Imagen por resonancia magnética (MRI) de un Meningioma cerebral.

Fuente: USA National Cancer Institute, Gov. (Disponible en: <https://www.cancer.gov/rare-brain-spine-tumor/tumors/meningioma>)

La Figura 11, muestra una imagen MRi de un Meningioma medianamente desarrollado, frecuentemente los de grado II y III se manifiestan como una masa en la capa externa que reviste el tejido del cerebro. En ocasiones el Meningioma maligno también se puede diseminar a otras partes del sistema nervioso central por medio del líquido cefalorraquídeo e incluso con posibilidad de alcanzar el tejido óseo.

6.2 PITUITARIO

El tumor pituitario también se muestra como uno de los tumores que más incidencia presenta en los diagnósticos médicos, solo en los Estados Unidos en el año 2015 se detectaron unos 13.210 casos de este tipo de tumor (Siegel, Miller , & Jemal , 2017). Este tipo de cáncer tumoral es complejo en cuanto a su manifestación ya que podría mostrarse asintómicamente, o presentar síntomas secundarios o por disfunción endocrina.

Algunos de los principales síntomas impactan directamente el campo visual de las personas afectando la visión, también se pueden mencionar las cefaleas y las hidrocefalias en casos más extremos debido a la secreción de líquido péptido activo (Trincado Martínez, 2003), este tumor no tiene una forma indefinida ya que varía según su naturaleza y su grado de desarrollo y crecimiento. Este tipo de tumor suele alojarse en la región de las glándulas pituitarias localizadas en la base del cerebro, el impacto de este mal es ampliamente peligroso debido que las glándulas pituitarias producen hormonas que afectan las funciones de otras glándulas del cuerpo.

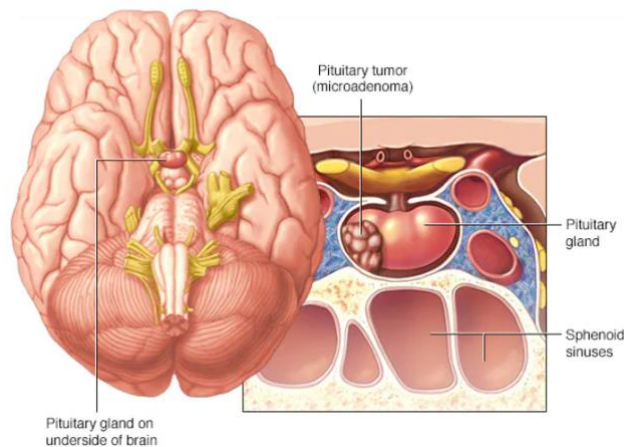


Figura 12: Vista superior y lateral de microadenoma pituitario localizado en la base de la glándula pituitaria.

Fuente: Mayo Clinic (Disponibles en: <https://www.mayoclinic.org/es-es/diseases-conditions/pituitary-tumors/symptoms-causes/syc-20350548#dialogId42104319>)

La *Figura 12* nos muestra un tumor pituitario localizado en la base de las glándulas pituitarias manifestándose como un tejido que se forma y crece en zona afectando las funciones pituitarias las cuales son fundamentales en el desarrollo de las actividades cerebrales y

endocrinas. La siguiente imagen muestra como se ve una MRI con vista superior y lateral de un tumor pituitario

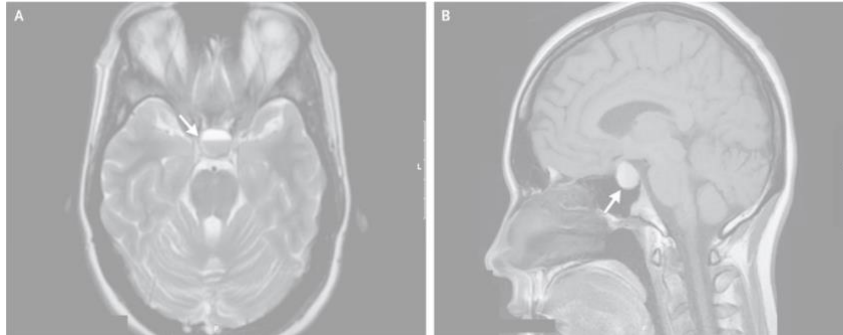


Figura 13: Imágenes obtenidas por MRI de un tumor pituitario A) Vista Superior B) Vista lateral izquierda.

Fuente: The New England Journal of Medicine (Disponible en: <https://www.nejm.org/doi/full/10.1056/nejmicm0900500>)

La *Figura 13*, nos muestra una imagen MRi de un paciente con tumor pituitario, en el cuadro de la derecha observamos la localización del tumor desde una vista superior, y el cuadro de la derecha nos muestra el mismo tumor desde la vista lateral.

6.3 GLIOMA

Otro de los principales tumores del sistema nervioso central, cuya información sobre las células que causan esta enfermedad aún se desconocen, aunque es cierto que se han desarrollado estudios profundos a lo largo de la historia con el fin de especificar su origen y se han expuesta ciertas hipótesis basadas en evidencia clínica, no obstante, al igual que sucede con los Meningiomas y tumores pituitarios, no es una tarea simple generar modelos predictivos debido a su heterogeneidad y variabilidad en la forma en que se presentan éstos tumores.

La incidencia de los gliomas es de aproximadamente 5 a 10 casos por cada 100,000 habitantes y comprenden tumores desde tumores de bajo grado o benignos hasta los más complejos y malignos por sus características fisiológicas en tamaño, forma, región donde se forman y tiempo de desarrollo en el cerebro humano, los cuales pueden ocasionar daños colaterales como cefaleas, cambios mentales, trastornos de lenguaje, alteraciones en la visión, disfunción endocrina, entre otros (Reyes Oliveiros & Lema Bouzas, 2007).

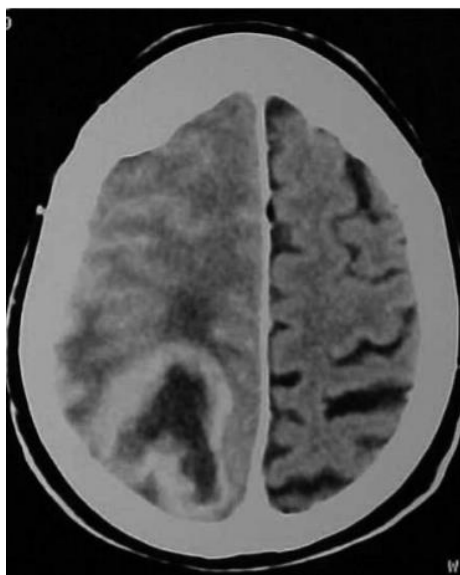


Figura 14: Imagen de Tomografía Axial Computarizada de un Glioma localizado en el lóbulo axial.

Fuente: Caracterización neuropsicológica de pacientes con glioma tratados en el instituto de cancerología de Medellín.

(Disponible en: https://www.acnweb.org/acta/2008_24_1_13.pdf)

La *Figura 14* nos muestra un glioma mayor de grado IV, el cual hace referencia a un tumor de unos 3 centímetros localizado en el lóbulo axial, este tipo paciente presentan tejidos cancerosos en una región muy sensible del cerebro cuyo tratamiento puede ser mixto dependiendo del diagnóstico médico, ya que puede requerir de una quimioterapia, radioterapia o cirugía.

6.4 IMÁGENES OBTENIDAS POR RESONANCIA MAGNÉTICA (MRI).

Las imágenes obtenidas por resonancia magnética se han utilizado en el campo de la medicina desde hace más de 40 años y aún continúan empleándose a la fecha actual para fines médicos debido a la gran cantidad de ventajas que ofrece esta técnica. Una imagen de resonancia magnética (MRi) no es más que un procedimiento en el que se aplica un campo magnético y ondas de radio controladas por computador para poder ver tejidos internos sin tener que invadir ningún tejido, órgano, hueso o parte del cuerpo.

El Doctor en Medicina Jaume Gili define la resonancia magnética como:

“un fenómeno físico por el cual ciertas partículas como los electrones, protones y los núcleos atómicos con un número impar de protones (Z) y/o un número impar de neutrones (N) pueden absorber selectivamente energía de radiofrecuencia” (Gili, 2002)

El fenómeno ocurre en presencia de campos magnéticos potentes, y es posible gracias a que algunos tejidos tienen una respuesta fisiológica al someterse a radiación temporal, razón por la cual se considera una técnica de obtención de imagen no-invasiva y que puede facilitar altas resoluciones en la imagen, hecho que facilita en gran manera el proceso de diagnóstico médico.

Las *MRI* se obtienen de dispositivos compuestos de dos grandes imanes en forma tubular para que el paciente puede entrar dentro de ellos y una vez que se emite un campo magnético, las moléculas de agua presentes en el cuerpo se configuran en otra posición, luego, por medio de la radiación de ondas de radio el tejido emite una respuesta que, aunque es muy pequeña, es suficientemente potente para crear una imagen a partir de dicha respuesta.

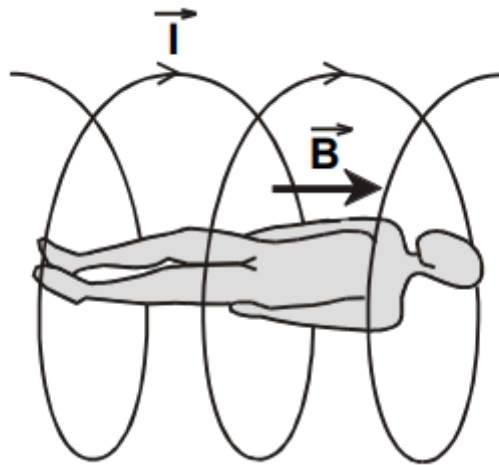


Figura 15: Interpretación física del campo magnético creado por el dispositivo generador durante el proceso de obtención de imagen.

Fuente: Introducción Biofísica a la Resonancia Magnética en Neuroimagen. pág 2.1

La *Figura 15* muestra el campo magnético \mathbf{B} inducido al que se somete un paciente durante la fase de obtención de resonancia, a lo largo de un conductor en forma de solenoide por donde circula una corriente eléctrica \mathbf{I} . Desde luego, esta técnica puede ser altamente peligrosa si no se siguen las precauciones de seguridad rigurosamente, para ello es necesario estar libre de cualquier elemento metálico durante la sesión, si el paciente cuenta con alguna prótesis metálica dentro de su cuerpo, marcapasos, implantes, etc, se debe consultar al médico el paso a seguir, la acción de informarlo al personal técnico de radiología es fundamental.

La *Figura 16* muestra la representación gráfica de un dispositivo utilizado para producir imágenes *MRI*.

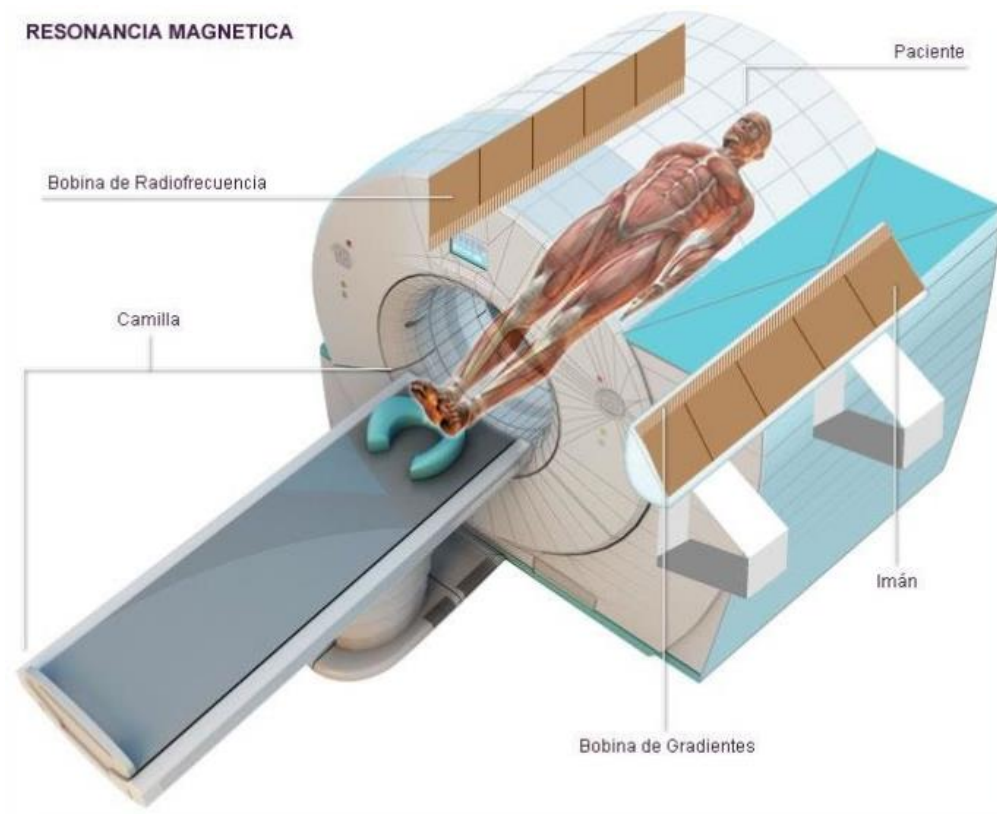


Figura 16: Representación gráfica de una máquina para MRI

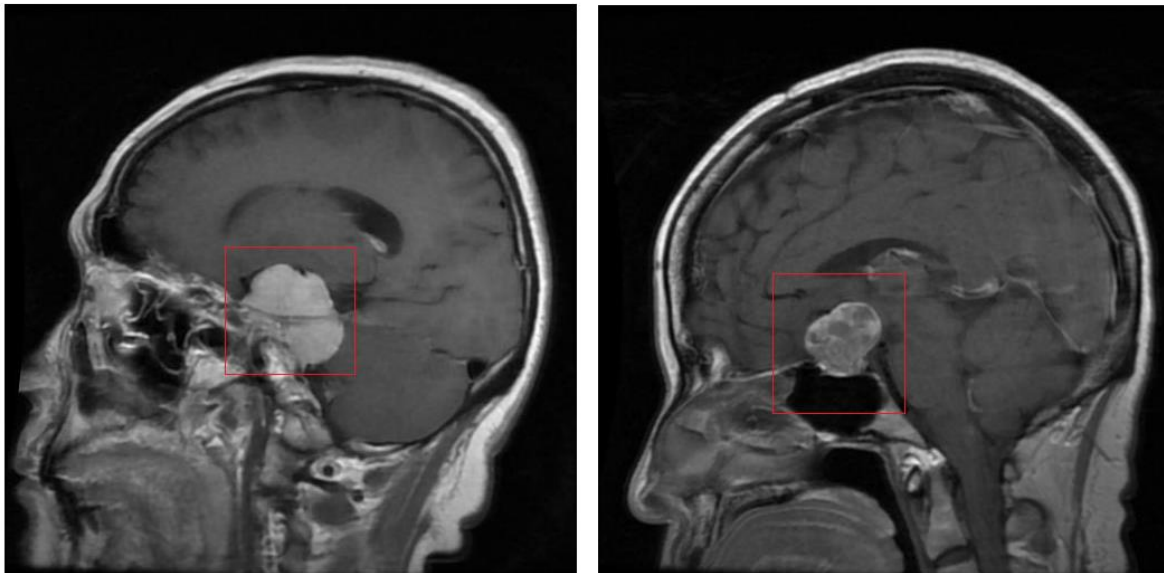
Fuente: ResearchGate (Disponible en: https://www.researchgate.net/figure/La-prueba-de-resonancia-magnetica-nuclear-RMN-consiste-en-usar-un-poderoso-campo_fig4_329020406)

Tal y como lo podemos ver, el dispositivo no es más cámara tubular en donde el paciente debe acostarse siguiendo rigurosamente las instrucciones de los técnicos radiólogos, retirando del cuerpo todo dispositivo metálico o ropa con accesorios metálicos, las bobinas actúan como imanes, al aplicarse una corriente eléctrica generan un campo magnético que viaja con una dirección conocida, al interactuar con las ondas de radio que hacen que los tejidos del organismo se sensibilicen es como se logra la obtención de la imagen *MRI*.

6.5 NATURALEZA DE UN TUMOR CEREBRAL

Anteriormente se hizo referencia a la particularidad de los tumores cerebrales de caracterizarse por ser ampliamente heterogéneos, no existen patrones definidos que determinen su forma, su tamaño o localización dentro del tejido cerebral. Esto representa el principal reto de esta investigación, ya que dos tumores diferentes pueden tener una localización, tamaño y forma muy similares, por ende, entrenar una red neuronal de manera efectiva no es una tarea fácil.

A continuación, vemos en la *Figura 17* un caso real de dos tumores (Meningioma y Pituitario), con una anatomía muy similar, el tamaño y forma es muy parecido en ambos casos, el punto de localización es muy similar también, ambas imágenes pertenecen a pacientes distintos.



MENINGLIOMA

PITUITARIO

Figura 17: Similitud anatómica de un tumor Meningioma y Pituitario.

Fuente: Figshare Dataset (Disponible en: https://figshare.com/articles/dataset/brain_tumor_dataset/1512427)

Por otra parte, sabemos que los tumores pueden presentarse síntomas desde etapas tempranas, o bien, existe la posibilidad que el paciente manifieste síntomas hasta fases avanzadas del tumor, esto demuestra lo complejo de realizar un pronóstico efectivo desde la primera vez.

GLIOMAS

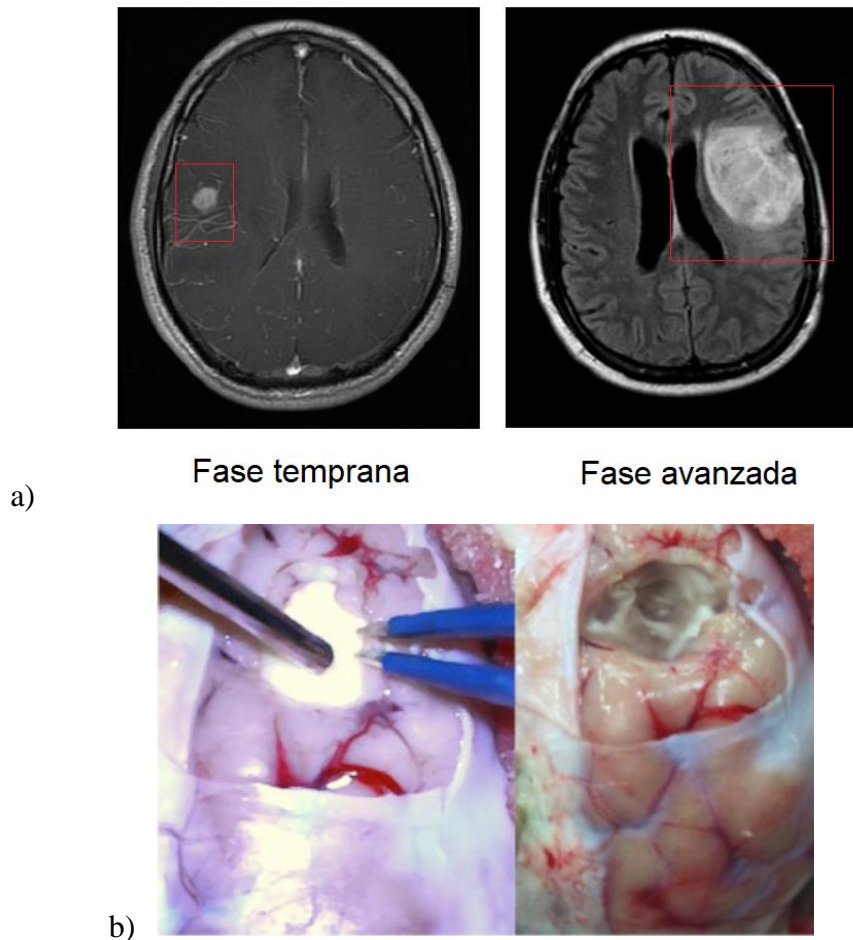


Figura 18: a) Fase temprana y Avanzada de un tumor cerebral Glioma. b) Imagen real del proceso de extirpación de un tumor Glioma avanzado en un paciente

Fuente: Figshare Dataset (Disponible en: https://figshare.com/articles/dataset/brain_tumor_dataset/1512427)

La *Figura 18a*, demuestra la diferencia de tamaños que puede llegar a representar el mismo tumor en diferentes etapas de desarrollo, es evidente que un tumor en etapa avanzada tiene altas posibilidades de llegar a producir un daño irreversible en el paciente o bien llegar a ser letal, de ahí surge el interés por parte de la comunidad médica de llegar a poder detectar en etapas tempranas esta enfermedad, ya que el tiempo de detección puede salvar la vida de una persona que padece este mal.

Por otra parte, la *Figura 18b* muestra el proceso de extirpación de un tumor Glioma avanzado en un paciente, por medio del cual un médico neurocirujano procese a cortar la membrana que protege al cerebro para poder tener acceso al tejido cancerígeno. Estas cirugías suelen ser muy complejas por la zona en donde se encuentra el cáncer, por lo que se requiere un alto nivel de conocimiento y experiencia en la práctica.

CAPÍTULO VII: DESARROLLO E IMPLEMENTACIÓN DEL SISTEMA

La implementación del sistema se fundamenta en un diseño basado en *Big Data* por medio del uso de *datasets* que nos permiten entrenar la red de manera eficiente, entre más imágenes de entrada usemos para realizar el entrenamiento mejor será el resultado y el rendimiento. Este trabajo muestra la utilización de diferentes *CNN* previamente entrenadas con el fin de mostrar cuál proporciona mejores resultados en la ejecución, de igual manera se generará un modelo de creación propia de igual manera para poder comparar los efectos en la salida. Como se ha mencionado en secciones anteriores, el proyecto pretende plantear el uso de diferentes técnicas que permitan contrastar los diferentes resultados experimentales.

7.1 DATASETS UTILIZADOS EN LA IMPLEMENTACION

Los *datasets* representan uno de los parámetros más importantes con los que se desarrollará nuestra red neuronal, ya que como se ha mencionado anteriormente, nos permite dar datos de entrada a nuestra red para que pueda reconocer imágenes según sus características. El principal reto de este desarrollo se justifica en el hecho de que los tumores cerebrales suelen tener formas, tamaños similares e igualmente localización dentro de la zona cerebral en regiones similares, por lo que debemos tener un modelo de *CNN* suficientemente preciso para que logre diferenciar entre uno y otro tumor.

Para este desarrollo investigativo se han utilizado los siguientes *datasets* disponibles de forma libre en la web que se detallarán a continuación:

7.1.1 DATASET DE RENDIMIENTO MEJORADO

- Enhanced Performance of Brain Tumor Classification via Tumor Region Augmentation and Partition (Cheng, et al., 2015)

Disponible en: https://figshare.com/articles/dataset/brain_tumor_dataset/1512427

La estructura de este *dataset* agrupa los tumores según los tipos de tejido que forman el tumor y se le llama tipo de tejido de la Región de Interés (ROI: *region of interest* por sus siglas en inglés) y se centra en tres tipos de tumor: Meningioma, Glioma y Pituitario obtenidos por resonancia magnética, que, dicho sea de paso, es la técnica más popular y convencional en los diagnósticos de tumores cerebrales.

7.1.2 DATASET KAGGLE

- Classification of brain tumors (MRI) (Chakrabarty & Kanchan, 2020).

Disponible en: <https://www.kaggle.com/sartajbhuvaji/brain-tumor-classification-mri>

Kaggle es una plataforma web de libre acceso concebida para la creación e intercambio de set de datos para poder entrenar redes neuronales, esto permite a la comunidad de desarrollo tener acceso a más de 50,000 datasets con más de 400,000 notebooks para poder crear, mejorar, experimentar y poner a prueba los proyectos de *Machine Learning* y *Deep learning* (Kaggle, 2010).

7.1.3 GITHUB DATASETS

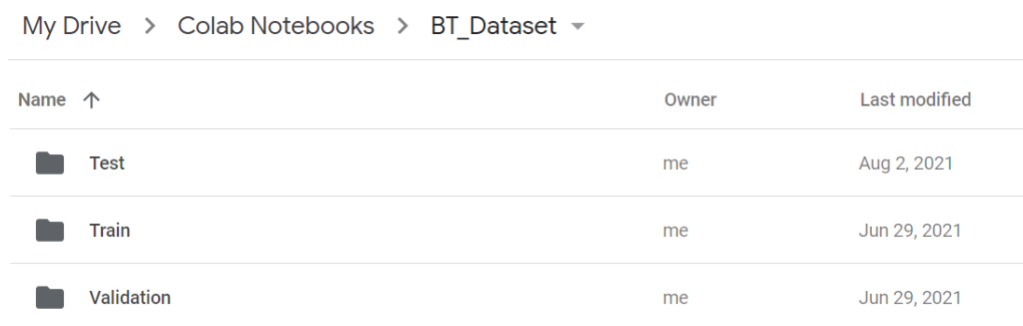
- GitHub Brain Tumor Detector (Ali Habib, 2019)

Disponible en: <https://github.com/MohamedAliHabib/Brain-Tumor-Detection>

Otro medio de libre acceso que permite subir a su plataforma proyectos para intercambio e interacción entre la comunidad de desarrolladores, dentro de sus principales ventajas podemos mencionar no solo el hecho de ser *open source*, sino también, que permite colgar en la nube archivos, *datasets*, repositorios, entre otras cosas, necesarias para la implementación de proyectos.

7.2 PREPARACIÓN DE LOS DATASET

Tal y como se indicó anteriormente, los *datasets* son de suma importancia para el correcto entrenamiento de nuestra red neuronal, de este set de datos dependen, entre otras cosas, la eficiencia con que nuestro modelo de red neuronal va a trabajar, a mayor eficiencia, mayor porcentaje de confiabilidad tendremos en los resultados de predicción. En esta ocasión, se ha utilizado *Google Colab* usando la interfaz de *Google Drive*, es en *Google Drive* que almacenaremos nuestra colección de imágenes que para nuestros propósitos serán imágenes. A continuación, observamos en la siguiente imagen la forma en como hemos distribuido nuestro set de datos.



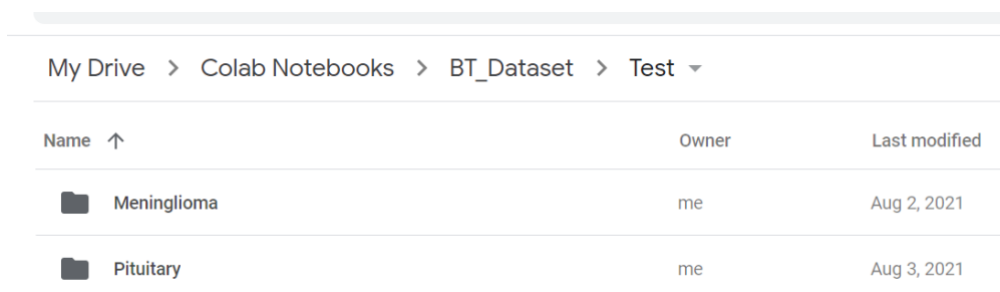
My Drive > Colab Notebooks > BT_Dataset ▾

Name ↑	Owner	Last modified
Test	me	Aug 2, 2021
Train	me	Jun 29, 2021
Validation	me	Jun 29, 2021

Figura 19: distribución de carpetas en Google Drive.

Fuente: Google Colab (Elaboración propia)

Como se puede ver en la *Figura 19*, se ha dividido el *dataset* en 3 diferentes carpetas: Train, Validation y Test, de forma que se ha utilizado el 70% del total de imágenes para entrenamiento, 20% validación y 10% para testeo. Dentro de cada una de las carpetas se encuentran cada una de las clases que corresponden a los tipos de tumor que se quiere predecir de la siguiente forma:



My Drive > Colab Notebooks > BT_Dataset > Test ▾

Name ↑	Owner	Last modified
Meninglioma	me	Aug 2, 2021
Pituitary	me	Aug 3, 2021

Figura 20: Estructura interna de cada de una de las carpetas para Entrenamiento, Validación y Testeo.

Fuente: Google Colab (Elaboración propia)

Para este trabajo haremos una combinación de datasets y veremos el resultado en la implementación, los trabajos que se han realizado con estos datasets normalmente no suelen combinar los sets de datos, pero en esta ocasión haremos una depuración del datasets en donde eliminaremos las imágenes de baja resolución y les combinaremos para ver el efecto en la salida.

7.3 OVERFITTING

La regla del 70-20-10 que hemos utilizado en nuestra red no es definitiva para todos los casos y todas las CNN, esto es solo una elección aleatoria, sin embargo, algunas bibliografías sugieren la distribución de los *datasets* de una u otra manera para evitar el “*overfitting*” o sobreajuste, término utilizado en estadística para describir cuando un modelo contiene demasiada información sobre una muestra o clase al punto que solo es capaz de aprender datos de esa clase pero al presentarse otro dato no es capaz de reconocerlo ya que no tiene la suficiente información sobre esa clase. Tal y como lo ha citado Pablo Argibay en el informe sobre Estadística Avanzada: el problema del sobreajuste y el método de descripciones mínimas:

“Se define la paradoja del sobreajuste como: los modelos complejos contienen más información sobre los datos de la muestra, pero menos información sobre los datos futuros (predicción)” (Argibay, 2011)

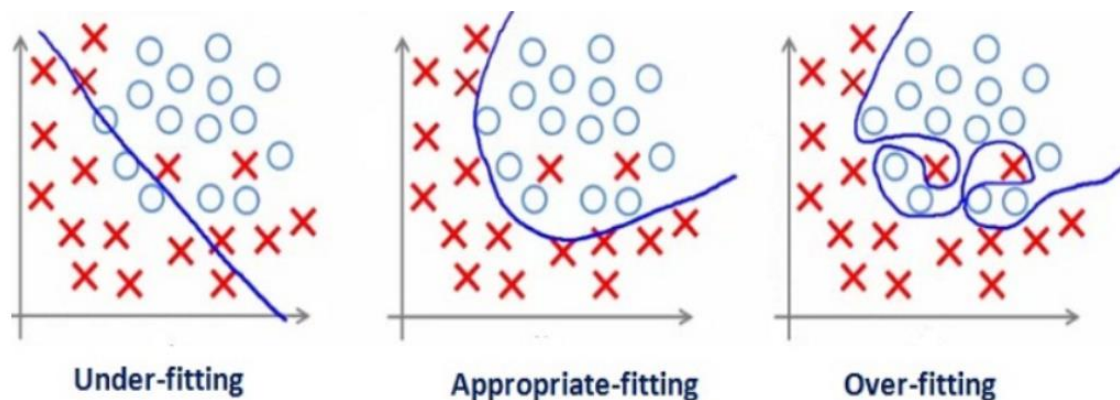


Figura 21: Comportamiento de una función con sobreajuste.

Fuente: Dream Learning (Disponible en: <https://dreamlearning.ai/neural-network-from-scratch-elementos-indispensables-teoria-optimizadores-inicializaciones-regularizadores-y-otros-hyperparametros-parte-11/>)

Como podemos observar en la *Figura 21*, se aborda el problema del sobreajuste, este se debe principalmente a que el algoritmo o función recibe demasiados datos de una o varias clases, es decir, cuando se da un sobre entrenamiento, por tanto, cuando debe reconocer otros

patrones o características en la imagen, el sistema no será capaz de reconocerlos, lo ideal sería tener una función que sea capaz de clasificar de manera correcta las diferentes clases. Por otra parte, en la imagen de la izquierda también tenemos el caso de “*underfitting*” al cual tenemos que evitar también, ya que también nos proporciona una lectura poco exacta en la salida.

El caso del *underfitting* sucede cuando el algoritmo no ha sido capaz de aprender de forma efectiva cada una de las diferentes clases, esto se puede ser ocasionado por la calidad de las imágenes en los datasets o porque no hemos introducido una cantidad suficiente de datos de entrada para que el algoritmo sea capaz de aprender eficientemente dichas clases. Al igual que en el caso del *overfitting*, el *underfitting* podría generar falsos positivos o falsos negativos en la lectura de la salida de nuestro sistema.

Lo ideal sería intentar que la función sea similar a la imagen que observamos en el centro de la figura, en la cual vemos como la función es capaz de discriminar las diferentes clases de forma eficiente, aunque si bien es cierto, llegar a alcanzar el 100% de efectividad es realmente una tarea compleja, se pueden tener lecturas muy cercanas, cuanto más cerca del 100%, mejor será nuestra predicción.

7.4 USO DE CNN PRE-ENTRENADAS Y TRANSFER LEARNING

El uso de redes neuronales pre-entrenadas es una acción realmente útil para efectos de entrenar nuestro modelo, se debe tomar en cuenta que existen trabajos previos que han demostrado a través de resultados reproducibles que es posible obtener un nivel de efectividad cercano al 97%, modelos que son bastante exactos si se comparan con los primeros desarrollos de *CNN*. Más adelante se discutirán algunas de las principales *CNN* pre-entrenadas que se utilizarán en este trabajo, por ahora, solo será necesario decir que se puede implementar el uso de las principales arquitecturas de *CNN* conocidas como *VGG16* e *Inception* para reproducir los resultados y poder hacer un análisis comparativo de sus ventajas y desventajas.

Con el paso del tiempo las *CNN* son cada vez más utilizadas en diferentes campos de la ciencias y la tecnología, y esto ha hecho que cada vez se requiera un nivel más alto en la precisión de los sistemas que utilizan ésta técnica, por ello se introducido de *fine-tuning* o ajuste fino, en el cual se utilizan redes neuronales altamente exactas que han pasado por un proceso inmensamente meticuloso de entrenamiento, utilizando bases de datos enormes, lo cual permite ajustar los pesos de las capas de la red neuronal, de modo que ese modelo se pueda utilizar en otras redes (Kang, Cho, Yoon, Park, & Kim, 2021).

Esto facilita un proceso de entrenamiento mucho más rápido y preciso, concediendo un mejor aprovechamiento del tiempo, así como aumentando la confiabilidad de la red. Estos modelos hacen la tarea de clasificación y segmentación en clases de imágenes mucho más heterogéneas, como es el caso nuestro con los *datasets* de Tumores Cerebrales.

7.5 LA CONVOLUCIÓN

La convolución en una red neuronal es una técnica ampliamente usada en procesamiento de imágenes para extracción de características de una imagen, éste método se usa principalmente para la clasificación y segmentación, el termino convolución fue introducido por primera vez en 1980 por Kunihiko Fukushima, no obstante, 18 años después se mejoró éste método introduciendo un nuevo acercamiento por medio de la “programación hacia atrás” o *backpropagation* en la cual se utiliza la sumatoria de los errores en cada de las capas de la red neuronal por medio del cálculo del gradiente descendiente para posteriormente compararlo con la salida deseada. Este método se reproduce de forma automática y permite a la red neuronal reconocer características o patrones en una imagen con un tamaño definido de píxeles (Wu, 2017).

La convolución en una imagen de tamaño A x B píxeles se realiza por medio de la aplicación de filtros que esencialmente son más pequeños en tamaño que la propia imagen, la *Figura 22*, nos muestra este efecto.

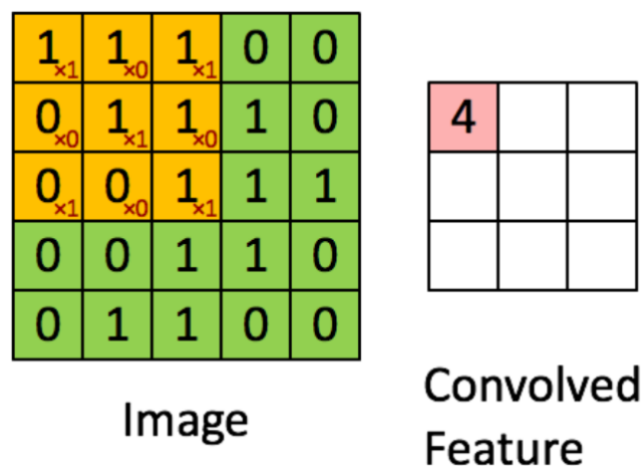


Figura 22: Convolución aplicando un filtro 3 x 3 píxeles para una imagen de 5 x 5 píxeles.

Fuente: Ice Cream Labs. (Disponible en: <https://medium.com/@icecreamlabs/3x3-convolution-filters-a-popular-choice-75ab1c8b4da8>)

En ella podemos observar cómo se la aplicación del filtro nos da una nueva salida en una imagen más pequeña, esta representación muestra una imagen de una sola dimensión, normalmente las imágenes son de 3 dimensiones por cada uno de los canales RGB, esto nos permite la extracción de algunas características en cada una de las capas de nuestra red.

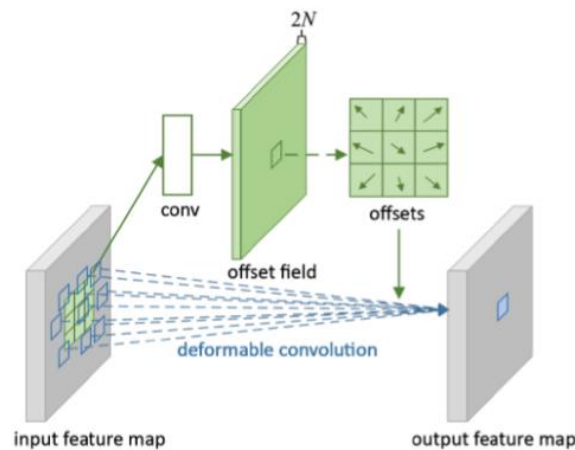


Figura 23: Actuación de una convolución en una imagen

Fuente: Archana Blog. (Disponible en: <https://archana1998.github.io/post/rahul-sukthankar/>)

En la *Figura 23*, observamos como es el efecto de la convolución aplicada en una imagen, el filtro que mencionamos anteriormente se aplica sobre un segmento determinado de una imagen, y es esta acción la que nos permite extraer las características, acción que a posteriori nos permitirá reconocer a cuál categoría pertenece la imagen, y de esta manera como la clasificación o segmentación en imágenes se lleva a cabo. Esta técnica es válida para aplicarla en secuencias de vídeo ya que se componen de varias imágenes por segundo, desde luego se requiere un altísimo recurso computacional para poder procesar el algoritmo, pero es esencia el mismo efecto que se produce en la salida.

7.6 VGG16

VGG16 se trata de una red neuronal convolucional, la cual ha sido entrenada en un subset de *ImageNet*, la cual utiliza una colección de más de 14 millones de imágenes pertenecientes a más de 20 mil clases distintas, alcanzando resultados con más del 92% de efectividad en la clasificación de imágenes (Tammina, 2019). Esta arquitectura es ampliamente utilizada para la técnica de *transfer learning* y su estructura la observamos a continuación en la *Figura 24*:

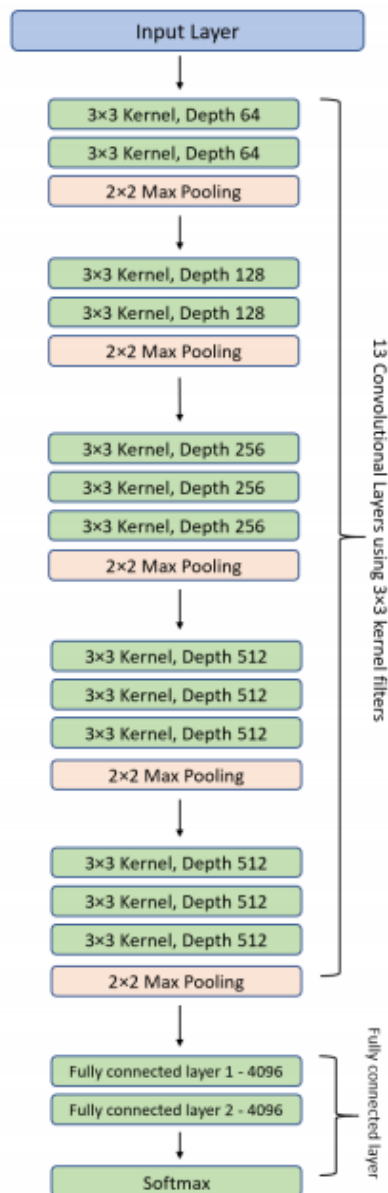


Figura 24: Arquitectura del modelo VGG16 con 2 capas totalmente conectadas.

Fuente: International Journal of Scientific and Research Publications (Disponible en: [Transfer learning using VGG-16 with Deep Convolutional Neural Network for Classifying Images \(ijsrp.org\)](https://www.ijsrp.org/)).

La estructura tiene dos capas convolucionales en la entrada, con un filtro de 64 de profundidad con un kernel 3 x 3, seguido de 2 capas con un filtro de 128 de profundidad con un kernel 3 x 3, seguidamente tiene 2 capas con un filtro de 256 de profundidad con un kernel 3 x 3, luego 6 capas con un filtro de 512 de profundidad con un kernel 3 x 3 separadas en 3 capas en un bloque y 3 en otro, finalmente dos capas totalmente conectadas con un módulo de salida con activación softmax, cada uno de los módulos tiene un capa de activación MaxPoolig de 2 x 2.

Layer	Patch size	Input size
conv×2	3×3/1	3×224×224
pool	2×2	64×224×224
conv×2	3×3/1	64×112×112
pool	2×2	128×112×112
conv×3	3×3/1	128×56×56
pool	2×2	256×56×56
conv×3	3×3/1	256×28×28
pool	2×2	512×28×28
conv×3	3×3/1	512×14×14
pool	2×2	512×14×14
fc	25088×4096	25088
fc	4096×4096	4096

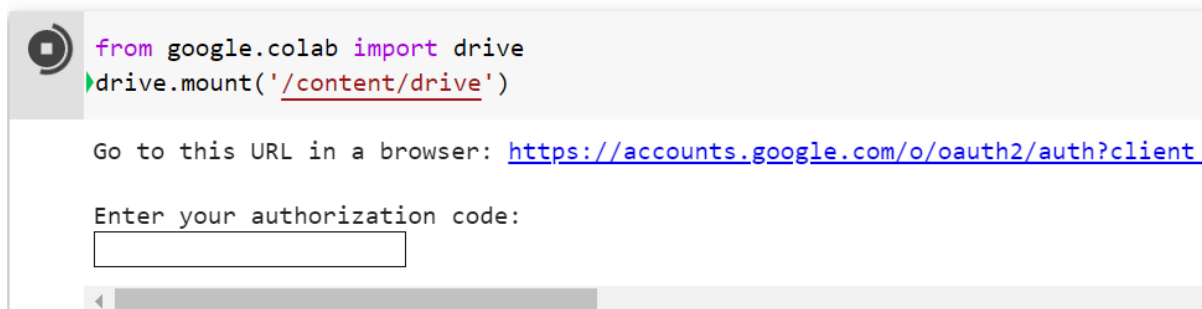
Figura 25: Cuadro resumen de la arquitectura VGG16.

Fuente: International Journal of Scientific and Research Publications (Disponible en: [Transfer learning using VGG-16 with Deep Convolutional Neural Network for Classifying Images \(ijsrp.org\)](https://www.ijsrp.org/)).

La *Figura 25*, nos muestra lo anteriormente mencionado, el tamaño estándar de la imagen es de 224 x 224 px, por lo que debemos ajustar y preprocesar la imagen antes de someterla al proceso de entrenamiento y validación de la red neuronal.

7.7 INTEGRACIÓN DEL ENTORNO DE GOOGLE COLAB

Tal y como se había mencionado en las secciones 5.1 y 5.2 el entorno de trabajo que hemos elegido para desarrollar el algoritmo es *Google Colab* para lo cual se necesita solamente tener una cuenta de *Google* (puede ser *Gmail* o cualquier otro servicio de *Google*). Una de las grandes ventajas de utilizar este servicio es que la integración es sencilla, a continuación, se muestra el proceso para poder usar los *datasets* que hemos almacenado en *Google Drive* en nuestro entorno de desarrollo de *Google Colab*. En la siguiente *Figura 26* se muestra como iniciar el proceso de habilitación de *Google Drive* al entorno de *Colab*.



```
from google.colab import drive
drive.mount('/content/drive')
```

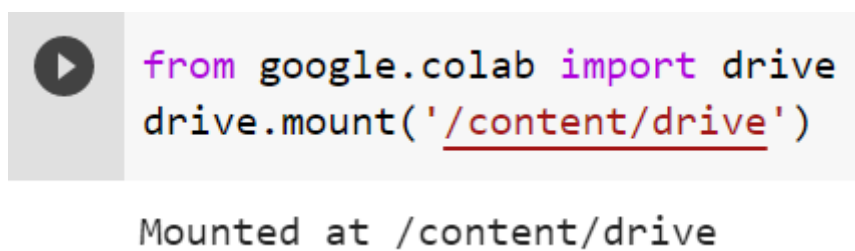
Go to this URL in a browser: <https://accounts.google.com/o/oauth2/auth?client>

Enter your authorization code:

Figura 26: Integración de Google Drive con el entorno de Google Colab.

Fuente: Google Colab (Elaboración propia)

Al introducir el comando de integración que se encuentra documentado en la guía oficial de uso de *Google Colab* (Google Inc., 2018), se habilita una clave de autorización a la que se tiene acceso por medio de un enlace que genera de manera automática, una vez introducida dicha clave, podemos empezar a utilizar el entorno de *Google Colab* con la interfaz de *Jupyter Notebook* (Ver sección 5.3).



```
from google.colab import drive
drive.mount('/content/drive')
```

Mounted at /content/drive

Figura 27: Mensaje de integración satisfactoriamente completado.

Fuente: Google Colab (Elaboración propia)

En la *Figura 27*, observamos como el montaje de *Google Drive* está satisfactoriamente hecho, a partir de este momento podemos utilizar los datasets e información almacenados en nuestro drive, que justamente es dónde tenemos nuestro set de datos almacenados, de manera que tengamos acceso a cada una de las carpetas en donde están guardadas las imágenes que utilizamos como referencia para entrenar nuestra red, estas imágenes vienen en presentaciones de diferentes tamaños y resoluciones, ya que son datasets mixtos que se han recopilado de distintas bases de datos (Ver sección 7.1), por tanto no son imágenes homogéneas, y deberán ser tratadas posteriormente para estandarizar el tamaño y de ésta manera el algoritmo pueda actuar sobre ellas con un patrón en común.

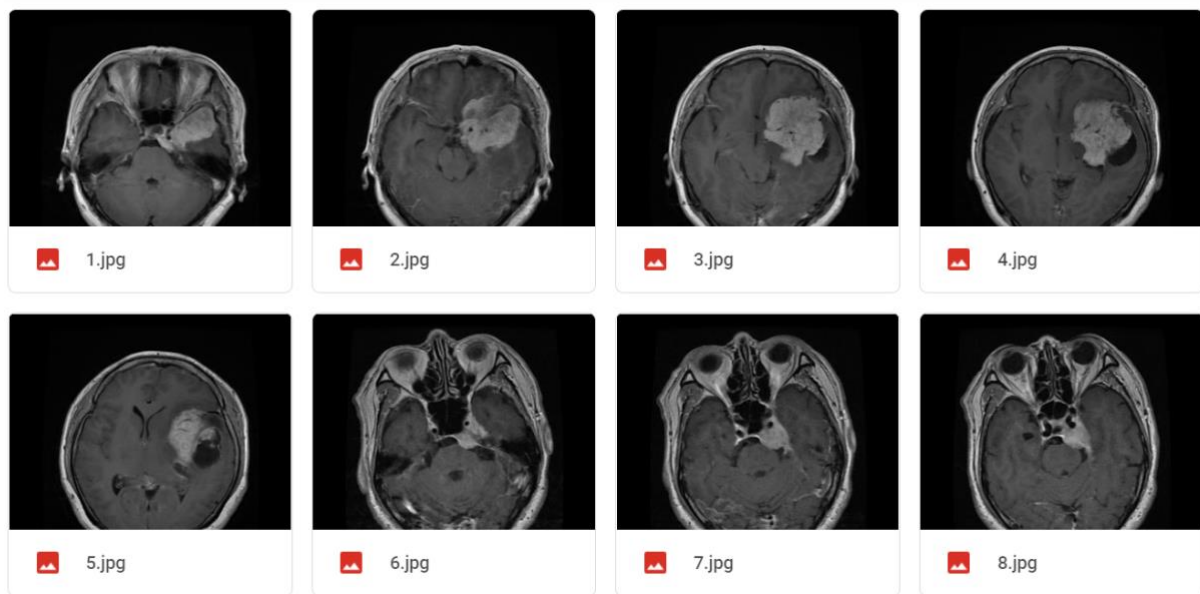


Figura 28: Ejemplo de almacenamiento de datasets para entrenamiento, tipo de tumor Meningioma.

Fuente: Google Colab (Elaboración propia)

En la *Figura 28*, se muestra un ejemplo real de almacenaje de imágenes en una carpeta que justamente es nuestro *dataset*, en este caso particular, cada imagen está en formato *.JPG* debido a que es un formato significativamente ligero de manera que podemos almacenar una cantidad importante de imágenes en nuestro *Google Drive*.

CAPÍTULO VIII: RESULTADOS EXPERIMENTALES

En este capítulo se tratarán los resultados obtenidos durante la ejecución de los algoritmos creados para la clasificación de los diferentes tipos de tumores cerebrales, mismos que se explicaron con detalle en la sección 6, a continuación, se expondrán las principales particularidades del proceso de desarrollo, así como el efecto que se produce en la salida.

8.1 USO DE UNIDAD DE PROCESAMIENTO GRÁFICO (GPU)

En la sección 5.1 se indicó que una de las principales ventajas del uso de *Google Colab*, es que nos provee una GPU de forma gratuita, la cual se usa para aumentar la capacidad de procesamiento gráfico en el tratamiento de imágenes y vídeos, esto representa una importante característica para nuestros efectos de nuestra implementación.

```
+-----+
| NVIDIA-SMI 470.57.02    Driver Version: 460.32.03    CUDA Version: 11.2    |
+-----+
| GPU  Name          Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf   Pwr:Usage/Cap|      Memory-Usage | GPU-Util  Compute M. |
|                                           MIG M.         |
+-----+-----+
|   0   Tesla T4             Off   | 00000000:00:04:0 Off |   0          0      |
| N/A   33C    P8             9W / 70W |  0MiB / 15109MiB |   0%      Default  |
|                                           N/A          |
+-----+-----+

+-----+
| Processes:                                |
| GPU  GI    CI          PID  Type   Process name          GPU Memory |
|   ID  ID  ID                   |              Usage   |
+-----+-----+
| No running processes found              |
+-----+-----+
```

Figura 29: Información del GPU brindada por Google Colab.

Fuente: Google Colab (Elaboración propia)

Tal y como observamos en la *Figura 29*, *Google Colab* está asignando una *GPU Tesla T4 de NVIDIA*, esta unidad se creó para fines de implementación de proyectos de IA ofreciendo un alto rendimiento para en tratamiento de imágenes, videos y todo lo referente al procesamiento gráfico computacional, entendiendo que las exigencias tecnológicas modernas demandan alta capacidad de procesamiento y calidad. Las especificaciones técnicas las podemos consultar en la siguiente imagen (NVIDIA, 2019).

SPECIFICATIONS

GPU Architecture	NVIDIA Turing
NVIDIA Turing Tensor Cores	320
NVIDIA CUDA® Cores	2,560
Single-Precision	8.1 TFLOPS
Mixed-Precision (FP16/FP32)	65 TFLOPS
INT8	130 TOPS
INT4	260 TOPS
GPU Memory	16 GB GDDR6 300 GB/sec
ECC	Yes
Interconnect Bandwidth	32 GB/sec
System Interface	x16 PCIe Gen3
Form Factor	Low-Profile PCIe
Thermal Solution	Passive
Compute APIs	CUDA, NVIDIA TensorRT™, ONNX

Figura 30: Especificaciones técnicas de la GPU NVIDIA T4.

Fuente: NVIDIA Oficial. (Disponible en: <https://www.nvidia.com/en-us/data-center/tesla-t4/>)

Por otro lado, también podemos observar que inicialmente se ha asignado solamente el uso de una GPU, suficiente para poder desarrollar nuestro algoritmo.

```
▶ physical_devices = tf.config.experimental.list_physical_devices('GPU')
print("GPUs Available: ", len(physical_devices))
tf.config.experimental.set_memory_growth(physical_devices[0], True)

GPUs Available: 1
```

Figura 31: Muestreo de GPU disponibles para iniciar con la ejecución de código.

Fuente: Google Colab (Elaboración propia)

Sin embargo, el uso de esta aplicación de *Google* también ofrece otras GPU almacenadas en sus servidores como se puede apreciar en la *Figura 31*, si la sesión con la que se está trabajando se termina o se interrumpe luego de transcurrido cierta cantidad de tiempo, *Google Colab* podría ofrecer una nueva GPU en la siguiente sesión, un ejemplo de ello lo podemos observar en la siguiente imagen.

```

Tue Aug 24 14:27:17 2021
+-----+
| NVIDIA-SMI 470.57.02    Driver Version: 460.32.03    CUDA Version: 11.2    |
+-----+
| GPU  Name          Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf    Pwr:Usage/Cap|      Memory-Usage | GPU-Util  Compute M. |
|                                           MIG M.         |
+-----+-----+
|   0   Tesla K80           Off      | 00000000:00:04:0  Off  |           0         |
| N/A   71C    P8             33W / 149W |      0MiB / 11441MiB |           0%      Default |
|                                           MIG M.         |
|                                           N/A           |
+-----+-----+

Processes:
+-----+
| GPU  GI  CI           PID  Type  Process name          GPU Memory |
| ID   ID  ID                                   Usage     |
+-----+-----+
| No running processes found |
+-----+

```

Figura 32: Asignación de GPU Tesla K80 en nueva sesión de Google Colab.

Fuente: Google Colab (Elaboración propia)

En esta ocasión, *Google Colab* nos ha asignado una GPU modelo Tesla K80 tal y como se puede ver en la *Figura 32*, GPU fabricada por *NVIDIA* al igual que la Tesla T4, ambas muy similares en cuanto a capacidad de procesamiento, las diferencias más importantes radican en su precio ya que la GPU modelo Tesla K80 es mucho más antigua que la T4, por tanto, se construyeron con arquitecturas distintas.

8.2 PARÁMETROS INICIALES

Antes de iniciar con la fase de entrenamiento de la red neuronal, se deben definir algunos parámetros de suma relevancia ya que estos deberán determinar el tratamiento que le daremos a nuestra red neuronal, alguno de los principales parámetros que debemos establecer son:

- Tamaño de la imagen: En esta ocasión estandarizaremos el tamaño de cada una de las imágenes a 224 x 224 píxeles ya que cada una de dichas imágenes que provienen de diversos *datasets* obtenidos en diferentes sitios web (Referirse a la sección 7.1) tienen tamaños y resoluciones diferentes.
- Cantidad de imágenes que se utilizarán en el muestreo: En nuestro caso utilizaremos inicialmente un muestreo o *batch size* de 16. El *batch size* es la cantidad de imágenes sobre las que actuará el algoritmo en cada una de las épocas de aprendizaje y tiene una

correlación muy importante con la tasa de aprendizaje o *learning rate* según estudios como el que se demuestra en “*El efecto del batch size en la generalización de las redes neuronales convolucionales en un dataset de histopatología*” (Ibrahim & Castelli, 2020). En la *Figura 33* se pueden observar los siguientes resultados.

Test AUC		
Batch size	Adam LR = 0.0001	Adam LR = 0.001
16	0.9677	0.9144
32	0.9636	0.9332
64	0.9616	0.9381
128	0.9567	0.9432
256	0.9585	0.9652

Figura 33: Efecto en el rendimiento de aprendizaje utilizando el optimizador ADAM para diferentes batch size.

Fuente: ScienceDirect (Disponible en:

https://www.researchgate.net/publication/337816564_Analysis_of_various_optimizers_on_deep_convolutional_neural_network_model_in_the_application_of_hyperspectral_remote_sensing_image_classification)

- **Clases:** Las pruebas iniciales las realizaremos con 2 clases para poder hacer el ajuste de los parámetros, una vez conseguido resultados satisfactorios procederemos a añadir más clases para corroborar el rendimiento y así poder indagar si el producto es el mismo.

8.3 USO DE LIBRERÍAS

Para el desarrollo de este trabajo se hará uso de las librerías que ofrece *TensorFlow* y *Keras*, esto nos permitirá desarrollar las principales funciones de nuestra aplicación, cada una de estas librerías tiene una competencia especial para desarrollar una u otra operación dentro del desarrollo del código.

```
[2] import numpy as np
import tensorflow as tf

from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.models import load_model

from tensorflow.keras.layers import Activation, Dense, Flatten, BatchNormalization, Conv2D, MaxPool2D, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.metrics import categorical_crossentropy
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.metrics import confusion_matrix

import itertools
import os
import shutil
import random
import glob
import matplotlib.pyplot as plt
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
%matplotlib inline
```

Figura 34: Uso de librerías de TensorFlow y Keras para el desarrollo del sistema.

Fuente: Google Colab (Elaboración propia)

Como muestra la *Figura 34*, podemos ver que se importan las siguientes librerías:

- a) TensorFlow: Esencial para las funciones propias de la red neuronal, sus capas, activación, optimizadores y funciones de graficación.
- b) Numpy: Utilizada para los cálculos algebraicos y numéricos que conllevan las principales operaciones que se desarrollarán.
- c) Itertools: Necesarias para crear interacciones de bucles dentro del ambiente Python.
- d) OS: Que permite hacer integración con el sistema operativo de los ordenadores con los que se desarrolla el sistema.
- e) Shutil: Ofrece funciones de alto nivel para la administración de archivos con los que se trabajará.
- f) Random: Tiene que ver con el manejo numérico y su distribución en las distintas configuraciones de pilas.
- g) Glob: Una librería pequeña pero necesaria para poder buscar y gestionar archivos en el sistema, cuando estos son heterogéneos en cuanto tipo de archivo, extensión, etc.
- h) Matplotlib: Para generación de gráficas a partir de datos obtenidos.
- i) Warning: Para desplegar mensajes de error y advertencia cuando se produce una falla en la ejecución, o bien, cuando existe alguna amenaza de posible daño a algún archivo en particular.

8.4 PRUEBAS INICIALES

Para entrar de lleno a la práctica, hemos elegido un *dataset* pequeño con solamente dos clases para poder determinar el efecto en la implementación de nuestra primera red neuronal, para ello hemos elegido una pequeña porción de datos del set de *Kaggle* (ver sección 7.1.2), en la cual hemos elegido 440 imágenes para entrenamiento, 120 imágenes para validación y 60 para test. Las clases con las que trabajaremos en esta ocasión son Meningioma y No Tumor, en la ejecución del código podemos corroborar que la elección de las imágenes es la correcta, como se muestra en la *Figura 35*.

```
Found 840 images belonging to 2 classes.  
Found 240 images belonging to 2 classes.  
Found 120 images belonging to 2 classes.
```

Figura 35: Total de imágenes elegidas para Entrenamiento, Validación y Prueba, para ambas clases.

Fuente: Google Colab (Elaboración propia)

Efectivamente, comprobamos el total de imágenes en la figura anterior, en donde tenemos *datasets* balanceados por cantidad total de imágenes. La cantidad de imágenes utilizados en las carpetas dedicadas para entrenamiento, validación y test no debe ser necesariamente igual para cada clase, esta es una decisión aleatoria que en nuestro caso la hemos destinado así para que no haya diferencias en las tasas de aprendizaje que más adelante aplicaremos.

Seguidamente, procedemos a aplicar un filtro que viene dado por las librerías de *Keras* y *TensorFlow*, el cual ayuda al algoritmo poder identificar y resaltar mejor las características más sobresalientes de las imágenes de nuestro *dataset*, con el fin único de que nuestra red neuronal pueda hacer una clasificación correcta basado en sus características. La siguiente figura muestra la aplicación de este filtro en cada imagen.

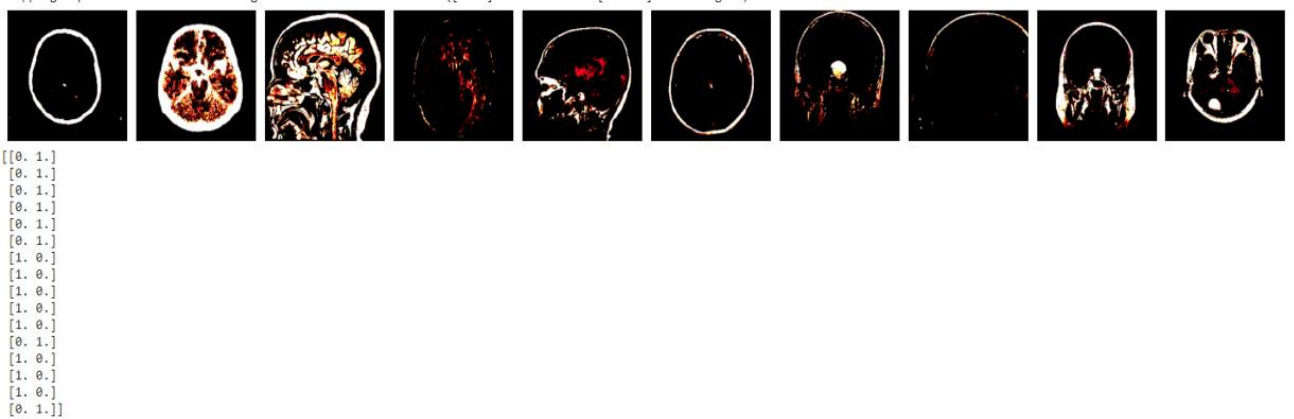


Figura 36: Aplicación de filtro RGB para poder extraer características de la imagen con predicción de clases.

Fuente: Google Colab (Elaboración propia)

La *Figura 36* nos muestra la aplicación del filtro para poder extraer las principales características de la imagen, esto le permite al algoritmo poder reconocer los patrones que a posteriori permitirán hacer una ejecución de aprendizaje más efectiva, es importante señalar que las imágenes antes de este paso venían con resoluciones y tamaños diferentes, al realizar esta acción, homogenizamos la imagen para estandarizar la entrada de nuestra red neuronal.

Por otra parte, los *arrays* que vemos bajo las imágenes, representan a cuál clase pertenece cada una de las imágenes, de la forma:

Meningioma [0. 1.], en clasificación binaria: 01

No Tumor: [1. 0.], en clasificación binaria: 00

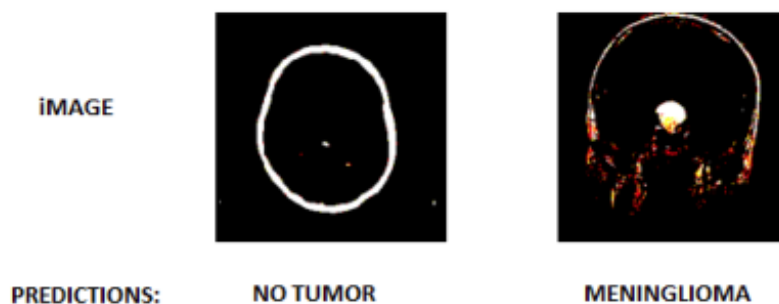


Figura 37: Clasificación de imagen según su clase.

Fuente: Google Colab (Elaboración propia)

En la *Figura 37* podemos detallar mejor como la aplicación del filtro nos permite observar de mejor forma la presencia o ausencia de un cerebral, la imagen de la izquierda claramente ejemplifica un cerebro sano mientras que la imagen de la derecha muestra un cerebro con tumor, a esta acción se le llama pre-procesamiento, y existen diferentes modos y filtros para aplicar en esta etapa, la documentación oficial de *Keras* nos dice que podemos aplicar alguna de las siguientes modalidades: *Grayscale*, *RGB* y *RGBA*, si la imagen va a ser convertida en 1, 3 o 4 canales (Keras Doc, 2015).

En esta ocasión utilizaremos un modelo de elaboración propia, el cual resumimos en el siguiente cuadro.

```
Model: "sequential_1"
```

Layer (type)	Output Shape	Param #
conv2d_10 (Conv2D)	(None, 224, 224, 64)	1792
conv2d_11 (Conv2D)	(None, 224, 224, 64)	36928
max_pooling2d_4 (MaxPooling2D)	(None, 112, 112, 64)	0
conv2d_12 (Conv2D)	(None, 112, 112, 128)	73856
conv2d_13 (Conv2D)	(None, 112, 112, 128)	147584
max_pooling2d_5 (MaxPooling2D)	(None, 56, 56, 128)	0
conv2d_14 (Conv2D)	(None, 56, 56, 256)	295168
conv2d_15 (Conv2D)	(None, 56, 56, 256)	590880
max_pooling2d_6 (MaxPooling2D)	(None, 28, 28, 256)	0
conv2d_16 (Conv2D)	(None, 28, 28, 512)	1180160
conv2d_17 (Conv2D)	(None, 28, 28, 512)	2359808
max_pooling2d_7 (MaxPooling2D)	(None, 14, 14, 512)	0
flatten_1 (Flatten)	(None, 100352)	0
dense_3 (Dense)	(None, 4096)	411045888
dense_4 (Dense)	(None, 4096)	16781312
dense_5 (Dense)	(None, 2)	8194
Total params: 432,520,770		
Trainable params: 432,520,770		
Non-trainable params: 0		

Figura 38: Resumen del modelo creado en la prueba inicial.

Fuente: Google Colab (Elaboración propia)

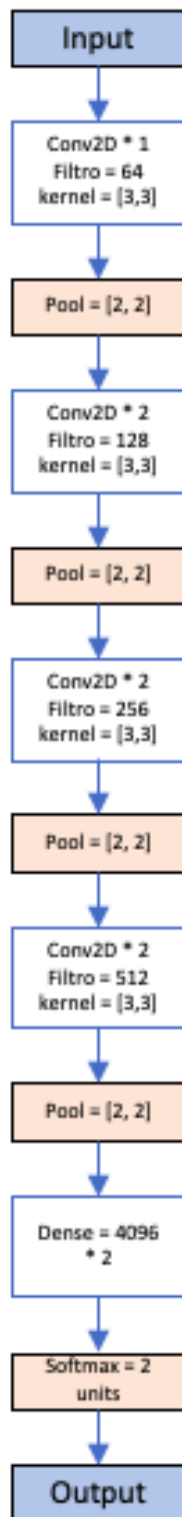


Figura 39: Diagrama de bloques de modelo creado.

Fuente: Elaboración propia.

Tal y como observamos en la *Figura 38* y *Figura 39*, el modelo que hemos creado genera un total de aproximadamente 432 Millones de parámetros entrenables, distribuidos de la forma expresada en la figura 30, donde aplicamos el concepto de Conv2D y Pool que a continuación detallamos:

- Conv2D: Este tipo de capa está disponible en las librerías de *Keras* y *TensorFlow*, crea un vector de sesgo que produce un tensor a la salida de dicha capa, tal y como lo dice la documentación oficial de *Keras*, cuando se utiliza en la primera capa provee un argumento llamado `input_shape` que define la forma de la entrada y los canales.
- MaxPooling2D: Una capa de igual manera bi-dimensional que reduce la muestra de la entrada a lo largo de sus dimensiones espaciales, tomando el valor máximo sobre una ventana de entrada. De igual manera el pooling crea una nueva salida resultante la cual permite la extracción de características de una imagen tal y como se mencionó en la sección 7.5.
- Activación por capas: Cada una de las capas de una red neuronal deben producir una salida que, a su vez, será la entrada de la siguiente capa, las funciones de activación más empleadas en CNN son:
 - a) Sigmoide: Cuyos valores de salida se encuentran entre 1 y 0, y vienen definidas por la ecuación:

Ecuación 3	$\sigma(z) = \frac{1}{1 + e^{-z}}$
------------	------------------------------------

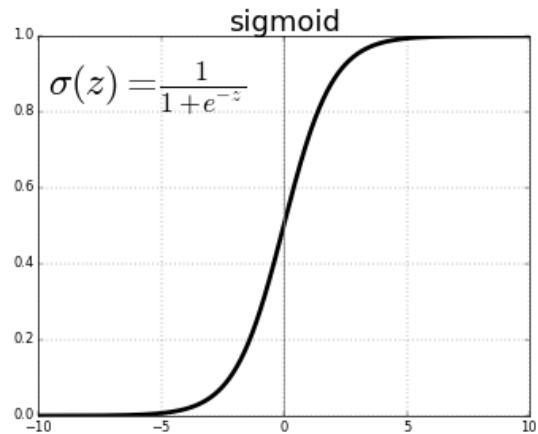


Figura 40: Función de activación sigmoidea*

b) ReLU: Que produce una salida activa positiva ante cualquier entrada positiva.

Ecuación 4	$f(x) = \max(0, x) = \begin{cases} 0 & \text{para } x < 0 \\ x & \text{para } x \geq 0 \end{cases}$
------------	---

* Fuente: ml4a.github.com (Disponible en https://ml4a.github.io/ml4a/es/neural_networks/)

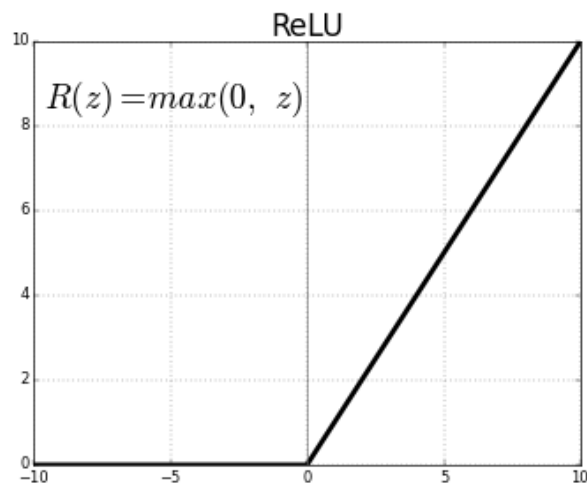


Figura 41: Función de activación ReLU**

c) Softmax: Cuando la suma total de todas las posibles salidas es igual a 1.

Ecuación 5	$f(z)_j = \frac{e^{-x_j}}{\sum_{k=1}^K e^{z_k}}$
------------	--

* Fuente: ml4a.github.com (Disponible en https://ml4a.github.io/ml4a/es/neural_networks/)

** Fuente: ml4a.github.com (Disponible en https://ml4a.github.io/ml4a/es/neural_networks/)

d) *tanh*: comprende valores estrictamente menores a 1 y estrictamente mayores a -1.

Ecuación 6	$f(x) = \frac{2}{1 + e^{-2x}} - 1$
------------	------------------------------------

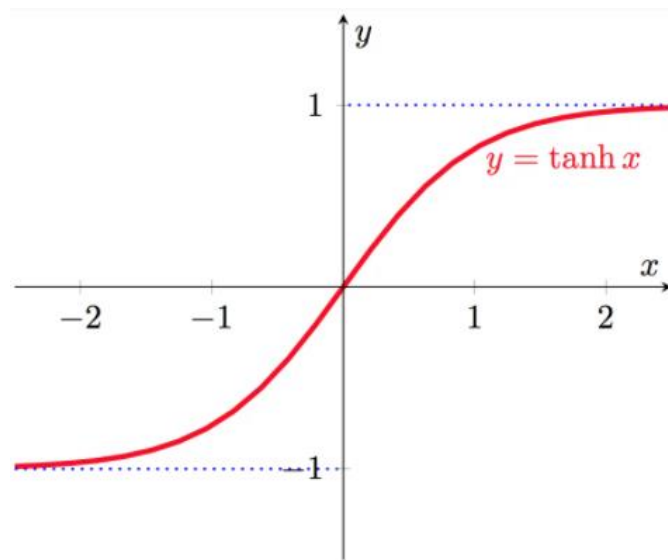


Figura 42: Funcion de activacion tanh.

Fuente: JournalDev (Disponible en: <https://www.journaldev.com/49083/tanh-activation-function>)

La última capa de nuestro modelo de red neuronal tiene una función de activación *softmax* debido a su propiedad de generalización de regresión logística, la cual hará un cálculo de todas las probabilidades de cada clase dentro de un total de posibilidades.

Al proceder a entrenar el sistema nos percatamos que el modelo no es exacto, y tiene un patrón de comportamiento inestable durante el proceso de entrenamiento, la explicación a este fenómeno es el *overfitting*, como hemos señalado anteriormente, el *dataset* que estamos utilizando es muy pequeño y no es suficiente teniendo en cuenta que las imágenes *MRI* (Resonancia Magnética) son muy heterogéneas debido a la gran diversidad de formas, tamaños y aspectos de los tumores cerebrales, y en este caso de los meningiomas.

A continuación, en la *Figura 43* se muestra la exactitud en el proceso de validación y en las pérdidas obtenidas durante el proceso de entrenamiento, para ello hemos elegido 15 épocas.

```

117/117 - 21s - loss: 0.0020 - accuracy: 1.0000 - val_loss: 0.0715 - val_accuracy: 0.9722
Epoch 48/50
117/117 - 21s - loss: 0.0019 - accuracy: 1.0000 - val_loss: 0.0673 - val_accuracy: 0.9741
Epoch 49/50
117/117 - 21s - loss: 0.0018 - accuracy: 1.0000 - val_loss: 0.0664 - val_accuracy: 0.9741
Epoch 50/50
117/117 - 21s - loss: 0.0016 - accuracy: 1.0000 - val_loss: 0.0561 - val_accuracy: 0.9815

```

Figura 43: Exactitud en la validación y comportamiento de las pérdidas durante el proceso de entrenamiento.

Fuente: Google Colab (Elaboración propia)

La exactitud en el proceso de validación es bastante aceptable, sin embargo, tal y como observamos, no es el comportamiento que esperamos, ya que suponemos una exactitud ascendente cada vez que superamos una época de entrenamiento.

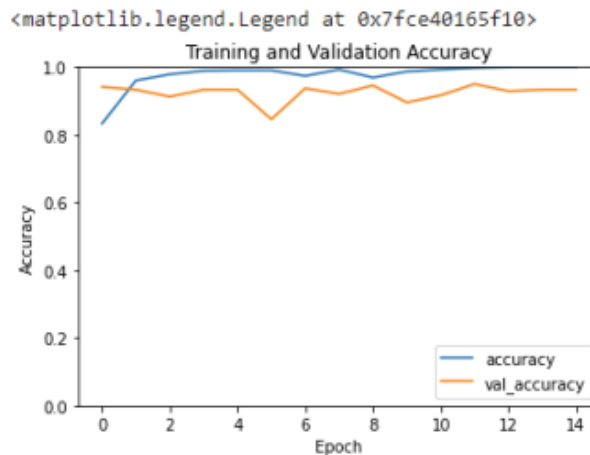


Figura 44: Gráfica de precisión en proceso de entrenamiento y validación.

Fuente: Google Colab (Elaboración propia)

La Figura 44, nos muestra una muy buena actuación del algoritmo durante el proceso de entrenamiento, llegando a alcanzar el 100% en la exactitud, caso similar lo tenemos durante el proceso de validación en donde se obtienen resultados cercanos a 94% de precisión, nada mal tomando en cuenta que estamos trabajando con un *dataset* relativamente pequeño.

Por otro lado, tenemos la gráfica que representa las pérdidas para el entrenamiento y validación en la Figura 45, tal como y podemos observar en la siguiente figura, el comportamiento durante el entrenamiento es el esperado, durante las primeras épocas la pérdida es claramente grande, sin embargo, a medida que avanzan las épocas, las pérdidas van disminuyendo, hasta alcanzar valores cercanos a cero.

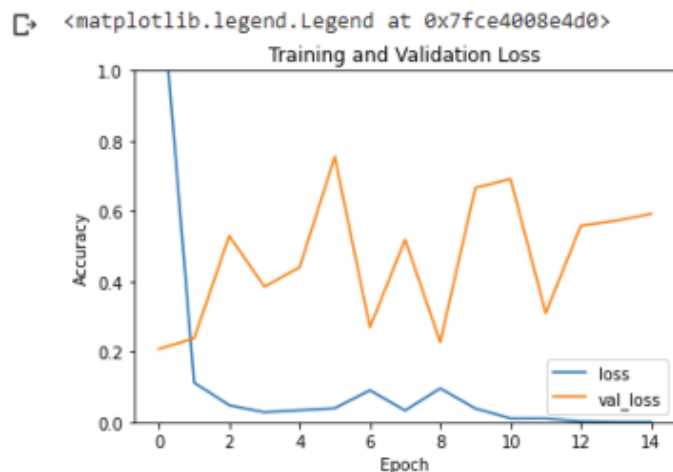


Figura 45: Gráfica de pérdidas durante el proceso de entrenamiento y validación.

Fuente: Google Colab (Elaboración propia)

Una de las funciones que nos ofrece la librería de *Keras* y *Tensorflow* es la llamada *matriz de confusión*, que nos permite ver más claramente los resultados obtenidos de una manera más visual, en nuestro caso, aplicamos esta función sobre nuestro *dataset* de Test. Por medio de la *matriz de confusión* nos podemos percatar de la cantidad de imágenes que detecta de manera correcta y de manera falsa.

La interpretación más adecuada que podríamos darle a la *matriz de confusión* un arreglo lineal de todas las posibilidades que puede tener una imagen para ser correcta o erróneamente clasificada.

Matriz de confusión para una variable de salida dicotómica (clasificación)		Resultado de la predicción	
		Sí	No
Valor real de la clase	Sí	Verdadero positivo	Falso Negativo
	No	Falso positivo	Verdadero negativo

Figura 46: Arreglo de una matriz de confusión

Fuente: MicroMedix (Disponible en: <https://micromedix.me/2018/01/27/cardiopatias-en-las-cosas-del-corazon-no-basta-con-una-segunda-opinion/matriz-de-confusion/>)

En la *Figura 46*, podemos ver el arreglo de una matriz de confusión, en donde el cuadrante superior izquierdo y el inferior derecho nos muestran las predicciones acertadas, ya sea verdadero positivo o verdadero negativo para un valor real perteneciente a una clase, mientras que los cuadrantes superior derecho e inferior izquierdo nos muestra las predicciones no acertadas por el sistema, sean estas falso positivo o falso negativo. Esta ayuda visual es muy útil ya que facilita la interpretación de las predicciones de las imágenes de Test, además que nos ayuda a conocer de una forma sencilla y directa si el sistema es exacto o impreciso.

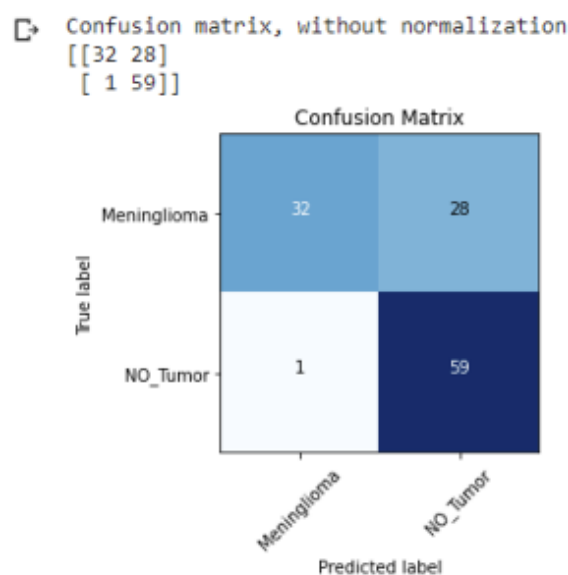


Figura 47: Matriz de confusión para las clases de Meninglioma y NO_Tumor.

Fuente: Google Colab (Elaboración propia)

La *Figura 47*, muestra las predicciones hechas de nuestra red neuronal sobre la clase NO_Tumor, éstas son muy precisas ya que solamente el 1.66% de las predicciones no fue acertada, solo una imagen no la consigue clasificar bien, ya que ocurren fenómenos que pueden ser falso positivos o falso negativos, sin embargo, las predicciones hechas para la clase Meninglioma son claramente inferiores en cuanto a la calidad de la predicción, de hecho, el 46.66% de las predicciones estuvieron erróneas lo que representa prácticamente la mitad del total de las imágenes para Test, esto genera una enorme imprecisión en la salida, haciendo que la aplicación son muy poco confiable.

8.5 PROCESO MEJORADO DE ENTRENAMIENTO

Tal y como lo hemos apreciado, el hecho de trabajar *datasets* pequeños suele producir *underfitting* durante el proceso de entrenamiento y validación, por ende, los resultados están propensos a verse afectados en la salida ya que el sistema no es capaz de “aprender” de forma efectiva ya sea en una o en varias clases, por tanto, cuando introducimos una nueva imagen a la red neuronal para que pueda clasificarla basado en las predicciones verdaderas, los resultados son mediana o altamente imprecisos. Por esta razón, ahora introduciremos un set de datos más grande con las mismas clases del ejemplo anterior (Ver sección 8.3) y veremos el efecto.

La arquitectura de nuestra red es en esencia la misma que hemos aplicado en la sección 8.4, y que hemos implementado para extracción de características en cada una de sus capas, a continuación, la *Figura 49* nos muestra cada una de las capas y bloques que forman nuestra red neuronal, este formato se encuentra disponible en la librería de Keras y la obtenemos por medio de la librería de matplotlib como se muestra en la *Figura 48*.

```
[ ] from keras.utils.vis_utils import plot_model
    plot_model(model, to_file='model_plot.png', show_shapes=True, show_layer_names=True)
```

Figura 48: Función de matplotlib para obtener modelo de red neuronal por bloques.

Fuente: Google Colab (Elaboración propia)

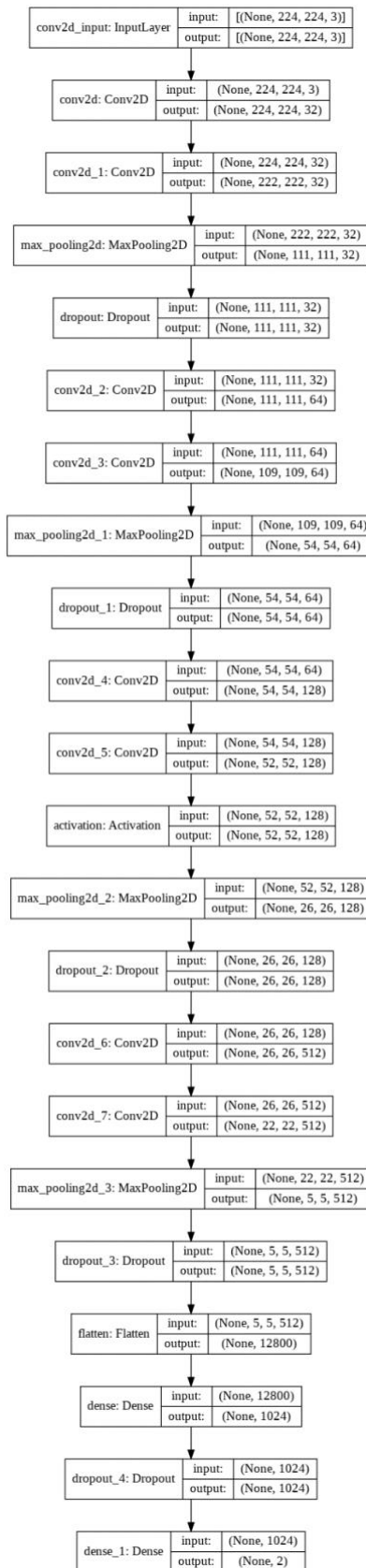


Figura 49

Fuente: Google Colab (Elaboración propia)

Otro aspecto para tomar en cuenta es que aplicaremos un ajuste en los *datasets*, ya que en el ejemplo anterior se utilizó un *dataset* pequeño (Ver *Figura 35*) en dónde los resultados fueron muy deficientes. En esta ocasión utilizaremos un *dataset* más grande, y de igual manera haremos una clasificación de imágenes para eliminar aquellas con baja resolución, ya que pueden producir un proceso de entrenamiento y aprendizaje deficiente, que es justamente lo que se debe evitar.

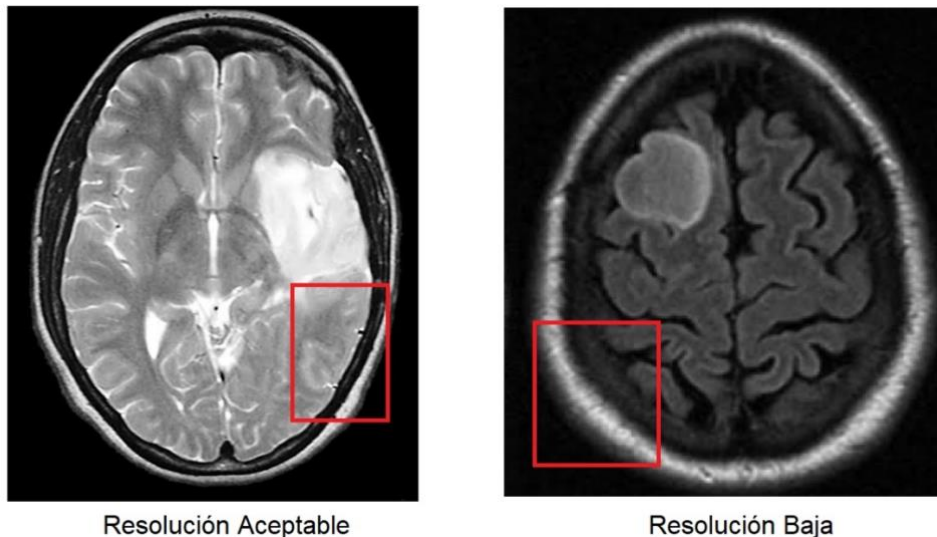


Figura 50: MRI de Gliomas con distintas resoluciones.

Fuente: Figshare Dataset (Disponible en: https://figshare.com/articles/dataset/brain_tumor_dataset/1512427)

La *Figura 48*, nos muestra las diferencias de resolución que podemos encontrar en el *dataset*, la calidad en ambas imágenes es notablemente desigual y cuando introducimos estas imágenes a nuestra red neuronal para que inicie el proceso de entrenamiento y validación, producimos un deficiente aprendizaje de la red neuronal, haciendo que los resultados tengan una confiabilidad muy baja y por ende un rendimiento insuficiente.

La imagen de la izquierda es evidentemente muy superior en cuanto a calidad y resolución, y en su contraparte, la imagen de la derecha es lo opuesto, si observamos detalladamente la sección del cuadro rojo, vemos una difuminación pronunciada en los márgenes perimetrales que pueden provocar confusiones en el proceso de extracción de características. Por otro lado, se debe tomar en cuenta que el formato que estamos utilizando es el formato JPG.

Utilizar este tipo de formato tiene la ventaja que son archivos livianos, por tanto, son más fáciles de manipular y almacenar en disco o en la nube, pero tienen la gran desventaja de tener una baja resolución, y a pesar de que las redes neuronales se diseñan tomando en cuenta estas variables, siempre existe el riesgo de producir *under-fitting* o entrenamiento deficiente debido a la calidad de imágenes. Una vez que se han depurado los *datasets* eliminando las imágenes con mala resolución, procedemos a organizar de nuevo las carpetas para Entrenamiento, Validación y Test, y ejecutamos el código.

Otro aspecto que se debe mencionar es que antes de iniciar el proceso de entrenamiento vamos a realizar el pre-procesado de las imágenes para que el sistema pueda entrenar la red de una mejor forma y así tener mayor facilidad para poder extraer sus características.

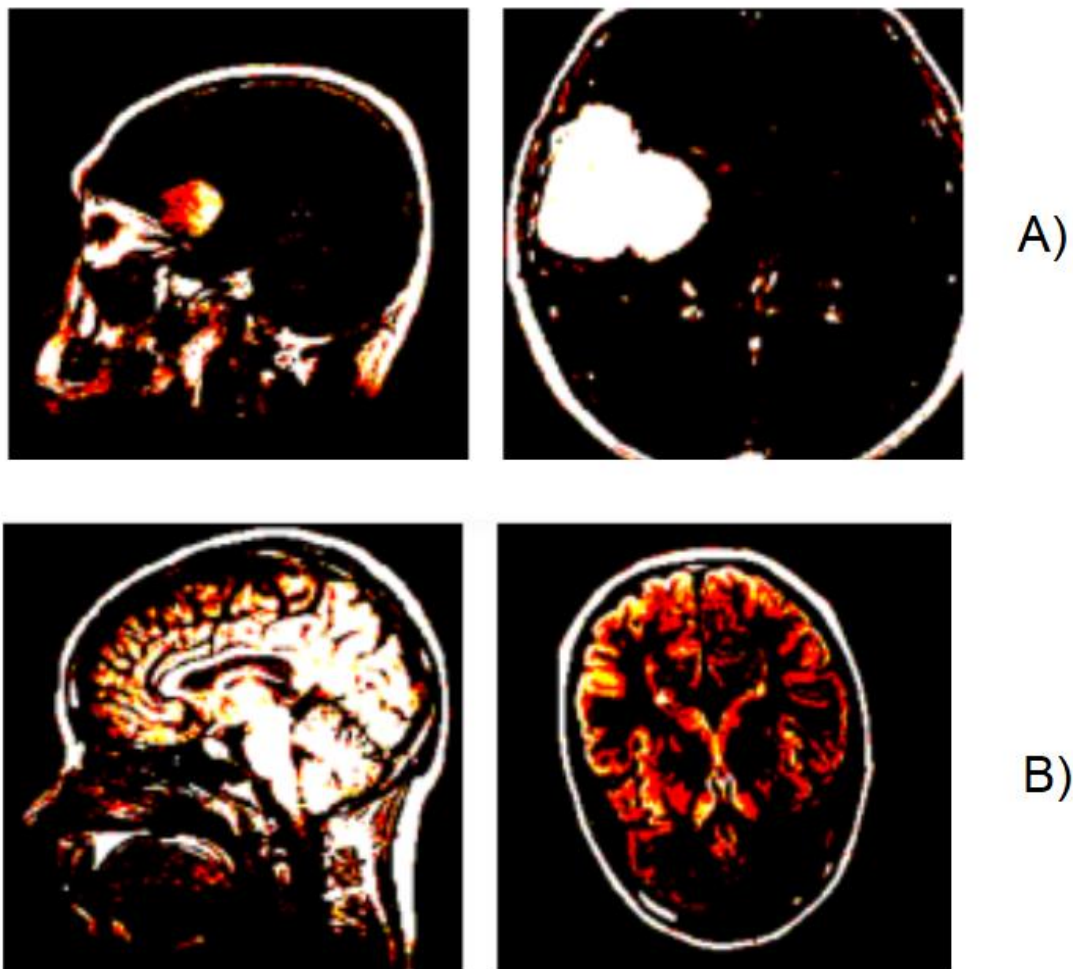


Figura 51: Imágenes pre-procesadas antes de iniciar el entrenamiento de la red neuronal. A) Cerebro con tumor en vista lateral y frontal. B) Cerebro sin presencia de tumores en vista lateral y frontal.

Fuente: Google Colab (Elaboración propia)

La *Figura 52*, nos muestra la cantidad de imágenes que utilizaremos en esta siguiente prueba, las clases son las mismas con las que estuvimos trabajando en la prueba anterior, Meningioma y NO_Tumor. En esta ocasión utilizaremos 1860 imágenes para Entrenamiento (930 por cada clase), 540 imágenes para Validación (270 por cada clase) 260 imágenes para Test (130 por cada clase).

```
Found 1860 images belonging to 2 classes.  
Found 540 images belonging to 2 classes.  
Found 260 images belonging to 2 classes.
```

Figura 52: Experimentación con nuevo dataset.

Fuente: Google Colab (Elaboración propia)

Otra técnica que aplicaremos es el *data augmentation* del cual hablaremos más adelante, por ahora, solo diremos que permite la modificación de imágenes de su forma original a una nueva configuración modificada, en la que podemos girarla, ampliarla o disminuirla, rotar el sentido original, entre otras opciones más con el fin de poder ampliar la cantidad de imágenes en el *dataset* para realizar el entrenamiento de nuestra red neuronal. A continuación, podremos observar el proceso de entrenamiento y validación al entrenar nuestra red neuronal con estas nuevas condiciones, de igual forma hemos aumentados el número de épocas para apreciar mejorar las gráficas de pérdida y aprendizaje, la *Figura 53* nos muestra dichos resultados.

```

Epoch 2/30
117/117 - 27s - loss: 0.4335 - accuracy: 0.8226 - val_loss: 0.5794 - val_accuracy: 0.6481
Epoch 3/30
117/117 - 27s - loss: 0.3271 - accuracy: 0.8720 - val_loss: 0.2963 - val_accuracy: 0.8778
Epoch 4/30
117/117 - 27s - loss: 0.2261 - accuracy: 0.9194 - val_loss: 0.2344 - val_accuracy: 0.9222
Epoch 5/30
117/117 - 27s - loss: 0.1595 - accuracy: 0.9435 - val_loss: 0.2100 - val_accuracy: 0.9241
Epoch 6/30
117/117 - 27s - loss: 0.1433 - accuracy: 0.9452 - val_loss: 0.2957 - val_accuracy: 0.8704
Epoch 7/30
117/117 - 27s - loss: 0.1047 - accuracy: 0.9656 - val_loss: 0.3452 - val_accuracy: 0.8741
Epoch 8/30
117/117 - 27s - loss: 0.0534 - accuracy: 0.9855 - val_loss: 0.3685 - val_accuracy: 0.9185
Epoch 9/30
117/117 - 27s - loss: 0.0681 - accuracy: 0.9796 - val_loss: 0.2661 - val_accuracy: 0.9148
Epoch 10/30
117/117 - 27s - loss: 0.0650 - accuracy: 0.9790 - val_loss: 0.3440 - val_accuracy: 0.9093
Epoch 11/30
117/117 - 27s - loss: 0.0521 - accuracy: 0.9833 - val_loss: 0.2053 - val_accuracy: 0.9259
Epoch 12/30
117/117 - 27s - loss: 0.0234 - accuracy: 0.9925 - val_loss: 0.1645 - val_accuracy: 0.9481
Epoch 13/30
117/117 - 27s - loss: 0.0255 - accuracy: 0.9919 - val_loss: 0.3113 - val_accuracy: 0.9241
Epoch 14/30
117/117 - 27s - loss: 0.0462 - accuracy: 0.9833 - val_loss: 0.7274 - val_accuracy: 0.8685
Epoch 15/30
117/117 - 27s - loss: 0.0687 - accuracy: 0.9774 - val_loss: 0.1751 - val_accuracy: 0.9389
Epoch 16/30
117/117 - 27s - loss: 0.0204 - accuracy: 0.9935 - val_loss: 0.3382 - val_accuracy: 0.9333
Epoch 17/30
117/117 - 27s - loss: 0.0311 - accuracy: 0.9914 - val_loss: 0.1647 - val_accuracy: 0.9519
Epoch 18/30
117/117 - 27s - loss: 0.0159 - accuracy: 0.9962 - val_loss: 0.2137 - val_accuracy: 0.9352
Epoch 19/30
117/117 - 27s - loss: 0.0118 - accuracy: 0.9941 - val_loss: 0.2748 - val_accuracy: 0.9426
Epoch 20/30
117/117 - 27s - loss: 0.0068 - accuracy: 0.9968 - val_loss: 0.5863 - val_accuracy: 0.8889
Epoch 21/30
117/117 - 27s - loss: 0.0359 - accuracy: 0.9882 - val_loss: 0.3489 - val_accuracy: 0.9333
Epoch 22/30
117/117 - 27s - loss: 0.0063 - accuracy: 0.9984 - val_loss: 0.3994 - val_accuracy: 0.9278
Epoch 23/30
117/117 - 27s - loss: 0.0105 - accuracy: 0.9957 - val_loss: 0.5951 - val_accuracy: 0.9074
Epoch 24/30
117/117 - 27s - loss: 0.0152 - accuracy: 0.9941 - val_loss: 0.4436 - val_accuracy: 0.9204
Epoch 25/30
117/117 - 27s - loss: 0.0293 - accuracy: 0.9903 - val_loss: 0.1770 - val_accuracy: 0.9426
Epoch 26/30
117/117 - 27s - loss: 0.0130 - accuracy: 0.9952 - val_loss: 0.1697 - val_accuracy: 0.9648
Epoch 27/30
117/117 - 27s - loss: 0.0098 - accuracy: 0.9973 - val_loss: 0.2219 - val_accuracy: 0.9463
Epoch 28/30
117/117 - 27s - loss: 0.0035 - accuracy: 0.9989 - val_loss: 0.2932 - val_accuracy: 0.9519
Epoch 29/30
117/117 - 27s - loss: 0.0074 - accuracy: 0.9968 - val_loss: 0.2596 - val_accuracy: 0.9259
Epoch 30/30
117/117 - 27s - loss: 0.0890 - accuracy: 0.9747 - val_loss: 0.1904 - val_accuracy: 0.9333

```

Figura 53: Resumen del modelo creado en la segunda prueba con dataset más grande.

Fuente: Google Colab (Elaboración propia)


```
↳ Confusion matrix, without normalization
[[130  0]
 [ 3 127]]
```

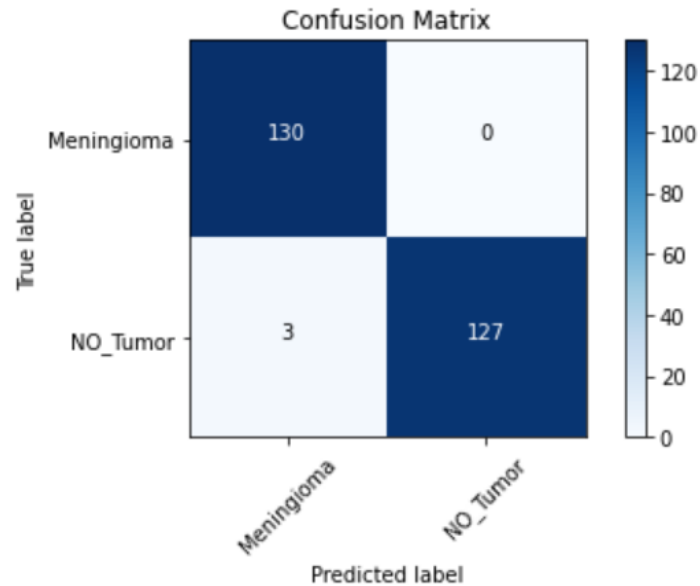


Figura 56: Matriz de confusión para las clases de Meningioma y NO_Tumor con nuevo dataset.

Fuente: Google Colab (Elaboración propia).

Tal y como se muestra en la *Figura 54*, la exactitud en la predicción realizada sobre la clase **Meningioma** es 100% precisa, mientras que para **NO_Tumor** es de 2,3%, es decir, tomando en cuenta la naturaleza de las imágenes con las que se está trabajando, así como las posibles deficiencias en cuanto a resolución y calidad de estas en el *dataset*, podemos asegurar que tenemos un sistema altamente confiable.

Las gráficas de pérdidas y exactitud en los procesos de entrenamiento y validación demuestran que se obtuvieron resultados muy satisfactorios, lo cual indica que el aumento en la cantidad de imágenes del *dataset* juega un papel esencial en nuestra red neuronal e impacta directamente la calidad en las predicciones y resultados.

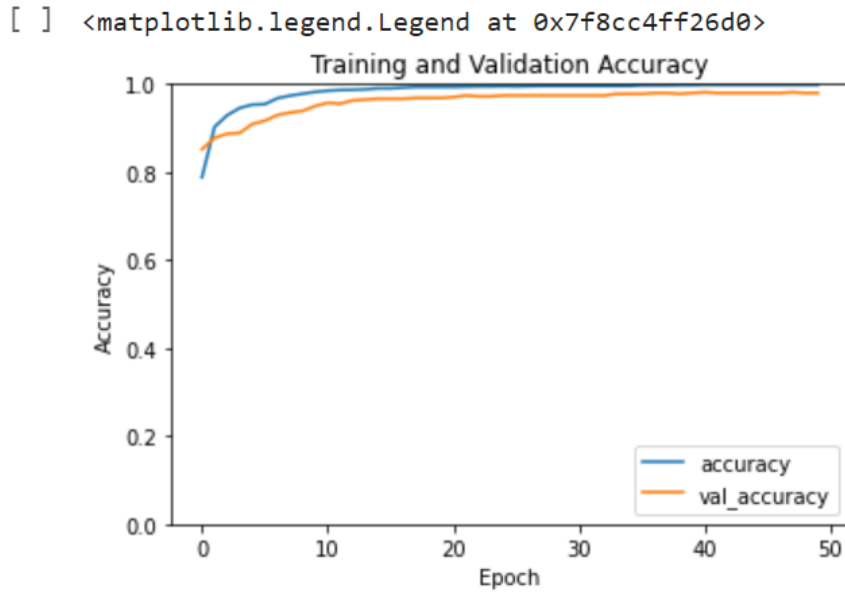


Figura 57: Gráfica de precisión de los procesos entrenamiento y validación.

Fuente: Google Colab (Elaboración propia).

La *Figura 57*, nos muestra la forma en la que la red neuronal aprende progresivamente en cada interacción o época, la salida es justamente la que se espera ya que tras cada época la tasa de aprendizaje se hace más precisa y la pérdida se hace menor, mientras que la *Figura 58* nos muestra las pérdidas para el entrenamiento y validación, las cuales se hacen más pequeñas tras cada interacción, esto demuestra claramente que la red neuronal está ajustada correctamente y que el aprendizaje se ha hecho de manera correcta, la diferencia física entre las imágenes de las dos clases con las que estamos trabajando hace aún más exacta la salida del sistema, ya que hay una importante diferencia entre un cerebro con tumor a uno que no lo tiene.

Sin embargo, estos resultados no pueden tomarse en cuenta como parámetro real para las predicciones, porque la red está entrenada con una base de datos de pocas imágenes, esto genera una predicción poco confiable ya que posiblemente exista *under-fitting* y el efecto en la salida puede estar sesgado.

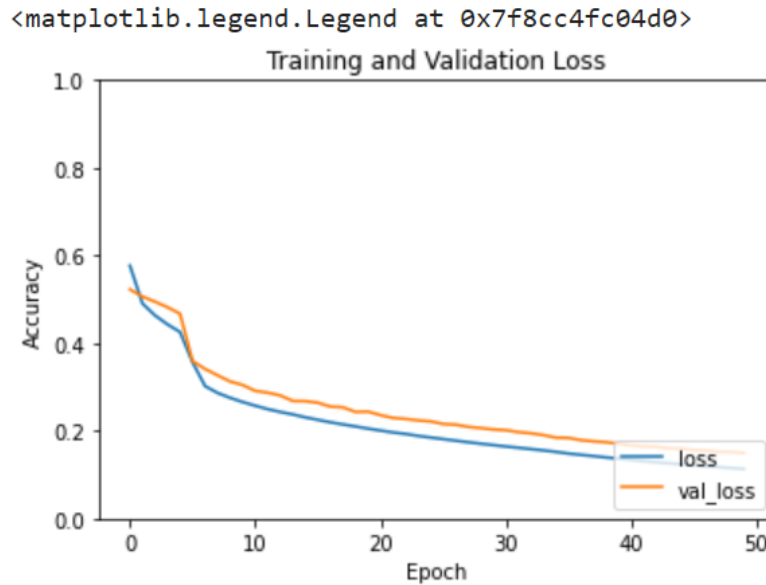


Figura 58: Gráfica de pérdidas de los procesos entrenamiento y validación.

Fuente: Google Colab (Elaboración propia).

8.6 DATA AUGMENTATION

Como se ha mencionado anteriormente la técnica del “*data augmentation*” ha venido a brindar una importante ayuda a quienes trabajan con modelos neuronales, ya que muchas veces los *datasets* suelen ser el principal problema o limitante para construir una red neuronal precisa, funcional y exacta. Sin embargo, se debe ser sumamente cuidadoso con las modificaciones que se realicen sobre las imágenes, ya que se deben preservar las características esenciales de la imagen sin llegar a distorsionarlas al punto que provoquemos el efecto inverso, es decir, un over-fitting, lo cual es precisamente lo que se quiere evitar (Wong, Gatt, Stamatescu, & McDonnell, 2016).

El *data augmentation* nos permite no solo aumentar la cantidad de imágenes en un *dataset*, sino que nos facilita modificar la imagen orgánicamente dándonos la oportunidad de modificar muchas variables para evitar llegar a confundir el proceso de entrenamiento, algunas de las principales características que nos permite modificar esta función son:

- **rotation_range:** Permite girar la imagen hacia la derecha o izquierda sobre su centro.
- **width_shift_range:** Esta variable habilita el rango de desplazamiento del ancho de la imagen.
- **height_shift_range:** Esta variable habilita el rango de desplazamiento de la altura de la imagen.
- **shear_range:** Esta opción nos permite modificar el ángulo (en grados) de corte en sentido antihorario.
- **zoom_range:** Variable que nos permite hacer un acercamiento a la imagen.
- **channel_shift_rango:** Es un rango para cambios de canal de color aleatorio.
- **horizontal_flip:** Para girar horizontalmente de forma aleatoriamente la imagen.
- **validation_split:** Para girar verticalmente de forma aleatoriamente la imagen.

Es importante destacar que la función de *data augmentation* la hemos aplicado sobre las imágenes designadas para entrenamiento, nunca debe aplicarse sobre las imágenes asignadas para Test, a continuación, se muestra la configuración que hemos elegido para nuestra red en la *Figura 59*:

```
train_generator = ImageDataGenerator(rotation_range=15,
                                     width_shift_range=0.1,
                                     height_shift_range=0.1,
                                     shear_range=0.20,
                                     zoom_range=0.1,
                                     channel_shift_range=10,
                                     horizontal_flip=True,
                                     preprocessing_function=preprocess_input)
```

Figura 59: Configuración de la función de data augmentation de nuestra cnn.

Fuente: Google Colab (Elaboración propia).

Al aplicar el *data augmentation* obtendremos una salida como la que vemos en la *Figura 60*, en dónde utilizamos la imagen original (marcada en el cuadro rojo) y modificamos los parámetros que vimos en la *Figura 59* para conseguir la salida que no es otra cosa más que la imagen original con cambios en su posición, zoom, corte, etc.

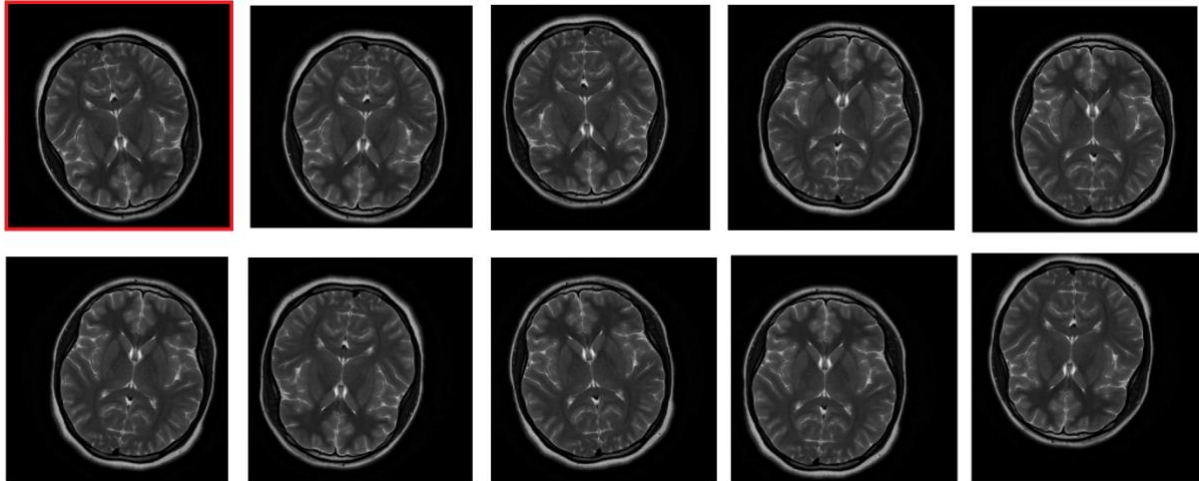


Figura 60: Resultado de aplicar el data augmentation sobre una imagen.

Fuente: Google Colab (Elaboración propia).

8.7 USO DE TRANSFER LEARNING

En la Sección 7.4 se trató el tema del Transfer Learning, en donde se asegura que esta técnica se aplica en escenarios de aprendizaje automático supervisado en donde se intenta utilizar el conocimiento previo de una red neuronal que ha sido entrenada con miles de clases e imágenes, demostrando una exactitud y un rendimiento altamente preciso, lo cual nos permite ahorrar una cantidad considerable de tiempo en el proceso de entrenamiento de nuestra cnn.

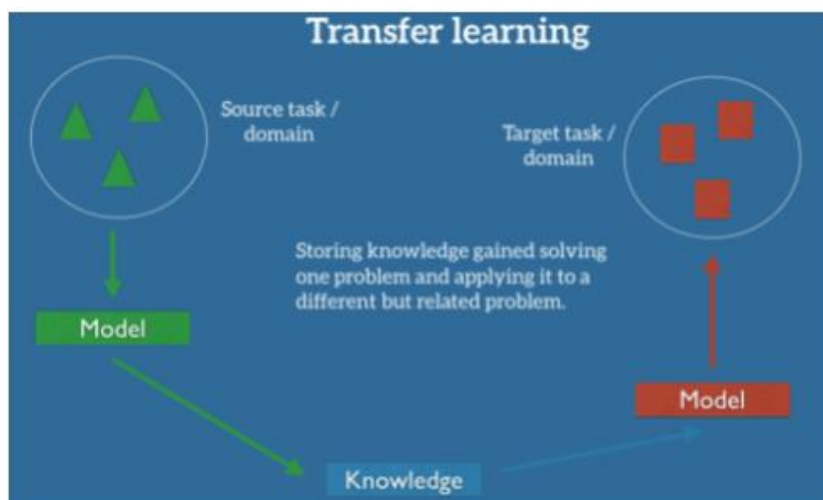


Figura 61: Modelo de Transfer Learning aplicado a nuevo modelo.

Fuente: ReDer.io (Disponible en: <https://ruder.io/transfer-learning/index.html#whatistransferlearning>)

La Figura 61 demuestra lo anteriormente dicho, en donde el conocimiento conseguido previamente aplicado en otro modelo es aprovechado en nuevo modelo, y en donde las clases, datasets y tipo de problema son completamente distintos, sin embargo, los pesos de las capas en la red neuronal se almacenas para reproducirlos en un nuevo problema.

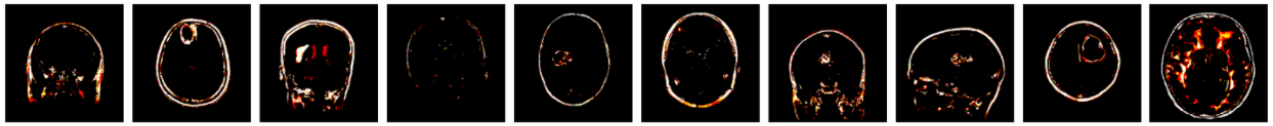
En esta ocasión, utilizaremos la red llamada VGG16 que se explicó en la Sección 7.6, en donde añadiremos las clases de *Gioma* y *Pituitario* que se discutieron en el Capítulo 6, para ello hemos elegido la misma cantidad de imágenes para Entrenamiento, Validación y Test, la Figura 62 nos muestra la distribución del datasets según la cantidad de clases:

```
Found 3720 images belonging to 4 classes.
Found 1080 images belonging to 4 classes.
Found 520 images belonging to 4 classes.
```

Figura 62: Dataset de 4 clases para implementación con Transfer Learning.

- Fuente: Google Colab (Elaboración propia).

- **Entrenamiento:** 3720 imágenes (930 por cada clase)
- **Validación:** 1080 imágenes (270 por cada clase)
- **Test:** 520 imágenes (130 por cada clase)



```

[[0. 0. 1. 0.]
 [1. 0. 0. 0.]
 [0. 0. 0. 1.]
 [0. 0. 0. 1.]
 [0. 0. 0. 1.]
 [0. 0. 1. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]
 [0. 0. 0. 1.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 1. 0. 0.]
 [0. 1. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 0. 1.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]

```

Figura 63: Etiquetado por clases para Transfer Learning

Fuente: Google Colab (Elaboración propia).

En la Figura 63, vemos las etiquetas que se asignan a cada clase, en esta ocasión tenemos 4 posibilidades en la predicción de la siguiente forma:

- **0.0.0.1:** Meningioma
- **0.0.1.0:** NO_Tumor
- **0.1.0.0:** Pituitary
- **1.0.0.0:** Glioma

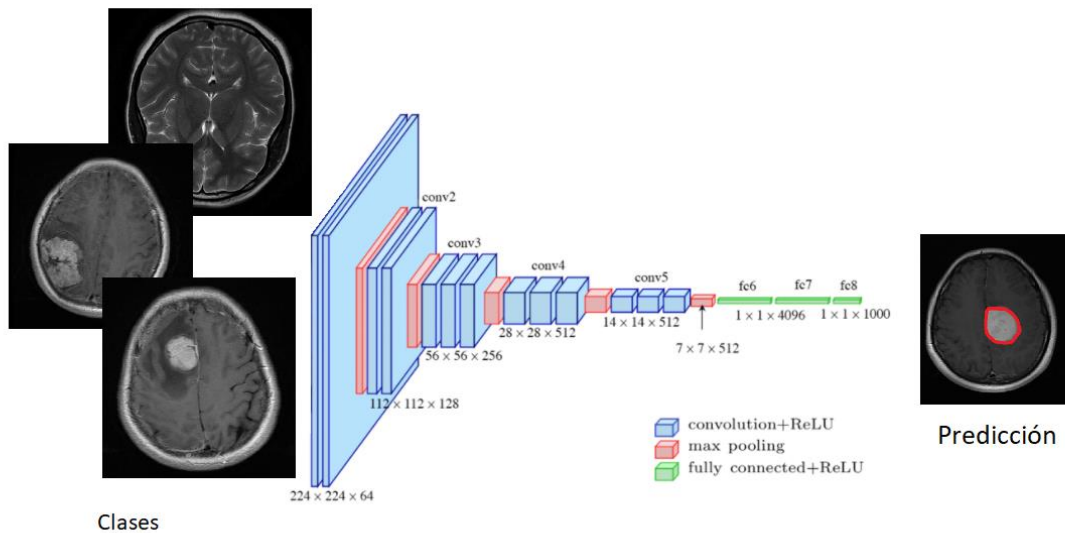


Figura 64: Modelo VGG utilizado para Transfer Learning.

Fuente: Elaboración propia

La *Figura 64* nos muestra el modelo que implementaremos en esta ocasión, el cual se trata de la red neuronal VGG16, la cual recibe en la entrada las diferentes imágenes de 4 clases diferentes, pasan por la capa de entrada en un formato ya ajustado a la forma (224, 224, 3) para estandarizar su tamaño, luego se aplica un max pooling para dar paso a la siguiente capa, en la que se aplica un filtro de (112, 112, 128), después de pasar por dos capas de convolución, se le aplica un nuevo filtro *max pooling* para entrar en la siguiente capa de dimensiones (56, 56, 256). Posteriormente, luego de aplicar un filtro (de tamaño 28, 28, 512) entran en acción 3 capas subsecuentes de convolución y activación ReLU. Seguidamente, aparecen las capas de (14, 14, 512) luego de que se aplica el filtro *max pooling*. Finalmente, se da paso a las 3 últimas capas densas de dimensión (1, 1, 4096), luego de pasar por un filtro de (7, 7, 512), estas tres últimas capas reducen la dimensionalidad a 1 debido a que este punto es donde se procede a la extracción de las características de la imagen, gracias a esto podemos realizar una clasificación apropiada de clases.

En esta ocasión, al utilizar la técnica de Transfer Learning, originalmente descargamos el modelo que se encuentra disponible en el siguiente enlace:

https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels.h5

una vez descargado el modelo veremos que el modelo cuenta con 3 capas tipo Dense como lo muestra la *Figura 65*, dicho modelo utiliza la última capa para predicciones, sin embargo, se encuentra abierta para ajustar la cantidad de clases de nuestro modelo.

fc1 (Dense)	(None, 4096)	102764544
fc2 (Dense)	(None, 4096)	16781312
predictions (Dense)	(None, 1000)	4097000
=====		
Total params: 138,357,544		
Trainable params: 138,357,544		
Non-trainable params: 0		

Figura 65: Últimas 3 capas del modelo VGG16

Fuente: Google Colab (Elaboración propia).

El total parámetros que se pueden entrenar son aproximadamente 140 millones de parámetros, así que ajustaremos la última capa de salida para que reconozca las 4 clases de nuestra CNN.

fc1 (Dense)	(None, 4096)	102764544
fc2 (Dense)	(None, 4096)	16781312
dense (Dense)	(None, 4)	16388
=====		
Total params: 134,276,932		
Trainable params: 16,388		
Non-trainable params: 134,260,544		

Figura 66: Últimas 3 capas del modelo VGG16 con ajuste de la capa de salida.

Fuente: Google Colab (Elaboración propia).

Ahora tenemos solamente 16 mil parámetros “entrenables” ya que hemos limitado el sistema a solamente las 4 clases de interés, la exactitud del sistema lo podemos observar en la siguiente *Figura 67*:

```

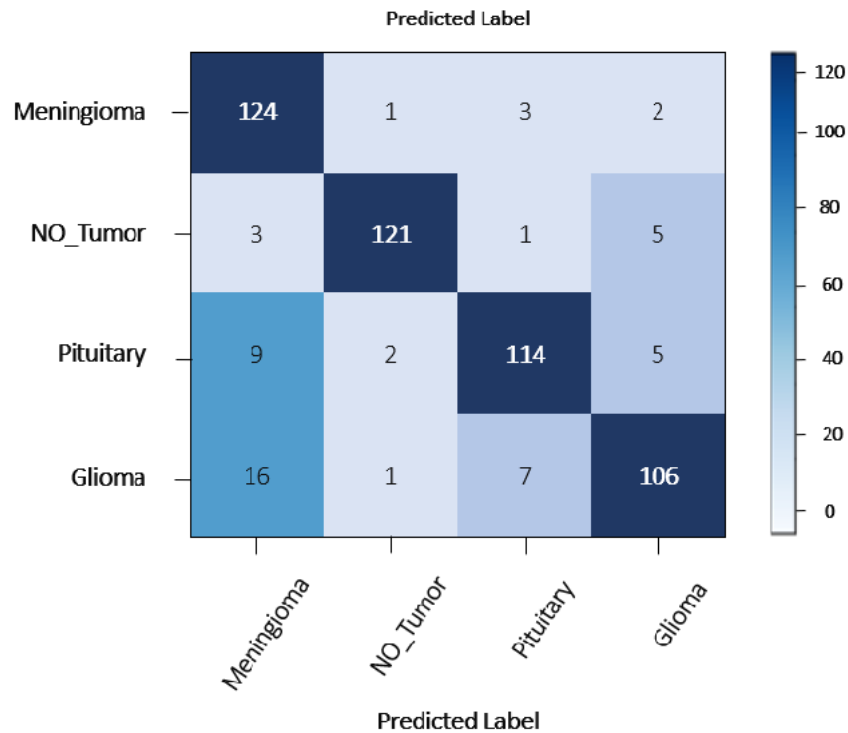
27/27 - 9s - loss: 0.0088 - accuracy: 1.0000 - val_loss: 0.3645 - val_accuracy: 0.8667
Epoch 46/50
27/27 - 9s - loss: 0.0086 - accuracy: 1.0000 - val_loss: 0.3655 - val_accuracy: 0.8667
Epoch 47/50
27/27 - 9s - loss: 0.0083 - accuracy: 1.0000 - val_loss: 0.3692 - val_accuracy: 0.8667
Epoch 48/50
27/27 - 9s - loss: 0.0081 - accuracy: 1.0000 - val_loss: 0.3612 - val_accuracy: 0.8667
Epoch 49/50
27/27 - 9s - loss: 0.0078 - accuracy: 1.0000 - val_loss: 0.3856 - val_accuracy: 0.8542
Epoch 50/50
27/27 - 9s - loss: 0.0074 - accuracy: 1.0000 - val_loss: 0.3784 - val_accuracy: 0.8625

```

Figura 67: Rendimiento de entrenamiento en exactitud y pérdidas con transfer learning.

Fuente: Google Colab (Elaboración propia)

Ahora, observamos que el sistema aprende de forma correcta, no obstante, se produce un rendimiento ligeramente inferior al que teníamos cuando implementamos la red neuronal con 2 clases, la matriz de confusión nos muestra que para algunas clases las predicciones no son tan exactas.



a)

```
[ ] evaluation = model.evaluate(train_batches)
print('Test Accuracy: {}'.format(evaluation[1]))
```

b)

```
59/59 [=====] - 35s 589ms/step - loss: 0.2294 - accuracy: 0.8817
Test Accuracy: 0.8817204236984253
```

Figura 68: a) Matriz de confusión para CNN de 4 clases., b) Evaluación de la exactitud en el proceso de entrenamiento.

Fuente: Google Colab (Elaboración propia)

La Figura 68a nos muestra las predicciones de la CNN, tal y como vemos, hemos perdido precisión en dichas predicciones, la principal justificación de este efecto se debe a la similitud física entre clases, tal y como hemos mencionado, el principal reto de este trabajo es tener que gestionar la enorme similitud que existe tal y como lo podemos apreciar en la siguiente imagen. Por otro lado, la Figura 68b muestra la precisión en el proceso de entrenamiento el cual alcanza el 88.17%.

La exactitud en las predicciones son las siguientes:

- **Meningioma:** 96.9%
- **No_Tumor:** 93,8%
- **Pituitary:** 90%
- **Glioma:** 83,8%

También es importante mencionar que el sistema ahora pierde precisión a medida que añadimos clases nuevas, sin embargo, sigue siendo una lectura bastante aceptable si lo comparamos con otros trabajos realizados con estos mismos *datasets* (ver sección 4.3). A continuación, miraremos un caso que real que se justifica lo dicho:

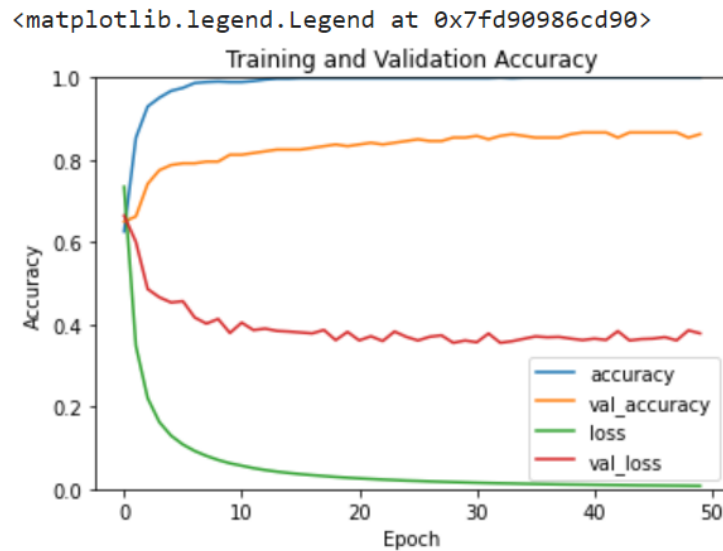


Figura 69: Gráficas de rendimiento en exactitud y pérdidas para el proceso de entrenamiento y validación de la cnn.

Fuente: Google Colab (Elaboración propia).

Por otro lado, la *Figura 69* refleja lo dicho anteriormente, las gráficas de tendencia del entrenamiento y validación del sistema muestran como a pesar de tener un buen proceso de aprendizaje y pérdidas durante el entrenamiento, las gráfica refleja lo opuesto durante el proceso de validación mostrando un bajo rendimiento, por lo que podemos asegurar que los resultados no son tan buenos en comparación al ejemplo anterior ya que las predicciones de algunas clases (meningioma principalmente) no son muy buenas.

MRI vista craneal posterior

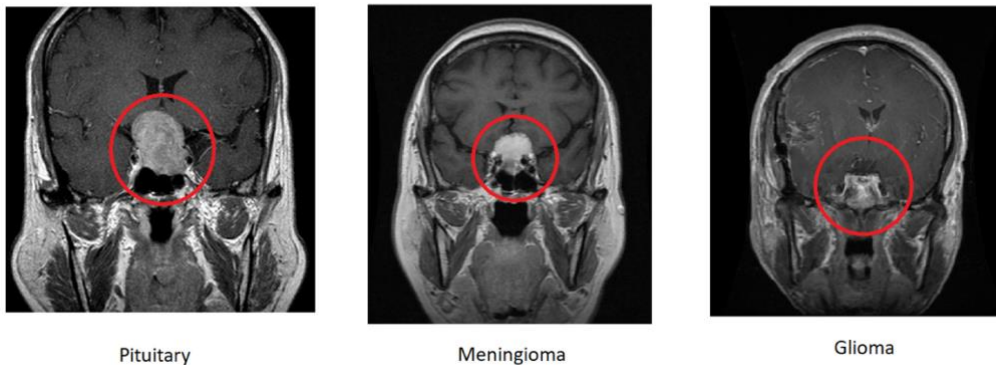


Figura 70: Similitud entre imágenes de distintas clases.

Fuente: Figshare Dataset (Disponible en: https://figshare.com/articles/dataset/brain_tumor_dataset/1512427)

La *Figura 70*, muestra la similitud que existe entre los diferentes tipos de tumores en algunas imágenes *MRI* del dataset, esto porque el tejido cancerígeno se encuentra localizado en una posición similar en el cerebro y la forma del tumor suele ser semejante en todos los casos, esto claramente impacta la precisión de la red neural al momento de entrenamiento, por otra parte, la resolución de algunas imágenes en estos *datasets* no son de la mejor calidad, y a pesar que hemos eliminado las imágenes más distorsionadas, siguen existiendo un porcentaje de imágenes con deficiencias.

Sumado a esto, hay que tomar en cuenta que los *datasets* son una recopilación de imágenes con distintas vistas como lo vemos en la *Figura 71*, lo que añade aún mayor heterogeneidad.

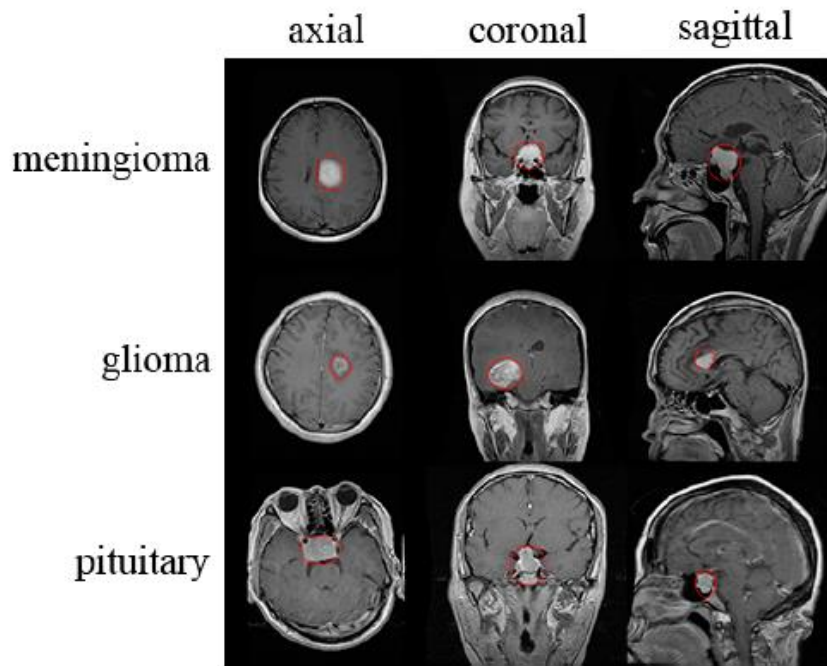


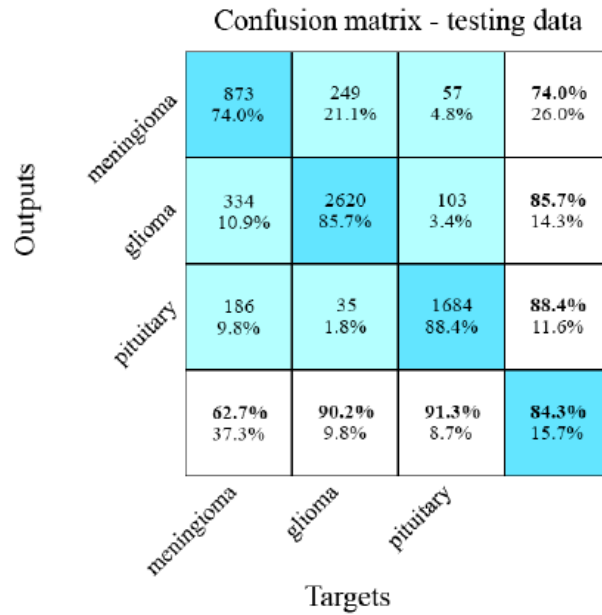
Figura 71: Vistas en imágenes MRi de diferentes ángulos en datasets.

Fuente: Classification of Brain Tumors from MRi Images Using a Convolutional Neural Network (pág 4)

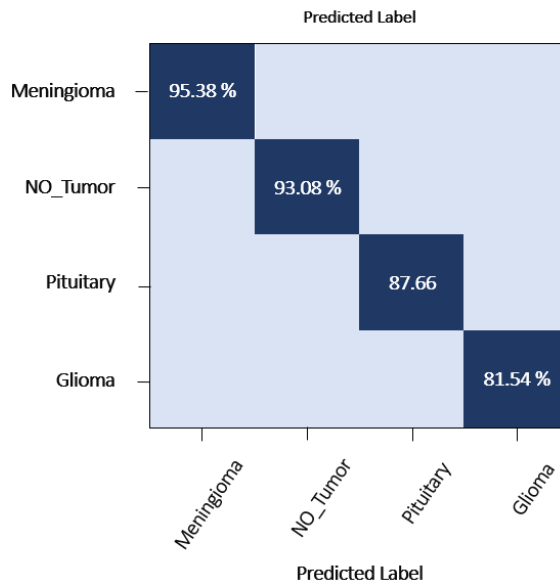
Esto sucede porque los tumores no suelen localizarse en puntos específicos en el tejido cerebral, al contrario, se deben tomar diferentes imágenes desde diferentes vistas para poder tener un mejor criterio a la hora de dar el diagnóstico final, el análisis médico debe tomar en cuenta cada detalle en la anatomía del tumor y por ello es que se tiene esta variabilidad en las imágenes.

8.8 ANALISIS COMPARATIVO CON OTROS ESTUDIOS

Los resultados expuestos en la sección anterior nos demuestran que el rendimiento es bueno si tomamos en cuenta los detalles a los que nos referimos y que representan el mayor reto de este trabajo, ya que las bases de datos disponibles de forma libre son muy limitadas y existen pocas. El trabajo del que hablamos en la sección 4.3 llamado “*Classification of Brain Tumors from MRi Images Using a Convolutional Neural Network*” nos demuestra que a pesar de los esfuerzos por ajustar al máximo los modelos de red neuronal para lograr excelentes resultados y con una alta precisión, no es tarea sencilla y requiere ciertas mejoras en los *datasets* para poder alcanzar mejores resultados.



a)



b)

Figura 72: a) Matriz de Confusión obtenida en estudio previo. B) Porcentaje de lecturas verdaderas positivas en las predicciones de nuestro sistema.

Fuente: Classification of Brain Tumors from MRi Images Using a Convolutional Neural Network (pág 8)

Tal y como observamos en la *Figura 72a*, la matriz de confusión para el estudio “*Classification of Brain Tumors from MRi Images Using a Convolutional Neural Network*” la precisión obtenida con *data augmentation* del *dataset* de *Figshare* muestra un rendimiento ligeramente inferior a nuestro caso de estudio, sin embargo, sigue siendo muy similar a nuestros resultados mostrados en la *Figura 68*, con la diferencia que el modelo predictivo que se propone en este estudio posiciona al Meningioma como la clase con menos precisión a la salida con

74.0% de exactitud, seguido por el glioma con un 85.7% y por el último el tumor pituitario con el 88.4%. Estos resultados fueron obtenidos con el siguiente modelo de red neuronal:

Layer No.	Layer Name	Layer Properties
1	Image Input	256 × 256 × 1 images
2	Convolutional	16 5 × 5 × 1 convolutions with stride [2 2] and padding 'same'
3	Rectified Linear Unit	Rectified Linear Unit
4	Dropout	50% dropout
5	Max Pooling	2 × 2 max pooling with stride [2 2] and padding [0 0 0]
6	Convolutional	32 3 × 3 × 16 convolutions with stride [2 2] and padding 'same'
7	Rectified Linear Unit	Rectified Linear Unit
8	Dropout	50% dropout
9	Max Pooling	2 × 2 max pooling with stride [2 2] and padding [0 0 0]
10	Convolutional	64 3 × 3 × 32 convolutions with stride [1 1] and padding 'same'
11	Rectified Linear Unit	Rectified Linear Unit
12	Dropout	50% dropout
13	Max Pooling	2 × 2 max pooling with stride [2 2] and padding [0 0 0]
14	Convolutional	128 3 × 3 × 64 convolutions with stride [1 1] and padding 'same'
15	Rectified Linear Unit	Rectified Linear Unit
16	Dropout	50% dropout
17	Max Pooling	2 × 2 max pooling with stride [2 2] and padding [0 0 0]
18	Fully Connected	1024 hidden neurons in fully connected (FC) layer
19	Rectified Linear Unit	Rectified Linear Unit
20	Fully Connected	3 hidden neurons in fully connected layer
21	Softmax	softmax
22	Classification Output	3 output classes, "1" for meningioma, "2" for glioma, and "3" for a pituitary tumor

Figura 73: Modelo de Red Neuronal aplicado en estudio.

Fuente: Classification of Brain Tumors from MRi Images Using a Convolutional Neural Network (pág 4)

Este modelo se compone de un total de 22 capas, mientras que nuestro diseño original se componía de un total de 17 como lo mostraba la *Figura 38*, en donde claramente notamos que a pesar del aumento de datos y el diseño de la red, el aprendizaje llega a un punto de saturación que, para el caso de los *datasets* que hemos utilizado, aun cuando intentemos seguir aplicando la técnica de la *data augmentation* y ajustando más la CNN, no llegaremos a conseguir mayores cambios, para lo cual, podemos recomendar el uso de *datasets* más diversos y con mayor calidad de resolución de las imágenes con el objetivo de tener una precisión mayor en la salida.

Mientras que la *Figura 72b* nos muestra los porcentajes de lecturas verdaderas positivas que tiene nuestro sistema, como vemos la lectura con el porcentaje más bajo de exactitud sigue siendo ligeramente mejor que la propuesta en el estudio de Milica M. Badža y Marko Č. Barjaktarovic, el cual nos deja en evidencia que el aumento de datos puede llegar a saturarse sin producir efectos de mejora.

CAPÍTULO IX: DESARROLLO DE APLICACIÓN

Una vez que hemos logrado construir un sistema cuya salida muestre resultados reproducibles y por supuesto, con el rendimiento esperado, procederemos a construir una aplicación web que le dará un valor agregado basado en las recomendaciones que obtenido de la Asociación de Afectados por Tumores Cerebrales en España (ASATE), en la cual los especialistas aseguran que al tratarse de un campo médico muy sensible y en el que se requiere muchísima experiencia tanto en el conocimiento didáctico como en el práctico, aún no nos encontramos en el nivel adecuado como para confiar 100% en las predicciones hechas por una red neuronal, incluso cuando esta sea altamente precisa.

Un error en la predicción por mínimo que sea podría traer consecuencias negativas para el paciente, en primer lugar porque el tratamiento para uno u otro tumor no es el mismo en todos los casos, segundo porque la localización física en el cerebro podría ser muy similar entre las distintas clases de tumores y esto podría arrojar predicciones equívocas y tercero porque se requiere un estudio multidisciplinar para dar un diagnóstico exacto, dónde se deben involucrar especialistas del área de Radiología, Oncología y Neurociencia.

9.1 RETROALIMENTACIÓN PREOFESIONAL

Para obtener una opinión que apoye nuestro estudio, hemos entrevistado al personal de asesoría Médica de la Asociación de Afectados por Tumores Cerebrales de España (ASATE), quienes nos ha dado su opinión desde el punto de vista médico y operativo, con el objetivo de construir un sistema que sea de provecho para quienes harían uso de este sistema, los principales puntos que hemos obtenido de la entrevista ya los hemos estudiado previamente y otras opiniones secundarias las exponemos a continuación:

- Los tumores cerebrales se han estudiado desde mucho tiempo atrás, dicho estudio ha venido evolucionando a medida que aparecen nuevas tendencias tecnológicas en distintas áreas como la visión por computador, procedimientos quirúrgicos asistidos, robótica asistencial para detección y tratamiento del cáncer.

- La comunidad médica debe apoyarse en una serie de recomendaciones establecidas por la organización mundial de la salud (OMS) y por entes gubernamentales cuando se trata de procedimientos en donde se requiere intervención de un anestesiólogo, cirujano o cualquier especialista en intervenciones invasivas y/o suministro de medicamentos a pacientes.
- Nuestra propuesta se trata de un nuevo sistema aún no testado en centros de salud a nivel local o internacional.
- El uso de redes neuronales para predicción de tumores cerebrales viene a apoyar la investigación, sin embargo, aún no puede ser utilizado como procedimiento oficial para diagnóstico médico.
- El sistema que hemos propuesto podría ser utilizado como una sugerencia que apoye al personal médico encargado de dar diagnósticos, esto vendría a acortar tiempos en el análisis de imágenes para dar un diagnóstico.
- Las predicciones dadas por el sistema podrían no solo ayudar a reducir los tiempos de espera, sino que también, podría reducir el error humano en la interpretación en la primera fase del análisis.
- Las aplicaciones más estandarizadas utilizan servicios web, esto por su facilidad de uso y la amplia gama de funciones que se pueden implementar, de igual manera, con el auge de los servicios en la nube que se ofrecen para almacenamiento, se puede utilizar bases de datos exclusivas y dedicadas para almacenamiento de las imágenes obtenidas y así, poder habilitar otros servicios anexos.
- Debido que en la mayoría de centros médicos, las MRi suelen almacenarse en bases de datos locales, los primeros análisis se realizan sobre imágenes digitales, aplicando diferentes filtros para poder analizar diferentes escenarios según forma, tamaño, etc.

9.2 DESARROLLO DE APLICACIÓN

Gracias a este input técnico, se ha decidido desarrollar una aplicación sencilla capaz de proveer las predicciones hechas por nuestro modelo, para ello haremos uso de una herramienta llamada *Flask*, la cual es un microframework de *Python*, tal y como se explicó en la *sección 5.10* en la que nuestro servicio web hará un *request* de tipo HTTP a un web server que almacenará nuestro modelo creado y que su vez, nos dará el resultado de las predicciones sobre la imagen que queramos analizar.

Esto es básicamente un servicio de solicitud y respuesta, en el que nuestro back-end será el servicio Flask y el front-end la aplicación web por medio de los lenguajes HTTP y JAVASCRIPT, así que lo que verá el usuario será las predicciones del modelo desde la web, pero la programación que está oculta al usuario será el microframework de Python que no vemos desde el lado del usuario. La *Figura 71*, muestra el funcionamiento de la aplicación propuesta.



Figura 74: Esquema de funcionamiento de la aplicación propuesta.

Fuente: Elaboración propia.

Para habilitar el uso de esta aplicación web, debemos tomar en cuenta que el usuario no tiene conocimiento del lenguaje de programación en el que se desarrolló el modelo de red neuronal y de hecho, no debería saberlo ya que su interés se enfoca en el servicio web por medio del cual obtendrá las predicciones del modelo neuronal, esto significa que debemos crear un sistema capaz de interpretar nuestro modelo (independientemente del lenguaje) el cual hemos obtenido en el entorno de programación de *Google Colab*, la aplicación web no puede ser interpretada no puede ser en el entorno *Colab*.

9.2.1 DESARROLLO DE APLICACIÓN BACK-END.

Previamente comentamos la estructura de nuestra aplicación (Ver *Figura 74*), la cual se basa en dos aplicaciones por separado, en este caso hablaremos del *Back-End*, que no es otra cosa más que nuestra aplicación *Flask* que está escrita en Python, a esta aplicación le hemos llamado *app_cnn_predict4.py* la cual hace un llamado a nuestro modelo obtenido previamente de nuestra red neuronal cuyo nombre es *Model_4clases.h5*. Para ello hemos creado un entorno dentro de nuestro repositorio de *Jupyter Notebooks* con las apps de *Flask* como lo apreciamos en la siguiente *Figura 75*.

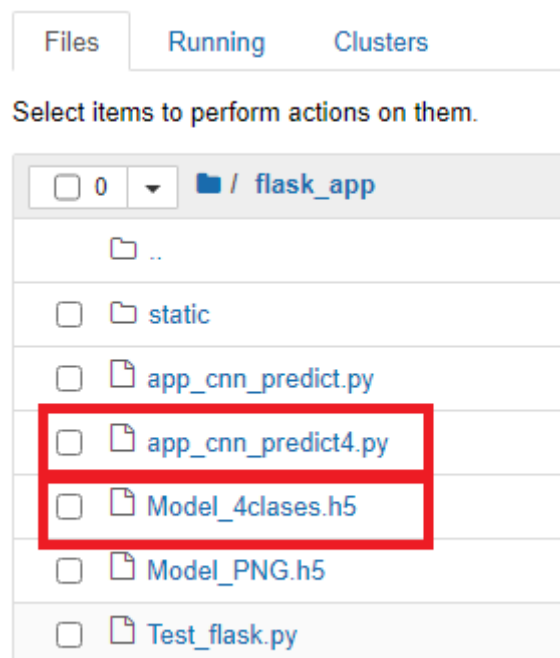


Figura 75: Aplicación Flask y repositorio de modelo de red neuronal.

Fuente: Elaboración propia (Ambiente de desarrollo de Jupyter Notebook)

Una vez que carga este modelo hacemos un pre-procesamiento de la imagen para definir tamaño, escala de colores y convertir la imagen a un array, luego hacemos una codificación que se le asigna a una imagen con la llave llamada 'image' desde JSON la cual se almacena en la variable de mensaje, para posteriormente decodificar dicha imagen en *io.BytesIO* dentro de la imagen codificada, no en la memoria, luego definimos las posibilidades por clase para hacer la predicción por medio de un *numpy array*, lo anteriormente dicho se muestra en la siguiente *Figura 76*.

```
def get_model():
    global model
    model = load_model('Model_4clases.h5')
    print(" * Modelo cargado correctamente!")

def preprocess_image(image, target_size):
    if image.mode != "RGB":
        image = image.convert("RGB")
    image = image.resize(target_size)
    image = img_to_array(image)
    image = np.expand_dims(image, axis=0)
    return image
```

Figura 76: Pre-procesamiento de imagen dentro de la aplicación de Flask llamada app_cnn_predict4.py

Fuente: Elaboración propia (Ambiente de desarrollo de Jupyter Notebook)

9.2.2 DESARROLLO DE APLICACIÓN FRONT-END

El *Front-End* será el complemento de nuestra aplicación web escrita en *HTML*, cuya función es hacer un llamado a la aplicación flask que contiene el modelo de red neuronal que hemos construido, para habilitar el entorno *Flask*, lo hacemos como se muestra en la siguiente *Figura 77*.

```
Select Command Prompt - python -m flask run --host=0.0.0.0

C:\Users\personal\flask_app>set FLASK_APP=app_cnn_predict4.py
C:\Users\personal\flask_app>python -m flask run --host=0.0.0.0
* Serving Flask app "app_cnn_predict4.py" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off

* Modelo cargado correctamente!
* Running on all addresses.
  WARNING: This is a development server. Do not use it in a production deployment.
* Running on http://10.211.55.11:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [18/Sep/2021 14:30:45] "GET /static/cnn_predict4.html HTTP/1.1" 304 -
127.0.0.1 - - [18/Sep/2021 14:30:46] "GET /favicon.ico HTTP/1.1" 404 -
```

Figura 77: Habilitación de entorno Flask.

Fuente: Elaboración propia (consola CMD Windows 10)

Una vez habilitado nuestro entorno Flask, dentro de nuestra aplicación *HTML* nos valdremos de una librería de JavaScript pequeña y sencilla, pero con una gran cantidad de funciones llamada JQuery (jQuery, 2006), el repositorio se encuentra disponible en: <https://jquery.com/download/> en el que nos valdremos del API disponible en: <https://code.jquery.com/jquery-3.3.1.min.js> , que nos permitirá hacer un llamado a una función llamada *prediction* que contiene la imagen de cada clase, como lo observamos en la *Figura 78*.

```
<p style="font-weight:bold">Selecciona la imagen en formato .png y pulsa boton de prueba</p>
<input id="image-selector" type="file">
<button id="predict-button">Prueba</button>
<p style="font-weight:bold">Prediccion 4 clases</p>
<p> Meningioma: <span id="men-prediction"></span></p>
<p> NO Tumor: <span id="notu-prediction"></span></p>
<p> Glioma: <span id="gli-prediction"></span></p>
<p> Pituitary: <span id="pit-prediction"></span></p>
<img id="selected-image" src=""/>

<script src="https://code.jquery.com/jquery-3.3.1.min.js"></script>
<script>
```

Figura 78: Acceso a librería JQuery para hacer el llamado a la función "prediction"

Fuente: Elaboración propia (Ambiente de desarrollo de Jupyter Notebook)

Por último, le diremos a nuestra app que haga un “POST” para que publique las predicciones de cada una de las clases y así ver de forma numérica, cada una de las probabilidades que tiene una imagen de presentar uno u otro tipo de Tumor, tal y como se expone en la Figura 79.

```
console.log(message);
$.post("http://127.0.0.1:5000/cnn_predict4", JSON.stringify(message), function(response){
  $("#men-prediction").text(response.prediction.men.toFixed(6));
  $("#notu-prediction").text(response.prediction.notu.toFixed(6));
  $("#gli-prediction").text(response.prediction.gli.toFixed(6));
  $("#pit-prediction").text(response.prediction.pit.toFixed(6));
  console.log(response);
});
```

Figura 79: Publicación en entorno web en la app escrita en HTML.

Fuente: Elaboración propia (Ambiente de desarrollo de Jupyter Notebook).

Estos repositorios los almacenaremos dentro de una carpeta que hemos llamado “static” la cual contiene nuestra app de *html*, como observamos en la Figura 80.

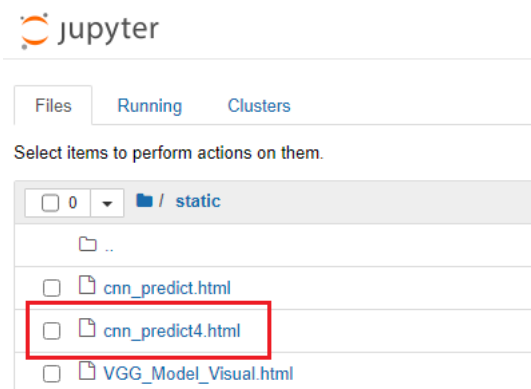


Figura 80: Repositorio de aplicaciones html dentro nuestro repositorio maestro.

Fuente: Elaboración propia (Ambiente de desarrollo de Jupyter Notebook).

9.2.3 EJECUCIÓN DE LA APLICACIÓN WEB.

En las anteriores secciones se explicó el desarrollo de nuestra aplicación, ahora demostraremos los resultados una vez que hemos habilitado la app, para ello accedemos al web server de Flask por medio de la dirección: http://127.0.0.1:5000/static/cnn_predict4.html , y nos abrirá la siguiente interfaz que observamos en la Figura 81.

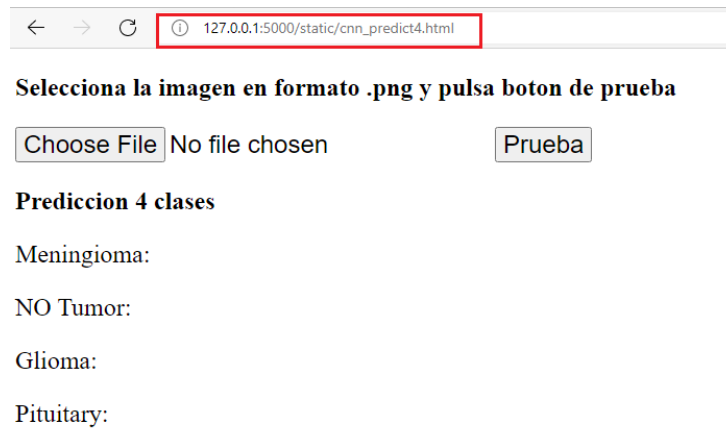
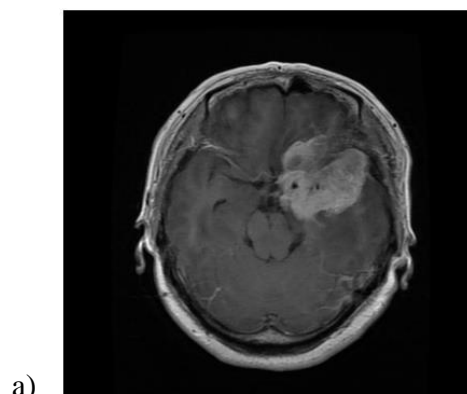
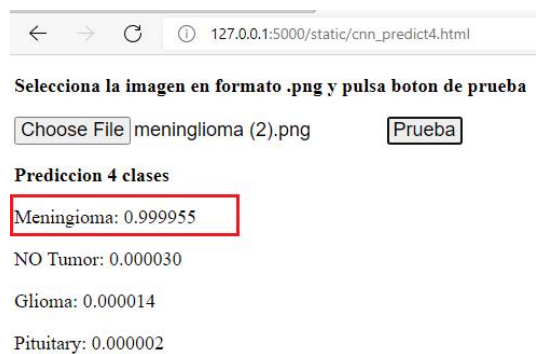


Figura 81: Acceso a la aplicación web.

Fuente: Elaboración propia (Entorno http).

Al elegir una imagen que puede estar almacenada dentro de nuestro ordenador o servidor con las imágenes *MRi* obtenidas de algún *dataset*, procederemos a presionar el botón de Prueba y obtendremos los resultados de las predicciones por clases, lo anterior lo apreciamos en la *Figura 82*, la cual representa las predicciones para cada una de las clases en diferentes imágenes MRi, de la siguiente manera: a) Meningioma, b) No Tumor, c) Glioma y d) Pituitario.



← → ↻ ⓘ 127.0.0.1:5000/static/cnn_predict4.html

Selecciona la imagen en formato .png y pulsa boton de prueba

Choose File NOT (62).png

Prueba

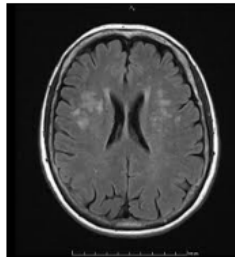
Prediccion 4 clases

Meningioma: 0.000086

NO Tumor: 0.999819

Glioma: 0.000094

Pituitary: 0.000001



b)

← → ↻ ⓘ 127.0.0.1:5000/static/cnn_predict4.html

Selecciona la imagen en formato .png y pulsa boton de prueba

Choose File glioma (23).png

Prueba

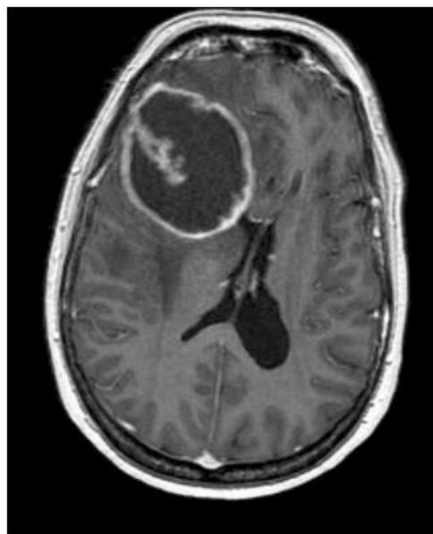
Prediccion 4 clases

Meningioma: 0.038257

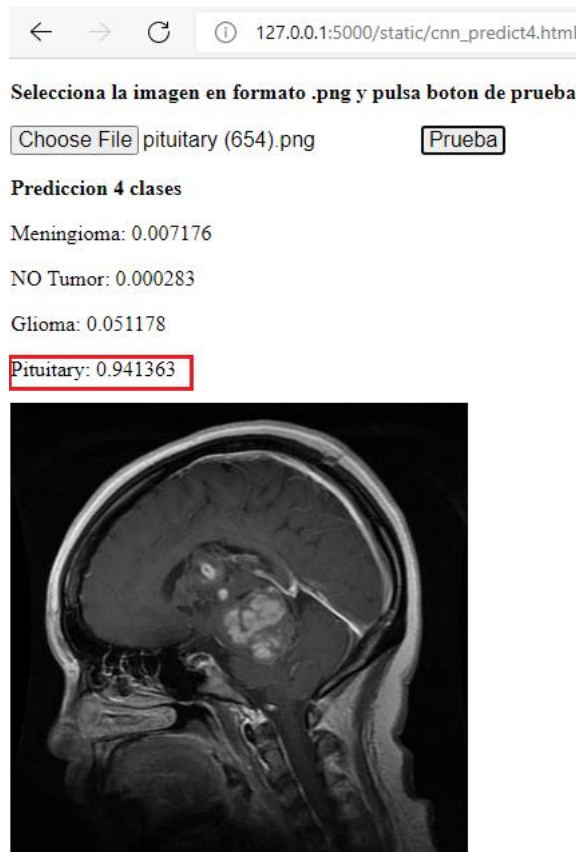
NO Tumor: 0.018373

Glioma: 0.943328

Pituitary: 0.000042



c)



d)

Figura 82: Predicciones obtenidas en nuestra aplicación web.

Fuente: Elaboración propia (Entorno http).

La *Figura 82* nos muestra un ejemplo de predicción para cada clase, en ella hemos podido comprobar que la exactitud en el proceso de entrenamiento y validación demuestra resultados consistentes para cada clase. Para hacer una prueba hemos tomado imágenes en las que previamente ya conocíamos su clase y hemos retado el sistema para comprobar las predicciones y tiene una alta precisión en la lectura.

9.2.4 EJECUCIÓN DE APLICACIÓN VISUAL

Adicionalmente a nuestra aplicación web, hemos desarrollado otra aplicación que permite ver los resultados de las predicciones con un formato visual en el que se muestran gráficos de barras y de pastel para tener una idea un poco más intuitiva de cada una de las predicciones.

Para nosotros valdremos de las librerías de *D3.js* de Java Script disponibles de forma gratuita en la web, esta herramienta permite agregar valor a nuestros proyectos de Machine Learning, sabemos que la visualización de resultados no solo es útil para la interpretación de los efectos del sistema en la salida, sino que también es una forma más dinámica de ver dichos resultados. *D3.js* permite manipular documentos basados en datos y es amigable con los lenguajes de *HTML*, *SVG* y *CSS* combinando de esta manera componentes de visualización y datos (D3js, 2011).

```
<script src="https://code.jquery.com/jquery-3.3.1.min.js"></script>
<script src="https://d3js.org/d3.v5.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/crossfilter/1.3.12/crossfilter.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/dc/3.0.3/dc.min.js"></script>
</script>
```

Figura 83: Habilitación de librería para funcionalidad D3js para aplicación gráfica.

Fuente: Elaboración propia (Entorno http).

En la *Figura 83* se muestran la forma en como hemos habilitado las librerías de D3js y CDjn para la aplicación que hemos propuesto. A continuación, veremos los resultados.

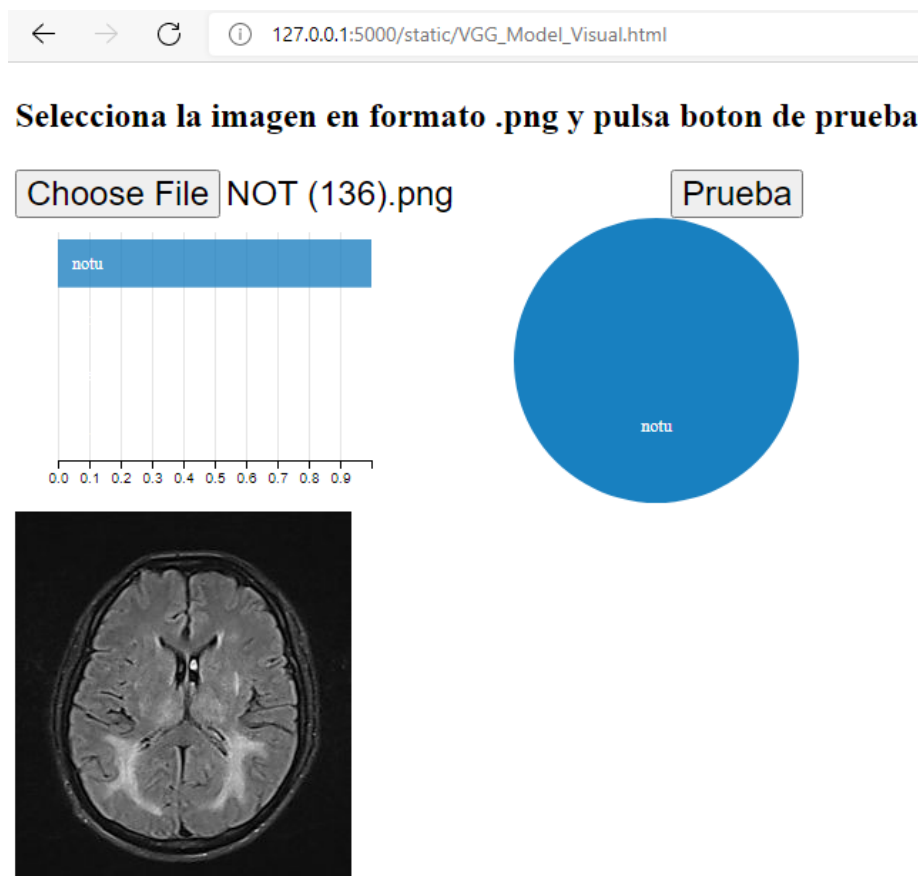


Figura 84: Aplicación visual para predicción de clases. Ejemplo con una imagen MRi sin tumor.

Fuente: Elaboración propia (Entorno http).

La *Figura 84*, nos permite apreciar el resultado que obtenemos a la salida de nuestra aplicación web visual, en ella hemos seleccionado una imagen que representa un cerebro sin tumor, para el cual el resultado reflejado tanto en el gráfico de barras horizontales como el gráfico de pastel, muestran que la imagen tiene un 99.99% de probabilidad de que sea un cerebro sin tumor, esto se corresponde con los resultados obtenidos en la aplicación de predicciones que vemos en la *Figura 85*.

Selecciona la imagen en formato .png y pulsa boton de prueba

NOT (136).png

Prediccion 4 clases

Meningioma: 0.000001

NO Tumor: 0.999999

Glioma: 0.000000

Pituitary: 0.000001

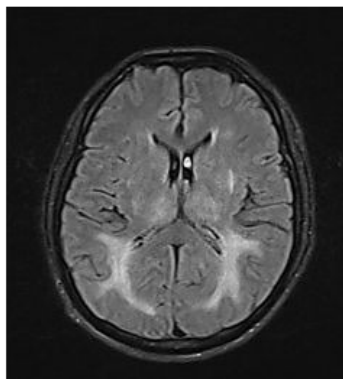


Figura 85: Resultados numéricos de predicciones sobre la misma imagen utilizada en el ejemplo anterior.

Fuente: Elaboración propia (Entorno http).

Ahora, analizaremos una imagen con un índice de confusión un poco más elevado, en la que una imagen MRi de un tumor pituitario tiene mucha semejanza con otras clases, por la posición donde se encuentra el tejido cancerígeno, el tamaño del tumor, y la forma. El resultado lo analizamos a continuación.

Selecciona la imagen en formato .png y pulsa boton de prueba

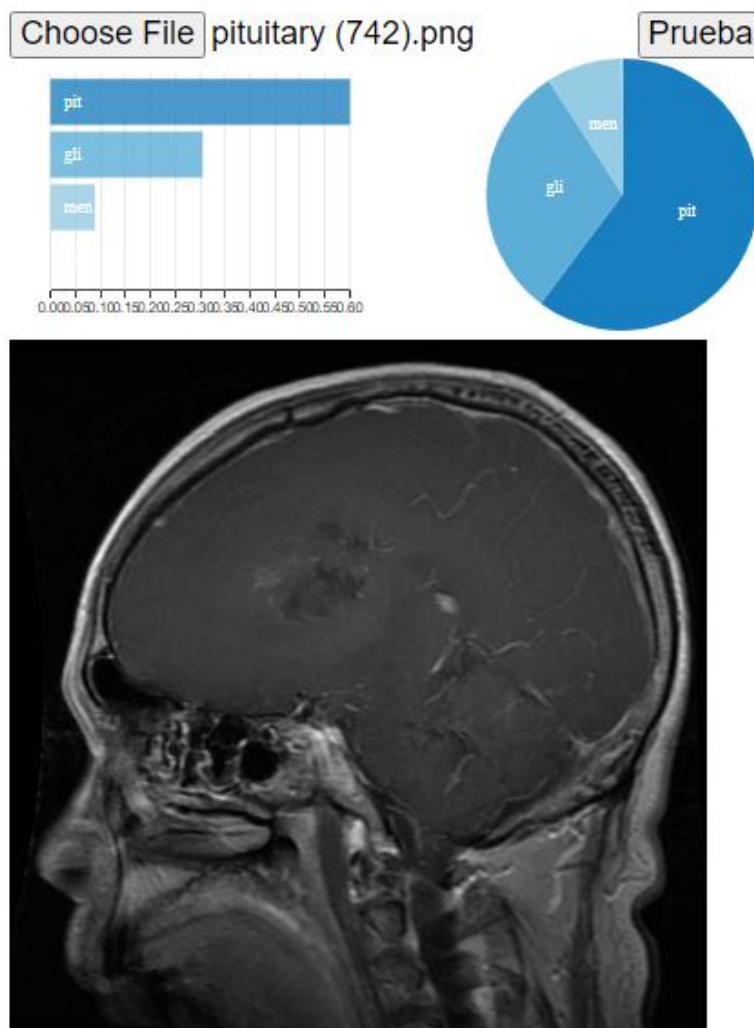


Figura 86: Aplicación visual para un Tumor Pituitario con semejanza con otras clases.

Fuente: Elaboración propia (Entorno http).

Observamos en la *Figura 86* que a pesar que la imagen MRi de un tumor Pituitario tiene alta semejanza con otra, las predicciones siguen siendo compatibles con las predicciones de una forma precisa.

Selecciona la imagen en formato .png y pulsa boton de prueba

Choose File pituitary (742).png

Prueba

Prediccion 4 clases

Meningioma: 0.089641

NO Tumor: 0.001447

Glioma: 0.306253

Pituitary: 0.602659

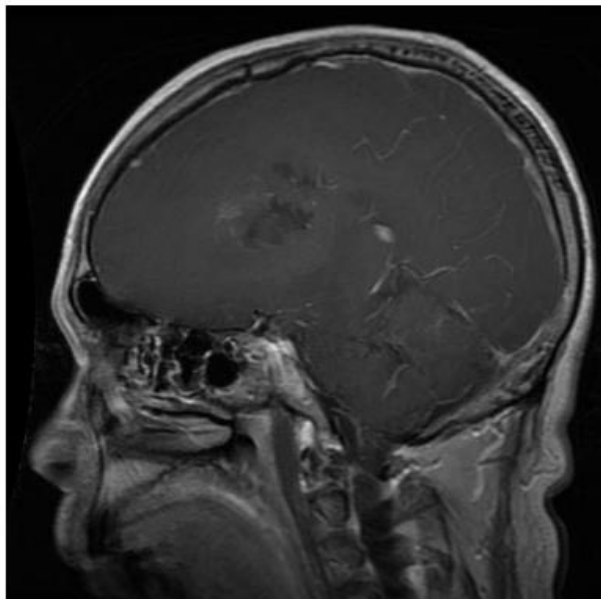


Figura 87: Análisis numérico de predicciones para la misma imagen MRI con Tumor Pituitario.

Fuente: Elaboración propia (Entorno http).

Tal y como lo apreciamos en la Figura 87, vemos que las predicciones muestran un 60.26% de probabilidad para el tumor pituitario y el porcentaje restante de 39.74% clases lo componen las demás clases, lo cual demuestra lo compatible de los resultados ya que la mayor probabilidad la tiene el tumor Pituitario.

CONCLUSIONES.

El desarrollo de este trabajo aporta conocimiento en diferentes áreas tecnológicas y científicas, con la intención principal de ayudar a la sociedad en general en un área vital como lo es la salud. La idea es que a futuro los procesos de diagnóstico sean más rápidos y más eficientes, permitiendo que los pacientes que sufren de este tipo de males puedan obtener su terapia médica lo más pronto posible con el fin de evitar complicaciones a futuro, ya que el derecho fundamental de cada individuo dentro de la sociedad a mejorar su calidad de vida debe ser factible para todos sin excepción.

Y por la parte técnica también hemos visto que a pesar de que no existe una vasta cantidad de bases de datos disponibles para fines investigativos, se pueden obtener resultados muy positivos, no obstante, la oportunidad de mejora es muy grande, no solo a nivel de los datasets, sino que también a nivel de técnicas de Deep Learning que permitan obtener resultados aún más confiables.

Con la finalización de este proyecto, podemos asegurar que se han cumplido los objetivos propuestos al inicio del trabajo, ya que no solo se han conseguido buenos resultados con el material disponible para entrenar nuestra red neuronal, sino que hemos añadido un valor adicional con la implementación de una aplicación web aplicable en un escenario real y que en definitiva ayudará mucho al personal médico con el proceso de diagnóstico médico.

Adicionalmente, algunas de las principales observaciones técnicas que podemos plasmar con este trabajo son:

- Los datasets disponibles de forma gratuita proveen la suficiente información para entrenar una red neuronal con resultados aceptables, estos datasets se puede combinar para producir un mejor efecto en los resultados, ya que permite la cantidad de información disponible para los procesos de entrenamiento y validación de nuestra red neuronal.
- La técnica de Transfer Learning pone en evidencia que es sumamente útil para procesos de entrenamiento de una red neuronal, los resultados reflejan un importante nivel de exactitud y por ende, de confiabilidad.
- La técnica del Data Augmentation también es de suma importancia cuando lo que se busca es mejorar los resultados en las predicciones.

- Las funciones que ofrece Keras y TensorFlow son de gran utilidad cuando se trabajó con redes neuronales para Machine y Deep Learning.
- La utilización de Google Colab ha sido fundamental para el desarrollo de este trabajo, ya que gracias a esta herramienta hemos podido hacer una GPU muy poderosa, lo cual ha facilitado mucho la optimización en el tiempo de entrenamiento, sin dejar de lado el hecho que es gratuito y la podemos integrar con Google Drive, así que esto nos aporta la ventaja de la portabilidad, en donde no dependemos de una sola estación de trabajo, sino que podemos utilizar cualquier ordenador con conexión a internet, sin necesidad tener que descargar los datasets en cada estación de trabajo.
- La combinación de datasets nos ha permitido mejorar la exactitud de los resultados.
- La creación de un modelo de red neuronal nos ha permitido llevar las predicciones hechas por el sistema a una aplicación web, que se acerca más a la realidad de un escenario de trabajo, en un centro médico.
- La implementación de Flask nos ha sido de gran utilidad dado que hemos utilizado Python como lenguaje de desarrollo, y Flask al ser escrito en Python nos permite crear una aplicación en el mismo entorno.
- La implementación de nuestra red neuronal nos ha demostrado que a pesar de que los tumores cerebrales tienen una gran heterogeneidad, se pueden construir modelos confiables en las predicciones, lo cual, nos deja en evidencia que independientemente de las clases de imágenes que estemos utilizando, siempre se puede hacer un ajuste fino con el fin de mejorar la calidad en las predicciones.
- Al igual que en cualquier área tecnológico y científica siempre se puede evaluar la posibilidad de mejorar los resultados. En nuestro caso, está claro que la mejora en la calidad de los datasets sería una enorme oportunidad de crear sistemas aún más confiables.

SIGUIENTES PASOS POR REALIZAR

Tal y como lo hemos expuesto en las conclusiones, también es necesario decir que una posible segunda etapa de este proyecto a implementar a futuro incluye algunas mejoras que exponemos a continuación:

- Se recomienda mejorar la calidad de los datasets, esta tarea requiere de colaboración por parte del personal médico. Una adecuada gestión de los datasets impactaría positivamente los resultados y, por tanto, la precisión y la confiabilidad sería mejor.
- Si bien es cierto, la tarea de diagnóstico médico de tumores cerebrales requiere un profundo análisis multidisciplinar, sin embargo, si se logra mejorar los resultados en la predicción, estaríamos a las puertas de cambiar algunas técnicas para el diagnóstico médico de este tipo de cáncer.
- De igual forma, la implementación de esta propuesta se puede llevar a la práctica en la predicción de otros tipos de tumores donde la anatomía podría variar, sin embargo, con los datasets adecuados se podría utilizar este modelo para construir una red neuronal robusta y con resultado excelentes.
- Con el auge de las aplicaciones móviles, también se podrían crear modelos con funcionalidad móvil.
- El uso de otros formatos de imágenes médicas también podría aportar una considerable mejora para los procesos de entrenamiento de una red neuronal. Como es conocido, recientemente se han implementado otras técnicas para obtención de imágenes médicas que podrían implementarse en el área de la neurociencia, área sumamente compleja por la naturaleza anatómica del sistema nervioso central y por su gran importancia en el desempeño de las funciones fisiológicas del ser humano.

BIBLIOGRAFÍA

- [1] Ali Habib, M. (31 de Agosto de 2019). *Github*. Obtenido de Brain Tumor Detector with Tensorflow & Keras: <https://github.com/MohamedAliHabib/Brain-Tumor-Detection>
- [2] Argibay, P. (2011). *Estadística avanzada: el problema del sobreajuste y el método de descripciones mínimas*. Buenos Aires, Argentina: Revistas del Hospital Italiano.
- [3] ASATE. (21 de Febrero de 2021). *Asociacion de Afectado por Tumores Cerebrales en España*. Obtenido de Sabías que: <http://www.asate.es/tumores-cerebrales/sabias-que/>
- [4] Braininvestigators. (8 de Octubre de 2018). *Mind Decoding Co*. Obtenido de <https://www.braininvestigations.com/neurociencia/inteligencia-artificial-tandem/>
- [5] Chakrabarty, N., & Kanchan, S. (24 de Mayo de 2020). *Kaggle*. Obtenido de Brain Tumor Classification (MRI): <https://www.kaggle.com/sartajbhuvaji/brain-tumor-classification-mri>
- [6] Cheng, J. (02 de Abril de 2017). *Brain Tumor Dataset Figshare*. Obtenido de <https://doi.org/10.6084/m9.figshare.1512427.v5>
- [7] Cheng, J., Huang, W., Cao, S., Yang, R., Yang, W., Yun, Z., . . . Feng, Q. (2015). *Enhanced Performance of Brain Tumor Classification via Tumor Region Augmentation and Partition*. Southern Medical University. Guangzhou, China: Southern Medical University.
- [8] D3js. (11 de Febrero de 2011). *Data-Driven Documents*. Obtenido de <https://d3js.org/>
- [9] Flores López, R., & Fernández F, J. (2008). *Las Redes Neuronales Artificiales: Fundamentos Teóricos y aplicaciones prácticas*. León: Netbiblio.
- [10] García Benítez, S., López Molina, J., Ramos Trejo, O., Palacios Morales, I., & Mora Méndez, L. (2016). *Redes Neuronales en la exploración geotécnica y el*

- [11] *modelado paramétrico: hacia entregables más competitivos*. Mérida, MX: Sociedad Mexicana de Ingeniería Geotécnica.
- [12] García Fenoll, I. (2010). *Aportaciones a la segmentación y caracterización de imágenes médicas 3D*. Sevilla: Universidad de Sevilla.
- [13] Gili, J. (2002). Introducción Biofísica a la Resonancia Magnética en Neuroimagen. En J. Gili. Barcelona: Scribd.
- [14] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. Massachusetts: The MIT Press.
- [15] Google Inc. (Febrero de 2018). *Google Colab*. Obtenido de Jupyter Notebook Projects: https://colab.research.google.com/notebooks/intro.ipynb?utm_source=scs-index
- [16] Grinberg, M. (2014). Flask Web Development. En *Developing Web Application with Python* (págs. 3-4). Sebastopol, CA: O'Reilly Media Inc.
- [17] IAARbook. (2018 de Marzo de 2018). *IAARBook Github*. Obtenido de <https://iaarbook.github.io/deeplearning/>
- [18] Ibrahim, K., & Castelli, M. (2020). *The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset*. Lisboa, Portugal: ScienceDirect.
- [19] jQuery. (2 de Marzo de 2006). *jQuery Documentation*. Obtenido de <https://jquery.com/download/>
- [20] Kaggle. (2010). *Kaggle Datasets*. Obtenido de www.kaggle.com/
- [21] Kang, Y., Cho, N., Yoon, J., Park, S., & Kim, J. (2021). *Transfer Learning of a Deep Learning Model for Exploring Tourists' Urban Image Using Geotagged Photos*. Seúl, Korea: Department of Social Studies, Ewha Womans University, Seoul.

- [22] Keepcoding. (2018). *Feature Engineering Made Easy*. Mumbai: Packt. Obtenido de <https://keepcoding.io/blog/que-son-datasets/>
- [23] Keras. (March de 2015). *Keras About*. Obtenido de <https://keras.io/about/>
- [24] Keras Doc. (27 de Marzo de 2015). *Image data preprocessing*. Obtenido de Keras: <https://keras.io/api/preprocessing/image/>
- [25] Milica M. , B., & Barjatarovic, M. (2020). *Classification of Brain Tumors from MRI Images Using a Convolutional Neural Network*. Belgrade, Serbia: University of Belgrade.
- [26] NVIDIA. (Marzo de 2019). *NVIDIA UK*. Obtenido de <https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/tesla-t4/t4-tensor-core-datasheet-951643.pdf>
- [27] Oracle Inc. (2016). *Oracle*. Obtenido de <https://www.oracle.com/es/big-data/what-is-big-data/>
- [28] Organización Mundial de la Salud. (2016). *Departamento de Cirug a, Radiolog a y Medicina F sica*. Obtenido de <http://www.oc.lm.ehu.es/Departamento/OfertaDocente/Teledocencia/Cruces/Cirugia1/NcE0Z.%20Clasificaci%C3%B3n%20tumores%20SNC%20OMS%202016%20Imprimir.pdf>
- [29] Ozdemir, S., & Susarla, D. (2018). *Feature Engineering Made Easy*. Mumbai: Packt.
- [30] Reyes Oliveiros, F., & Lema Bouzas, M. (2007). *Gliomas del enc falo*. Santiago de Compostela: Fundaci n Pedro Barri  de la Maza .
- [31] Siegel, R., Miller , K., & Jemal , A. (05 de Enero de 2017). *American Cancer Society*. Obtenido de <https://acsjournals.onlinelibrary.wiley.com/doi/full/10.3322/caac.21387>

- [32] Sociedad Española de Oncología Médica. (18 de Diciembre de 2019). *SEOM Tumores Cerebrales*. Obtenido de <https://seom.org/info-sobre-el-cancer/tumores-cerebrales?start=2>
- [33] Tammina, S. (2019). *Transfer learning using VGG-16 with Deep Convolutional Neural Network for Classifying Images*. Hyderabad, India: International Journal of Scientific and Research Publications (IJSRP).
- [34] TensorFlow. (2015). *Large-scale machine learning on heterogeneous systems* . Obtenido de <https://www.tensorflow.org/?hl=es-419>
- [35] Trincado Martínez, P. (Octubre de 2003). *Clinica Las Condes*. Obtenido de Unidad de Endocrinología: http://www.clc.cl/clcprod/media/contenidos/pdf/MED_14_1/TumoresPituitarios.pdf
- [36] USA National Cancer Institute . (26 de Marzo de 2020). *Centro de Investigación del Cáncer* . Obtenido de NCI-Connect: <https://www.cancer.gov/rare-brain-spine-tumor/tumors/meningioma>
- [37] Wiemels, J., Wrensch, M., & Claus, E. (2010). *Epidemiology and etiology of meningioma*. San Francisco, CA: Springer.
- [38] Wong, S., Gatt, A., Stamatescu, V., & McDonnell, M. (2016). *Understanding data augmentation for classification: when to warp?* Edinburgh, Australia: Information Technology and Mathematical Sciences University of South Australia.
- [39] Wu, J. (2017). *Introduction to Convolutional Neural Networks*. Nanjing, China: Nanjing University, China LAMDA Group.

ANEXO A. GLOSARIO

ASATE: Asociación de Afectados por Tumores Cerebrales en España.

SEOM: Sociedad Española de Oncología Médica.

OMS: Organización Mundial de la Salud.

MRi: Imagen de Resonancia Magnética.

TAC: Tomografía Axial Computarizada.

SPECT: Single Photon Emission Computer Tomography.

PET: Photo Emission Tech

IA: Inteligencia Artificial.