2021

CLASIFICACIÓN DE TUMORES CEREBRALES MENINGIOMA, GLIOMA, PITUITARY A PARTIR DE IMÁGENES DE RESONANCIA MAGNÉTICA MEDIANTE WAVELET E INTELIGENCIA ARTIFICIAL EN RASPBERRY PI4

ACREDITADA INSTITUCIONALMENTE

¡Seguimos avanzando!

INGENIERIA ELECTRÓNICA

OSCAR JULIÁN MOGOTOCORO GELVEZ

UNIVERSIDAD DE PAMPLONA | FORMANDO LIDERES PARA LA CONSTRUCCIÓN DE UN NUEVO PAIS EN PAZ

61 AÑOS

“La mente es un tipo de flor, debe regarse día a día con conocimientos nuevos, existen limitaciones para ella cuando terminamos por aceptarlas, y es ahí cuando empieza a marchitarse”

“Temo el día de no cometer un solo error, será un día sin la oportunidad de aprender, será un día en vano, será un día desperdiciado”

UNIVERSIDAD DE PAMPLONA
"Formando líderes para la construcción de un nuevo país en paz"

CLASIFICACIÓN DE TUMORES CEREBRAL MENINGIOMA, GLIOMA, PITUITARY A PARTIR DE IMÁGENES DE RESONANCIA MAGNÉTICA MEDIANTE WAVELET E INTELIGENCIA ARTIFICIAL EN RASPBERRY PI4

Autor:
OSCAR JULIÁN MOGOTOCORO GELVEZ

Director:
M.S LUIS ENRIQUE MENDOZA

Ingeniería Electrónica
Departamento de Ingenierías Eléctrica, Electrónica, Sistemas y Telecomunicaciones
Facultad de Ingenierías y Arquitectura
Universidad de Pamplona
Pamplona, 20 diciembre del 2021

A IVAN POR SU INCONDICIONAL APOYO

Tabla de contenido

TABLA DE FIGURAS

LISTA DE TABLAS

LISTA DE ECUACIONES

1.

Prefacio

AGRADECIMIENTOS

El conocimiento viene de los cielos, agradezco a DIOS por brindarme sabiduría, inteligencia, y acompañamiento en los momentos difíciles, a mis padres pedro Antonio, y rosita gelvez por su sacrificio diario sin ellos no sería nada, a mi hermano ivan por su apoyo constante y por ofrecer su sacrificio para que pueda consolidar este sueño llamado ingeniería electrónica, a mis hermanos, este logro no es solo mío es de todos, siempre será fácil recordar las personas que influyen en nuestras vidas como el profesor Luis enrique Mendoza que aceptó dirigir mi proyecto de grado, por creer en mis capacidades y en la posibilidad de terminar este proyecto, agradecimiento al profesor José Daniel Ramírez corzo por lograr transmitir nobleza en sus clases, antes de ser mi jurado es uno de los profesores que más admiro, a mis compañeros de estudio a esteban torres, a Sebastián Nossa y a Esneyder acosta de ahora en adelante en esta vida antes de considerarlos colegas los considero amigos. Agradecimiento a Karen por sus consejos y por seguirme la idea de cada locura.

Y mi Agradecimiento favorito a Tatiana calderón por acompañar este proceso, por recordarme día a día mis bases espirituales, por aconsejarme en los momentos difíciles, gracias por convertirte en ese ángel que alumbra mi camino.

RESUMEN

La clasificación de los tumores cerebrales se realiza mediante biopsia, que no suele realizarse antes de la cirugía cerebral definitiva. La mejora de la tecnología y el aprendizaje automático puede ayudar a los radiólogos en el diagnóstico de tumores sin medidas invasivas [1]. Este proyecto está enfocado en clasificar 3 tipos de tumor cerebral a partir de imágenes de resonancia magnética, mediante técnicas matemáticas como lo son el análisis de señales con transformada de wavelet, y aplicación de técnicas de inteligencia artificial con el fin de diferenciar 3 tipos de tumor cerebral, tumor meningioma, tumor glioma, tumor pituitary, el sistema se desarrollará mediante 4 fases principales las cuales son: adquisición de imágenes de resonancia magnética potenciadas con contraste ponderadas en T1, procesamiento de imágenes, reconocimiento de los tipos de tumor, e implementación del método basado en inteligencia artificial. En la fase de adquisición de datos se recolectarán imágenes de los 3 tipos de tumor cerebral esta base de datos se obtendrá a partir de consultas bibliográficas y de páginas web la cual pretende contribuir en la fase de procesamiento de imágenes. En la fase de procesamiento de imágenes se reconocerán patrones característicos de cada tipo de tumor y se acudirá a técnicas de inteligencia artificial para lograr clasificar los 3 tipos de tumor. En la fase de reconocimiento de los tipos de tumor se validarán los métodos de inteligencia artificial que logran diferenciar los 3 tipos de tumor, en la fase de implementación del sistema se creará una interfaz gráfica amigable con el radiólogo escrita en Python y asequible desde la Raspberry pi4.

Palabras clave:

REDES NEURONALES, WAVELET, TUMOR CEREBRAL, INTELIGENCIA ARTIFICIAL, RESONANCIA MAGNÉTICA, RASPBERRY PI4.

1.

Introducción

1. INTRODUCCIÓN

Un diagnóstico temprano del cáncer prolonga la vida del paciente y previene la muerte del paciente, además que el cáncer es uno de las dos principales muertes en el mundo cada vez más la inteligencia artificial está ayudando en procedimientos médicos, diagnóstico de enfermedades, algunos de los tumores cerebrales primarios son tumores cerebrales glioma, tumores cerebrales meningioma, la diferencia más amplia en este tipo de tumores es que pueden ser benignos como los tumores meningioma o malignos como los tumores cerebrales glioma, debido a esto es importante diferenciarlos para evaluación médica en pacientes. El diagnostico más común es la resonancia magnética sin embargo es susceptible a la subjetividad humana y una gran cantidad de datos es difícil para observación humana, por ello la aplicación de la transformada discreta de wavelet puede reducir el impacto de la subjetividad humana.

La aplicación de nuevas tecnologías inteligentes como el aprendizaje automático y la inteligencia artificial ha logrado impactos significativos en la medicina, brindando herramientas a ramas de la medicina basadas en imagenología, el mayor percance al diagnosticar tumores cerebrales radica en la experiencia del radiólogo, un diagnóstico certero y efectivo ayuda a controlar el crecimiento del tumor cerebral.

El objetivo de este proyecto radica en el examen de clasificar tres tipos de tumor primario cerebral a partir de una base de datos asimétrica o desequilibrada con una red neuronal convolucional que ha mostrado excelentes resultados por un previo procesamiento de imágenes, se presenta una arquitectura de inteligencia artificial basada en redes neuronales convolucionales previamente entrenada con imágenes de resonancia magnética potenciadas con contraste T1, y aplicación de la transformada discreta de wavelet. El rendimiento de la red neuronal convolucional se prueba con dos bases de datos, una base de datos normal, y una base de datos aumentada, los resultados se muestran en métricas de precisión con respecto a las fases de entrenamiento y validación.

El entrenamiento de redes neuronales amplias requiere de equipos de cómputo sofisticados y de alto costo económico, por ello se presenta el uso de una red neuronal entrenada en una máquina virtual que pone a disposición Google a través de Google colab, en la que se gestionan tarjetas gráficas modernas, y procesadores basados en tensores, de manera gratuita y libre y se exporta el modelo entrenado a un sistema embebido del tamaño de una tarjeta de crédito para apoyar el diagnóstico de tres tipos de tumor cerebral.

1.1. PROBLEMA

El cáncer es la segunda causa principal de muerte a nivel mundial, según la Organización Mundial de la Salud (OMS) [2]. La detección temprana del cáncer puede prevenir la muerte, pero esto no siempre es posible. A diferencia del cáncer, un tumor puede ser benigno, precarcinoma o maligno. Los tumores benignos difieren de los malignos en que los benignos generalmente no se diseminan a otros órganos y tejidos y pueden extirparse quirúrgicamente [3].

Cuando se sospecha la presencia de un cuerpo extraño en el cerebro se debe escoger un método para confirmarlo, la resonancia magnética permite caracterizar el comportamiento del tumor y realizar un diagnóstico diferencial del mismo y de procesos infecciosos o inflamatorios. Actualmente la resonancia magnética cuenta con métodos modernos y funcionales para apoyar el diagnóstico, tales como la espectroscopia, la difusión, la perfusión y la técnica BOLD [4]. La detección temprana de tumores cerebrales depende principalmente de la experiencia del radiólogo [5], de allí nace nuestro auge por ofrecer al radiólogo una alternativa para apoyar su diagnóstico, la creación de una herramienta a la que pueda acudir y ofrecer un tratamiento al paciente, donde está a su vez este fundamentada por la experiencia de otros radiólogos.

1.2. JUSTIFICACIÓN

Generalmente las técnicas basadas en inteligencia artificial requieren de equipos de cómputo de alto presupuesto, con ello nuestro principal objetivo es ofrecer al especialista una herramienta para apoyar su diagnóstico económica, portátil y asequible. La importancia de desarrollar redes más pequeñas también está relacionada con la posibilidad de implementar el algoritmo en plataformas móviles, lo que es importante para el diagnóstico en los países en desarrollo [6]. Otro factor importante en la toma de decisiones médicas es el tiempo que requiere el radiólogo para observar las imágenes y ofrecer un diagnóstico certero, que depende en gran medida de su experiencia, esta herramienta le ofrecerá la posibilidad de acompañar un diagnóstico preciso en menor tiempo, además que estará complementada con el diagnóstico de otros radiólogos expertos. Está herramienta económica, portátil se complementará con una interfaz gráfica diseñada en Python, e integrada en un sistema embebido con procesador, de código abierto y del tamaño de una tarjeta de crédito el sistema embebido raspberry pi 4.

1.3. OBJETIVOS

1.3.1. OBJETIVO GENERAL

- Clasificar tumores cerebrales meningioma, glioma, pituitary a partir de imágenes de resonancia magnética mediante wavelet e inteligencia artificial en raspberry pi4.

1.3.2. OBJETIVOS ESPECÍFICOS

- Revisar las fuentes bibliográficas acerca del uso de técnicas matemáticas, métodos computacionales, para la clasificación de tumores cerebrales.

- Aplicar la transformada discreta WAVELET para analizar las imágenes de resonancia magnética de tumores cerebrales.

- proponer e implementar un sistema de inteligencia artificial basado en redes neuronales para la identificación de tres tumores cerebrales meningioma, glioma, pituitary en una Raspberry pi 4.

- Programar una interfaz gráfica en Python, mediante la cual se pueda evaluar nuevas imágenes y ofrecer un diagnóstico certero sobre el tipo de tumor cerebral.

1.4. ACOTACIONES

Para el entrenamiento y evaluación del proyecto se usará el lenguaje de programación de alto nivel, y de código abierto Python ya que a diferencia de otros softwares como MatLab ofrece herramientas de inteligencia artificial más fluidas y con una comunidad más diversa. La creación de la interfaz gráfica y el empaquetado en la raspberry pi4 es posible gracias a Python además que ofrece a los usuarios múltiples librerías para la gestión de un entorno gráfico.

En la actualidad la clasificación de todos los tumores cerebrales exigiría equipos de cómputo bastante elevados en costos por ello pretendemos trabajar con solamente 3 tipos de tumor cerebral, además que nuestra propuesta se basa en el desarrollo e implementación en un equipo económico, y además portátil como lo es la raspberry pi 4, esta fue creada con la posibilidad de ser la computadora portátil más económica en el mercado.

2.

Marco Teórico

2. MARCO TEÓRICO

2.1. ESTADO DEL ARTE.

CLASIFICACIÓN DE LOS TUMORES CEREBRALES A PARTIR DE IMÁGENES DE RESONANCIA MAGNÉTICA MEDIANTE UNA RED NEURONAL CONVOLUCIONAL. Un algoritmo de aprendizaje automático que ha logrado resultados sustanciales en la segmentación y clasificación de imágenes es la red neuronal convolucional (CNN). Presentamos una nueva arquitectura de CNN para la clasificación de tumores cerebrales de tres tipos de tumores. La red desarrollada es más simple que las redes reentrenadas ya existentes, y fue probada en imágenes de resonancia magnética mejoradas por contraste ponderadas en T1. El rendimiento de la red se evaluó utilizando cuatro enfoques: combinaciones de dos 10-Métodos de validación cruzada y dos bases de datos. La capacidad de generalización de la red se probó con uno de los métodos de 10 veces, la validación cruzada por temas, y la mejora se probó mediante el uso de una base de datos de imágenes aumentada. El mejor resultado para el método de validación cruzada de 10 veces se obtuvo para la validación cruzada de registros para el conjunto de datos aumentados y, en ese caso, la precisión fue del 96,56%. Con una buena capacidad de generalización y una buena velocidad de ejecución, la nueva arquitectura CNN desarrollada podría utilizarse como una herramienta eficaz de apoyo a la toma de decisiones para los radiólogos en el diagnóstico médico [1].

Preprocesamiento de imágenes y aumento de datos: Las imágenes de resonancia magnética de la base de datos eran de diferentes tamaños y se proporcionaron en formato int16. Estas imágenes representan la capa de entrada de la red, por lo que se normalizaron y se redimensionaron a 256 × 256 píxeles. Para aumentar el conjunto de datos, transformamos cada imagen de dos maneras. La primera transformación fue la rotación de la imagen en 90 grados. La segunda transformación fue voltear imágenes verticalmente [13].

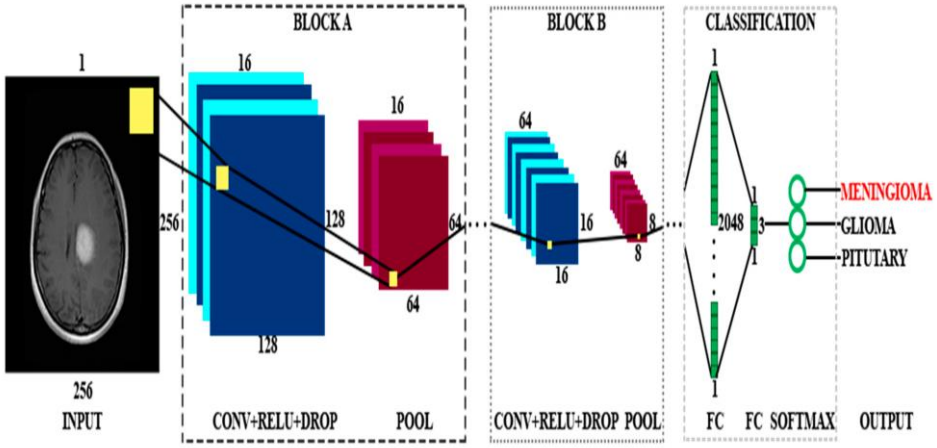


Figura 1. Representación esquemática de la arquitectura de la red neuronal convolucional (CNN) [1]

CLASIFICACIÓN DE LOS TUMORES CEREBRALES, El cáncer constituye la segunda causa de muerte en Chile; si bien los tumores malignos de encéfalo constituyen el 1.2% del cáncer en Chile, presentan alto impacto social por su mal pronóstico. En los últimos 20 años se ha desarrollado abundante investigación que ha llevado a dilucidar importantes mecanismos genéticos y epigenéticos de los tumores cerebrales. La nueva edición del año 2016 de la "Clasificación de tumores primarios del sistema nervioso central de la Organización Mundial de la Salud (OMS)" incorpora por primera vez la necesidad de contar con marcadores de biología molecular para la correcta clasificación de tumores astrocíticos, oligodendrogliales y méduloblastoma. Esperamos que la iniciativa de estructurar esta enfermedad en entidades con mecanismos biológicos comunes, nos permita un desarrollo futuro de terapias dirigidas o terapia personalizada con mayor efectividad para esta devastadora enfermedad [14].

GLIOMAS DE BAJO GRADO	DIAGNÓSTICO OMS 2016	GRADO OMS
Astrocitoma difuso	Astrocitoma difuso IDH mutado	Grado II
	Astrocitoma difuso IDH nativo	Grado II
	Astrocitoma difuso NOS	Grado II
Oligodendroglioma	Oligodendroglioma IDH mutado y 1p19q codeletado	Grado II
	Oligodendroglioma NOS	Grado II
Oligoastrocitoma	Oligoastrocitoma NOS	Grado II
GLIOMAS ALTO GRADO		
Astrocitoma anaplásico	Astrocitoma anaplásico IDH mutado	Grado III
	Astrocitoma anaplásico IDH nativo	Grado III
	Astrocitoma anaplásico NOS	Grado III
Oligodendroglioma anaplásico	Oligodendroglioma anaplásico IDH mutado y 1p/19q codeletado	Grado III
	Oligodendroglioma anaplásico NOS	Grado III
Oligoastrocitoma anaplásico	Oligoastrocitoma NOS	Grado III
Glioblastoma	Glioblastoma IDH mutado	Grado IV
	Glioblastoma IDH nativo	Grado IV
	Glioblastoma NOS	Grado IV
MEDULOBLASTOMA		
Meduloblastoma genéticamente definido	Meduloblastoma SHH activado y p53 mutado	Grado IV
	Meduloblastoma SHH activado y p53 nativo	Grado IV
	Meduloblastoma no WNT no SHH	Grado IV
Meduloblastoma histológicamente definido	Meduloblastoma clásico	Grado IV
	Meduloblastoma desmoplástico/nodular	Grado IV
	Meduloblastoma con nodularidad extensa	Grado IV
	Meduloblastoma de células grandes/anaplásico	Grado IV
	Meduloblastoma NOS	Grado IV

Tabla 1. EXTRACTO CLASIFICACIÓN DE TUMORES DEL SISTEMA NERVIOSO CENTRAL, OMS AÑO 2016 [14].

CLASIFICACIÓN MEDIANTE REDES NEURONALES DE APRENDIZAJE PROFUNDO PARA TUMORES CEREBRALES, El aprendizaje profundo es un nuevo campo de aprendizaje automático que ha ganado mucho interés en los últimos años. Se aplicó ampliamente a varias aplicaciones. y ha demostrado ser una poderosa herramienta de aprendizaje automático para muchos de los problemas complejos. En este artículo usamos Deep Neural Network clasificador que es una de las arquitecturas DL para clasificar un conjunto de datos de 66 resonancias magnéticas cerebrales en 4 clases, p. normal, glioblastoma, sarcoma y Tumores de carcinoma broncogénico metastásico. El clasificador se combinó con la transformada de ondícula discreta (DWT), la poderosa característica herramienta de extracción y análisis de componentes principales (PCA) y la evaluación del desempeño fue bastante buena en todo el desempeño [15].

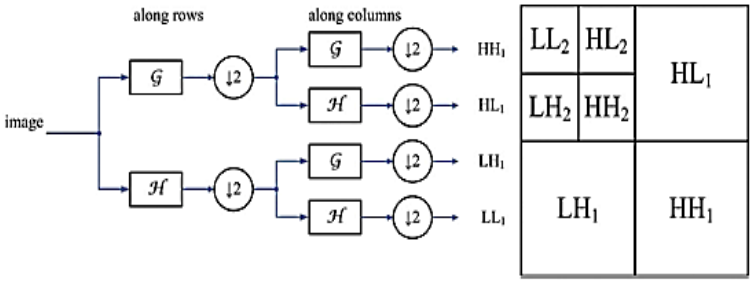


Figura 2. Niveles de descomposición DWT de una imagen [15].

DETECCIÓN DE TUMORES EN IMÁGENES DE RESONANCIA MAGNÉTICA CEREBRAL UTILIZANDO EL MÉTODO CONVOLUCIONAL HÍBRIDO ARQUITECTURA DE RED NEURONAL, El tumor cerebral es uno de los tipos de cáncer peligrosos y mortales que se observan en adultos y niños. Temprano y preciso El diagnóstico de tumor cerebral es importante para el proceso de tratamiento. Es un paso importante para que los especialistas detecten el tumor cerebral utilizando sistemas asistidos por computadora. Estos sistemas permiten a los especialistas realizar la detección de tumores más fácilmente. Sin embargo, también se evitan los errores cometidos con los métodos tradicionales. En este trabajo, tiene como objetivo diagnosticar el tumor cerebral mediante imágenes de resonancia magnética. Los modelos de CNN, una de las redes de aprendizaje profundo, se utilizan para proceso de diagnóstico. La arquitectura Resnet50, uno de los modelos de CNN, se utiliza como base. Las últimas 5 capas del Se eliminó el modelo Resnet50 y se agregaron 8 capas nuevas. Con este modelo se obtiene un valor de precisión del 97,2%. Además, los resultados se obtienen con los modelos Alexnet, Resnet50, Densenet201, InceptionV3 y Googlenet. De todos estos modelos, el modelo desarrollado con el mayor rendimiento ha clasificado las imágenes de tumores cerebrales. Como resultado, Cuando se analiza en otros estudios de la literatura, se concluye que el método desarrollado es eficaz y puede ser utilizado en sistemas asistidos por computadora para detectar tumores cerebrales [16].

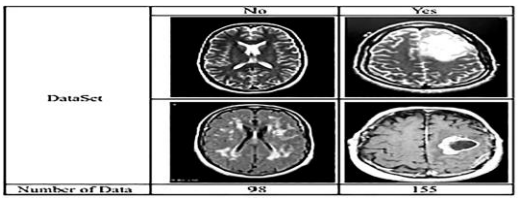


Figura 3. Imágenes de resonancia magnética con tumor cerebral y sin tumor [16]

2.2 TUMOR CEREBRAL

Un tumor cerebral es un crecimiento de células anormales en el tejido del cerebro. Los tumores pueden ser benignos (no cancerosos) o malignos (con células cancerígenas que crecen muy rápido). Algunos son primarios, o sea, que comienzan en el cerebro. Otros son metastásicos, o sea, que comenzaron en alguna otra parte del cuerpo y llegan al cerebro [17].

Un tumor es una masa de células anómalas. La mayoría de las células normales envejecen o se dañan, mueren y son reemplazadas por células nuevas. A veces, este proceso no tiene un buen resultado. Las nuevas células se forman cuando el cuerpo no las necesita, y las células viejas o dañadas no mueren como deberían. Esas células adicionales pueden formar tumores. Los tumores que comienzan en el cerebro se llaman tumores cerebrales primarios. Personas de todas las edades pueden desarrollar este tipo de tumores, incluidos los niños. Además, se pueden formar de diferentes maneras [18].

El cáncer que se ha diseminado al cerebro desde otra parte del cuerpo se llama tumor cerebral metastásico. Los tumores cerebrales metastásicos son mucho más comunes que los tumores primarios. Tanto los tumores cerebrales primarios como los metastásicos pueden causar síntomas similares. Los síntomas dependen principalmente de dónde se encuentra el tumor en el cerebro [18].

2.2.1. SINTOMATOLOGÍA

Los tumores en el cerebro pueden mostrar varios síntomas. Estos son algunos de los más comunes [17].

- Dolores de cabeza, generalmente por la mañana
- Náusea y vómitos
- Cambios en la capacidad para hablar, escuchar o ver
- Problemas de equilibrio o al caminar
- Problemas con el pensamiento o la memoria
- Debilidad o sentir muchas ganas de dormir
- Cambios en el estado de ánimo o conducta
- Convulsiones [17].

2.2.2. ANTECEDENTES MÉDICOS Y EXAMEN MÉDICO

Si los signos o los síntomas sugieren que usted podría tener un tumor encefálico o de médula espinal, su médico le hará preguntas acerca de su antecedente médico, enfocándose en los síntomas y cuándo estos comenzaron. El médico también revisará la función del encéfalo y la médula espinal, mediante el examen de los reflejos, la fuerza muscular, la visión, el movimiento de los ojos y la boca, la coordinación, el equilibrio y el estado de alerta [19].

2.3 MÉTODOS DE DETECCIÓN TUMOR CEREBRAL

Si los resultados del examen son anormales, es posible que le recomienden a un neurólogo (un doctor especializado en el tratamiento médico de las enfermedades del sistema nervioso) o un neurocirujano (un doctor especializado en el tratamiento quirúrgico de enfermedades del sistema nervioso) para realizar un examen neurológico más detallado u otras pruebas [19].

2.3.1. ANÁLISIS POR IMÁGENES

Su médico puede recomendar uno o más de estos estudios por imágenes. Estos estudios utilizan rayos X, imanes potentes o sustancias radiactivas para crear imágenes del encéfalo y la médula espinal. Las imágenes por resonancia magnética (MRI) y las tomografías computarizadas (CT) son utilizadas con más frecuencia para detectar las enfermedades del encéfalo. Si hay un tumor encefálico, estos estudios casi siempre lo mostrarán. A menudo, los médicos también obtienen una idea sobre el tipo de tumor que podría ser, basándose en cómo luce y dónde está localizado en el encéfalo [19].

2.3.1.1 Imagen por resonancia magnética

Las imágenes por resonancia magnética (MRI) son muy útiles para examinar el encéfalo y la médula espinal. Además, se consideran la mejor manera para detectar tumores en estas áreas. Por lo general, las imágenes que proveen son más detalladas que las de una tomografía computarizada (descrita más adelante). Sin embargo, no muestran los huesos del cráneo tan bien como las tomografías. Por lo tanto, es posible que en la MRI no se muestren los efectos de los tumores en el cráneo. En estos estudios se utiliza ondas de radio e imanes potentes (en lugar de rayos X) para producir imágenes. Para ayudar a mostrar mejor los detalles, es posible que un material de contraste, llamado gadolinio, se inyecte en una vena antes de realizar el estudio [19].

La Resonancia magnética es, por tanto, un método en el que se van a obtener distintas imágenes referentes a distintos cortes que se van a realizar en el cuerpo humano. El objetivo buscado con las imágenes de RM es medir las señales que provienen de los núcleos de los átomos en presencia de campos magnéticos. Las imágenes de resonancia magnética son anatómicas y funcionales, por tanto, permiten observar la morfología y tejidos y los procesos dinámicos que suceden en los mismos [7].

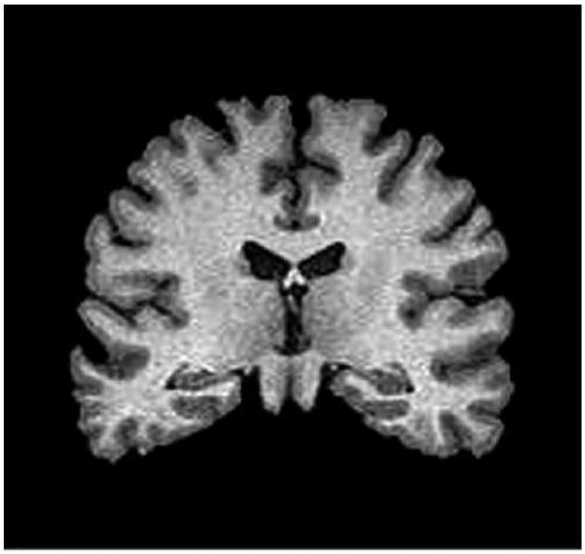


Figura 4. Imagen de resonancia magnética cerebral [7].

Varios tipos especiales de MRI pueden ser útiles en algunas situaciones, tal como:

- **Angiografía por resonancia magnética (MRA) y venografía por resonancia magnética (MRV):** estos tipos especiales de MRI se puede emplear para observar los vasos sanguíneos en el encéfalo. Este estudio resulta muy útil antes de la cirugía para ayudar al cirujano a planear la operación [19].

- **Espectroscopia por resonancia magnética:** este estudio puede hacerse como parte de una MRI. Mide los cambios bioquímicos en un área del encéfalo (que se muestran en los resultados parecidos a una gráfica llamada espectro), aunque también se pueden producir imágenes básicas). Tras comparar los resultados de un tumor con los del tejido encefálico normal, a veces puede ayudar a determinar el tipo de tumor (o qué tan rápido es probable que crezca), aunque una biopsia del tumor a menudo sigue siendo necesaria para obtener un diagnóstico preciso. Además, la espectroscopia por resonancia magnética se puede usar después del tratamiento para ayudar a determinar si un área que aún luce anormal en otro estudio es un tumor remanente o si es más probable que se trate de tejido cicatricial [19].

- **Perfusión por resonancia magnética:** para este estudio, también conocido como perfusión por MRI, se inyecta rápidamente sustancia de contraste en una vena. Luego se obtiene un tipo especial de imagen por resonancia magnética para observar la cantidad de sangre que pasa por las diferentes partes del encéfalo y el tumor. Los tumores a menudo tienen un suministro mayor de sangre que las áreas normales del encéfalo. Un tumor que crece más rápidamente podría necesitar más sangre. La perfusión por MRI le puede dar a los médicos una idea de cuál es el mejor lugar para hacer una biopsia. También se puede usar después del tratamiento para ayudar a determinar si un área que aún luce anormal es un tumor remanente o si es más probable que se trate de tejido cicatricial [19].

- **MRI funcional (fMRI):** este estudio identifica diminutos cambios en el flujo sanguíneo en una parte activa del encéfalo. Se puede usar para determinar qué parte del encéfalo maneja una función, tal como el habla, el pensamiento, la sensación o el movimiento. Los médicos pueden usar la MRI funcional para ayudar a determinar qué partes del encéfalo deben evitar cuando se planea la cirugía o la radioterapia. Este estudio es similar a la MRI convencional, salvo que se le solicitará que realice tareas específicas (como contestar preguntas simples o mover sus dedos) mientras se lleva a cabo el estudio [19].

2.3.1.2 Tomografía computarizada

La tomografía computarizada (TC) es un estudio de radiografía que produce imágenes transversales detalladas de su encéfalo y de su médula espinal (u otras partes del cuerpo). A diferencia de una radiografía convencional, una tomografía computarizada (TC) crea imágenes detalladas de los tejidos blandos del cuerpo. La TC no se usa tan frecuentemente como la MRI para detectar tumores de encéfalo o médula espinal, pero pueden ser útiles en algunos casos. Se puede usar si la MRI no es una opción (como en las personas que tienen mucho sobrepeso o personas que le temen a los lugares cerrados). Además, la TC muestra un mayor detalle de las estructuras de los huesos cercanas al tumor. Al igual que con la MRI, puede que se administre una inyección de una sustancia de contraste a través de una vía endovenosa (dentro de la vena) antes del estudio (aunque se utiliza un colorante diferente para las TC). Esto ayuda a delinear mejor cualquier tumor que pueda estar presente [19].

- **Angiografía por TC (TCA):** para este estudio, a usted se le inyecta un material de contraste a través de una vía endovenosa mientras se encuentra en el escáner de la TC. El escáner crea imágenes detalladas de los vasos sanguíneos en el encéfalo, lo que puede ayudar a los médicos a planear la cirugía. En algunos casos, la angiografía por TC puede proveer mejores detalles de los vasos sanguíneos del tumor y sus alrededores que la angiografía por MR [19].

2.3.1.3 Tomografía por emisión de positrones

Para realizar una tomografía por emisión de positrones (TEP) se le inyecta una sustancia ligeramente radiactiva (generalmente un tipo de azúcar conocido como FDG) que se acumula principalmente en las células tumorales. Después se usa una cámara especial para crear una imagen de las áreas de radiactividad en el cuerpo. La imagen no es tan detallada como en una TC o una MRI, pero la información obtenida puede ser útil para determinar la probabilidad de que las áreas anormales vistas en otras pruebas (como una MRI) sean tumores. Este estudio es más probable que sea útil para tumores de rápido crecimiento (tumores de alto grado) que para tumores de crecimiento más lento. Este estudio también es útil después del tratamiento para ayudar a determinar si un área que aún luce anormal en una MRI es un tumor remanente o si es más probable que se trate de tejido cicatricial. El tumor remanente se podría ver en la TEP, pero no el tejido cicatricial [19].

2.3.1.4 Radiografía de tórax

Puede que se tome una radiografía de tórax para localizar tumores en los pulmones si se detecta un tumor en el encéfalo. Esto se debe a que la mayoría de los tumores en el encéfalo de los adultos en realidad comenzaron en otro órgano (con más frecuencia los pulmones) y luego se propagaron al encéfalo. Este estudio se puede hacer en el consultorio de su médico, en un centro de radiología para pacientes ambulatorios o en un hospital [19].

2.3.2. BIOPSIA DE TUMOR DE ENCÉFALO O DE MÉDULA ESPINAL

Los estudios por imágenes, tales como las imágenes por resonancia magnética y la tomografía computarizada, pueden mostrar un área anormal que probablemente sea un tumor de encéfalo o de médula espinal. Pero estos estudios no siempre pueden indicar exactamente el tipo de tumor. A menudo, esto solo se logra mediante la extracción de parte del tejido del tumor en un procedimiento que se conoce como biopsia [19].

Una biopsia puede ser un procedimiento de por sí o puede ser parte de una cirugía para extraer el tumor. A veces, un tumor puede parecer tan característicamente obvio en una MRI que no se necesita una biopsia, especialmente si el tumor está en una parte del cerebro que dificultaría la biopsia (como el tronco encefálico). En pocos casos, una biopsia no es necesaria si una TEP o una espectroscopia por MR ofrece suficiente información [19].
Los dos tipos de biopsias principales son:

2.3.2.1 Biopsia estereotáctica con aguja

Este tipo de biopsia se puede usar si, de acuerdo con los estudios por imágenes, los riesgos de la cirugía para extirpar el tumor podrían ser muy altos (como algunos tumores en áreas vitales, aquellos tumores que están muy profundos dentro del encéfalo, u otros tumores que probablemente no se puedan extraer de forma segura con cirugía), pero una muestra

sigue siendo necesaria para hacer un diagnóstico. Para la biopsia, el paciente puede estar dormido (bajo anestesia general) o despierto. Si el paciente está despierto, el neurocirujano inyecta un anestésico local en las áreas de la piel sobre el cráneo para adormecerlas. (El cráneo y el encéfalo no experimentan dolor) [19].

La biopsia se puede hacer de dos maneras principales:

- Un método consiste en realizar una MRI o una CT, y luego usar cualquiera de los marcadores (cada uno del tamaño de una moneda de diez centavos) colocados en diferentes partes del cuero cabelludo, o contornos faciales y del cuero cabelludo, para crear un mapa del interior de la cabeza. Luego se hace una incisión (corte) en el cuero cabelludo y se taladra un orificio pequeño en el cráneo. Luego se utiliza un sistema de guía por imagen para dirigir una aguja hueca en el tumor para extraer pequeños fragmentos de tejido [19].

- En un abordaje que se utiliza con menos frecuencia, se fija un marco rígido a la cabeza. Una MRI o una CT a menudo se usa junto con el armazón para ayudar al neurocirujano a guiar la aguja hueca en el tumor. Esto también requiere de una incisión (corte) en el cuero cabelludo y de un orificio pequeño en el cráneo [19].

El tejido extraído se envía a un patólogo (un médico especializado en el diagnóstico de enfermedades mediante pruebas de laboratorio). Algunas veces puede que sea necesario que un neuropatólogo, un patólogo especializado en enfermedades del sistema nervioso, examine el tejido. El patólogo examina el tejido con un microscopio (y puede que haga otras pruebas de laboratorio) para determinar si el tumor es benigno o maligno (canceroso) y exactamente qué tipo de tumor está presente. Esto es muy importante para determinar el pronóstico de una persona y el mejor curso del tratamiento. Es posible que se pueda hacer un diagnóstico preliminar el mismo día, aunque a menudo toma al menos varios días obtener un diagnóstico final [19].

2.3.2.2 Biopsia quirúrgica o abierta (craneotomía)

Es posible que el neurocirujano no haga una biopsia con aguja si los estudios por imágenes muestran que el tumor probablemente se pueda tratar con cirugía. En lugar de esto, se puede hacer una operación llamada craneotomía (descrita en Cirugía para los tumores de encéfalo y de médula espinal en adultos) para remover todo o la mayor parte del tumor. (Si la extirpación de todo el tumor probablemente dañaría las estructuras importantes cercanas, se podría proceder con la extirpación de la mayor parte del tumor, conocida como cirugía citorreductora). Para obtener un diagnóstico preliminar, un patólogo examina de inmediato pequeñas muestras del tumor, mientras el paciente se encuentra en el quirófano (sala de operaciones). Esto puede ayudar a guiar el tratamiento, incluyendo si se debe hacer

cirugía adicional en ese momento. En la mayoría de los casos, se obtiene un diagnóstico final dentro de varios días [19].

2.3.3. PUNCIÓN LUMBAR

Este estudio se usa principalmente para buscar células cancerosas en el líquido cefalorraquídeo que rodea el encéfalo y la médula espinal. Para esta prueba, usted se acuesta sobre un costado en una cama o una camilla con sus rodillas hacia su pecho. El médico primero adormece un área en la parte baja de la espalda cerca de la médula espinal. Entonces se introduce una pequeña aguja hueca entre los huesos de la columna vertebral para extraer algo de líquido. Este líquido se envía a un laboratorio para ser examinado y saber si contiene células cancerosas. También se pueden hacer otras pruebas en el líquido Por lo general, las punciones lumbares son muy seguras, pero los médicos tienen que asegurarse de que la prueba no cause un descenso drástico en la presión de líquido dentro del cráneo, lo que podría causar problemas graves. Por esta razón, los estudios por imágenes, como la TC o la MRI, se hacen primero [19].

2.3.4. PRUEBAS DE ORINA Y SANGRE

Estas pruebas de laboratorio rara vez son parte del diagnóstico real de tumores encefálicos y de médula espinal, pero se pueden hacer para saber cuán bien están funcionando el hígado, los riñones y algunos otros órganos. Esto resulta especialmente importante antes de planear la cirugía. Si usted está recibiendo quimioterapia, se realizarán análisis de sangre rutinariamente para verificar los recuentos sanguíneos y para saber si el tratamiento está afectando a otras partes de su cuerpo [19].

2.4 TIPOS DE TUMOR CEREBRAL

2.4.1 Glioma.

El glioma es un tipo de tumor que se desarrolla en el cerebro y la médula espinal. Los gliomas comienzan en el soporte viscoso (células gliales) que rodea las células nerviosas y las ayuda a funcionar. Tres tipos de células gliales pueden producir tumores. Los gliomas se clasifican según el tipo de célula glial involucrada en el tumor, así como las características genéticas del tumor [8].

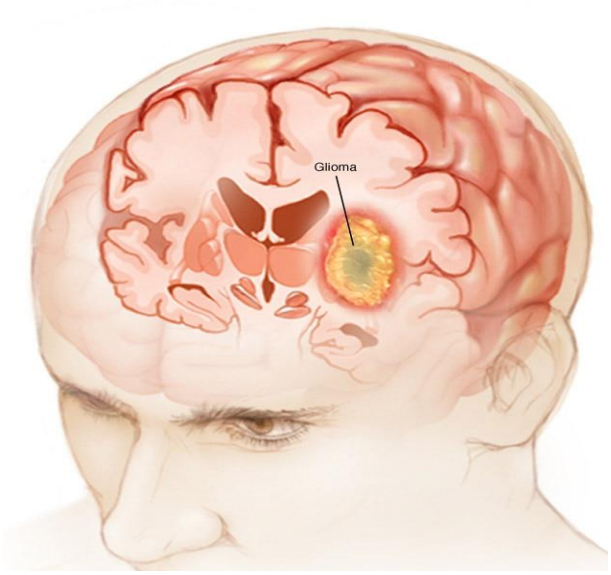


Figura 5. Zona Cerebral donde se ubica el tumor glioma [8].

En la figura 5 se aprecia como Los gliomas comienzan en el soporte viscoso (células gliales) que rodea las células nerviosas en el cerebro [8].

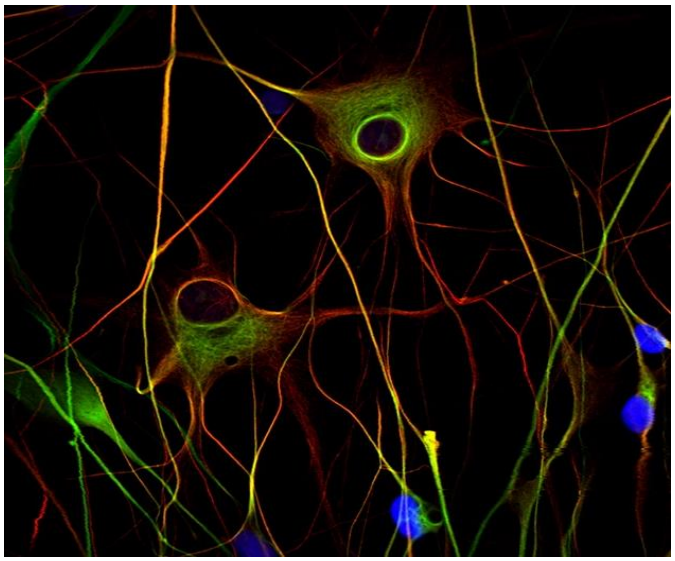


Figura 6. Células de glioma malignas [8].

En la figura 6 se aprecian Células de glioblastoma multiforme (tumor cerebral maligno). Las células tienen formas irregulares con ramificaciones que pueden expandirse dentro del cerebro [8].

2.4.2 Meningioma.

Un meningioma es un tumor que surge de las meninges, que son las membranas que rodean el cerebro y la médula espinal. Aunque técnicamente no es un tumor cerebral, se lo incluye en esta categoría porque puede comprimir o presionar el cerebro, los nervios y los vasos adyacentes. El meningioma es el tipo de tumor más común que se forma en la cabeza. Dado que la mayoría de los meningiomas crecen lentamente, a menudo sin signos y síntomas significativos, no siempre requieren un tratamiento inmediato y se pueden controlar a lo largo del tiempo [8].

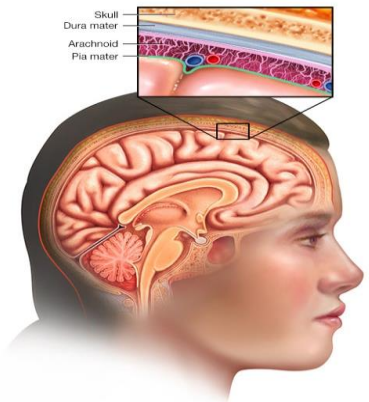


Figura 7. Zona Cerebral donde se ubica el tumor meningioma [8].

En la figura 7 se aprecian tres capas de membranas conocidas como meninges que protegen el cerebro y la médula espinal. La capa interna delicada es la piamadre. La capa del medio es la aracnoidea, una estructura como una tela de araña llena de un líquido que amortigua el cerebro. La fuerte capa externa es denominada dura madre [8].

2.4.3 Pituitarios.

Los tumores pituitarios son crecimientos anormales que se desarrollan en la glándula pituitaria. Algunos tumores pituitarios generan demasiadas hormonas que regulan las funciones importantes del cuerpo. Algunos tumores pituitarios pueden hacer que la glándula pituitaria produzca niveles más bajos de hormonas La mayoría de los tumores pituitarios son crecimientos no cancerosos (benignos). Estos permanecen en la glándula pituitaria o en los tejidos que la rodean y no se diseminan a otras partes del cuerpo [8].

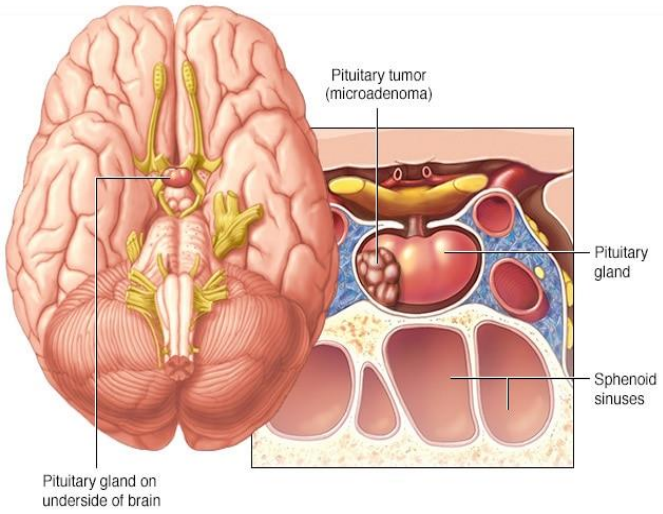


Figura 8. Zona Cerebral donde se ubica el tumor pituitario [8].

En la figura 8 se aprecia Un tumor hipofisario que se forma en la hipófisis cerca del cerebro que puede causar cambios en los niveles hormonales del cuerpo. Esta ilustración muestra un tumor más pequeño (microadenoma) [8].

2.5 IMÁGENES POTENCIADAS EN T1

El tiempo T1 es el tiempo que tarda la magnetización longitudinal, en el eje z, en recuperar el 63% de su valor inicial. Cuando se tiene un T1 corto significa que se tiene una liberación de la energía rápida, ya que se alcanza rápidamente el estado de equilibrio. Es decir, el tiempo T1 nos da información acerca de la facilidad de liberación energética. Una imagen T1 (figura 9) es aquella en la que la intensidad de la señal es proporcional a la facilidad en la relajación [4].

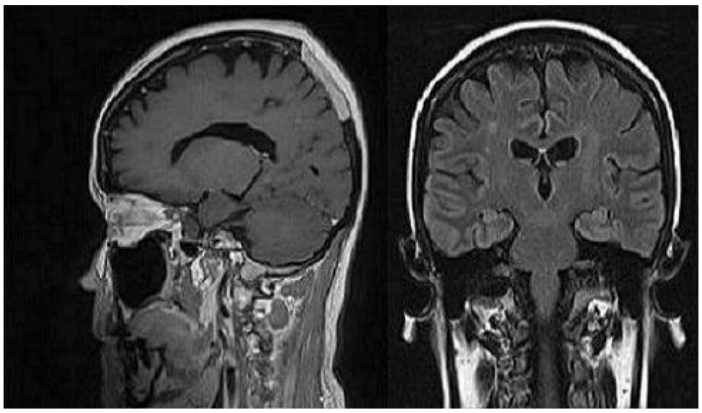


Figura 9. Imágenes de resonancia magnética potenciadas en T1 [4].

2.6 WAVELET EN IMÁGENES.

La técnica de Wavelet consiste en hacer una transformación a la imagen y por tanto se analiza la textura en el dominio transformado. Esta técnica consiste en analizar el contenido en frecuencia de una imagen dentro de las diferentes escalas de la imagen. Al realizar este análisis se obtienen unos coeficientes de wavelet que representan diferentes escalas y diferentes direcciones de frecuencia. Cuando se aplica la transformada de Wavelet sobre una imagen se asocia a cada pixel un conjunto de números, que son los coeficientes de Wavelet, estos caracterizan el contenido en frecuencia de la imagen en ese punto sobre un conjunto de escalas. A partir de los coeficientes de Wavelet se podrán calcular los parámetros de textura para de esta manera realizar el análisis de textura [4]. En la figura 10 se aprecia la aplicación de la transformada discreta de wavelet con varios niveles de descomposición a la imagen de la figura 4 de una resonancia magnética.

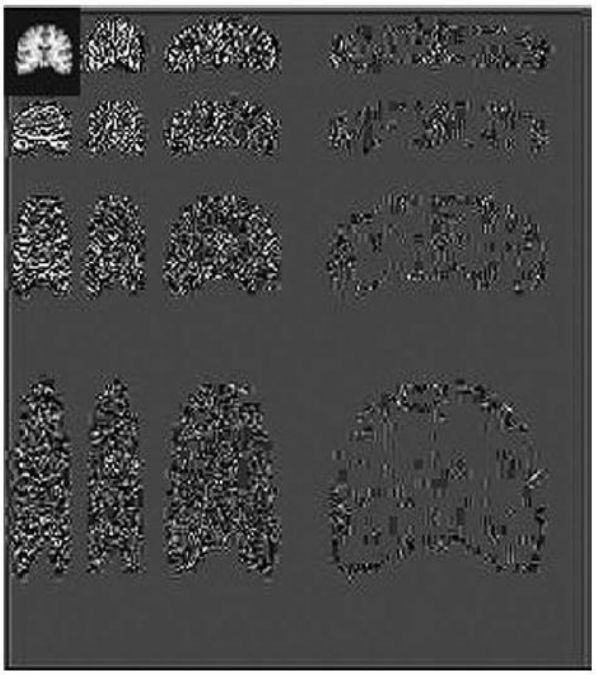


Figura 10. Transformada Wavelet de la figura 4 [4].

2.6.1 TRANSFORMADA DISCRETA DE WAVELET BIDIMENSIONAL.

La Transformada Wavelet se utiliza para la compresión tanto de imágenes como de audio. En el proceso de análisis de la Wavelet, las señales son representadas utilizando un grupo de funciones básicas producidas por el desplazamiento y el escalado de una función madre o función principal. La transformada Wavelet es una descomposición de una señal en frecuencias, La DWT bidimensional se ha aplicado en análisis multi-resolución visión por computador, y compresión de imágenes [21].

La DWT bidimensional trabaja sobre una señal 2-D como puede ser una imagen. utilizamos filtros 2-D Estos filtros 2-D pueden ser separables o no separables, donde un filtro 2-D f(n1, n2) es separable si es expresable como f(n1, n2)= f1(n1) f2(n2). La DWT 2-D separable descompone una imagen Si(n1, n2) en la imagen promedio y tres imágenes detalle, de acuerdo con las expresiones [21].

$$S_{i+1}(n_1, n_2) = \sum_{k1}\sum_{k2} g(k_1)g(k_2)S_i(2n_1 - k_1, 2n_2 - k_2)$$

$$W^1{}_{i+1}(n_1, n_2) = \sum_{k1}\sum_{k2} g(k_1)h(k_2)S_i(2n_1 - k_1, 2n_2 - k_2)$$

$$W^2{}_{i+1}(n_1, n_2) = \sum_{k1}\sum_{k2} h(k_1)g(k_2)S_i(2n_1 - k_1, 2n_2 - k_2)$$

$$W^3{}_{i+1}(n_1, n_2) = \sum_{k1}\sum_{k2} h(k_1)h(k_2)S_i(2n_1 - k_1, 2n_2 - k_2)$$

donde H(z) y G(z) son los filtros Wavelet 1-D. La señal $S_{i+1}(n_1, n_2)$ es un suavizado de baja resolución de la imagen $S_i(n_1, n_2)$ Este suavizado se calcula desde $S_i(n_1, n_2)$ mediante un filtro pasa-baja y diezmando por 2 a lo largo de filas y columnas. Las señales $W^1{}_{i+1}(n_1, n_2), W^2{}_{i+1}(n_1, n_2), W^3{}_{i+1}(n_1, n_2)$ contienen el detalle de $S_i(n_1, n_2)$ El nivel 1 de la DWT 2-D se calcula como muestra la figura 11.

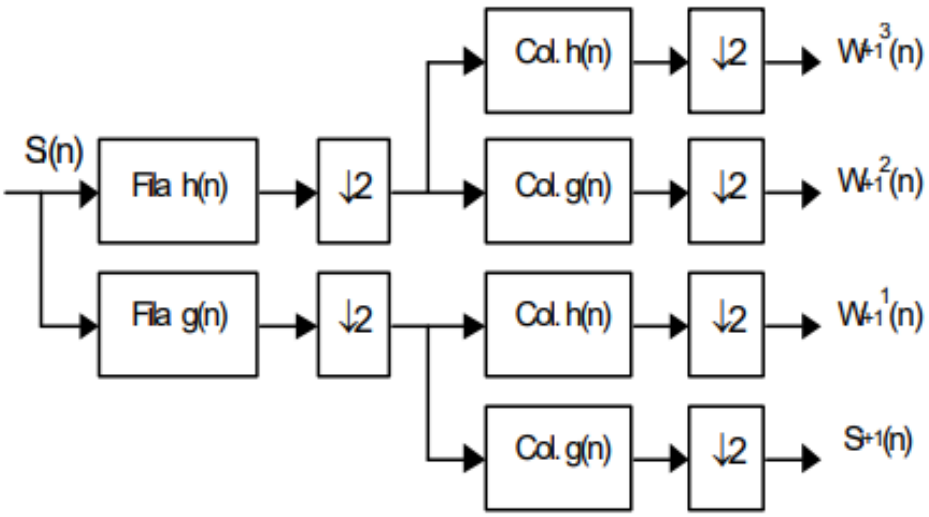


figura 11 diagrama de bloques del banco utilizado [21]

2.6.2 CALCULO DE LA DWT BIDIMENSIONAL EN MATLAB .

El algoritmo de descomposición de ondículas 2-D para imágenes es similar al caso unidimensional. Las funciones de escala y ondícula bidimensionales se obtienen tomando los productos tensoriales de las funciones de escala y ondícula unidimensionales. Este tipo de DWT bidimensional conduce a una descomposición de los coeficientes de aproximación en el nivel *j* en cuatro componentes: la aproximación en el nivel *j* + 1 y los detalles en tres orientaciones (horizontal, vertical y diagonal). El siguiente cuadro (figura 12), describe los pasos básicos de descomposición de imágenes [21].

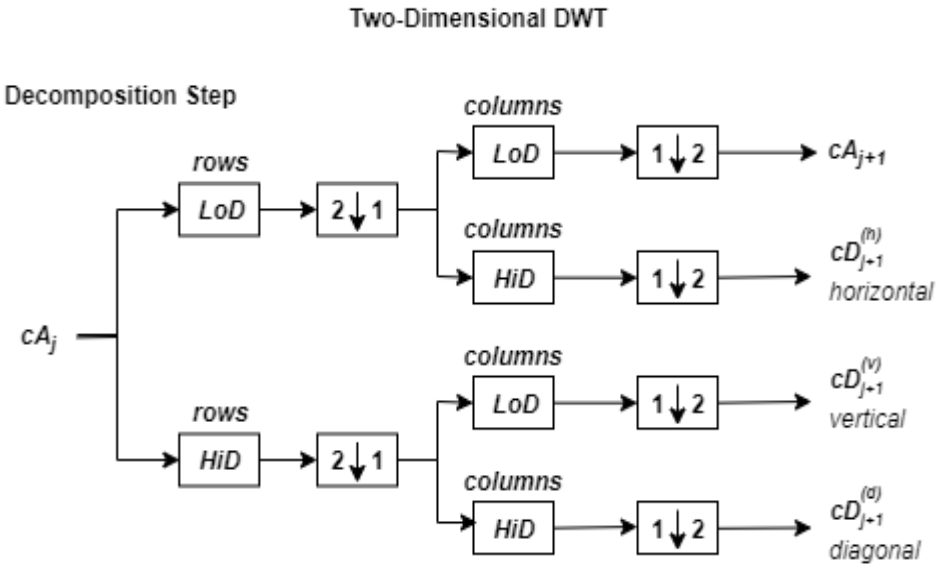


figura 12 Banco de filtros DWT Bidimensional [22].

La transformada de ondícula discreta (DWT) se computa con dwt2 este parámetro devuelve los coeficientes de la matriz de aproximación CA matriz de detalle coeficientes CH, CV y CD (horizontal, vertical, diagonal, respectivamente).

2.6.2.1 ONDITA MADRE HAAR

la ondícula de Haar o la wavelet de Haar es una cierta secuencia de funciones. Ahora se le reconoce como la primera wavelet conocido. Esta secuencia fue propuesta en 1909 por Alfred Haar. Haar usó estas funciones para dar un ejemplo de un sistema ortonormal contable para el espacio de las funciones de cuadrado integrable en la recta real. El estudio de las wavelets, e incluso el término "wavelet", no vinieron hasta mucho después. Como un caso especial de wavelet de Daubechies, también es llamado D2. La desventaja técnica de la wavelet de Haar es que no es continuo y por lo tanto no derivable [22].

La función wavelet madre de las funciones de Haar $\psi(t)$ puede ser descrita como:

$$\psi(t) = \begin{cases} 1 & 0 \le t \le \frac{1}{2} \\ -1 & \frac{1}{2} \le t \le 1 \\ 0 & en\ otros\ casos \end{cases}$$

Ecuación 1 Función wavelet madre Haar [22]

2.6.2.2 EFECTOS DE BORDE Y MODO DE EXTENSIÓN ZPD

Clásicamente, el DWT se define para secuencias con una longitud de alguna potencia de 2, y se necesitan diferentes formas de extender muestras de otros tamaños. Los métodos para extender la señal incluyen relleno de ceros, relleno suave, extensión periódica, y replicación del valor límites metrización) [21].

El algoritmo básico para el DWT no se limita a la longitud diádica y se basa en un esquema simple:

- convolución y reducción de resolución. Como es habitual, cuando se realiza una convolución en señales de longitud finita, surgen distorsiones de borde. Para hacer frente a las distorsiones de los bordes, el borde debe tratarse de manera diferente a las otras partes de la señal [21].

A menudo, es preferible utilizar esquemas simples basados en la extensión de la señal en los límites. Esto implica el cálculo de algunos coeficientes adicionales en cada etapa del proceso de descomposición para obtener una reconstrucción perfecta. Cabe señalar que se necesita una extensión en cada etapa del proceso de descomposición [21].

Zero-padding ('zpd'): este método asume que la señal es cero fuera del soporte original. La desventaja del relleno con ceros es que las discontinuidades se crean artificialmente en la frontera[21].

2.7 PROCESAMIENTO DE IMÁGENES.

Al conjunto de técnicas y procesos para descubrir o hacer resaltar información contenida en una imagen usando como herramienta principal una computadora se le conoce como Procesamiento digital de imágenes (PDI). Hoy en día, el PDI es un área de investigación muy específica en computación y está muy relacionada con el procesamiento digital de señales. Esta Relación estriba en el hecho de que en esencia el PDI es una forma muy especial del Procesamiento digital de señales en dos o tres dimensiones [20].

2.7.1 APLICACIONES EN LA MÉDICINA.

Durante los últimos 15 años, un número creciente de técnicas referentes a imágenes digitales ha sido introducido en la práctica médica. Muchos radiólogos y personal de laboratorios médicos conocen y manipulan imágenes digitales como las producidas por Tomografía Asistida por Computadora (tac), resonancia magnética y por métodos de medicina nuclear [20].

Debido al desarrollo, y gran factibilidad de ejecución de las computadoras, las imágenes en medicina, que eran tradicionalmente grabadas sobre películas, ahora pueden manipularse en forma digital. El desarrollo de las técnicas de almacenamiento de imágenes digitales en medicina ha estado acompañado por un incremento en el uso de herramientas de manipulación de imágenes [20].

Estas últimas se pueden dividir en dos categorías

- Herramientas de propósito general para procesar imágenes, las cuales se utilizan para manipular y modificar la presentación de las imágenes. éstas pueden ser: ajuste de la intensidad y contraste; compactación de la imagen y rotación de la misma; filtros para suavizar y resaltar imágenes, y algoritmos para la extracción de propiedades como textura, y otros.

- Técnicas para el análisis y técnicas de medición en la evaluación cuantitativa de las imágenes.

2.8 INTELIGENCIA ARTIFICIAL

La inteligencia artificial (IA) tiene por objetivo el estudio y el análisis del comportamiento humano en los ámbitos de la comprensión, de la percepción, de la resolución de problemas y de la toma de decisiones con el fin de poder reproducirlos con la ayuda de un computador. De esta manera, las aplicaciones de la IA se sitúan principalmente en la simulación de actividades intelectuales del hombre. Es decir, imitar por medio de máquinas, normalmente electrónicas, tantas actividades mentales como sea posible, y quizás llegar a mejorar las capacidades humanas en estos aspectos. Un programa de IA manipula informaciones simbólicas bajo la forma de conceptos, de objetos o reglas. [10]

2.8.1 RED DE NEURONAS

Una Red de Neuronas Artificiales (en adelante, RNA) es un paradigma de procesamiento de información inicialmente inspirado en el modo en el que lo hace el cerebro. El elemento clave de este paradigma es su estructura. Las RNA están compuestas por un cierto número de elementos de procesamiento o neuronas que trabajan al unísono para resolver un problema específico. Las redes neuronales actuales se basan en el modelo matemático de neurona propuesto por McCulloch y Pitts en 1943 [11].

En dicho modelo (Figura 13) cada neurona recibe un conjunto de entradas {x1, x2, xD} y devuelve una única salida y. Además, dentro de una RNA existen numerosas conexiones entre las distintas neuronas que la forman. Estas conexiones simulan las conexiones interneuronales del cerebro y, al igual que éstas, pueden establecerse con mayor o menor

intensidad. En el caso de las RNA está intensidad la determinan los pesos sinápticos (o simplemente pesos). De este modo, cada entrada xi de una neurona se encuentra afectada por un peso wi [11].

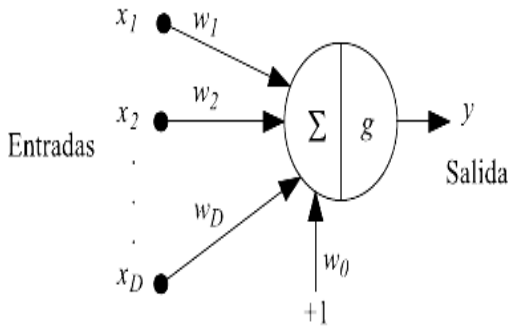


Figura 13. Modelo neuronal de McCulloch-Pitts [11].

2.8.2 REDES NEURONALES CONVOLUCIONALES

Las Redes Neuronales Convolucionales también conocidas por sus siglas en ingles de CNN, son un tipo de redes neuronales multicapa que se especializan en el reconocimiento de patrones en imágenes. Como las estructuras anteriores las neuronas artificiales y las redes neuronales artificiales las CNN están inspiradas en características bilógicas en este caso las redes convolucionales están basadas en el córtex visual de ojo humano. Se basan en la operación de la convolución, lo cual radica en filtrar una imagen utilizando una serie de kernels que consiste en tomar grupos de pixeles próximos y cada píxel de salida es combinación lineal de los píxeles de entrada el formato de entrada es el alto x ancho x profundidad de la imagen, ver figura 14. [23]

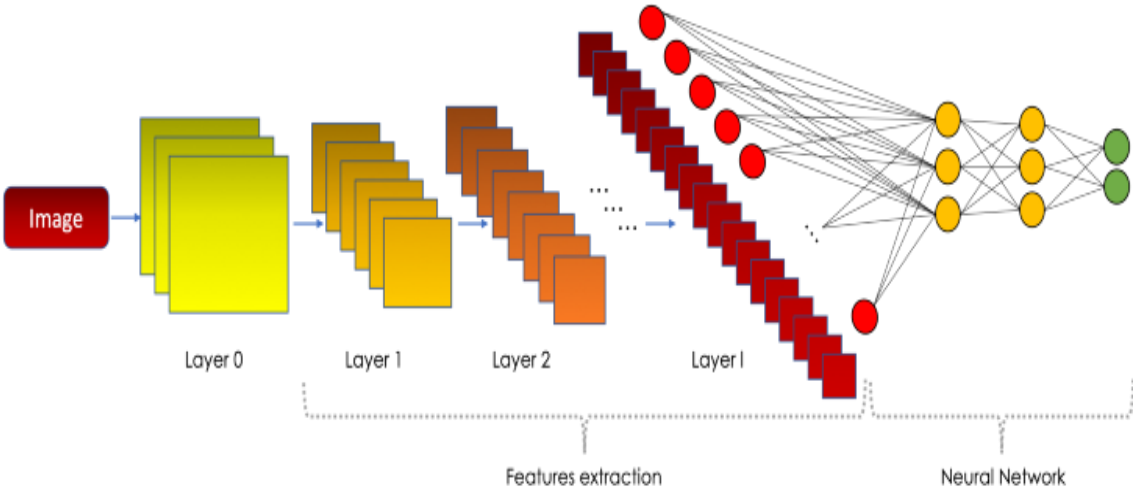


figura 14 Diagrama estructural de una red neuronal convolucional [23]

2.8.2.1 MAXPOOLING

Después de una operación de convolución, solemos realizar agrupaciones para reducir la dimensionalidad sin tocar la profundidad. Esto nos permite reducir el número de parámetros, lo que acorta el tiempo de entrenamiento y combate el sobreajuste. Las capas agrupadas reducen la resolución de cada mapa de características de forma independiente, reduciendo la altura y el ancho, manteniendo intacta la profundidad [23].

El tipo más común de agrupación es la agrupación máxima (maxPooling), que solo toma el valor máximo en la ventana de agrupación. Al contrario de la operación de convolución, la agrupación no tiene parámetros. Desliza una ventana sobre su entrada y simplemente toma el valor máximo en la ventana. Similar a una convolución, especificamos el tamaño de la ventana y el paso. En la figura 15 está el resultado de un maxPooling usando una ventana de 2x2 y un paso 2. Cada color denota una ventana diferente. Dado que tanto el tamaño de la ventana como el paso son 2, las ventanas no se superponen. [23]

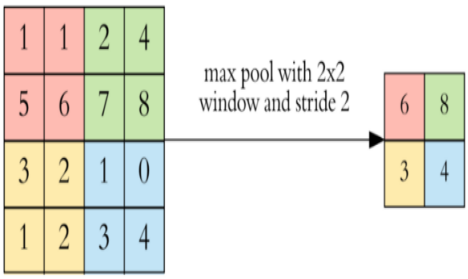


figura 15 MAXPOOLING 2X2 en una imagen 4x4 [23]

2.8.2.1 PADDING O RELLENO

El filtro de convolución debe estar contenido en la entrada (figura 16), Si queremos mantener la misma dimensionalidad, podemos usar el relleno o Padding para rodear la entrada con ceros.

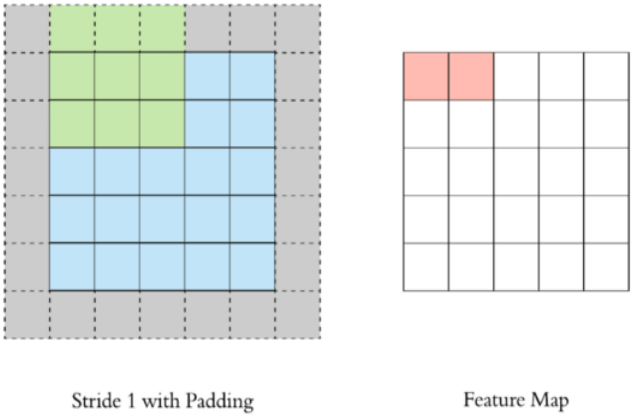


figura 16 stride y Padding "same" [23]

2.8.2.3 FUNCIONES DE ACTIVACIÓN

La función de activación devuelve una salida de acuerdo al conjunto de datos de entrada, esta función es de gran importancia en las redes neuronales artificiales debido a que le da al modelo propiedades no lineales logrando adaptarse a cualquier tipo de entrada, una red neuronal sin ningún tipo de función de activación no tendría sentido puesto que se comportaría como un modelo lineal incapaz de solucionar problemas no lineales [23].

- **RELU**: Es una de las funciones más utilizadas en el contexto actual puesto que consigue una optimización en la rapidez del aprendizaje de las redes neuronales, en la ecuación 3 si la función recibe valores de entrada negativos el resultado será cero, pero si el valor de argumento es mayor igual a cero mantendrá su valor. En comparación con otras funciones de activación RELU no tiene zonas de saturación esto significa que los pesos de conexión son los óptimos ya que al no tener variación de saturación hay una mayor flexibilidad en los pesos. El gradiente de esta función será cero en el segundo cuadrante y uno en el primer cuadrante. Cuando se tiene que la función es igual a cero y su derivada también lo es se genera lo que es la muerte de neuronas, a pesar que puede ser un inconveniente en algunos casos permite la regularización Dropout [23].

$$f(z) = \max(0, z) = \begin{cases} 0, & z < 0 \\ z, & z \geq 0 \end{cases}$$

Ecuación 2 Función de activación Relu

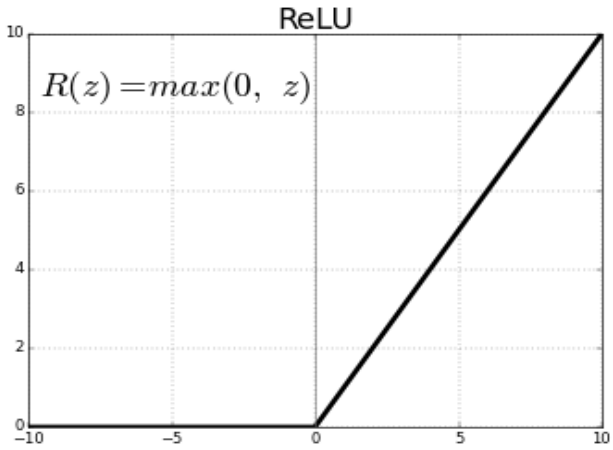


figura 17 Función de activación Relu [23]

- **SOFTMAX:** Esta función se implementa cuando hay múltiples clases y es necesario asignar probabilidades a cada una de las clases en la capa de salida del clasificador, Esta función de activación devuelve la distribución de probabilidad de cada una de las clases soportadas en el modelo. La función Softmax calcula la distribución de probabilidades del evento sobre 'n' eventos diferentes. En términos generales, esta

función calculará las probabilidades de cada clase objetivo sobre todas las clases objetivo posibles. Más tarde, las probabilidades calculadas serán útiles para determinar la clase objetivo para las entradas dadas. La principal ventaja de usar Softmax es el rango de probabilidades de salida. El rango será de 0 a 1, y la suma de todas las probabilidades será igual a uno. Si la función softmax utilizada para el modelo de clasificación múltiple devuelve las probabilidades de cada clase y la clase objetivo tendrá una probabilidad alta. ver ecuación 3. [23]

por lo que la salida será el que tenga la mayor probabilidad, cabe mencionar que la suma de todas las probabilidades será igual a uno. [23]

$$\Phi(x_i) = \frac{e^{x_i}}{\sum_{k=0}^{k} e^{k_j}}, i = 0, 1, 2, \dots k$$

Ecuación 3 Función de activación Softmax

2.9 MATLAB

MATLAB es una plataforma de programación y cálculo numérico utilizada por millones de ingenieros y científicos para analizar datos, desarrollar algoritmos y crear modelos. MATLAB® combina un entorno de escritorio perfeccionado para el análisis iterativo y los procesos de diseño con un lenguaje de programación que expresa las matemáticas de matrices y arrays directamente. disponible para las plataformas Unix, Windows, macOS y GNU/Linux. Entre sus prestaciones básicas se hallan la manipulación de matrices, la representación de datos y funciones, la implementación de algoritmos, la creación de interfaces de usuario (GUI) y la comunicación con programas en otros lenguajes y otros dispositivos hardware [22].

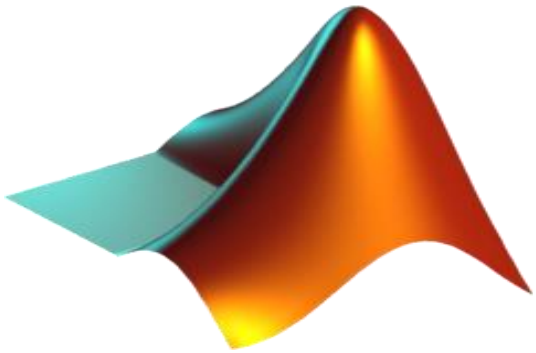


figura 18 icono de MATLAB [22]

2.10 GOOGLE COLABORATORY

Google colaboratory o más conocido como Google colab, permite codificar y ejecutar código **Python** en la nube. Las ventajas de esta plataforma es que no se necesita de ninguna configuración en la maquina pues solo es necesario el acceso a internet, permite el acceso de hardware especializado como GPUs y TPUs. Esta plataforma está pensada para estudiantes, científico de datos o investigadores de IA. La interfaz de Google colab es igual al jupyter notebook por lo tanto permite desarrollar código Python de manera dinámica posibilitando la ejecución de código como la escritura de texto junto a imágenes, HTML, latex y mucho más, lo anterior agrega una capa de interactividad tomando el código una forma de documento interactivo. [24]

La razón principal de uso de este entorno es la usabilidad de tarjetas gráficas de alto desempeño alojadas en la nube, y su ejecución por celdas, evitando ejecutar todo el algoritmo a la vez y optimizando el uso de memoria RAM. Solo es necesario una cuenta de Google y almacenamiento de los datos a procesar alojados en drive, cada vez que se ingresa a Google colab el usuario puede gestionar una máquina virtual por 12 horas, sin costo alguno, hay una versión paga de Google colab sin embargo para el desarrollo de este proyecto la versión gratuita satisface todos nuestros requerimientos.

2.10.1 UNIDAD DE PROCESAMIENTO GRÁFICO (GPU)

GPU por sus siglas en ingles Unidad de procesamiento de gráfico, es un tipo de procesador especializado en el procesamiento de gráficos y operaciones de como flotante. Una GPUs se basa en una placa o chip de silicio que alberga una serie de millones de transistores implementando una arquitectura especifica. Una forma sencilla de comprender la diferencia entre una GPU y una CPU es comparar la forma en que procesan las tareas (figura 19). Una CPU tiene unos cuantos núcleos optimizados para el procesamiento en serie secuencial, mientras que una GPU cuenta con una arquitectura en paralelo enorme que consiste de mieles de núcleos más pequeños y eficaces, y que se diseñaron para resolver varias tareas al mismo tiempo. gracias a esto la GPUs aligera la carga computacional del procesador central en aplicaciones como machine learning, ciencia de datos, etc. [25]

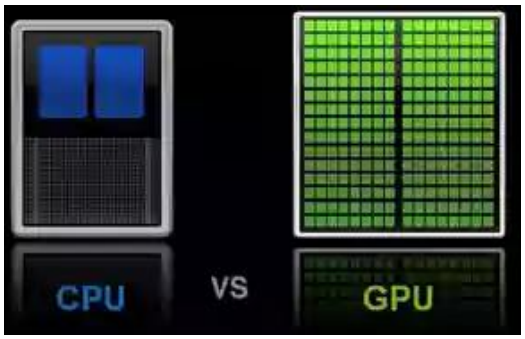


figura 19 Diferencias en la arquitectura por núcleos de una CPU y GPU [25]

2.10.2 PYTHON

Python es un potente lenguaje de programación fácil de aprender. Tiene estructuras de datos eficientes de alto nivel y un enfoque simple pero efectivo para la programación orientada a objetos. La elegante sintaxis y la escritura dinámica de Python, junto con su naturaleza interpretada, lo convierten en un lenguaje ideal para la creación de scripts y el rápido desarrollo de aplicaciones en muchas áreas en la mayoría de las plataformas. El intérprete de Python y la extensa biblioteca estándar están disponibles gratuitamente en formato fuente o binario para todas las plataformas principales desde el sitio web de Python. El mismo sitio también contiene distribuciones y referencias a muchos módulos, programas y herramientas de Python gratuitos de terceros, y documentación adicional[26].

figura 20. Google Colab para ejecución de código Python [26].

Al instalar el sistema operativo Raspbian por defecto la raspberry ya tiene instalado una versión de Python, muchos paquetes no están instalados y hay que instalarlos desde la consola de comandos como se verá en la sección ***, y otras Librerias no están disponibles directamente para la descarga en Raspbian sin embargo la comunidad de Python, y raspberry se han dedicado a publicar repositorios en GITHUB para la implementación de bibliotecas de Python en la raspberry, al tratarse de un sistema Debian basado en Linux, los repositorios se pueden clonar directamente desde GITHUB.

2.10.2.1 LIBRERIAS EN PYTHON DEDICADAS AL PROCESAMIENTO DE IMÁGENES.

- **OPENCV:** Es una biblioteca de código abierto de visión por computadora y de aprendizaje automático. Está escrita de forma nativa en c++, esta librería consta de más de 2500 algoritmos completamente funcionales y optimizados, que abarca un conjunto diverso de algoritmos de última generación. Estos algoritmos pueden emplearse para la detección y reconocimiento de rostros, identificar objetos, clasificar acciones humanas, rastreo de movimiento, detección de objetos y un largo etc. Tiene soporte e interfaces para múltiples entornos y lenguajes como lo son c++, Python, MATLAB. Es compatible con los sistemas operativos Windows, Linux, Android y mac OS. [27]

Para el tratamiento de imágenes a comprobar luego de creada la red neuronal usamos 3 funciones principales:

a) cv2.cvtColor(img,cv2.COLOR_BGR2RGB): Por defecto la librería OpenCV al cargar una imagen establece como canal principal de color el canal Azul, el segundario el canal verde y por último el canal rojo, pero la función cuando se le cargan las imágenes a la red neuronal está establecidas al formato RGB, con la función anterior transformamos el espacio de color BGR propio de OPENCV a formato RGB.

b) cv2.resize(img_color,(256,256)): Para lograr que la red neuronal clasifique la imagen, todo el conjunto de datos debe tener el mismo tamaño, por ello las imágenes se redimensionan a un tamaño de 256 pixeles de alto X 256 pixeles de ancho.

c) cv2.flip(img_resize,-1): Para comprobar la red neuronal con el conjunto de datos aumentado utilizamos la función flip de OpenCV que permite girar una imagen en el eje x la imagen 180 Grados cuando se establece -1.

2.10.2.2 LIBRERIAS EN PYTHON DEDICADAS A INTELIGENCIA ARTIFICIAL

- TENSORFLOW: Compila y entrena modelos innovadores sin sacrificar la velocidad ni el rendimiento. TensorFlow te brinda la flexibilidad y el control con funciones como la API funcional de Keras y la API de subclases de modelos para la creación de topologías complejas. Para lograr un prototipado fácil y una depuración rápida, usa la ejecución inmediata. TensorFlow también es compatible con un ecosistema de potentes bibliotecas de complementos y modelos para experimentar, entre los que se incluyen Ragged Tensors, TensorFlow Probability, Tensor2Tensor y BERT [28].

- KERAS: Keras es una biblioteca de Redes Neuronales de Código Abierto escrita en Python. Es capaz de ejecutarse sobre TensorFlow, Microsoft Cognitive Toolkit o Theano. Está especialmente diseñada para posibilitar la experimentación en más o menos poco tiempo con redes de Aprendizaje Profundo. ofrece API consistentes y simples, minimiza la cantidad de acciones del usuario necesarias para los casos de uso comunes y proporciona mensajes de error claros y procesables. También cuenta con una extensa documentación y guías para desarrolladores. Construido sobre TensorFlow 2.0 , Keras es un marco de trabajo sólido de la industria que puede escalar a grandes grupos de GPU o un pod completo de TPU. Puede exportar modelos de Keras a JavaScript para que se ejecuten directamente en el navegador, a TF Lite para que se ejecuten en iOS, Android y dispositivos integrados. También es fácil servir modelos de Keras a través de una API web [29].

2.10.2.3 LIBRERIAS EN PYTHON DEDICADAS A LA ESCRITURA DE INTERFAZ GRÁFICA

- **TKINTER:** Proporciona un conjunto de herramientas robusto e independiente de la plataforma para administrar ventanas. Disponible para desarrolladores a través del paquete Tkinter y sus extensiones, los módulos tkinter.tix y tkinter. El paquete Tkinter es una fina capa orientada a objetos encima de Tcl / Tk. Para usarlo tkinter, no necesita escribir código Tcl, pero deberá consultar la documentación de Tk y, ocasionalmente, la documentación de Tcl. Tkinter es un conjunto de envoltorios que implementan los widgets Tk como clases de Python. Las principales virtudes son que es rápido y que normalmente viene incluido con Python [30].

2.10.2.4 LIBRERIAS EN PYTHON DEDICADAS A PUBLICACIÓN Y DISTRIBUCIÓN DE APLICACIONES

- **PYINSTALLER**: Funciona con Python 3.5—3.9, crea ejecutables más pequeños gracias a la compresión transparente, es completamente multiplataforma y utiliza el soporte del sistema operativo para cargar las bibliotecas dinámicas, asegurando así una compatibilidad total. El objetivo principal de PyInstaller es ser compatible con paquetes de terceros listos para usar. Esto significa que, con PyInstaller, todos los trucos necesarios para que los paquetes externos funcionen ya están integrados dentro de PyInstaller, por lo que no se requiere la intervención del usuario. Nunca se le pedirá que busque trucos en wikis y aplique modificaciones personalizadas a sus archivos o scripts de configuración. Por ejemplo, bibliotecas como PyQt, Django o Matplotlib son totalmente compatibles , sin tener que manejar complementos o archivos de datos externos manualmente [31].

2.11 RASPBERRY PI4

La Raspberry pi 4 (Figura 21), es un sistema embebido diseñado para realizar funciones que permiten aportar inteligencia empleando procesadores digitales (CPU) en formato microprocesador, microcontrolador o procesador digital de señales DSP [12]. La tarjeta Raspberry PI es un dispositivo que cuenta con todas las características de un miniordenador personal que proporciona un procesador ARM, memoria RAM de 4gb DDR4, una GPU, puertos USB/UART y puertos de entradas y salidas digitales GPIO, audio, 2 salida de video micro HDMI y slot para tarjeta SD, y tiene un entorno de propósito general de programación de alto nivel, denominado Python, el cual cuenta con varias características generales que lo hacen especial, por ser fácil de leer y simple de implementar, además de ser un código abierto (de libre uso) [12].

Figura 21. Raspberry pi 4 modelo b [12].

2.11.1 CONFIGURACIÓN DEL SISTEMA OPERATIVO (S.O)

Desde la página oficial de raspberry y dirigiéndonos a la sección de descargas podemos seleccionar el sistema operativo a instalarse, este sistema operativo esta soportado para las versiones anteriores a la raspberry pi4, y se encuentran disponible con interfaz de usuario, basta con comprar una tarjeta microSD de tipo 10 de clase A tipo ultra con capacidad mayor a 10GB, ya que el sistema operativo va a utilizar 8 GB, una vez descargada la imagen debemos flashear la memoria con el sistema operativo, hay múltiples programas como lo es BALENAETCHER, previamente la memoria debe ser formateada en formato FAT32, cuando la capacidad de almacenamiento de la microSD es superior a 32GB, hay que emplear programas para dar formato a FAT32, el programa BALENAETCHER se le debe cargar el archivo .img y empezará a cargar el sistema operativo a la memoria, cuando se instala un sistema operativo en cualquier equipo con procesador la unidad de almacenamiento particiona las unidades de almacenamiento.

2.11.2 CONEXIÓN DE RASPBERRY PI4 A PANTALLA HDMI

Al iniciar por primera vez, el bootloader del sistema embebido reconoce la resolución de la pantalla, por ello es necesario que al encenderse la tarjeta la pantalla HDMI se encuentre encendida, seguido podemos notar como en el sistema operativo se encuentran instalados programas orientados a la enseñanza de programación, juegos, reproductores, navegadores, configuración y gestión de plataformas de conectividad como lo son una conexión a wifi, y una conexión a bluetooth.

2.11.3 CONEXIÓN DE RASPBERRY PI4 A INTERNET

Una vez conectados a nuestra red wifi, o por puerto ethernet podemos conectarnos a internet y/o acceder a la interfaz de comandos para instalar y actualizar paquetes o Librerias de Python.

2.11.4 CONEXIÓN POR ESCRITORIO REMOTO

Abrir el terminal de comandos en raspberry y ejecutar

$$sudo\ apt - get\ install\ xrdp$$

Una vez estemos conectados a internet, buscamos la dirección ip asignada a la tarjeta y buscamos el símbolo de sistema en Windows y ponemos la dirección ip, nos pedirá el nombre de host, y la clave, por defecto es “pi”, y “raspberry”, respectivamente.

2.11.5 INSTALACIÓN DE PYTHON IDE

Abrir el terminal de comandos en raspberry y ejecutar

$$sudo\ apt - get\ install\ python3$$
$$sudo\ apt - get\ install\ ide$$

$$sudo\ apt - get\ install\ python - ide$$

$$sudo\ apt - get\ install\ python3 - numpy$$

$$sudo\ apt - get\ install\ python3 - scipy$$

2.11.6 INSTALACIÓN DE PAQUETES DE PYTHON

- **Opencv:** Librería de visión artificial

  Abrir el terminal de comandos en raspberry y ejecutar

  $$sudo\ apt - get\ install\ libhdf5 - dev\ libhdf5 - serial - dev\ libatlas - base - dev\ libjasper - dev\ libqtgui4\ libqt4 - test$$

  $$pip3\ install\ opencv - contrib - python == 4.1.0.25$$

- **Tensorflow:** librería de inteligencia artificial

  Abrir el terminal de comandos en raspberry y ejecutar

  $$sudo\ apt - get\ install\ libhdf5 - dev\ libc - ares - dev\ libeigen3 - dev$$

$$sudo\ apt - get\ install\ libatlas - base - dev\ libatlas3 - base$$

$$sudo\ pip3\ install\ h5py == 2.10.0$$

$$sudo\ pip3\ install\ -U\ --user\ six\ wheel\ mock$$

$$wget\ https://github.com/Qengineering/Tensorflow - Raspberry - Pi/raw/master/tensorflow - 1.15.2 - cp37 - cp37m - linux\_armv7l.whl$$

$$sudo\ -H\ pip3\ install\ tensorflow - 1.15.2 - cp37 - cp37m - linux\_armv7l.whl$$
$$sudo\ apt - get\ install\ libhdf5 - dev\ libc - ares - dev\ libeigen3 - dev$$

$$sudo\ reboot$$

- **Keras:** librería de Optimización de inteligencia artificial

  Abrir el terminal de comandos en raspberry y ejecutar

  $$sudo\ pip3\ install\ keras$$

- **Tkinter:** librería de Creación de interfaces gráficas

  Abrir el terminal de comandos en raspberry y ejecutar

  $$sudo\ apt - get\ install\ python - tk$$

- **PyInstaller:** Librería de distribución y publicación de aplicaciones y ejecutables

  Abrir el terminal de comandos en raspberry y ejecutar

  $$git\ clone\ https://github.com/pyinstaller/pyinstaller$$

  $$sudo\ python3\ setup.py\ install$$

- **Picocom:** Librería para establecer comunicación UART entre raspberry pi4 y sim808

  Abrir el terminal de comandos en raspberry y ejecutar

  $$sudo\ apt - get\ install\ picocom$$

- **Pyrebase:** Librería para establecer comunicación entre Firebase y Raspberry pi 4 y raspberry pi3

Abrir el terminal de comandos en raspberry y ejecutar

$$sudo\ pip3\ install\ pyrebase$$

- **FSWebcam:** Librería para usar múltiples cámaras webcam de conexión USB
  Abrir el terminal de comandos en raspberry y ejecutar

$$sudo\ apt - get\ install\ fswebcam$$

2.11.7 ACTIVACIÓN DE PUERTOS GPIO

La raspberry proporciona 40 pines de comunicación de entrada y salida digital de propósito general, Antes de poder establecer una comunicación por cualquier protocolo de comunicación debemos activar la opción de puertos GPIO, y la opción de comunicación serial.

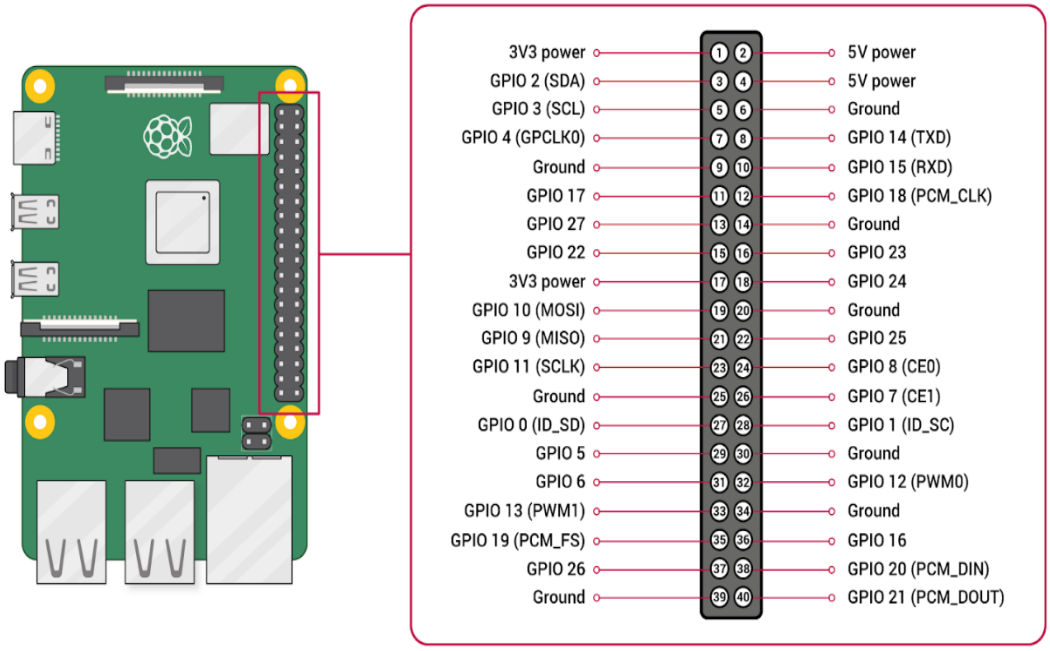


figura 22 Puertos GPIO de la raspberry pi4 [12]

Abrir el terminal de comandos en raspberry y ejecutar

$$sudo\ raspi - config$$

Seleccionamos las interfaces con las que vayamos a trabajar y automáticamente el sistema pide reiniciar para aplicar los ajustes.

2.11.8 PROTOCOLO DE COMUNICACIÓN UART

UART (universal asynchronous receiver / transmitter, por sus siglas en inglés), define un protocolo o un conjunto de normas para el intercambio de datos en serie entre dos dispositivos. UART es sumamente simple y utiliza solo dos hilos entre el transmisor y el receptor para transmitir y recibir en ambas direcciones. Ambos extremos tienen una conexión a masa. La comunicación en UART puede ser simplex (los datos se envían en una sola dirección), semidúplex (cada extremo se comunica, pero solo uno al mismo tiempo), o dúplex completo (ambos extremos pueden transmitir simultáneamente). En UART, los datos se transmiten en forma de tramas figura 23 [32].

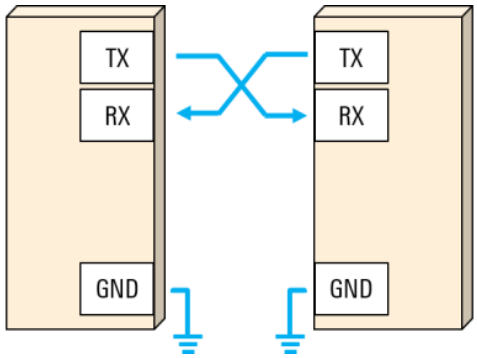


figura 23 conexionado de dos dispositivos a través del protocolo de comunicación UART [32]

2.12 FIREBASE

Firebase te ayuda a compilar y ejecutar apps exitosas Con el respaldo de Google y el aprecio de equipos de desarrollo de apps, desde startups hasta empresas mundiales está pensada para sincronización de datos en la nube gestión de almacenamiento, gestión de bases de datos, gestión de bases de datos en tiempo real, sincronización con aplicaciones web, aplicaciones móviles y el internet de las cosas (figura 24), para acceder a Firebase solo es necesario una cuenta de Google, y una conexión a internet, Firebase soporta múltiples plataformas y además se integra a múltiples aplicaciones para el desarrollador, el estudio científico, y además es de acceso gratuito, existe una versión pagada pero la versión original suministra 10 Gb de almacenamiento, y nuestro proyecto usa menos de ese almacenamiento.

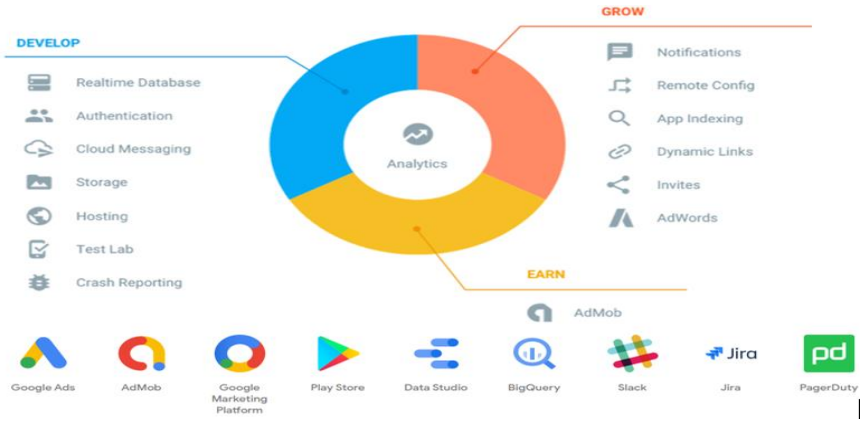


figura 24 Herramientas de Firebase y disposición con múltiples plataformas y aplicaciones web o aplicaciones móviles

3.

Metodología.

3. METODOLOGÍA

Para el desarrollo de este proyecto en primer lugar se realiza la carga del set de datos y su tratamiento en el software MatLab, los datos se encuentran en formato .mat y se tiene un total de 3064 objetos, donde cada objeto contiene información acerca de la etiqueta del tipo de tumor cerebral 1 para meningioma, 2 para tumor glioma y 3 para tumor pituitary. Además, la identificación de los pacientes, la imagen del tumor cerebral en formato int16, el borde del tumor almacenado en un vector de 4 posiciones, en donde la posición 1 indica la coordenada x1 de la zona donde se ubica el tumor, en la posición la coordenada y1, donde se encuentra el inicio del tumor, en la 3 posición la coordenada x2, y por último en la 4 posición se ubica la coordenada y2, además podemos encontrar una imagen binaria en donde se aprecia la región de cada tipo de tumor.

3.1 CARGA DE DATOS EN MATLAB

El conjunto de datos contiene 3064 imágenes de tumor cerebral con imágenes contrastadas en T1, que pertenecen a tres tipos de tumor, tumor meningioma (708 datos), tumor glioma (1426), y tumor pituitary (930 datos), cada dato contiene la etiqueta del tumor cerebral, la identificación del paciente, la imagen del tumor cerebral contrastada en T1, el borde del tumor y máscara binaria donde se aloja el tumor cerebral.

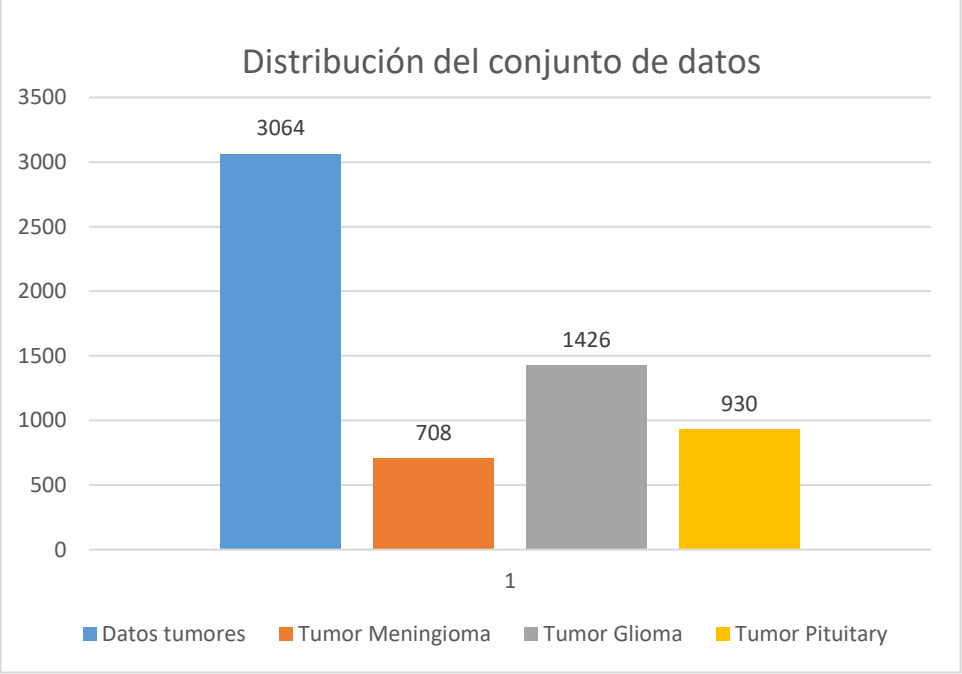


figura 25 Cantidad de datos de tumor cerebral

Para extraer la información de cada dato, fue necesario cargar los 3064 datos en MatLab y copiar los datos en celdas, para luego extraer la identificación del paciente, la imagen del tumor cerebral, el borde del tumor, y la máscara binaria donde se encuentra la región del tumor cerebral.

A continuación, podemos observar cómo se guarda la información en celdas para los primeros 5 datos, con su correspondiente etiqueta, para distinguir según el tipo de dato a que tipo de tumor pertenecen los datos. Similarmente se cargaron los 3064 datos.

```
close all,clear, clc;

%% DATA 1
load('1.mat');
t=cjdata;
L{1,t.label}=t;

load('2.mat');
t=cjdata;
L{2,t.label}=t;

load('3.mat');
t=cjdata;
L{3,t.label}=t;

load('4.mat');
t=cjdata;
L{4,t.label}=t;

load('5.mat');
t=cjdata;
L{5,t.label}=t;
```

Luego de cargados los 3064 datos, los guardamos en la carpeta raíz, pero al momento de cargarlos la memoria RAM se saturaba y se cerraba el programa, por lo que decidimos cargar los 2500 primeros datos en un archivo. Mat y a partir de los datos de 2500 hasta 3064 en otro archivo. Mat

Teniendo los dos tipos de datos cargamos en otro script como se muestra a continuación, ordenamos según la etiqueta de cada dato. Ya en 3 celdas tenemos ordenados los tumores cerebrales, en la primera celda guardamos los datos de los tumores meningioma, en la segunda celda guardamos los datos de los tumores glioma, y en la tercera celda guardamos los datos de los tumores pituitary

```
%%  CARGAR LOS PRIMEROS 2500 ARCHIVOS
load('data1.mat');

%%  CARGAR ARCHIVOS DE 2501-3064
load('data2.mat');

acum1=0; %% ACUMULADOR CANTIDAD DE DATOS TUMOR MENINGIOMA
acum2=0; %% ACUMULADOR CANTIDAD DE DATOS TUMOR GLIOMA
acum3=0; %% ACUMULADOR CANTIDAD DE DATOS TUMOR PITUITARY

%%  ORDENAMIENTO DE LOS PRIMEROS 2500 ARCHIVOS
for i=1:2500
    r1=L{i,1};
    r2=L{i,2};
    r3=L{i,3};
    [f1,c1]=size(r1);
    [f2,c2]=size(r2);
    [f3,c3]=size(r3);

    if(c1>0)
    acum1=acum1+1;
    X1{acum1,1}=L{i,1}; %%    CELDA DONDE SE ALMACENAN LOS ARCHIVOS DE TUMOR MENINGIOMA
    end

      if(c2>0)
      acum2=acum2+1;
      X2{acum2,1}=L{i,2}; %%    CELDA DONDE SE ALMACENAN LOS ARCHIVOS DE TUMOR GLIOMA
      end

      if(c3>0)
      acum3=acum3+1;
      X3{acum3,1}=L{i,3}; %%    CELDA DONDE SE ALMACENAN LOS ARCHIVOS DE TUMOR PITUITARY
      end
end

%%  ORDENAMIENTO DE LOS ARCHIVOS 2501-3064
for i=2501:3064
    r1=L2{i,1};
    r2=L2{i,2};
    [f1,c1]=size(r1);
    [f2,c2]=size(r2);

    if(c1>0)
    acum1=acum1+1;
    X1{acum1,1}=L2{i,1};
    end
```

```
        if(c2>0)
        acum2=acum2+1;
        X2{acum2,1}=L2{i,2};
        end
    end
```

En otro script cargamos la información de la primera celda para extraer los datos de los 708 tumores meningioma tales como las 708 imágenes, la identificación de todos los pacientes, en celdas guardamos los 708 vectores donde se encuentran las coordenadas del borde del tumor y las 708 imágenes de las máscaras donde se alojan las regiones del tumor cerebral, similarmente hicimos lo mismo para los 1426 datos de tumor glioma y los 930 datos de tumor hipofisary.

```
load('data_tumor_meningioma.mat');



L1=length(X1); %% CALCULAR LA LONGITUD DE LA VARIABLE DONDE SE GUARDAN LOS ARCHIVOS DE TUMOR MENINGIOMA
for i=1:L1
    da_me{i,1}=X1{i};
end

for i=1:L1
    Q=da_me{i,1};
    image_me{i,1}=Q.image;
    PID{i}=Q.PID;
    PID_me{i,1}=PID{i};
    border_me{i,1}=Q.tumorBorder;
    mask_me{i,1}=Q.tumorMask;
end

T = cell2table(PID_me);
filename = 'id pacientes tumor meningioma.xlsx';
writetable(T,filename,'Sheet',1,'Range','B1');
```

Al final exportamos en una tabla de Excel, la identificación de los pacientes con tumor cerebral meningioma, tumor cerebral glioma, y tumor cerebral hipofisary.

3.2 TRATAMIENTO DE IMÁGENES EN MATLAB

Una vez guardados las celdas, nos dirigimos a otro script y cargamos los datos .mat donde se encuentran las imágenes del tumor cerebral meningioma , y las imágenes de máscaras donde se aprecian las regiones donde se aloja el tumor cerebral, similarmente con las imágenes de tumor cerebral glioma, y máscara de región del tumor, y las imágenes de tumor cerebral pituitary y la máscara de la región donde se aprecia el tumor cerebral.

Las imágenes son de formato int32, y no todas tienen el mismo tamaño, por ello redimensionamos todas las imágenes a imágenes de 512 pixeles de alto X 512 pixeles de ancho.

```
load('data_images');
load('data_mask');

L1=length(image_me);

for i=1:L1
    image_me2{i}=imresize(image_me{i}, [512 512]);
end

for i=1:L1
    mask2{i,1}=imresize(mask_me{i}, [256 256]);
end

filtro = 'haar';
for i=1:L1
    t1=image_me2{i};
    [CA1{i,1},CH1{i,1},CV1{i,1},CD1{i,1}] = dwt2(t1,filtro,'mode','zpd');
end
```

3.2.1 ANALIZADOR DE WAVELET EN 2D

Matlab proporciona una interfaz gráfica llamada analizador de wavelet, donde podemos cargar una imagen y seleccionar un filtro (ondita madre), para apreciar las transformaciones del coefiente de aproximación, el coeficiente de aproximación vertical, coeficiente de aproximación de detalles y nos ofrece la posibilidad de descomponer en varios niveles la imagen, podemos seleccionar entre filtro Haar, Filtro Db, Filtro Sym, Filtro Coif, filtro Bior, filtro Rbio, filtro Dmey, y filtro Fk, además proporciona diferentes tipos de filtro.

A continuación, se muestra la aplicación de Todos los Filtros disponibles, en un nivel de descomposición y utilizamos el primer tipo de filtro implementado a una imagen de tumor meningioma.

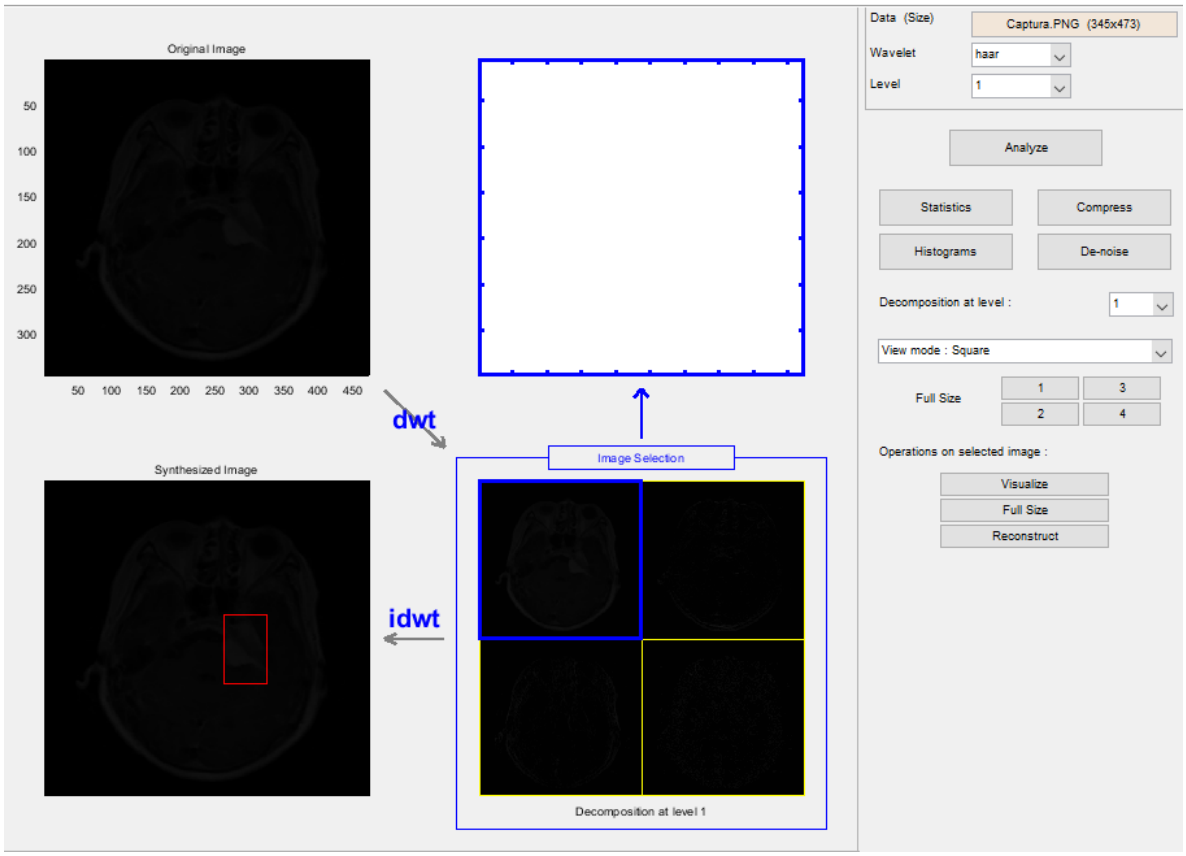


figura 26 Transformada Wavelet Filtro Haar

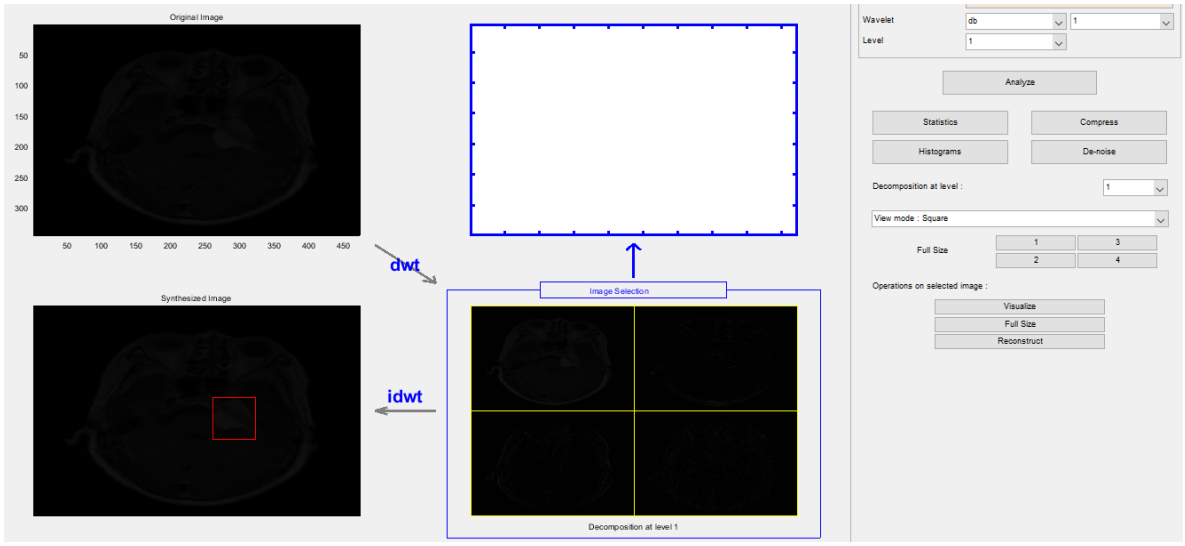


figura 27 Transformada Wavelet Filtro Db

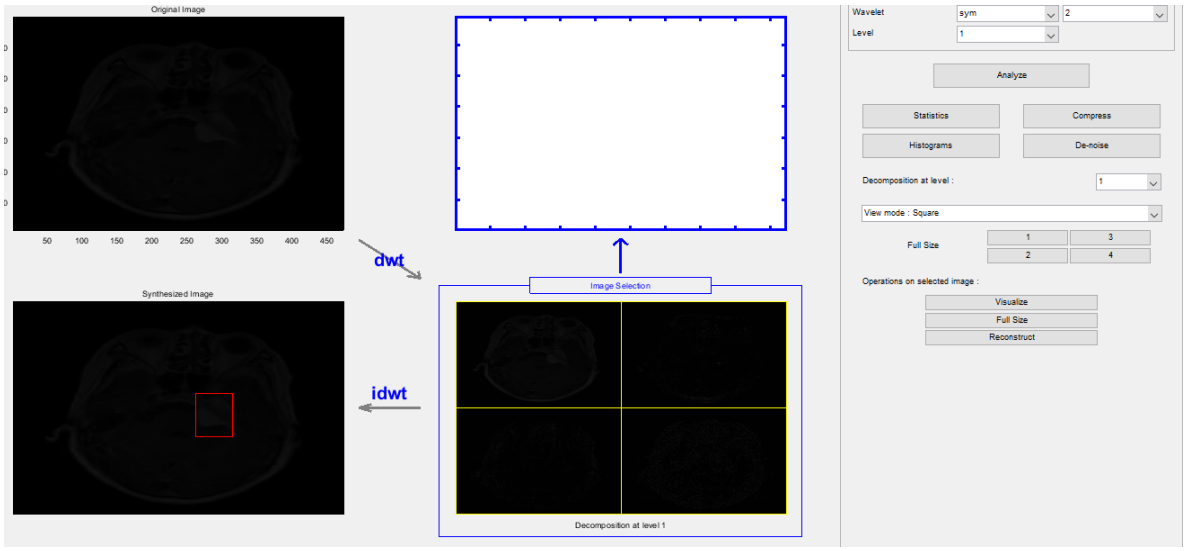


figura 28 Transformada Wavelet Filtro Sym

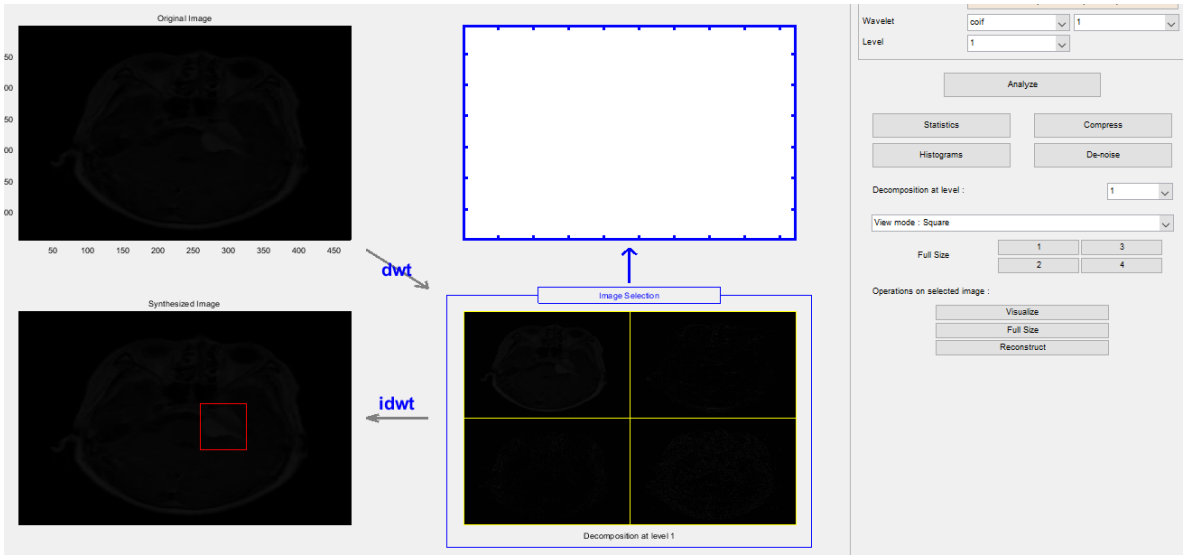


figura 29 Transformada Wavelet Filtro Coif

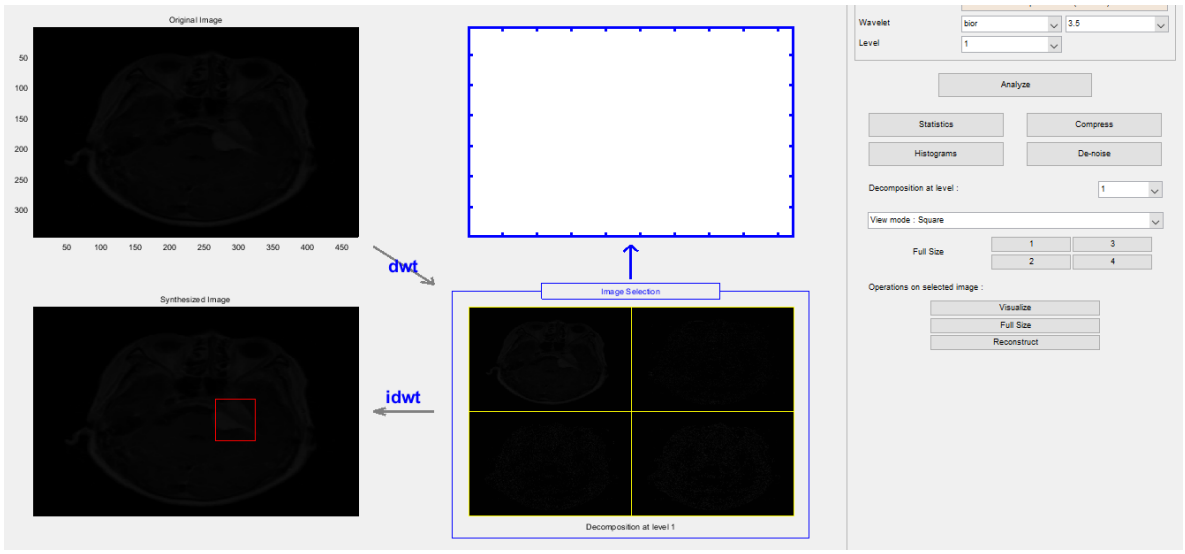


figura 30 Transformada Wavelet Filtro Bior

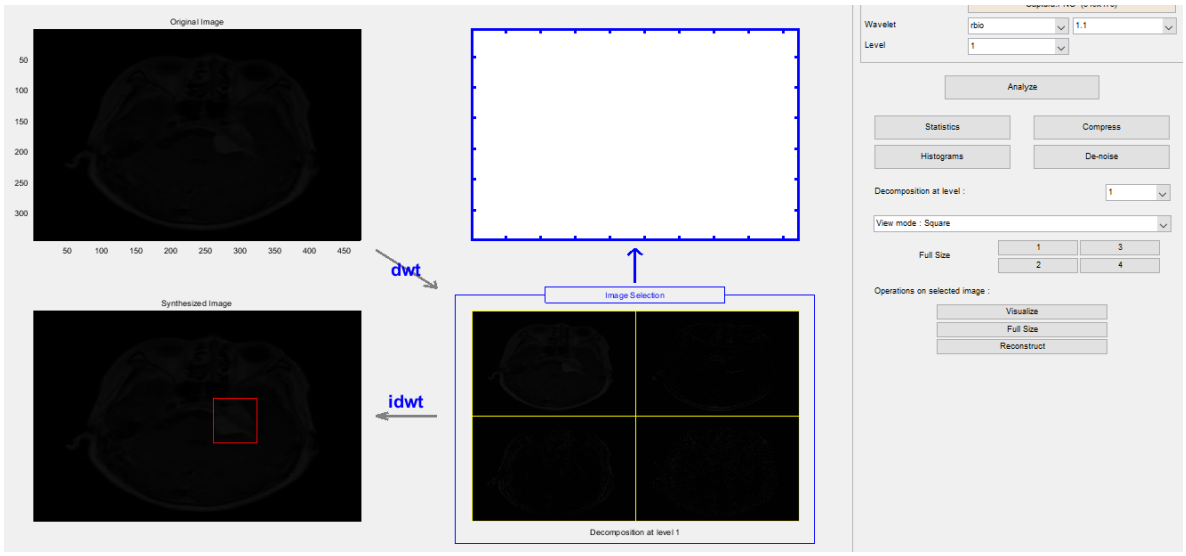


figura 31 Transformada Wavelet Filtro Rbio

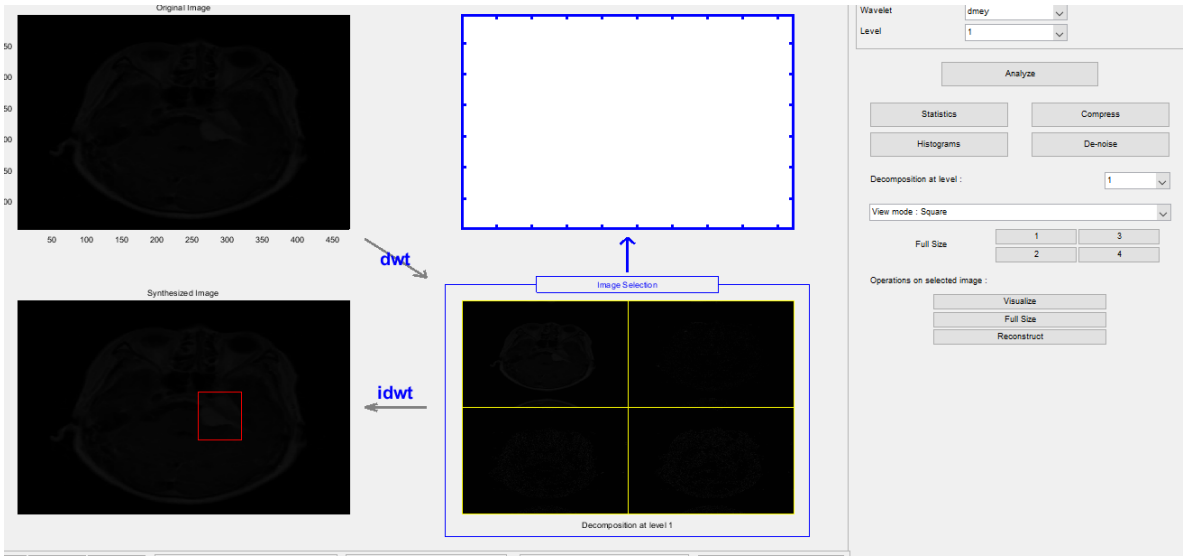


figura 32 Transformada Wavelet Filtro Dmey

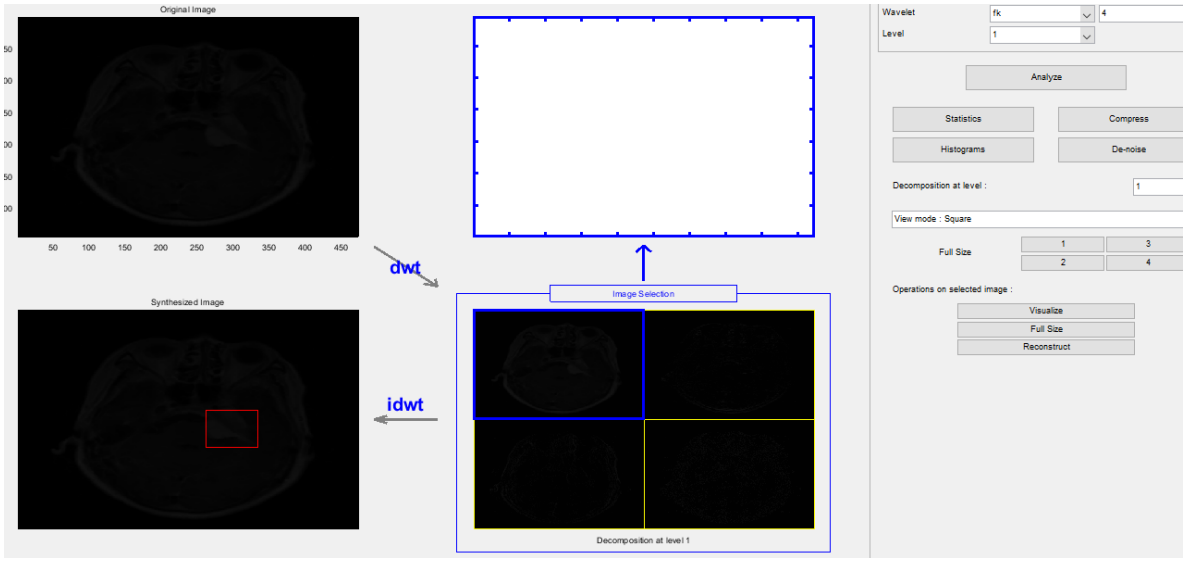


figura 33 Transformada Wavelet Filtro Fk

De todos los filtros aplicados a una imagen el filtro Haar fue el que mejor arrojó las características frecuenciales altas de la imagen, cuando aplicamos varios niveles de descomposición se degradaba la zona donde se alojaba el tumor cerebral, marcado con un cuadro rojo, que corresponde a la transformada inversa wavelet.

APLICACIÓN DE LA TRANSFORMADA DISCRETA WAVELET PARA ANALIZAR LAS IMÁGENES DE RESONANCIA MAGNÉTICA DE TUMORES CEREBRALES.

```
filtro = 'haar';
for i=1:L1
    t1=image_me2{i};
    [CA1{i,1},CH1{i,1},CV1{i,1},CD1{i,1}] = dwt2(t1,filtro,'mode','zpd');
end
```

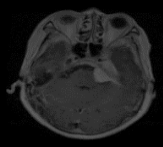
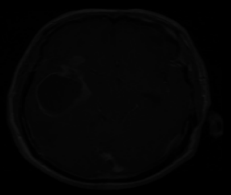
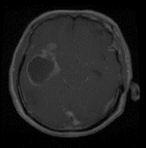
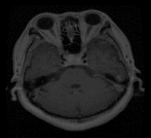
Imagen 512X512  *figura 34 Imagen Tumor Meningioma*	*imagen 256x256*  *figura 35 Transformada wavelet en tumor meningioma*
Imagen 512X512  *figura 36 Imagen Tumor Glioma*	*Imagen 256X256*  *figura 37 Transformada WAVELET en tumor glioma*
Imagen 512X512  *figura 38 Imagen Tumor Pituitary*	*Imagen 256X256*  *figura 39 Transformada WAVELET en tumor Pituitary*

Tabla 2 A la izquierda imágenes de tumor cerebral y a la derecha Transformadas de wavelet

Al aplicar la transformada discreta de wavelet, elegimos el coefiente de aproximaciones porque allí se encuentran las frecuencias altas a la vez que son realzadas, y las frecuencias bajas son suprimidas, a pesar de que la imagen de la transformada es la mitad de la imagen original, conserva las frecuencias altas del tumor cerebral.

Al aplicar las transformadas de wavelet se guardan las imágenes en una carpeta y las imágenes originales, así como las imágenes de las zonas donde se encuentra el tumor cerebral, y los datos de los pacientes en un archivo de Excel.

Al final obtendremos 3064 imágenes que corresponden a las transformadas discretas de cada tipo de tumor ordenadas según el tipo de tumor; tumor meningioma, tumor glioma, o tumor hipofisary.

3.3 CONJUNTO DE IMÁGENES

Para las muestras de entrenamiento, validación y prueba se considera el conjunto de imágenes de los tres tipos de tumor cerebral donde cada imagen corresponde a la transformada discreta de wavelet aplicado a las 3064 imágenes (figura 26).

3.3.1 AUMENTO DEL CONJUNTO DE IMÁGENES EN PYTHON

Inicialmente el conjunto de imágenes contiene 3064 imágenes de tres tipos de tumor, y se tomó la decisión de girar las 3064 imágenes 180° sin invertir su eje, para crear un conjunto de imágenes con 6128 imágenes.

Para girar las 3064 imágenes se utilizó la función de opencv Flip estableciendo -1 para solo girar las imágenes 180 grados sin invertir el eje , estas imágenes se guardan en otro directorio y se alimentan al conjunto de imágenes inicial.

```
input_images_path="C:/Users/User/Desktop/imagenes wavelet tumores/entrenamiento/glioma"

files_names = os.listdir(input_images_path)
print(files_names)

output_images_path = "C:/Users/User/Desktop/imagenes wavelet tumores/entrenamiento/glioma-new"
if not os.path.exists(output_images_path):
    os.makedirs(output_images_path)
    print("Directorio creado: ", output_images_path)
count = 1
for file_name in files_names:
    #print(file_name)
    '''
    if file_name.split(".")[-1] not in ["jpeg", "png"]:
        continue
    '''
    a=file_name
    image_path = input_images_path + "/" + file_name
    print(image_path)
    imagex = cv2.imread(image_path)
    inv=cv2.flip(imagex,-1)
    cv2.imwrite(output_images_path +"/" + (a), inv)
    count += 1

cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
input_images_path="C:/Users/User/Desktop/imagenes wavelet tumores/entrenamiento/meningioma"

files_names = os.listdir(input_images_path)
print(files_names)

output_images_path = "C:/Users/User/Desktop/imagenes wavelet tumores/entrenamiento/meningioma-new"
if not os.path.exists(output_images_path):
    os.makedirs(output_images_path)
    print("Directorio creado: ", output_images_path)
count = 1
for file_name in files_names:
    #print(file_name)
    '''
    if file_name.split(".")[-1] not in ["jpeg", "png"]:
        continue
    '''
    a=file_name
    image_path = input_images_path + "/" + file_name
    print(image_path)
    imagex = cv2.imread(image_path)
    inv=cv2.flip(imagex,-1)
    cv2.imwrite(output_images_path +"/" + (a), inv)
    count += 1

cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
input_images_path="C:/Users/User/Desktop/imagenes wavelet tumores/entrenamiento/pituitary"

files_names = os.listdir(input_images_path)
print(files_names)

output_images_path = "C:/Users/User/Desktop/imagenes wavelet tumores/entrenamiento/pituitary-new"
if not os.path.exists(output_images_path):
    os.makedirs(output_images_path)
    print("Directorio creado: ", output_images_path)
count = 1
for file_name in files_names:
    #print(file_name)
    '''
    if file_name.split(".")[-1] not in ["jpeg", "png"]:
        continue
    '''
    a=file_name
    image_path = input_images_path + "/" + file_name
    print(image_path)
    imagex = cv2.imread(image_path)
    inv=cv2.flip(imagex,-1)
    cv2.imwrite(output_images_path +"/" + (a), inv)
    count += 1

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Al final obtenemos la duplicación del conjunto de imágenes, copiamos las imágenes a la raspberry pi4 y separamos en 3 carpetas las imágenes, Para entrenar el modelo es necesario separarlas en tres carpetas, imágenes de entrenamiento figura 42, imágenes de validación figura 44, e imágenes de prueba figura 43, una vez conectados a internet accedemos al buscador y subimos las imágenes a Google drive, para poder gestionar las imágenes desde Google colab.

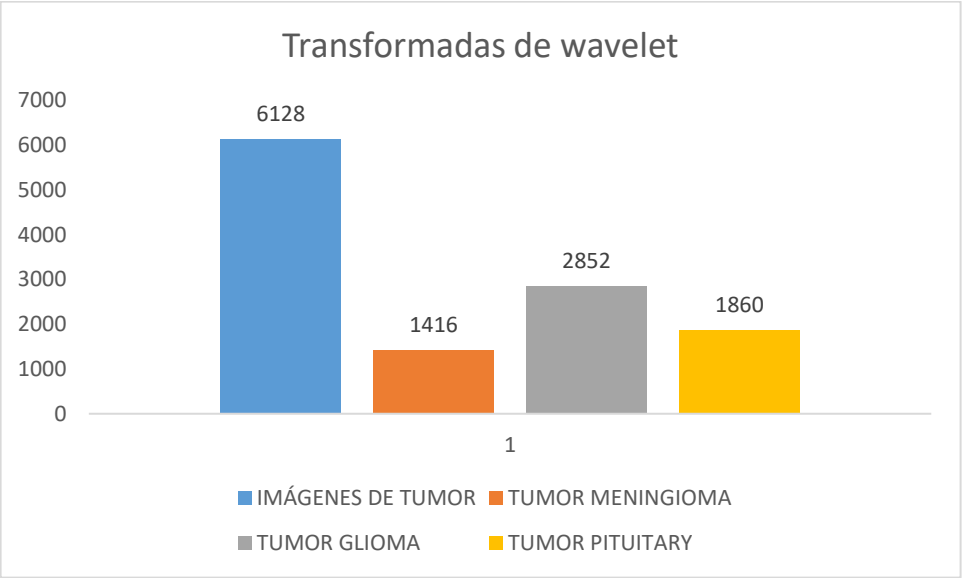


figura 40 Conjunto de imágenes aumentado

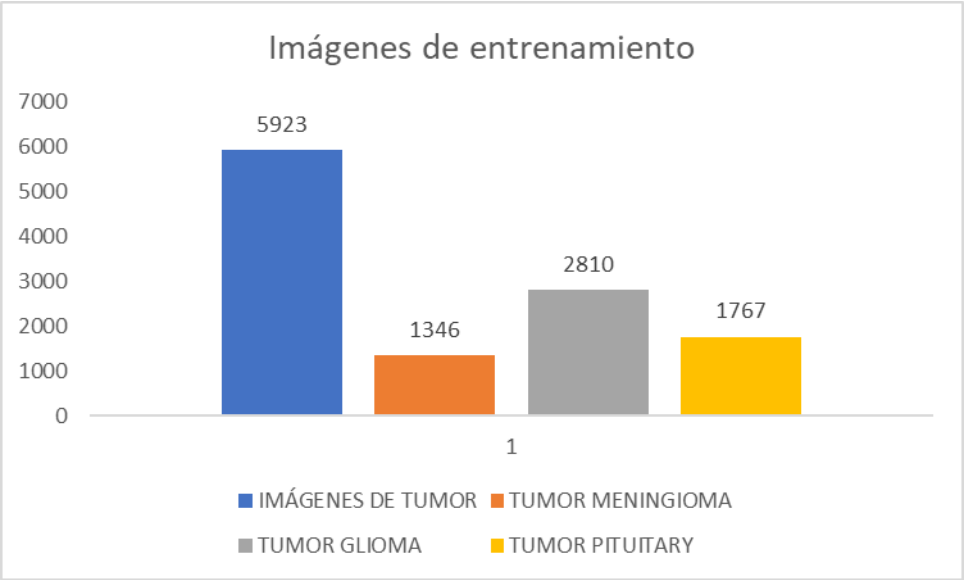


figura 41 Imágenes de entrenamiento

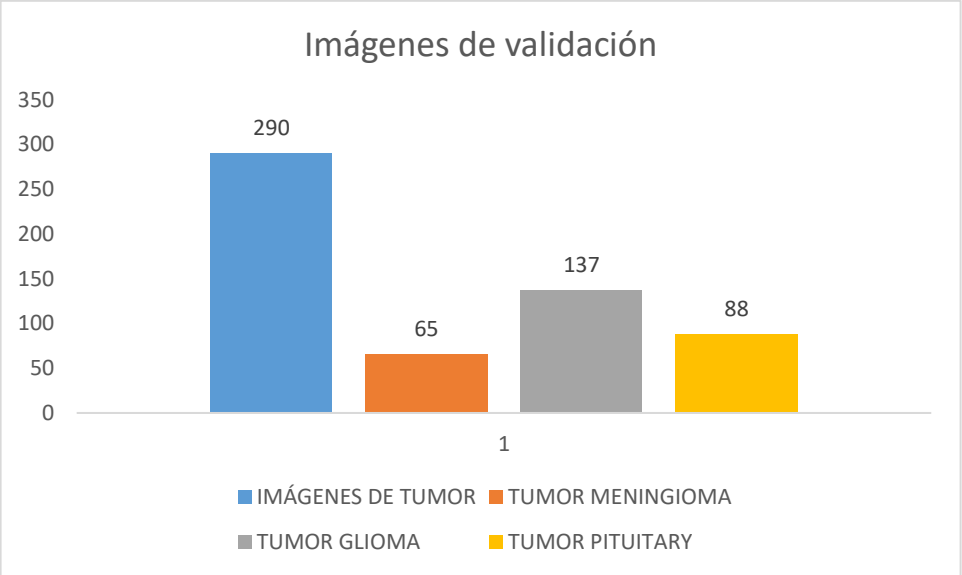


figura 42 Imágenes de Validación

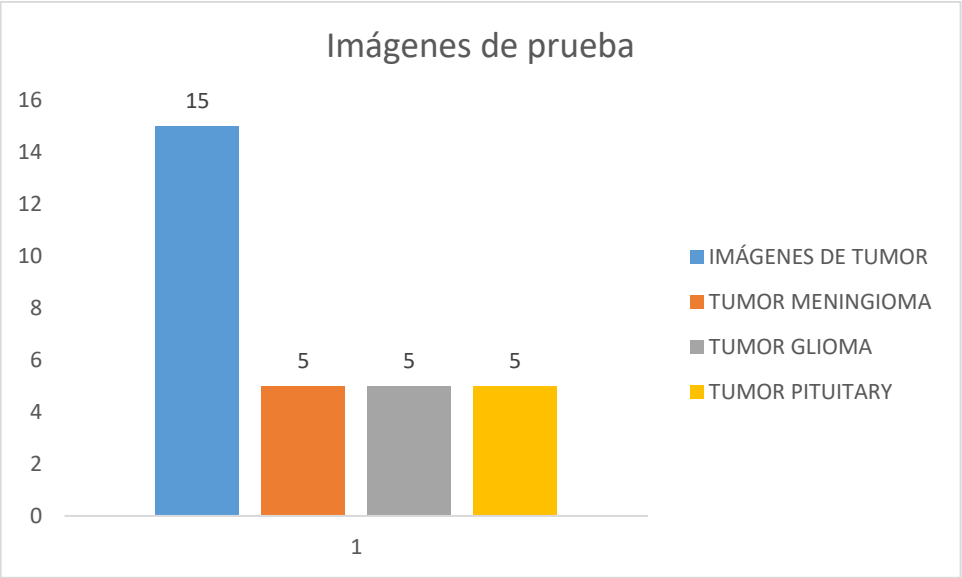


figura 43 imágenes de prueba

3.4 MODELO DE INTELIGENCIA ARTIFICIAL

Se decide usar redes neuronales convolucionales por su ventaja en la detección de rostros, clasificación de imágenes, detección de personas, detección de animales. Estas redes son muy útiles en la clasificación de imágenes debido a sus múltiples conexiones, y al estar basadas en filtros para extraer características de una imagen.

A continuación, se muestra el código en Google colab para guardar en una variable los datos de entrenamiento y los datos de validación, así como la carga de archivos desde drive a Google colab.

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

     Mounted at /content/drive
```

```
[ ]  !cp -r "/content/drive/MyDrive/datos_tumores_aumentado/entrenamiento" "/content/entrenamiento"
```

```
[ ]  !cp -r "/content/drive/MyDrive/datos_tumores_aumentado/validacion" "/content/validacion"
```

```
[ ]  data_entrenamiento = "/content/entrenamiento"
     data_validacion = "/content/validacion"
```

Se crearon dos modelos de red neuronal, uno en donde se ingresan las 6128 imágenes de los tres tipos de tumor cerebral, y otra red neuronal donde se ingresan las 6128 imágenes, pero se utiliza el aumento de datos.

3.4.1 MODELO CNN

Para el primer modelo se aplican técnicas de procesamiento de imágenes antes de entrenar el modelo. La clase de Keras imageDataGenerator es la encargada de aplicar una normalización a las imágenes, las transformadas están en formato uin16, la normalización cambia los datos de pixeles en un rango de 0 a 1, es necesario ya que el entrenamiento mejora los hiperparámetro de la red neuronal.

```
entrenamiento_datagen = ImageDataGenerator(rescale=1. / 255)
```

Con el método flow_from_directory de la clase ImageDataGenerator se generan imágenes desde un directorio en Keras, con este método se cargan las imágenes de entrenamiento y validación y se procesan en 3 clases, cada clase representa un tipo de tumor.

- Se redimensionan las imágenes en una escala de 256 x 256 con la función tarjet_size, todas las imágenes deben tener el mismo tamaño, esta redimensión disminuye el tiempo de ejecución del entrenamiento.

- El tamaño de lote se asigna en 32 esto indica que el entrenamiento se va ejecutando en lotes de 32 imágenes para evitar el sobreajuste al momento de entrenarse la red neuronal.

- Class_mode se estable en modo categorico ya que contamos con imágenes de tipo arreglo en 2 dimensiones.

```
entrenamiento_generador = entrenamiento_datagen.flow_from_directory(data_entrenamiento,
                                                                    target_size=(altura, longitud),
                                                                    batch_size=batch_size,
                                                                    class_mode='categorical')


validacion_generador = test_datagen.flow_from_directory(data_validacion,
                                                        target_size=(altura, longitud),
                                                        batch_size=batch_size,
                                                        class_mode='categorical')
```

```
Found 6116 images belonging to 3 classes.
Found 317 images belonging to 3 classes.
```

3.4.1.1 Implementación del modelo en Keras

la creación de la red neuronal convolucional se puede dividir en dos secciones:

- Definición de la estructura de la red neuronal convolucional con la sentencia model=sequential (), indicando una conexión jerárquica (una capa tras de otra).

- Con la función cnn.add () agregamos las capas a la red neuronal convolucional

```
cnn = Sequential()
tf.get_logger().setLevel('ERROR')
```

La sentencia tf.get_logger().setLevel("ERROR"), no deja imprimir mensajes de advertencia de la versión TensorFlow 1.15.2.

Se agrega al modelo una capa convolucional 2D, con filtros convolucionales de 32, debe ser igual al tamaño de lote, y un tamaño filtro = (3,3) que son los filtros kernels para que la red vaya extrayendo la información de cada imagen moviéndose 3X3 en cada imagen, esto creará un arreglo más pequeño pero obtendrá los patrones característicos de ese sector de transformada wavelet., se seleccionó la función de activación RELU por su capacidad de activación de valores positivos, además converge en el entrenamiento, como las imágenes de entrada de la red neuronal corresponden a imágenes RGB de 256 pixeles X 256 pixeles, la entrada de los tensores debe ser similar por ello input_shape=(256,256,3).

Luego de cada capa de convolución se agrega una capa de pooling de tipo MaxPooling2d esto reducirá a la mitad las dimensiones de las imágenes, pero manteniendo la profundidad con un tamaño_pool = (2,2).

```
filtrosConv1 = 32
tamano_filtro1 = (3, 3)
cnn.add(Convolution2D(filtrosConv1, tamano_filtro1, padding ="same", input_shape=(longitud, altura, 3), activation='relu'))
cnn.add(MaxPooling2D(pool_size=tamano_pool))
```

La salida del último bloque de maxPooling2d se aplana model.add(Flatten()), convirtiéndolo en un vector fila, las dimensiones del vector característico corresponde al multiplicar las dimensiones de salida de la capa de maxPooling2d_1 por la profundidad que equivale al número de filtros, esto es : (64*64)*64 = 262144, ver figura 45.

```
cnn.add(Flatten())
cnn.add(Dense(256, activation='relu'))
cnn.add(Dropout(0.5))
cnn.add(Dense(clases, activation='softmax'))
```

Se agrega una capa densa Dense(256, activation="relu"), para permitir agregar a la red neuronal 256 neuronas que aprenderán las diferencias de la capa aplanada y determinar la

similitud entre imágenes a partir del valor de sus pixeles, se usa una función relu de activación porque los datos de imágenes de entrada corresponden a pixeles de valor positivo.

Para evitar un posible sobreajuste del modelo en la etapa de clasificación se usó la técnica de dropout la cual en cada época desconectara el 50% de las conexiones, model.add(Dropout(0.5)), de las neuronas, esto consigue que la red neuronal no pueda aprender un solo camino para determinar separar las 3 clases, sino permite generalizar los datos de entrada.

Al final de la red neuronal convolucional se usaron 3 neuronas de salida con función de activación softmax, cada neurona arroja un porcentaje de acierto, y la función softmax es la encargada de determinar cuál neurona tuvo mayor porcentaje de activación y asignará un valor respecto a una de las 3 clases (3 tipos de tumor cerebral).

```
cnn.compile(loss='categorical_crossentropy',optimizer=optimizers.Adam(lr=lr),metrics=['accuracy'])
```

En la compilación, primero se configura la función de perdida, al tener tres clases de separabilidad se hace uso del Categorical_Crossentropy el cual calcula la perdida entre la clase y la predicción, loss='categorical_crossentropy'

Se usó el método de Adam (Estimación adaptativa por sectores), es adecuado para redes muy extensas y se configuró una tasa de aprendizaje, junto con métricas de precisión.

```
print[cnn.summary()]
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 256, 256, 32)      896
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 128, 128, 32)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 128, 128, 64)      8256
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 64, 64, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 262144)            0
_________________________________________________________________
dense (Dense)                (None, 256)               67109120
_________________________________________________________________
dropout (Dropout)            (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 771
=================================================================
Total params: 67,119,043
Trainable params: 67,119,043
Non-trainable params: 0
```

figura 44 Estructura red neuronal convolucional compilada.

3.4.2 MODELO CNN CON DATOS AUMENTADOS

Para aumentar el conjunto de imágenes con datos aumentados modificamos los datos de entrenamiento aleatoriamente aplicamos 3 técnicas de aumento de datos como lo fue:

Shear_range = 0.02 toma aleatoriamente imágenes y las distoriciona en un eje en un 2% de la imagen total.

Zoom_range = 0.02 toma aleatoriamente imágenes y hace un acercamiento del 2% de la imagen original

Horintol_flip = False, se deshabilita el giro de imágenes aleatorias ya que el conjunto de datos contiene imágenes giradas en 180°, este parámetro por defecto está configurado para estar activo.

```
entrenamiento_datagen = ImageDataGenerator(rescale=1. / 255), shear_range=0.02 ,zoom_range=0.02, horizontal_flip=False)

test_datagen = ImageDataGenerator(rescale=1. / 255)
test_datagen = ImageDataGenerator()

entrenamiento_generador = entrenamiento_datagen.flow_from_directory(data_entrenamiento,
                                                                    target_size=(altura, longitud),
                                                                    batch_size=batch_size,
                                                                    class_mode='categorical')


validacion_generador = test_datagen.flow_from_directory(data_validacion,
                                                        target_size=(altura, longitud),
                                                        batch_size=batch_size,
                                                        class_mode='categorical')
```

Con el método flow_from_directory de la clase ImageDataGenerator se generan imágenes desde un directorio en Keras, con este método se cargan las imágenes de entrenamiento y validación y se procesan en 3 clases, cada clase representa un tipo de tumor.

- Se redimensionan las imágenes en una escala de 256 x 256 con la función tarjet_size, todas las imágenes deben tener el mismo tamaño, esta redimensión disminuye el tiempo de ejecución del entrenamiento.

- El tamaño de lote se asigna en 32 esto indica que el entrenamiento se va ejecutando en lotes de 32 imágenes para evitar el sobreajuste al momento de entrenarse la red neuronal.

- Class_mode se estable en modo categorico ya que contamos con imágenes de tipo arreglo en 2 dimensiones.

3.4.2.1 Implementación del modelo en Keras

la creación de la red neuronal convolucional con datos aumentados dividir en:

- Definición de la estructura de la red neuronal convolucional con la sentencia modeloCNN2_AD=sequential (), indicando una conexión jerárquica (una capa tras de otra).

```
modeloCNN2_AD = tf.keras.models.Sequential([
```

```
tf.keras.layers.Conv2D(32, (4,4), activation='relu', padding ="same", input_shape=(longitud, altura, 3)),
```

Se agrega al modelo una capa convolucional 2D, con filtros convolucionales de 32, debe ser igual al tamaño de lote, y un tamaño filtro = (4,4) que son los filtros kernels para que la red vaya extrayendo la información de cada imagen moviéndose 4X4 en cada imagen, esto creará un arreglo más pequeño pero obtendrá los patrones característicos de ese sector de transformada wavelet, se seleccionó la función de activación RELU por su capacidad de activación de valores positivos, además converge en el entrenamiento, como las imágenes de entrada de la red neuronal corresponden a imágenes RGB de 256 pixeles X 256 pixeles, la entrada de los tensores debe ser similar por ello input_shape=(256,256,3).

```
tf.keras.layers.MaxPooling2D(2, 2)
```

Luego de cada capa de convolución se agrega una capa de pooling de tipo MaxPooling2d esto reducirá a la mitad las dimensiones de las capas convolucionales de 2, pero manteniendo la profundidad con un tamaño_pool = (2,2).

```
tf.keras.layers.Conv2D(64, (2,2), activation='relu'),
```

Se agrega al modelo una capa convolucional 2D, con filtros convolucionales de 64, tamaño de lote a la mitad, y un tamaño filtro = (2,2) que son los filtros kernels para que la red vaya extrayendo la información de cada imagen moviéndose 2X2 en cada imagen, esto creará un arreglo más pequeño pero obtendrá los patrones característicos de ese sector de transformada wavelet, se seleccionó la función de activación RELU por su capacidad de activación de valores positivos, además converge en el entrenamiento.

```
tf.keras.layers.MaxPooling2D(2, 2)
```

Luego de cada capa de convolución se agrega una capa de pooling de tipo MaxPooling2d esto reducirá a la mitad las dimensiones de las capas convolucionales de 2, pero manteniendo la profundidad con un tamaño_pool = (2,2).

```
tf.keras.layers.Conv2D(128, (2,2), activation='relu'),
```

Se agrega al modelo una capa convolucional 2D, con filtros convolucionales de 128, tamaño de lote de un cuarto, y un tamaño filtro = (2,2) que son los filtros kernels para que la red vaya extrayendo la información de cada imagen moviéndose 2X2 en cada imagen, esto creará un arreglo más pequeño pero obtendrá los patrones característicos de ese sector de transformada wavelet, se seleccionó la función de activación RELU por su capacidad de activación de valores positivos, además converge en el entrenamiento.

```
tf.keras.layers.MaxPooling2D(2, 2)
```

Luego de cada capa de convolución se agrega una capa de pooling de tipo MaxPooling2d esto reducirá a la mitad las dimensiones de las capas convolucionales de 2, pero manteniendo la profundidad con un tamaño_pool = (2,2).

La salida del último bloque de maxPooling2d se aplana modeloCNN_AD.add(Flatten()), convirtiéndolo en un vector fila, las dimensiones del vector característico corresponde al multiplicar las dimensiones de salida de la capa de maxPooling2d_17 por la profundidad que equivale al número de filtros, esto es : (31*31)*18 = 123008, ver figura **.

```
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(256, activation='relu'),
    tf.keras.layers.Dense(clases, activation='softmax')
```

Se agrega una capa densa Dense(256, activation=”relu”), para permitir agregar a la red neuronal 256 neuronas que aprenderán las diferencias de la capa aplanada y determinar la similitud entre imágenes a partir del valor de sus pixeles, se usa una función relu de activación porque los datos de imágenes de entrada corresponden a pixeles de valor positivo.

Para evitar un posible sobreajuste del modelo en la etapa de clasificación se usó la técnica de dropout la cual en cada época desconectara el 50% de las conexiones, model.add(Dropout(0.5)), de las neuronas, esto consigue que la red neuronal no pueda aprender un solo camino para determinar separar las 3 clases, sino permite generalizar los datos de entrada.

Al final de la red neuronal convolucional se usaron 3 neuronas de salida con función de activación softmax, cada neurona arroja un porcentaje de acierto, y la función softmax es la encargada de determinar cuál neurona tuvo mayor porcentaje de activación y asignará un valor respecto a una de las 3 clases (3 tipos de tumor cerebral).

```
modeloCNN2_AD.compile(optimizer='adam',
                      loss='categorical_crossentropy',
                      metrics=['accuracy'])
```

En la compilación, primero se configura la función de perdida, al tener tres clases de separabilidad se hace uso del Categorical_Crossentropy el cual calcula la perdida entre la clase y la predicción, loss='categorical_crossentropy'

Se usó el método de Adam (Estimación adaptativa por sectores), es adecuado para redes muy extensas y no se configuró una tasa de aprendizaje, junto con métricas de precisión.

```
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_15 (Conv2D)           (None, 256, 256, 32)      1568
_________________________________________________________________
max_pooling2d_15 (MaxPooling (None, 128, 128, 32)      0
_________________________________________________________________
conv2d_16 (Conv2D)           (None, 127, 127, 64)      8256
_________________________________________________________________
max_pooling2d_16 (MaxPooling (None, 63, 63, 64)        0
_________________________________________________________________
conv2d_17 (Conv2D)           (None, 62, 62, 128)       32896
_________________________________________________________________
max_pooling2d_17 (MaxPooling (None, 31, 31, 128)       0
_________________________________________________________________
dropout_5 (Dropout)          (None, 31, 31, 128)       0
_________________________________________________________________
flatten_5 (Flatten)          (None, 123008)            0
_________________________________________________________________
dense_10 (Dense)             (None, 256)               31490304
_________________________________________________________________
dense_11 (Dense)             (None, 3)                 771
=================================================================
Total params: 31,533,795
Trainable params: 31,533,795
Non-trainable params: 0
```

figura 45 Estructura red neuronal convolucional para datos aumentados compilada

3.5 ENTRENAMIENTO DE LOS DOS MODELOS DE INTELIGENCIA ARTIFICIAL

Google colab permite el entrenamiento de modelos de redes neuronales con la posibilidad de configurar el cuaderno, con tarjetas gráficas de alto desempeño modernas, basta con configurar el cuaderno como se muestra en la figura 47, y utilizar una función para acudir a la memoria de video virtual dedicada para el entrenamiento.

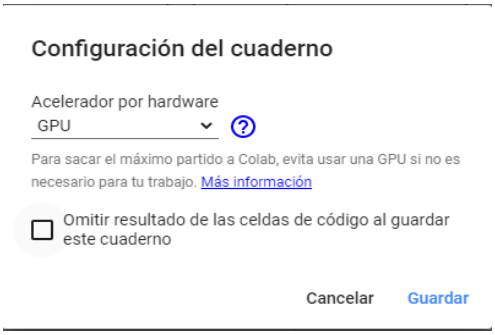


figura 46 Entorno de Google colab para elegir acelerador por hardware de tipo procesador gráfico

3.5.1 ENTRENAMIENTO MODELO CNN

La librería Tensorboard permite guardar los datos de entrenamiento, una vez terminado para verificar el aprendizaje de la red y sus pérdidas respecto a los datos de validación, para el entrenamiento del primer modelo utilizamos 5 pasos de validación por época, 10 épocas, y 1000 pasos de validación, además llamamos a la librería para almacenar los datos y después lanzar la aplicación para visualizar las gráficas de entrenamiento y validación. con la ayuda de la función de gestión de tiempo de la tarjeta gráfica.

```
#La variable de tensorboard se envia en el arreglo de "callbacks" (hay otros tipos de callbacks soportados)
#En este caso guarda datos en la carpeta indicada en cada epoca, de manera que despues
#Tensorboard los lee para hacer graficas
tensorboardcnn = TensorBoard(log_dir='logs/cnn')

import timeit

def entrenamiento_gpu():
  with tf.device('/device:GPU:0'):
    historial=cnn.fit_generator(entrenamiento_generador,
                                steps_per_epoch=5,
                                epochs=10,
                                validation_data=validacion_generador,
                                validation_steps=1000,
                                callbacks=[tensorboardcnn])
  return None

gpu_time = timeit.timeit('entrenamiento_gpu()', number=1, setup='from __main__ import entrenamiento_gpu')
```

3.5.2 ENTRENAMIENTO MODELO CNN2 CON DATOS AUMENTADOS

para el entrenamiento del segundo modelo utilizamos 5 pasos de validación por época, 10 épocas, y 1000 pasos de validación, además llamamos a la librería para almacenar los datos y después lanzar la aplicación para visualizar las gráficas de entrenamiento y validación, al final imprimimos el tiempo que tardó el entrenamiento con la ayuda de la función de gestión de tiempo de la tarjeta gráfica.

```
tensorboardCNN2_AD = TensorBoard(log_dir='logs/cnn2_AD')
import timeit

def entrenamiento_gpu():
  with tf.device('/device:GPU:0'):
    historial2=modeloCNN2_AD.fit(entrenamiento_generador,epochs=10,
                                 validation_data=validacion_generador,
                                 steps_per_epoch=5,
                                 validation_steps= 1000,
                                 callbacks=[tensorboardCNN2_AD])
  return None

gpu_time = timeit.timeit('entrenamiento_gpu()', number=1, setup='from __main__ import entrenamiento_gpu')
```

3.6 SELECCIÓN DEL MODELO DE INTELIGENCIA ARTIFICIAL

El modelo seleccionado es el modelo de inteligencia artificial con redes neuronales convolucionales con datos aumentados. Es un modelo más robusto y con más capas de convolución que el modelo 1, además que en el entrenamiento se utilizan técnicas de datos aumentados logrando que la red aprenda mejor a diferenciar entre las 3 clases, este modelo es más grande que el modelo 1, y por tanto sus pesos también lo son, el factor en contra de este modelo se apreció al momento de descargarlo y llevarlo a la raspberry pi 4 y además que es un modelo que se demora en clasificar una nueva imagen en relación con un modelo, pero es preferible esperar unos minutos más que obtener una clasificación errónea en poco tiempo.

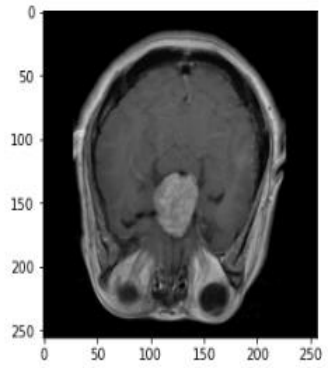
SISTEMA DE INTELIGENCIA ARTIFICIAL BASADO EN REDES NEURONALES PARA LA IDENTIFICACIÓN DE TRES TUMORES CEREBRALES MENINGIOMA, GLIOMA, PITUITARY EN UNA RASPBERRY PI 4.

```
import numpy as np
import tensorflow as tf
import cv2
import matplotlib.pyplot as plt

modeloCNN2_AD= tf.keras.models.load_model('modelo.h5') #Cargar el modelo de la RN
modeloCNN2_AD.load_weights('pesos.h5') #Cargar los pesos de la RN

def predict(A): #función para evaluar una imágen en la RN
  x=A
  x = np.expand_dims(x, axis=0)
  array = modeloCNN2_AD.predict(x)
  result = array[0] #En la primera posición del arreglo se encuentra la clase detectada
  answer = np.argmax(result)
  predictions_single = modeloCNN2_AD.predict(x)

  print(predictions_single)
  print(np.sum(predictions_single))
  print(np.argmax(predictions_single))
  if answer == 0:
    print("pred: Glioma")
  if answer == 1:
    print("pred: Meningioma")
  if answer == 2:
    print("pred: pituitary")
  return answer
```

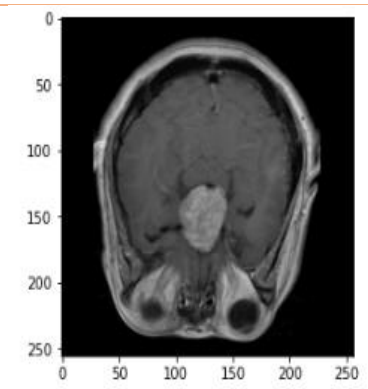



```
(256, 256, 3)
(256, 256, 3)
[[0. 0. 1.]]
1.0
2
pred: pituitary
[[0. 0. 1.]]
1.0
2
pred: pituitary
etiqueta 2
```

```
img_size=256 #tamaño de la imágen alto, ancho
img = cv2.imread("pituitary (4).png")
img_color= cv2.cvtColor(img,cv2.COLOR_BGR2RGB) #cambiar espacio de color BGR a
img_resize= cv2.resize(img_color,(img_size,img_size)) #redimensionar la imagen
img_resize=cv2.flip(img_resize,-1) # girar la imágen 180° sin inversión del eje

img = img_resize
plt.imshow(img)
plt.show()
print(img.shape)

A = np.array(img) #convertir la imagen a arreglo
print(A.shape)
predict(A) #acudir a la función para evaluar una nueva imagen
r=predict(A)
print("etiqueta",r) # etiqueta de la predicción
```




```
(256, 256, 3)
(256, 256, 3)
[[0. 0. 1.]]
1.0
2
pred: pituitary
[[0. 0. 1.]]
1.0
2
pred: pituitary
etiqueta 2
```

Tabla 3 A la derecha código de programación para evaluar la red neuronal, a la derecha clasificación de un tumor pituitary

3.7 COMUNICACIÓN ENTRE RASPBERRY PI4 Y RASPBERRY PI 3 CON FIREBASE.

figura 47 Comunicación entre raspberry pi4 y raspberry pi3 por medio de Firebase

Con una cuenta de Google abrimos en el buscador Firebase, y elegimos nueva consola, seguido activamos el almacenamiento de la nube, base de datos Firestore, base de datos en tiempo real, y la autenticación, ahora creamos una nueva aplicación.

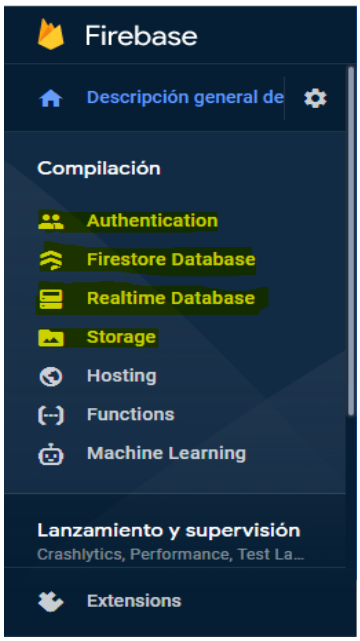


figura 48 Consola de Firebase

Ahora basta con seleccionar la configuración de la aplicación para obtener los links de acceso a la plataforma.

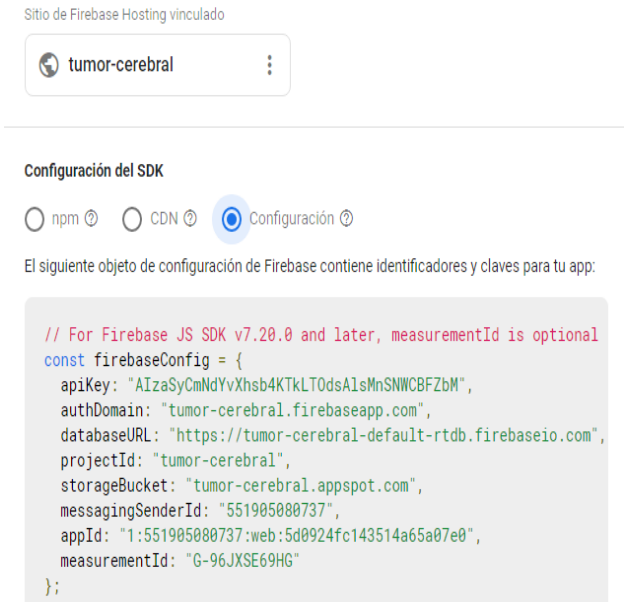


figura 49 Configuración para acceso desde un servicio externo al hosting de Firebase

Estos identificadores de acceso y claves fueron insertados en el entorno de programación de Python de la raspberry ver figura 50, para enviar o recibir datos a la nube desde raspberry pi4, o enviar datos a la nube de Firebase desde una raspberry pi3.

```
import pyrebase
import time
import cv2
import matplotlib as plt
import os
import numpy as np
import codecs, json
#Credenciales para la comunicación
config = {

 "apiKey": "AIzaSyCmNdYvXhsb4KTkLTOdsAlsMnSNWCBFZbM",
 "authDomain": "tumor-cerebral.firebaseapp.com",
 "databaseURL": "https://tumor-cerebral-default-rtdb.firebaseio.com",
 "projectId": "tumor-cerebral",
 "storageBucket": "tumor-cerebral.appspot.com",
 "messagingSenderId": "551905080737",
 "appId": "1:551905080737:web:5d0924fc143514a65a07e0",
 "measurementId": "G-96JXSE69HG"
}
#Inicializa la base de datos
firebase = pyrebase.initialize_app(config)
db = firebase.database()
storage = firebase.storage()

print("Enviando  Datos A Firebase Usando Raspberry Pi")
print("———————————————-")
```

figura 50 Comunicación con Firebase desde Raspberry pi4 en Python

En la figura 51 se muestra la forma de enviar una imagen a Firebase, la imagen hay que convertirla a una clase JSON para poder ser almacenada en la base de datos en tiempo real.

```
class NumpyEncoder(json.JSONEncoder):
   def default(self, obj):
      if isinstance(obj, np.ndarray):
         return obj.tolist()
      return json.JSONEncoder.default(self, obj)


a=cv2.imread("mask5.png")
a = np.array(a)
json_dump = json.dumps({'a': a}, cls=NumpyEncoder)


x=json_dump

print("Enviando Datos En Formato JSON")
db.child("Image1").child("1-set").set(x)
db.child("Image2").child("2-push").push(x)
db.child("Image3").child("3-push").push(x)
db.child("Image4").child("4-set").push(x)

firebase = pyrebase.initialize_app(config)
print("Datos Enviados")
db = firebase.database()
storage = firebase.storage()
```

figura 51 datos de una imagen enviados a Firebase mediante Raspberry pi4 en Python.

4.

Resultados.

4. RESULTADOS

4.1 RESULTADOS DE ENTRENAMIENTO

Se muestran los resultados correspondientes a los dos entrenamientos de cada red neuronal convolucional, el modelo con mejor precisión fue exportado a la raspberry pi4.

4.1.1 Modelo CNN

Los resultados de entrenamiento de la red neuronal convolucional modelo CNN se muestran en la figura 52, el tiempo de entrenamiento tardó aproximadamente una hora, y alcanzó porcentajes de precisión en el entrenamiento de 75,63%, con un porcentaje de precisión en la validación del 71,88%, este entrenamiento fue dispuesto con 5 pasos de validación por época, se establecieron 10 épocas de aprendizaje.

```
Epoch 1/10
4/5 [=======================>......] - ETA: 4s - loss: 2.2679 - acc: 0.3750 Epoch 1/10
5/5 [==============================] - 5s 1s/step - loss: 142.8472 - acc: 0.4625
5/5 [==============================] - 29s 6s/step - loss: 2.0904 - acc: 0.3875 - val_loss: 142.8472 - val_acc: 0.4625
Epoch 2/10
4/5 [=======================>......] - ETA: 3s - loss: 1.1147 - acc: 0.4688Epoch 1/10
5/5 [==============================] - 5s 1s/step - loss: 36.3536 - acc: 0.5000
5/5 [==============================] - 24s 5s/step - loss: 1.0975 - acc: 0.4750 - val_loss: 36.3536 - val_acc: 0.5000
Epoch 3/10
4/5 [=======================>......] - ETA: 3s - loss: 0.8756 - acc: 0.5469Epoch 1/10
5/5 [==============================] - 5s 1s/step - loss: 12.3491 - acc: 0.6250
5/5 [==============================] - 24s 5s/step - loss: 0.8904 - acc: 0.5437 - val_loss: 12.3491 - val_acc: 0.6250
Epoch 4/10
4/5 [=======================>......] - ETA: 3s - loss: 0.8392 - acc: 0.6484Epoch 1/10
5/5 [==============================] - 5s 1s/step - loss: 26.0460 - acc: 0.6625
5/5 [==============================] - 24s 5s/step - loss: 0.8286 - acc: 0.6562 - val_loss: 26.0460 - val_acc: 0.6625
Epoch 5/10
4/5 [=======================>......] - ETA: 3s - loss: 0.8559 - acc: 0.6094Epoch 1/10
5/5 [==============================] - 5s 1s/step - loss: 18.5636 - acc: 0.6500
5/5 [==============================] - 24s 5s/step - loss: 0.8414 - acc: 0.6125 - val_loss: 18.5636 - val_acc: 0.6500
Epoch 6/10
4/5 [=======================>......] - ETA: 3s - loss: 0.8088 - acc: 0.6328Epoch 1/10
5/5 [==============================] - 5s 1s/step - loss: 7.4496 - acc: 0.6562
5/5 [==============================] - 24s 5s/step - loss: 0.8165 - acc: 0.6375 - val_loss: 7.4496 - val_acc: 0.6562
Epoch 7/10
4/5 [=======================>......] - ETA: 3s - loss: 0.6563 - acc: 0.7266Epoch 1/10
5/5 [==============================] - 5s 1s/step - loss: 18.8562 - acc: 0.7312
5/5 [==============================] - 24s 5s/step - loss: 0.6940 - acc: 0.7125 - val_loss: 18.8562 - val_acc: 0.7312
Epoch 8/10
4/5 [=======================>......] - ETA: 3s - loss: 0.6146 - acc: 0.7656Epoch 1/10
5/5 [==============================] - 5s 1s/step - loss: 11.6158 - acc: 0.7188
5/5 [==============================] - 24s 5s/step - loss: 0.6355 - acc: 0.7563 - val_loss: 11.6158 - val_acc: 0.7188
Epoch 9/10
4/5 [=======================>......] - ETA: 3s - loss: 0.7651 - acc: 0.6953Epoch 1/10
5/5 [==============================] - 5s 1s/step - loss: 13.1450 - acc: 0.6750
5/5 [==============================] - 24s 5s/step - loss: 0.7889 - acc: 0.6750 - val_loss: 13.1450 - val_acc: 0.6750
Epoch 10/10
4/5 [=======================>......] - ETA: 3s - loss: 0.6774 - acc: 0.6875Epoch 1/10
5/5 [==============================] - 5s 1s/step - loss: 11.8834 - acc: 0.6625
5/5 [==============================] - 24s 5s/step - loss: 0.6564 - acc: 0.7000 - val_loss: 11.8834 - val_acc: 0.6625
```

figura 52 Métricas de entrenamiento y validación red convolucional CNN

4.1.2 Modelo CNN2 Datos Aumentados

Los resultados de entrenamiento de la red neuronal convolucional modeloCNN2_AD con aumento de datos se muestran en la figura 53, el tiempo de entrenamiento tardó aproximadamente una hora, y alcanzó porcentajes de precisión en el entrenamiento de 85%, con un porcentaje de precisión en la validación del 88,75%, este entrenamiento fue dispuesto con 5 pasos de validación por época, y se establecieron 10 épocas de aprendizaje.

```
Epoch 1/10
4/5 [=====================>......] - ETA: 4s - loss: 0.3914 - acc: 0.8516Epoch 1/10
5/5 [==============================] - 5s 942ms/step - loss: 7.3092 - acc: 0.8875
5/5 [==============================] - 27s 5s/step - loss: 0.4498 - acc: 0.8438 - val_loss: 7.3092 - val_acc: 0.8875
Epoch 2/10
4/5 [=====================>......] - ETA: 3s - loss: 0.5105 - acc: 0.7656Epoch 1/10
5/5 [==============================] - 5s 942ms/step - loss: 8.2416 - acc: 0.8813
5/5 [==============================] - 23s 5s/step - loss: 0.5087 - acc: 0.7500 - val_loss: 8.2416 - val_acc: 0.8813
Epoch 3/10
4/5 [=====================>......] - ETA: 3s - loss: 0.4256 - acc: 0.8359Epoch 1/10
5/5 [==============================] - 5s 936ms/step - loss: 10.9698 - acc: 0.8687
5/5 [==============================] - 23s 5s/step - loss: 0.4015 - acc: 0.8375 - val_loss: 10.9698 - val_acc: 0.8687
Epoch 4/10
4/5 [=====================>......] - ETA: 3s - loss: 0.4142 - acc: 0.8281Epoch 1/10
5/5 [==============================] - 5s 933ms/step - loss: 15.0695 - acc: 0.8625
5/5 [==============================] - 23s 5s/step - loss: 0.4049 - acc: 0.8250 - val_loss: 15.0695 - val_acc: 0.8625
Epoch 5/10
4/5 [=====================>......] - ETA: 3s - loss: 0.4338 - acc: 0.8125Epoch 1/10
5/5 [==============================] - 5s 923ms/step - loss: 27.5848 - acc: 0.8375
5/5 [==============================] - 23s 5s/step - loss: 0.4107 - acc: 0.8250 - val_loss: 27.5848 - val_acc: 0.8375
Epoch 6/10
4/5 [=====================>......] - ETA: 3s - loss: 0.3107 - acc: 0.8906Epoch 1/10
5/5 [==============================] - 5s 956ms/step - loss: 25.2614 - acc: 0.8313
5/5 [==============================] - 23s 5s/step - loss: 0.3462 - acc: 0.8687 - val_loss: 25.2614 - val_acc: 0.8313
Epoch 7/10
4/5 [=====================>......] - ETA: 3s - loss: 0.4635 - acc: 0.8047Epoch 1/10
5/5 [==============================] - 5s 921ms/step - loss: 34.3009 - acc: 0.8438
5/5 [==============================] - 22s 4s/step - loss: 0.4628 - acc: 0.8125 - val_loss: 34.3009 - val_acc: 0.8438
Epoch 8/10
4/5 [=====================>......] - ETA: 3s - loss: 0.3255 - acc: 0.8359Epoch 1/10
5/5 [==============================] - 5s 932ms/step - loss: 28.3926 - acc: 0.8562
5/5 [==============================] - 23s 5s/step - loss: 0.3143 - acc: 0.8500 - val_loss: 28.3926 - val_acc: 0.8562
Epoch 9/10
4/5 [=====================>......] - ETA: 3s - loss: 0.3665 - acc: 0.8047Epoch 1/10
5/5 [==============================] - 5s 930ms/step - loss: 25.1842 - acc: 0.8500
5/5 [==============================] - 23s 5s/step - loss: 0.4253 - acc: 0.7937 - val_loss: 25.1842 - val_acc: 0.8500
Epoch 10/10
4/5 [=====================>......] - ETA: 3s - loss: 0.5475 - acc: 0.8047Epoch 1/10
5/5 [==============================] - 5s 938ms/step - loss: 18.8451 - acc: 0.8500
5/5 [==============================] - 23s 5s/step - loss: 0.5193 - acc: 0.8188 - val_loss: 18.8451 - val_acc: 0.8500
```

figura 53 Métricas de entrenamiento y validación red convolucional CNN2 con aumento de datos

4.1.2 ELECCIÓN DE RED NEURONAL

En la gráfica de la figura 54 se aprecia como el modelo de la red neuronal convolucional con datos aumentados aprende más rápido a diferenciar entre un tipo de tumor y otro.

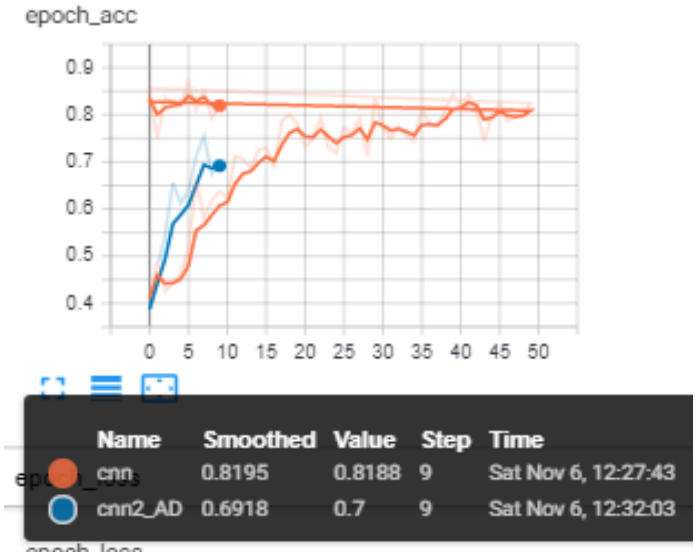


figura 54 Diferencia entre la precisión de entrenamiento del modelo CNN, y el modelo CNN2con aumento de datos.

En la gráfica de la figura 55 se aprecia como la red neuronal convolucional con datos aumentados línea azul al transcurrir las épocas disminuye sus pérdidas en comparación con la red convolucional CNN, por ello exportamos el modelo entrenado con datos entrenados a la raspberry pi4.

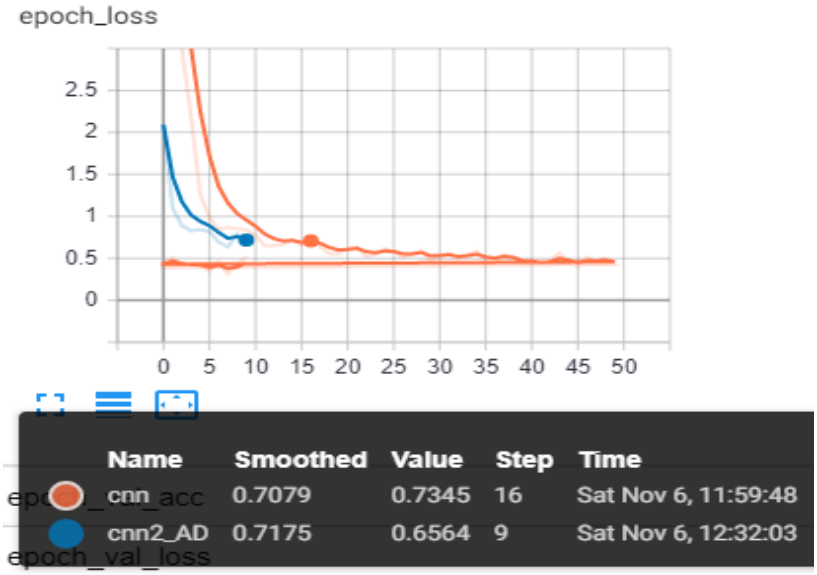


figura 55 Diferencia entre las pérdidas de entrenamiento del modelo CNN, y el modelo CNN2con aumento de datos

4.2 ENTRENAMIENTO DE LA RED CNN2 CON DATOS AUMENTADOS

La red creada fue entrenada con diferentes transformadas discretas de Wavelet aplicando wavelet madre Bior1.1, Coif1, Db1, Dmey, Fk6, Rbio1.1, Sym2, Haar. El conjunto de imágenes originales constaba de 3064 imágenes y cada set de imágenes fue subido a Google drive. Al final teníamos 8 conjuntos de imágenes cada uno con 3064 imágenes de los 3 tipos de tumor cerebral meningioma, glioma, pituitary, distribuidas en 95% de imágenes de entrenamiento y 5% imágenes de validación.

La red fue entrenada en Google colab con procesamiento gráfico con 10 épocas, 5 pasos por época, 50 pasos de validación.

4.2.1 Métricas Red Con Transformada Wavelet Madre Bior1.1

En la figura 56 se aprecian las métricas de precisión para la red CNN2 entrenada con las 3064 imágenes de los tres tipos de tumor, en el eje horizontal se aprecian las épocas, y el eje vertical la precisión notesé que solamente se alcanza un 53% de precisión en la red.

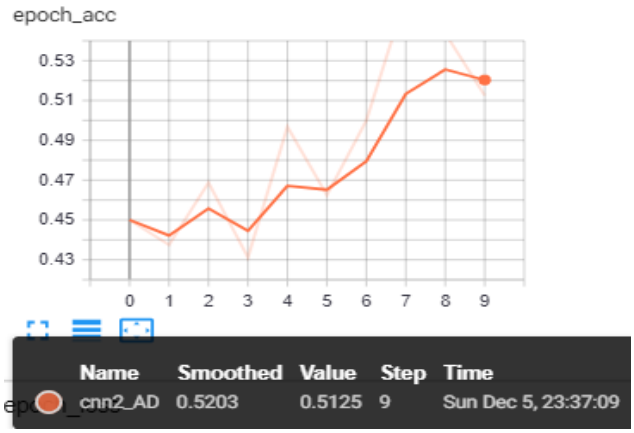


figura 56 Precisión red CNN2 con wavelet madre Bior 1.1

4.2.2 Métricas Red Con Transformada Wavelet Madre Coif1

En la figura 57 se aprecian las métricas de precisión para la red CNN2 entrenada con las 3064 imágenes de los tres tipos de tumor, en el eje horizontal se aprecian las épocas, y el eje vertical la precisión notesé que solamente se alcanza un 48% de precisión en la red.

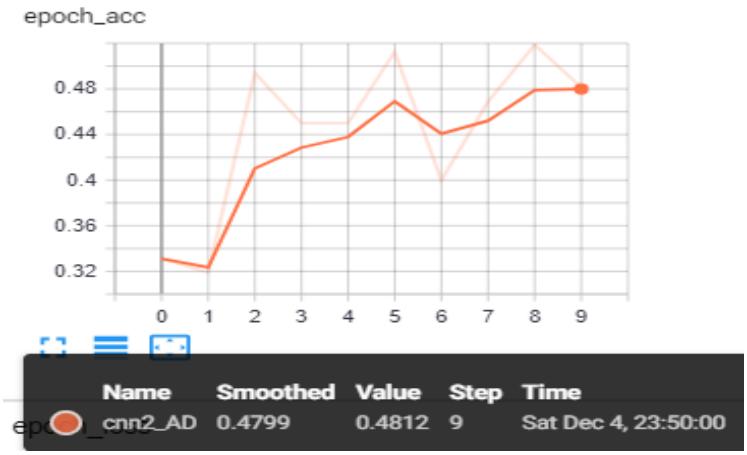


figura 57 Precisión red CNN2 con wavelet madre Coif1

4.2.3 Métricas Red Con Transformada Wavelet Madre Db1

En la figura 58 se aprecian las métricas de precisión para la red CNN2 entrenada con las 3064 imágenes de los tres tipos de tumor, en el eje horizontal se aprecian las épocas, y el eje vertical la precisión notesé que solamente se alcanza un 60% de precisión en la red.

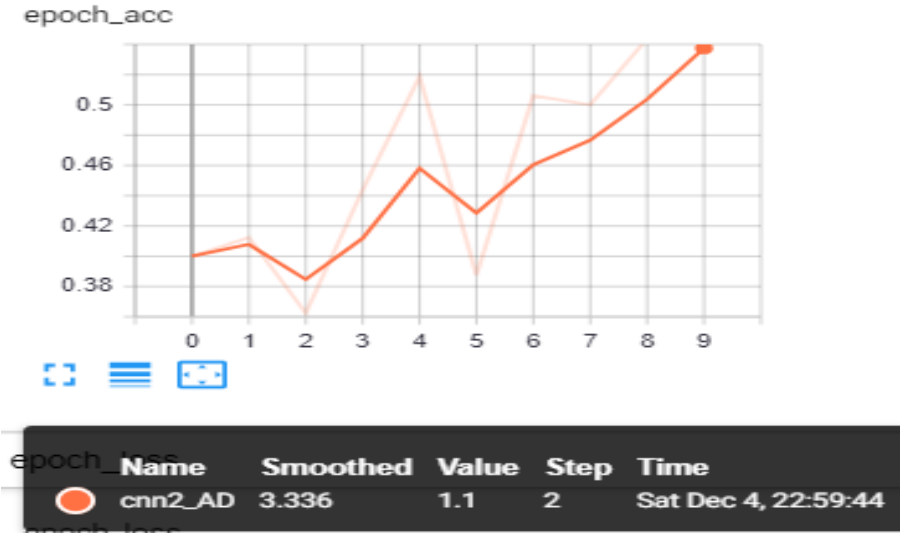


figura 58 Precisión red CNN2 con wavelet madre Db1

4.2.4 Métricas Red Con Transformada Wavelet Madre Dmey

En la figura 59 se aprecian las métricas de precisión para la red CNN2 entrenada con las 3064 imágenes de los tres tipos de tumor, en el eje horizontal se aprecian las épocas, y el eje vertical la precisión notesé que solamente se alcanza un 58% de precisión en la red.

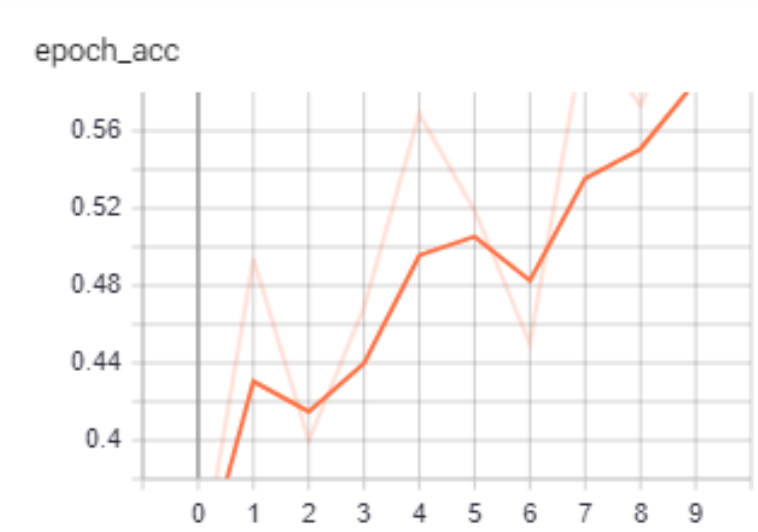


figura 59 Precisión red CNN2 con wavelet madre Dmey

4.2.5 Métricas Red Con Transformada wavelet Madre Fk6

En la figura 60 se aprecian las métricas de precisión para la red CNN2 entrenada con las 3064 imágenes de los tres tipos de tumor, en el eje horizontal se aprecian las épocas, y el eje vertical la precisión notesé que solamente se alcanza un 52.5% de precisión en la red.

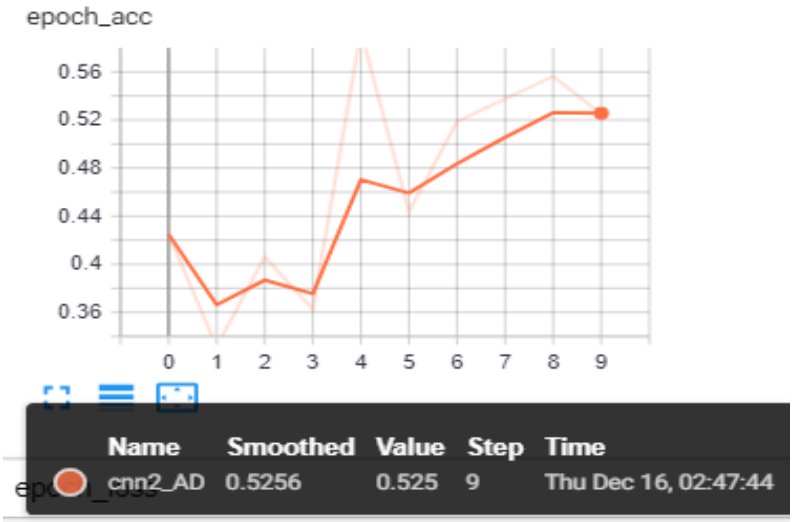


figura 60 Precisión red CNN2 con wavelet madre Fk6

4.2.6 Métricas Red Con Transformada wavelet Madre Haar

En la figura 61 se aprecian las métricas de precisión para la red CNN2 entrenada con las 3064 imágenes de los tres tipos de tumor, en el eje horizontal se aprecian las épocas, y el eje vertical la precisión notesé que alcanza un 69.18% de precisión en la red. Como la red entrenada con transformada Wavelet Madre Haar fue la que mejor resultados de precisión

arrojó se eligió el conjunto de imágenes para reentrenar la red con imágenes duplicadas. Ver sección 4.3

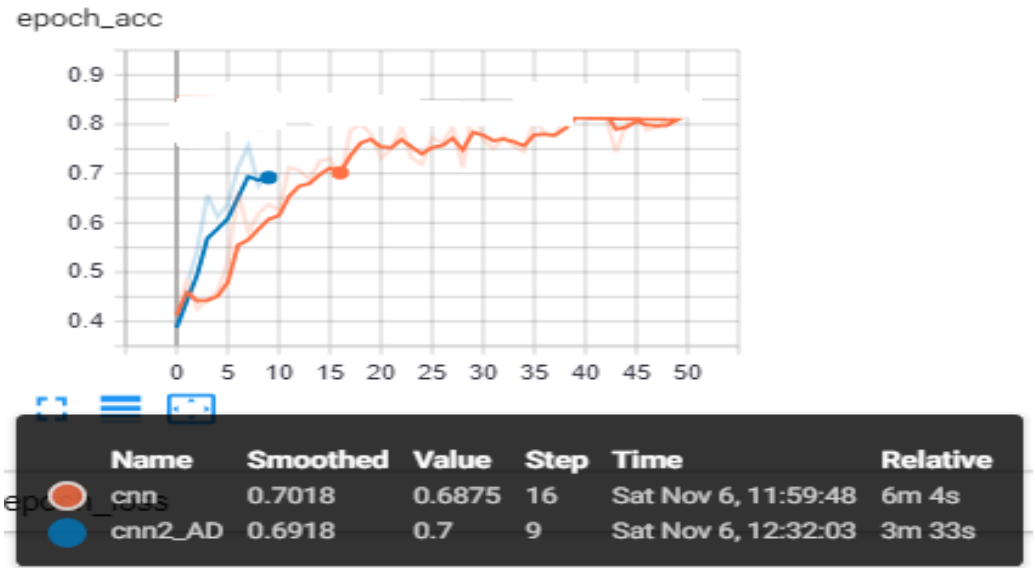


figura 61 Precisión red CNN2 con wavelet madre Haar

4.2.6 Métricas Red Con Transformada wavelet Madre Haar

En la figura 62 se aprecian las métricas de precisión para la red CNN2 entrenada con las 3064 imágenes de los tres tipos de tumor, en el eje horizontal se aprecian las épocas, y el eje vertical la precisión notesé que solamente se alcanza un 52% de precisión en la red.

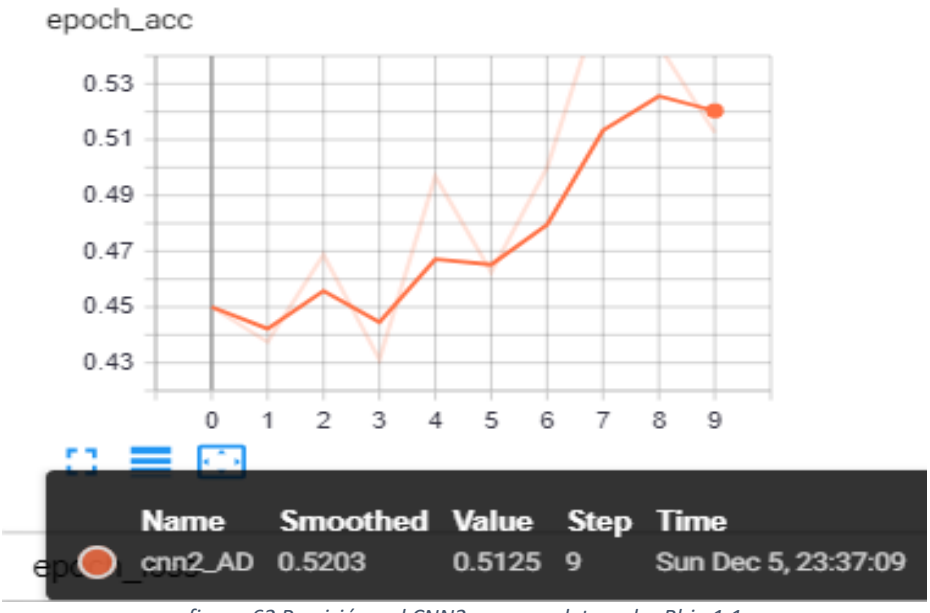


figura 62 Precisión red CNN2 con wavelet madre Rbio 1.1

4.2.7 Métricas Red Con Transformada wavelet Madre Sym2

En la figura 63 se aprecian las métricas de precisión para la red CNN2 entrenada con las 3064 imágenes de los tres tipos de tumor, en el eje horizontal se aprecian las épocas, y el eje vertical la precisión notesé que solamente se alcanza un 50% de precisión en la red.

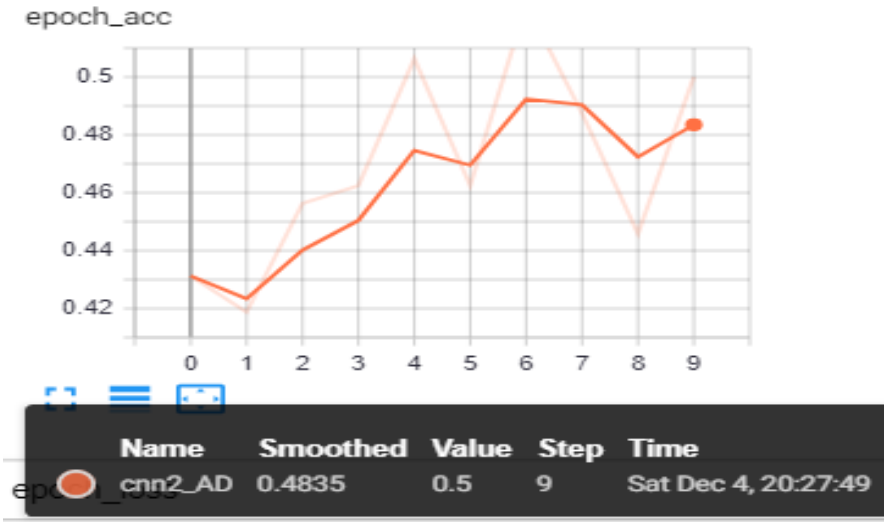


figura 63 Precisión red CNN2 con wavelet madre Sym2

4.3 REENTRENAMIENTO DE LA RED CNN2 CON DATOS AUMENTADOS

La red creada fue reentrenada con transformadas discretas de Wavelet aplicando wavelet madre Haar. El conjunto de imágenes originales constaba de 3064 imágenes y fueron duplicadas con imágenes giradas 180° sin inversión de eje, al final teníamos un set de datos de 6128 con los tres tipos de tumor cerebral meningioma, glioma, pituitary, distribuidas en 95% de imágenes de entrenamiento y 5% imágenes de validación, las imágenes de prueba 5 por cada tipo de tumor fueron restadas al set de entrenamiento. Al final teníamos 15 imágenes de prueba.

La red fue reentrenada en Google colab con procesamiento gráfico con 50 épocas, 10 pasos por época, 50 pasos de validación, 50 pasos por época de validación. Con aumento de datos 10% de acercamiento y 10% de giro horizontal aleatorio al conjunto de las 6128 imágenes.

En la figura 64 se muestran las métricas de entrenamiento y validación desde la época 1 a la época 15.

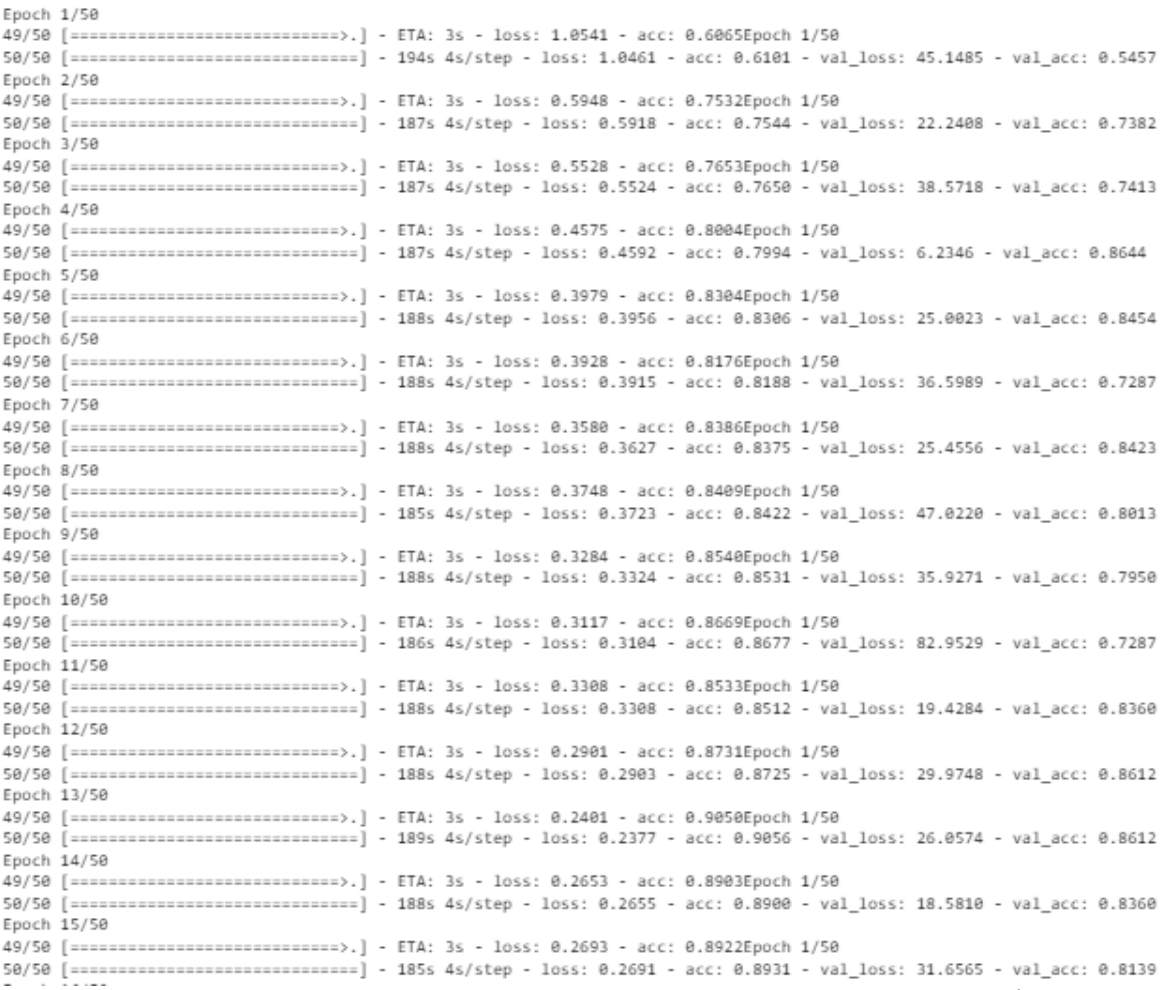


figura 64 Métricas de entrenamiento y validación Red CNN2 Reentrenada con Wavelet Madre Haar Época 1-15.

En la figura 65 se muestran las métricas de entrenamiento y validación desde la época 16 a la época 30.

```
Epoch 16/50
49/50 [============================>.] - ETA: 3s - loss: 0.2332 - acc: 0.8954Epoch 1/50
50/50 [==============================] - 188s 4s/step - loss: 0.2312 - acc: 0.8963 - val_loss: 40.4052 - val_acc: 0.8549
Epoch 17/50
49/50 [============================>.] - ETA: 3s - loss: 0.2318 - acc: 0.9039Epoch 1/50
50/50 [==============================] - 185s 4s/step - loss: 0.2337 - acc: 0.9020 - val_loss: 56.5656 - val_acc: 0.8675
Epoch 18/50
49/50 [============================>.] - ETA: 3s - loss: 0.2101 - acc: 0.9101Epoch 1/50
50/50 [==============================] - 187s 4s/step - loss: 0.2074 - acc: 0.9106 - val_loss: 39.3696 - val_acc: 0.8233
Epoch 19/50
49/50 [============================>.] - ETA: 3s - loss: 0.2243 - acc: 0.9082Epoch 1/50
50/50 [==============================] - 187s 4s/step - loss: 0.2249 - acc: 0.9081 - val_loss: 24.1680 - val_acc: 0.8107
Epoch 20/50
49/50 [============================>.] - ETA: 3s - loss: 0.2181 - acc: 0.9123Epoch 1/50
50/50 [==============================] - 185s 4s/step - loss: 0.2204 - acc: 0.9116 - val_loss: 31.6386 - val_acc: 0.8360
Epoch 21/50
49/50 [============================>.] - ETA: 3s - loss: 0.1823 - acc: 0.9254Epoch 1/50
50/50 [==============================] - 188s 4s/step - loss: 0.1810 - acc: 0.9262 - val_loss: 52.2047 - val_acc: 0.8612
Epoch 22/50
49/50 [============================>.] - ETA: 3s - loss: 0.1852 - acc: 0.9292Epoch 1/50
50/50 [==============================] - 188s 4s/step - loss: 0.1856 - acc: 0.9287 - val_loss: 34.2686 - val_acc: 0.8580
Epoch 23/50
49/50 [============================>.] - ETA: 3s - loss: 0.1883 - acc: 0.9184Epoch 1/50
50/50 [==============================] - 185s 4s/step - loss: 0.1852 - acc: 0.9200 - val_loss: 65.4054 - val_acc: 0.7886
Epoch 24/50
49/50 [============================>.] - ETA: 3s - loss: 0.1562 - acc: 0.9383Epoch 1/50
50/50 [==============================] - 190s 4s/step - loss: 0.1586 - acc: 0.9370 - val_loss: 83.5812 - val_acc: 0.8360
Epoch 25/50
49/50 [============================>.] - ETA: 3s - loss: 0.1730 - acc: 0.9228Epoch 1/50
50/50 [==============================] - 189s 4s/step - loss: 0.1729 - acc: 0.9231 - val_loss: 71.2653 - val_acc: 0.8423
Epoch 26/50
49/50 [============================>.] - ETA: 3s - loss: 0.1735 - acc: 0.9343Epoch 1/50
50/50 [==============================] - 188s 4s/step - loss: 0.1741 - acc: 0.9337 - val_loss: 35.8970 - val_acc: 0.8423
Epoch 27/50
49/50 [============================>.] - ETA: 3s - loss: 0.1633 - acc: 0.9375Epoch 1/50
50/50 [==============================] - 190s 4s/step - loss: 0.1622 - acc: 0.9375 - val_loss: 47.0525 - val_acc: 0.8612
Epoch 28/50
49/50 [============================>.] - ETA: 3s - loss: 0.1292 - acc: 0.9490Epoch 1/50
50/50 [==============================] - 189s 4s/step - loss: 0.1337 - acc: 0.9469 - val_loss: 68.3229 - val_acc: 0.8391
Epoch 29/50
49/50 [============================>.] - ETA: 3s - loss: 0.1408 - acc: 0.9539Epoch 1/50
50/50 [==============================] - 185s 4s/step - loss: 0.1402 - acc: 0.9536 - val_loss: 52.1430 - val_acc: 0.8675
Epoch 30/50
49/50 [============================>.] - ETA: 3s - loss: 0.1174 - acc: 0.9573Epoch 1/50
50/50 [==============================] - 188s 4s/step - loss: 0.1158 - acc: 0.9581 - val_loss: 51.2244 - val_acc: 0.8517
```

figura 65 Métricas de entrenamiento y validación Red CNN2 Reentrenada con Wavelet Madre Haar Época 16-30.

En la figura 66 se muestran las métricas de entrenamiento y validación desde la época 16 a la época 30.

```
Epoch 31/50
49/50 [============================>.] - ETA: 3s - loss: 0.0915 - acc: 0.9643Epoch 1/50
50/50 [==============================] - 188s 4s/step - loss: 0.0902 - acc: 0.9650 - val_loss: 87.2433 - val_acc: 0.8549
Epoch 32/50
49/50 [============================>.] - ETA: 3s - loss: 0.0917 - acc: 0.9662Epoch 1/50
50/50 [==============================] - 185s 4s/step - loss: 0.0915 - acc: 0.9656 - val_loss: 106.2272 - val_acc: 0.8076
Epoch 33/50
49/50 [============================>.] - ETA: 3s - loss: 0.0773 - acc: 0.9700Epoch 1/50
50/50 [==============================] - 188s 4s/step - loss: 0.0807 - acc: 0.9694 - val_loss: 146.6679 - val_acc: 0.7886
Epoch 34/50
49/50 [============================>.] - ETA: 3s - loss: 0.1133 - acc: 0.9598Epoch 1/50
50/50 [==============================] - 188s 4s/step - loss: 0.1117 - acc: 0.9606 - val_loss: 76.8030 - val_acc: 0.8738
Epoch 35/50
49/50 [============================>.] - ETA: 3s - loss: 0.1059 - acc: 0.9668Epoch 1/50
50/50 [==============================] - 187s 4s/step - loss: 0.1046 - acc: 0.9675 - val_loss: 67.0858 - val_acc: 0.8833
Epoch 36/50
49/50 [============================>.] - ETA: 3s - loss: 0.0706 - acc: 0.9739Epoch 1/50
50/50 [==============================] - 187s 4s/step - loss: 0.0712 - acc: 0.9737 - val_loss: 104.0451 - val_acc: 0.8580
Epoch 37/50
49/50 [============================>.] - ETA: 3s - loss: 0.0634 - acc: 0.9721Epoch 1/50
50/50 [==============================] - 185s 4s/step - loss: 0.0630 - acc: 0.9726 - val_loss: 78.5023 - val_acc: 0.8580
Epoch 38/50
49/50 [============================>.] - ETA: 3s - loss: 0.0780 - acc: 0.9719Epoch 1/50
50/50 [==============================] - 188s 4s/step - loss: 0.0767 - acc: 0.9725 - val_loss: 104.0007 - val_acc: 0.8549
Epoch 39/50
49/50 [============================>.] - ETA: 3s - loss: 0.0551 - acc: 0.9770Epoch 1/50
50/50 [==============================] - 188s 4s/step - loss: 0.0548 - acc: 0.9775 - val_loss: 82.4475 - val_acc: 0.8423
Epoch 40/50
49/50 [============================>.] - ETA: 3s - loss: 0.0736 - acc: 0.9739Epoch 1/50
50/50 [==============================] - 188s 4s/step - loss: 0.0724 - acc: 0.9744 - val_loss: 106.2246 - val_acc: 0.8360
Epoch 41/50
49/50 [============================>.] - ETA: 3s - loss: 0.0572 - acc: 0.9796Epoch 1/50
50/50 [==============================] - 189s 4s/step - loss: 0.0562 - acc: 0.9800 - val_loss: 100.6640 - val_acc: 0.8644
Epoch 42/50
49/50 [============================>.] - ETA: 3s - loss: 0.0584 - acc: 0.9818Epoch 1/50
50/50 [==============================] - 186s 4s/step - loss: 0.0581 - acc: 0.9816 - val_loss: 112.6905 - val_acc: 0.8328
Epoch 43/50
49/50 [============================>.] - ETA: 3s - loss: 0.0448 - acc: 0.9828Epoch 1/50
50/50 [==============================] - 188s 4s/step - loss: 0.0440 - acc: 0.9831 - val_loss: 104.4830 - val_acc: 0.8233
Epoch 44/50
49/50 [============================>.] - ETA: 3s - loss: 0.0530 - acc: 0.9815Epoch 1/50
50/50 [==============================] - 188s 4s/step - loss: 0.0661 - acc: 0.9806 - val_loss: 174.6660 - val_acc: 0.8454
```

figura 66 Métricas de entrenamiento y validación Red CNN2 Reentrenada con Wavelet Madre Haar Época 31-44.

En la figura 67 se muestran las métricas de entrenamiento y validación desde la época 16 a la época 30.

```
Epoch 45/50
49/50 [============================>.] - ETA: 3s - loss: 0.0948 - acc: 0.9701Epoch 1/50
50/50 [==============================] - 185s 4s/step - loss: 0.0933 - acc: 0.9707 - val_loss: 128.2141 - val_acc: 0.8265
Epoch 46/50
49/50 [============================>.] - ETA: 3s - loss: 0.0643 - acc: 0.9777Epoch 1/50
50/50 [==============================] - 184s 4s/step - loss: 0.0636 - acc: 0.9775 - val_loss: 118.3461 - val_acc: 0.8644
Epoch 47/50
49/50 [============================>.] - ETA: 3s - loss: 0.0419 - acc: 0.9831Epoch 1/50
50/50 [==============================] - 186s 4s/step - loss: 0.0435 - acc: 0.9828 - val_loss: 223.2253 - val_acc: 0.7697
Epoch 48/50
49/50 [============================>.] - ETA: 3s - loss: 0.0433 - acc: 0.9821Epoch 1/50
50/50 [==============================] - 187s 4s/step - loss: 0.0428 - acc: 0.9825 - val_loss: 153.6536 - val_acc: 0.8454
Epoch 49/50
49/50 [============================>.] - ETA: 3s - loss: 0.0428 - acc: 0.9872Epoch 1/50
50/50 [==============================] - 187s 4s/step - loss: 0.0426 - acc: 0.9875 - val_loss: 172.5272 - val_acc: 0.8202
Epoch 50/50
49/50 [============================>.] - ETA: 3s - loss: 0.0517 - acc: 0.9828Epoch 1/50
50/50 [==============================] - 189s 4s/step - loss: 0.0515 - acc: 0.9825 - val_loss: 158.0841 - val_acc: 0.8391
```

figura 67 Métricas de entrenamiento y validación Red CNN2 Reentrenada con Wavelet Madre Haar Época 45-50.

En la figura 68 se muestran las métricas de precisión de la red Reentrenada con Wavelet Madre Haar en el eje horizontal se muestran el transcurso de las épocas, y el eje vertical la precisión de la red al transcurrir las épocas se puede apreciar como la red aprende a clasificar los tres conjuntos de imágenes de cada tipo de tumor cerebral con una precisión del 98,25%. Con un tiempo de entrenamiento de 2 horas y 33 minutos. Este modelo fue importado desde Google colab a Raspberry pi4 para el cargue de nuevas imágenes.

```
modeloCNN2_AD.save("/content/modelo/modelo_tc.h5")
modeloCNN2_AD.save_weights("/content/modelo/pesos_tc.h5")
```

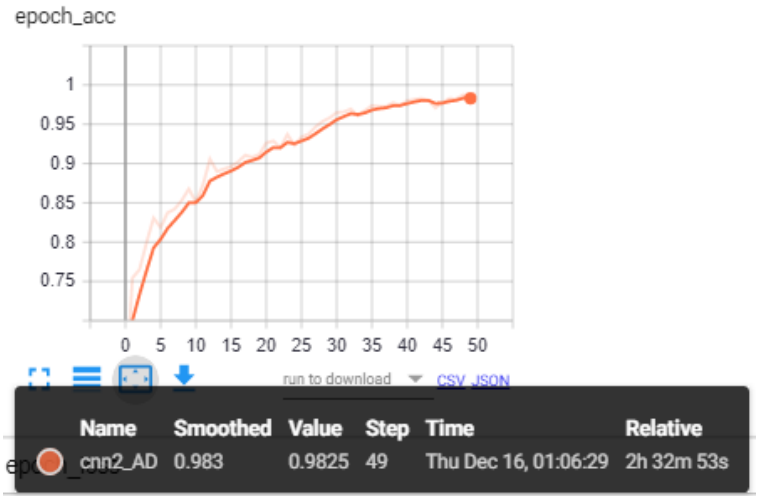


figura 68 métricas de precisión entrenamiento wavelet Haar 50 épocas

En la figura 69 se muestran las métricas de pérdidas de la red Reentrenada con Wavelet Madre Haar en el eje horizontal se muestran el transcurso de las épocas, y el eje vertical las pérdidas de la red al transcurrir las épocas notesé como la red hace casi nula la función de pérdidas al transcurrir las épocas.

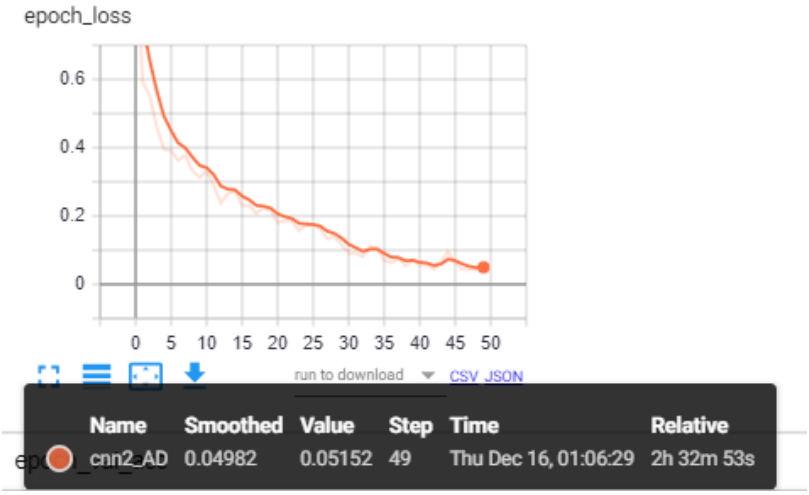


figura 69 métricas de Pérdidas entrenamiento wavelet Haar 50 épocas

4.4 ENTRENAMIENTO DE LA RED CNN2 CON DATOS AUMENTADOS

La red creada fue entrenada con las imágenes originales . El conjunto de imágenes originales constaba de 3064 imágenes con los tres tipos de tumor cerebral meningioma, glioma, pituitary, distribuidas en 95% de imágenes de entrenamiento y 5% imágenes de validación, las imágenes de prueba 5 por cada tipo de tumor fueron restadas al set de entrenamiento. Al final teníamos 15 imágenes de prueba.

La red fue reentrenada en Google colab con procesamiento gráfico con 50 épocas, 10 pasos por época, 50 pasos de validación, 50 pasos por época de validación. Con aumento de datos 10% de acercamiento y 10% de giro horizontal aleatorio al conjunto de las 3064 imágenes.

En la figura 70 se muestran las métricas de entrenamiento y validación desde la época 1 a la época 15.

```
Epoch 1/50
49/50 [============================>.] - ETA: 3s - loss: 1.5396 - acc: 0.5389Epoch 1/50
50/50 [==============================] - 212s 4s/step - loss: 1.5276 - acc: 0.5387 - val_loss: 130.2511 - val_acc: 0.4903
Epoch 2/50
49/50 [============================>.] - ETA: 3s - loss: 0.6347 - acc: 0.7246Epoch 1/50
50/50 [==============================] - 203s 4s/step - loss: 0.6342 - acc: 0.7245 - val_loss: 199.9126 - val_acc: 0.5677
Epoch 3/50
49/50 [============================>.] - ETA: 3s - loss: 0.5335 - acc: 0.7770Epoch 1/50
50/50 [==============================] - 202s 4s/step - loss: 0.5322 - acc: 0.7783 - val_loss: 149.5214 - val_acc: 0.6710
Epoch 4/50
49/50 [============================>.] - ETA: 3s - loss: 0.5011 - acc: 0.7883Epoch 1/50
50/50 [==============================] - 200s 4s/step - loss: 0.4994 - acc: 0.7906 - val_loss: 313.5375 - val_acc: 0.5935
Epoch 5/50
49/50 [============================>.] - ETA: 3s - loss: 0.4329 - acc: 0.8240Epoch 1/50
50/50 [==============================] - 201s 4s/step - loss: 0.4337 - acc: 0.8231 - val_loss: 481.4287 - val_acc: 0.6065
Epoch 6/50
49/50 [============================>.] - ETA: 3s - loss: 0.4074 - acc: 0.8316Epoch 1/50
50/50 [==============================] - 201s 4s/step - loss: 0.4091 - acc: 0.8300 - val_loss: 330.8632 - val_acc: 0.6129
Epoch 7/50
49/50 [============================>.] - ETA: 3s - loss: 0.3576 - acc: 0.8578Epoch 1/50
50/50 [==============================] - 204s 4s/step - loss: 0.3554 - acc: 0.8575 - val_loss: 296.7934 - val_acc: 0.6516
Epoch 8/50
49/50 [============================>.] - ETA: 3s - loss: 0.3624 - acc: 0.8540Epoch 1/50
50/50 [==============================] - 201s 4s/step - loss: 0.3624 - acc: 0.8537 - val_loss: 268.4771 - val_acc: 0.6452
Epoch 9/50
49/50 [============================>.] - ETA: 3s - loss: 0.3554 - acc: 0.8556Epoch 1/50
50/50 [==============================] - 199s 4s/step - loss: 0.3556 - acc: 0.8554 - val_loss: 144.8443 - val_acc: 0.6452
Epoch 10/50
49/50 [============================>.] - ETA: 3s - loss: 0.3265 - acc: 0.8754Epoch 1/50
50/50 [==============================] - 204s 4s/step - loss: 0.3245 - acc: 0.8760 - val_loss: 153.0966 - val_acc: 0.7355
Epoch 11/50
49/50 [============================>.] - ETA: 3s - loss: 0.3221 - acc: 0.8686Epoch 1/50
50/50 [==============================] - 200s 4s/step - loss: 0.3188 - acc: 0.8706 - val_loss: 271.3823 - val_acc: 0.6516
Epoch 12/50
49/50 [============================>.] - ETA: 3s - loss: 0.3056 - acc: 0.8792Epoch 1/50
50/50 [==============================] - 200s 4s/step - loss: 0.3056 - acc: 0.8791 - val_loss: 117.2773 - val_acc: 0.7677
Epoch 13/50
49/50 [============================>.] - ETA: 3s - loss: 0.2933 - acc: 0.8788Epoch 1/50
50/50 [==============================] - 202s 4s/step - loss: 0.2921 - acc: 0.8800 - val_loss: 277.2878 - val_acc: 0.5355
Epoch 14/50
49/50 [============================>.] - ETA: 3s - loss: 0.3195 - acc: 0.8773Epoch 1/50
50/50 [==============================] - 203s 4s/step - loss: 0.3187 - acc: 0.8766 - val_loss: 291.3168 - val_acc: 0.6194
Epoch 15/50
49/50 [============================>.] - ETA: 3s - loss: 0.2588 - acc: 0.8890Epoch 1/50
50/50 [==============================] - 203s 4s/step - loss: 0.2576 - acc: 0.8894 - val_loss: 136.3610 - val_acc: 0.6774
```

figura 70 Métricas de entrenamiento y validación Red CNN2 entrenada con imágenes originales épocas 1-15.

En la figura 71 se muestran las métricas de entrenamiento y validación desde la época 16 a la época 30.

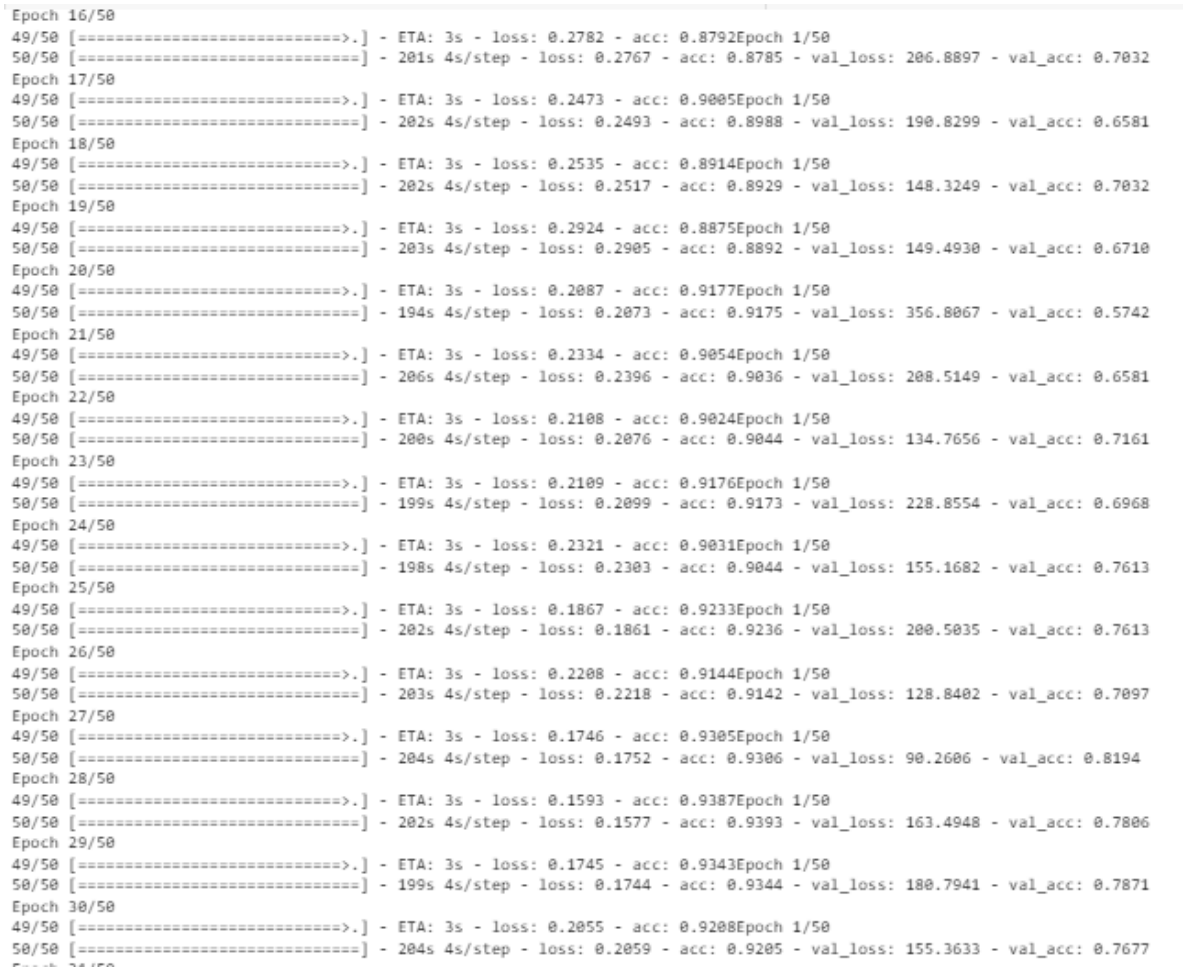


figura 71 Métricas de entrenamiento y validación Red CNN2 entrenada con imágenes originales épocas 16-30.

En la figura 72 se muestran las métricas de entrenamiento y validación desde la época 31 a la época 45.

```
Epoch 31/50
49/50 [============================>.] - ETA: 3s - loss: 0.1408 - acc: 0.9445Epoch 1/50
50/50 [==============================] - 202s 4s/step - loss: 0.1415 - acc: 0.9438 - val_loss: 93.6477 - val_acc: 0.8581
Epoch 32/50
49/50 [============================>.] - ETA: 3s - loss: 0.1599 - acc: 0.9342Epoch 1/50
50/50 [==============================] - 202s 4s/step - loss: 0.1596 - acc: 0.9343 - val_loss: 220.6554 - val_acc: 0.6323
Epoch 33/50
49/50 [============================>.] - ETA: 3s - loss: 0.1461 - acc: 0.9458Epoch 1/50
50/50 [==============================] - 201s 4s/step - loss: 0.1466 - acc: 0.9444 - val_loss: 131.7958 - val_acc: 0.8258
Epoch 34/50
49/50 [============================>.] - ETA: 3s - loss: 0.1743 - acc: 0.9324Epoch 1/50
50/50 [==============================] - 203s 4s/step - loss: 0.1721 - acc: 0.9337 - val_loss: 151.9622 - val_acc: 0.7742
Epoch 35/50
49/50 [============================>.] - ETA: 3s - loss: 0.1485 - acc: 0.9393Epoch 1/50
50/50 [==============================] - 201s 4s/step - loss: 0.1480 - acc: 0.9399 - val_loss: 230.2948 - val_acc: 0.7871
Epoch 36/50
49/50 [============================>.] - ETA: 3s - loss: 0.1346 - acc: 0.9413Epoch 1/50
50/50 [==============================] - 202s 4s/step - loss: 0.1347 - acc: 0.9413 - val_loss: 197.8016 - val_acc: 0.7613
Epoch 37/50
49/50 [============================>.] - ETA: 3s - loss: 0.1611 - acc: 0.9360Epoch 1/50
50/50 [==============================] - 200s 4s/step - loss: 0.1616 - acc: 0.9366 - val_loss: 219.3514 - val_acc: 0.8065
Epoch 38/50
49/50 [============================>.] - ETA: 3s - loss: 0.1312 - acc: 0.9554Epoch 1/50
50/50 [==============================] - 202s 4s/step - loss: 0.1330 - acc: 0.9544 - val_loss: 181.0734 - val_acc: 0.7548
Epoch 39/50
49/50 [============================>.] - ETA: 3s - loss: 0.1194 - acc: 0.9514Epoch 1/50
50/50 [==============================] - 202s 4s/step - loss: 0.1207 - acc: 0.9518 - val_loss: 104.5203 - val_acc: 0.8194
Epoch 40/50
49/50 [============================>.] - ETA: 3s - loss: 0.1425 - acc: 0.9439Epoch 1/50
50/50 [==============================] - 196s 4s/step - loss: 0.1420 - acc: 0.9438 - val_loss: 209.8638 - val_acc: 0.7419
Epoch 41/50
49/50 [============================>.] - ETA: 3s - loss: 0.1215 - acc: 0.9547Epoch 1/50
50/50 [==============================] - 204s 4s/step - loss: 0.1247 - acc: 0.9538 - val_loss: 188.1036 - val_acc: 0.7806
Epoch 42/50
49/50 [============================>.] - ETA: 3s - loss: 0.1143 - acc: 0.9546Epoch 1/50
50/50 [==============================] - 197s 4s/step - loss: 0.1146 - acc: 0.9537 - val_loss: 117.0446 - val_acc: 0.8194
Epoch 43/50
49/50 [============================>.] - ETA: 3s - loss: 0.1222 - acc: 0.9546Epoch 1/50
50/50 [==============================] - 199s 4s/step - loss: 0.1203 - acc: 0.9555 - val_loss: 210.2628 - val_acc: 0.7935
Epoch 44/50
49/50 [============================>.] - ETA: 3s - loss: 0.1166 - acc: 0.9597Epoch 1/50
50/50 [==============================] - 199s 4s/step - loss: 0.1157 - acc: 0.9599 - val_loss: 166.8310 - val_acc: 0.7613
Epoch 45/50
49/50 [============================>.] - ETA: 3s - loss: 0.1129 - acc: 0.9541Epoch 1/50
50/50 [==============================] - 201s 4s/step - loss: 0.1119 - acc: 0.9544 - val_loss: 246.4737 - val_acc: 0.7419
```

figura 72 Métricas de entrenamiento y validación Red CNN2 entrenada con imágenes originales épocas 31-45.

En la figura 73 se muestran las métricas de entrenamiento y validación desde la época 46 a la época 50.

```
Epoch 46/50
49/50 [============================>.] - ETA: 3s - loss: 0.1181 - acc: 0.9547Epoch 1/50
50/50 [==============================] - 199s 4s/step - loss: 0.1181 - acc: 0.9544 - val_loss: 201.3534 - val_acc: 0.7806
Epoch 47/50
49/50 [============================>.] - ETA: 3s - loss: 0.1068 - acc: 0.9681Epoch 1/50
50/50 [==============================] - 200s 4s/step - loss: 0.1072 - acc: 0.9674 - val_loss: 244.8072 - val_acc: 0.6968
Epoch 48/50
49/50 [============================>.] - ETA: 3s - loss: 0.0953 - acc: 0.9649Epoch 1/50
50/50 [==============================] - 199s 4s/step - loss: 0.0937 - acc: 0.9656 - val_loss: 165.9776 - val_acc: 0.7871
Epoch 49/50
49/50 [============================>.] - ETA: 3s - loss: 0.0957 - acc: 0.9642Epoch 1/50
50/50 [==============================] - 196s 4s/step - loss: 0.0955 - acc: 0.9643 - val_loss: 398.2548 - val_acc: 0.6710
Epoch 50/50
49/50 [============================>.] - ETA: 3s - loss: 0.1040 - acc: 0.9617Epoch 1/50
50/50 [==============================] - 200s 4s/step - loss: 0.1042 - acc: 0.9619 - val_loss: 202.6423 - val_acc: 0.7484
```

figura 73 Métricas de entrenamiento y validación Red CNN2 entrenada con imágenes originales épocas 46-50.

En la figura 74 se muestran las métricas de precisión de la red entrenada con Imágenes de resonancia magnética originales en el eje horizontal se muestran el transcurso de las épocas, y el eje vertical la precisión de la red al transcurrir las épocas se puede apreciar como la red aprende a clasificar los tres conjuntos de imágenes de cada tipo de tumor cerebral con una precisión del 96,19% con un tiempo de entrenamiento de 2 horas y 44 minutos.

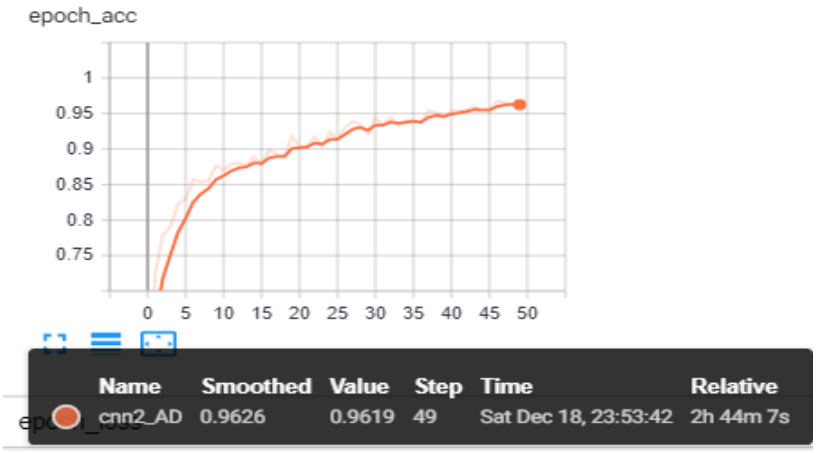


figura 74 métricas de precisión entrenamiento con 3064 Imágenes originales de resonancia magnética con 50 épocas.

En la figura 75 se muestran las métricas de pérdidas de la red entrenada con imágenes originales de resonancia magnética, en el eje horizontal se muestran el transcurso de las épocas, y el eje vertical las pérdidas de la red al transcurrir las épocas notesé como la red hace casi nula la función de pérdidas al transcurrir las épocas.

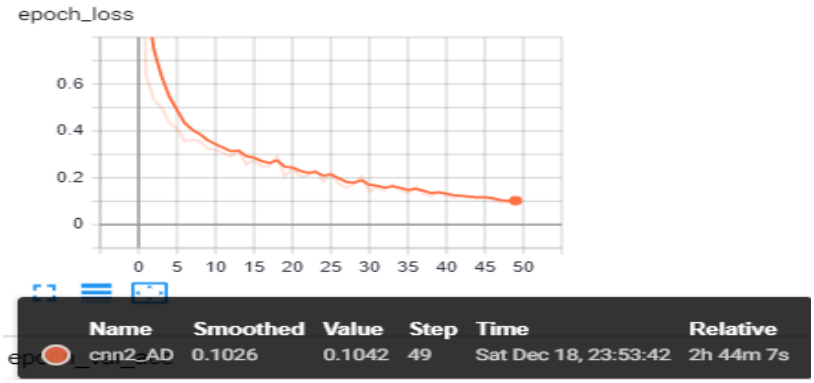


figura 75 métricas de pérdidas entrenamiento con 3064 Imágenes originales de resonancia magnética con 50 épocas.

4.5 TRANSFORMADAS DISCRETAS DE WAVELET EN IMÁGENES

A continuación, se muestran las transformadas wavelets de algunas imágenes de cada tipo de tumor cerebral, tumor glioma, tumor meningioma, y tumor pituitary.

4.5.1 Tumor Glioma con Transformada wavelet tipo Haar con un nivel de descomposición

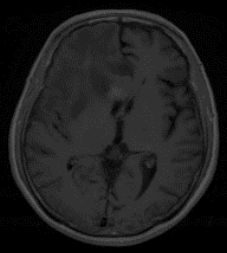
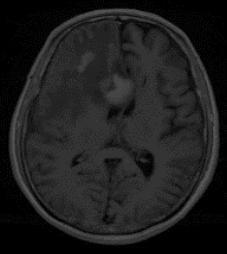
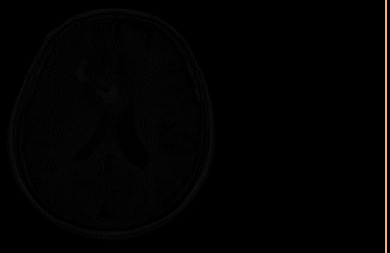
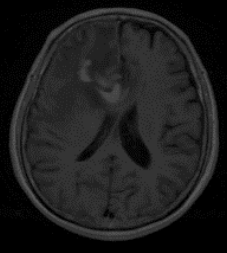
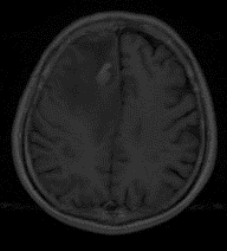
IMAGEN DE TUMOR CEREBRAL	TRANSFORMADA WAVELET
 512X512	 256X256
 512X512	 256X256
 512X512	 256X256
 512X512	 256X256

Tabla 4 Imágenes de tumor cerebral glioma y sus transformadas de wavelet

4.5.2 Tumor Meningioma con Transformada wavelet tipo Haar con un nivel de descomposición

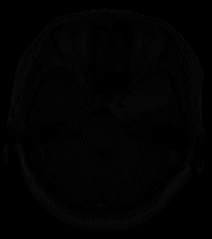
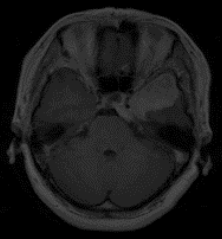
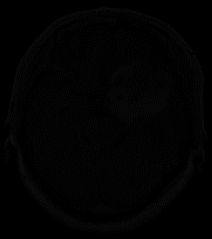
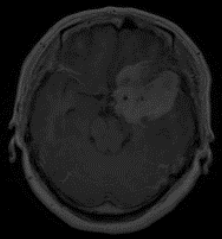
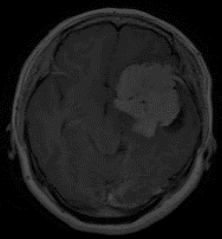
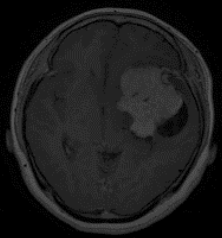
IMAGEN DE TUMOR CEREBRAL	TRANSFORMADA WAVELET
 512X512	 256X256
 512X512	 256X256
 512X512	 256X256
 512X512	 256X256

Tabla 5 Imágenes de tumor cerebral meningioma y sus transformadas de wavelet

4.5.3 Tumor Pituitary con Transformada wavelet tipo Haar con un nivel de descomposición

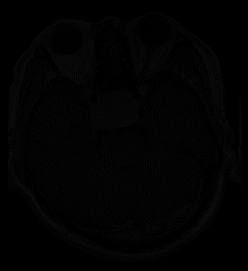
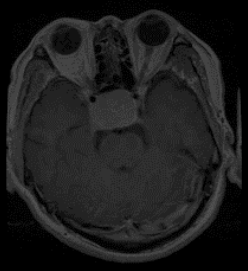
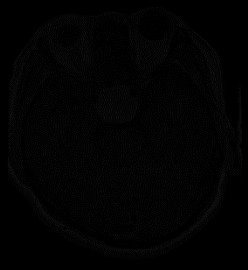
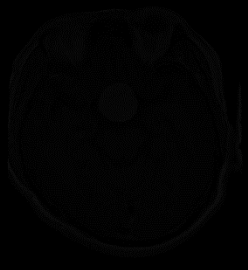
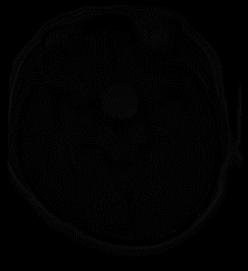
IMAGEN DE TUMOR CEREBRAL	TRANSFORMADA WAVELET
 512X512	 256X256
 512X512	 256X256
 512X512	 256X256
 512X512	 256X256

Tabla 6 Imágenes de tumor cerebral pituitary y sus transformadas de wavelet

4.5.4 Transformadas wavelet con los tres tipos de tumor cerebral.

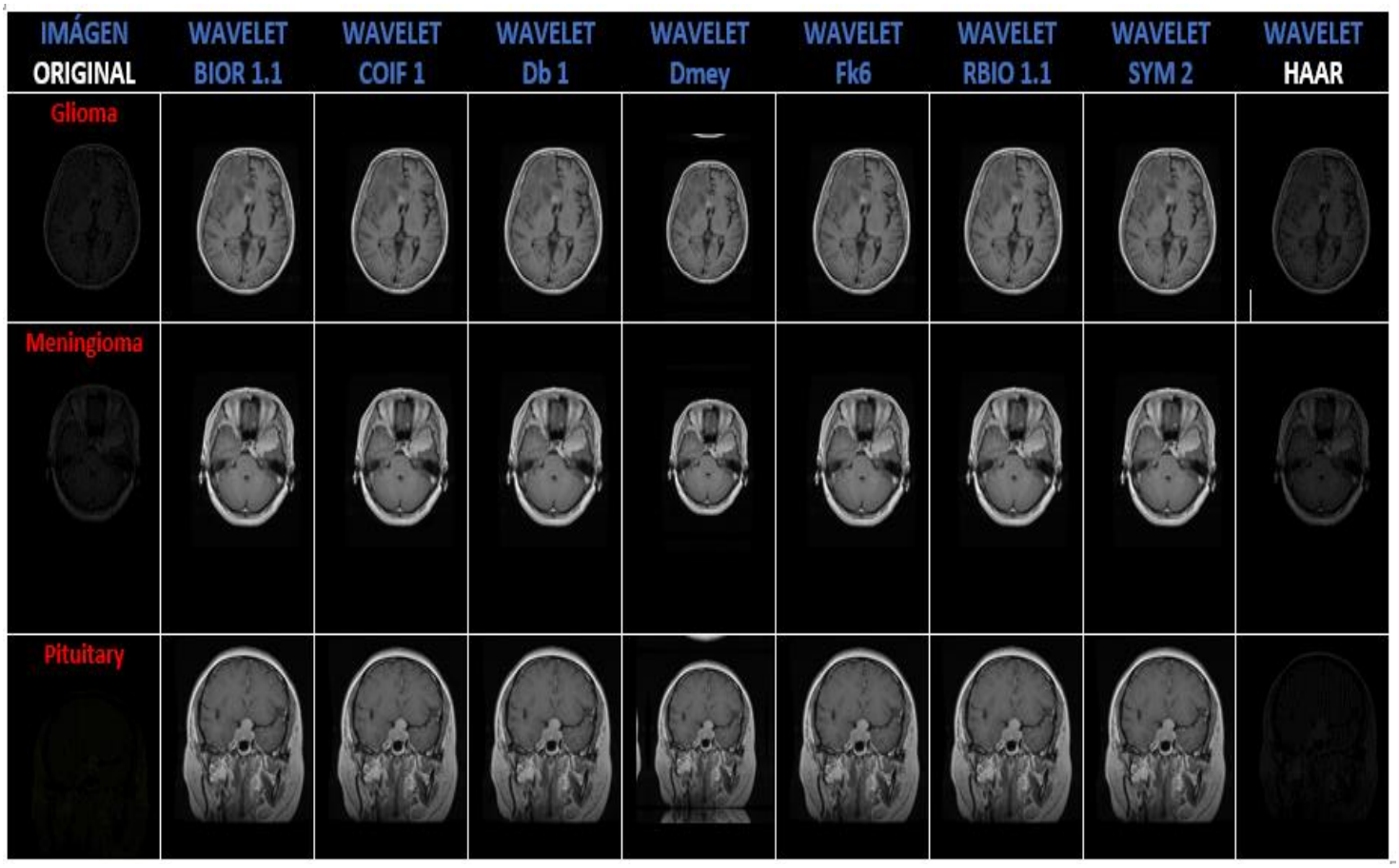


Tabla 7 Imágenes de tumor cerebral Glioma, Meningioma, Pituitary Con Transformadas wavelet

4.6 ZONA DEL TUMOR CEREBRAL

Se muestran a continuación las máscaras de la región del tumor cerebral, y se dibuja un rectángulo de color morado en la ubicación del tumor en el cerebro.

4.6.1 Tumor Glioma con ubicación de la zona del tumor cerebral

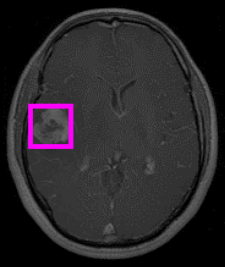
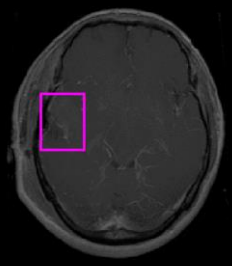
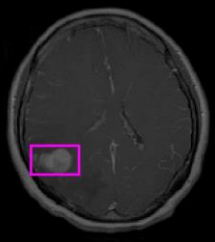
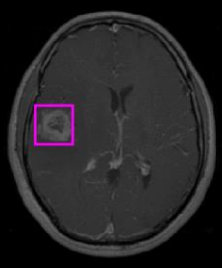
MÁSCARA DEL TUMOR	ZONA DEL TUMOR CEREBRAL
 256X256	 256X256
 256X256	 256X256
 256X256	 256X256
 256X256	 256X256

Tabla 8 Máscaras de tumores glioma y zonas donde se encuentran los tumores en el cerebro

4.6.2 Tumor Meningioma con ubicación de la zona del tumor cerebral

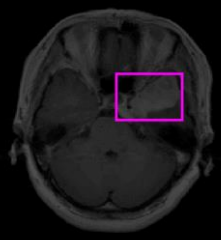
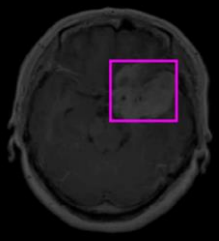
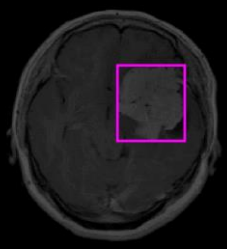
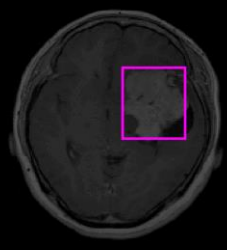
MÁSCARA DEL TUMOR	ZONA DEL TUMOR CEREBRAL
 256X256	 256X256
 256X256	 256X256
 256X256	 256X256
 256X256	 256X256

Tabla 9 Máscaras de tumores meningioma y zonas donde se encuentran los tumores en el cerebro

4.6.3 Tumor Pituitary con ubicación de la zona del tumor cerebral

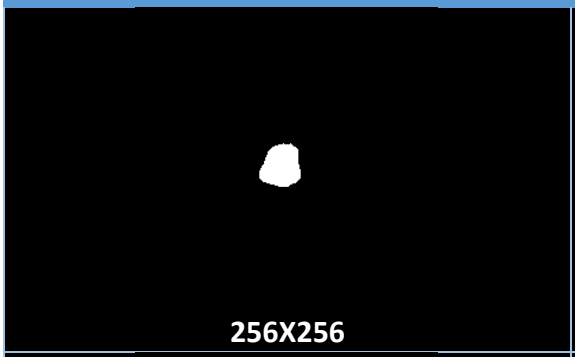
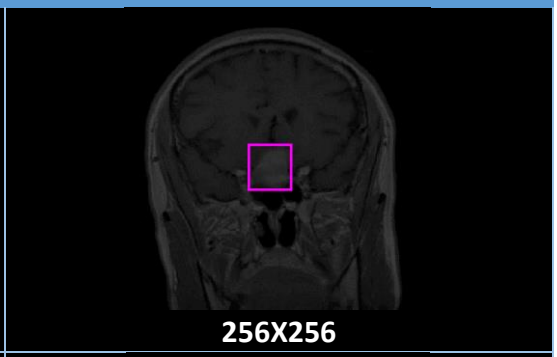
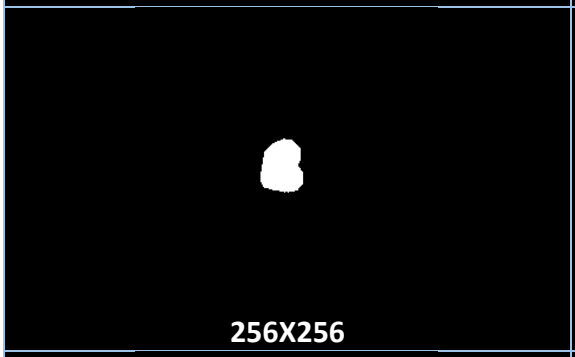
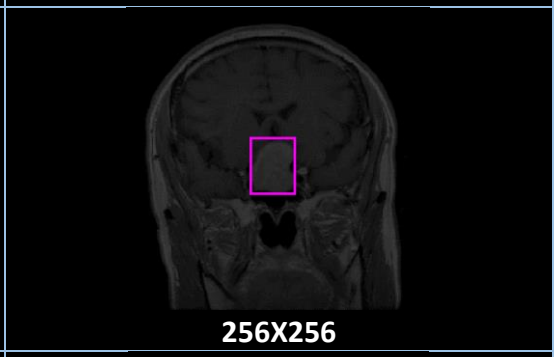
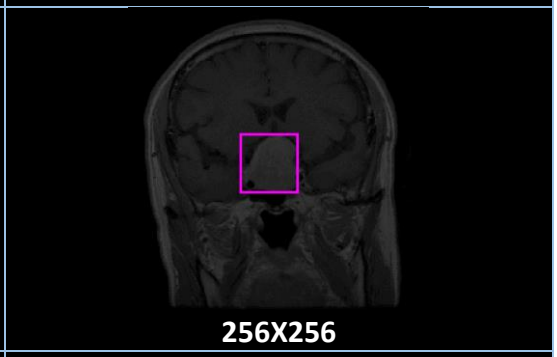
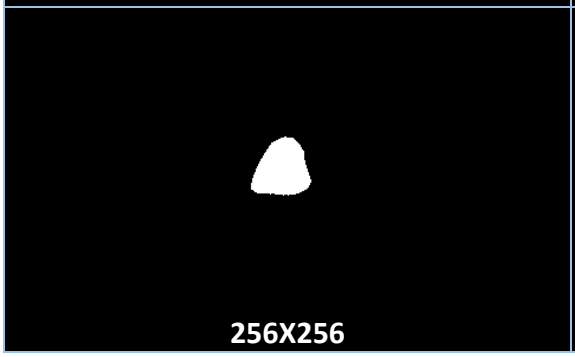
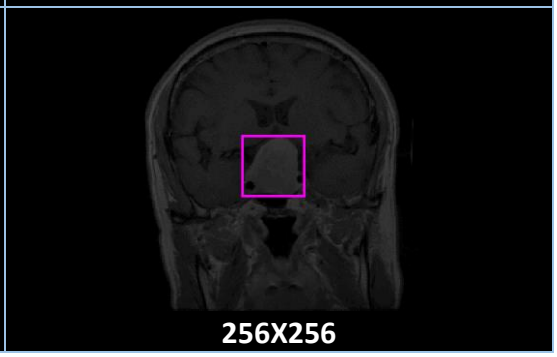
MÁSCARA DEL TUMOR	ZONA DEL TUMOR CEREBRAL
 256X256	 256X256
 256X256	 256X256
 256X256	 256X256
 256X256	 256X256

Tabla 10 Máscaras de tumores pituitary y zonas donde se encuentran los tumores en el cerebro

4.7 REGIÓN DEL TUMOR CEREBRAL

Se muestran a continuación las máscaras de la región del tumor cerebral, y se dibuja la región donde se aloja el tumor cerebral.

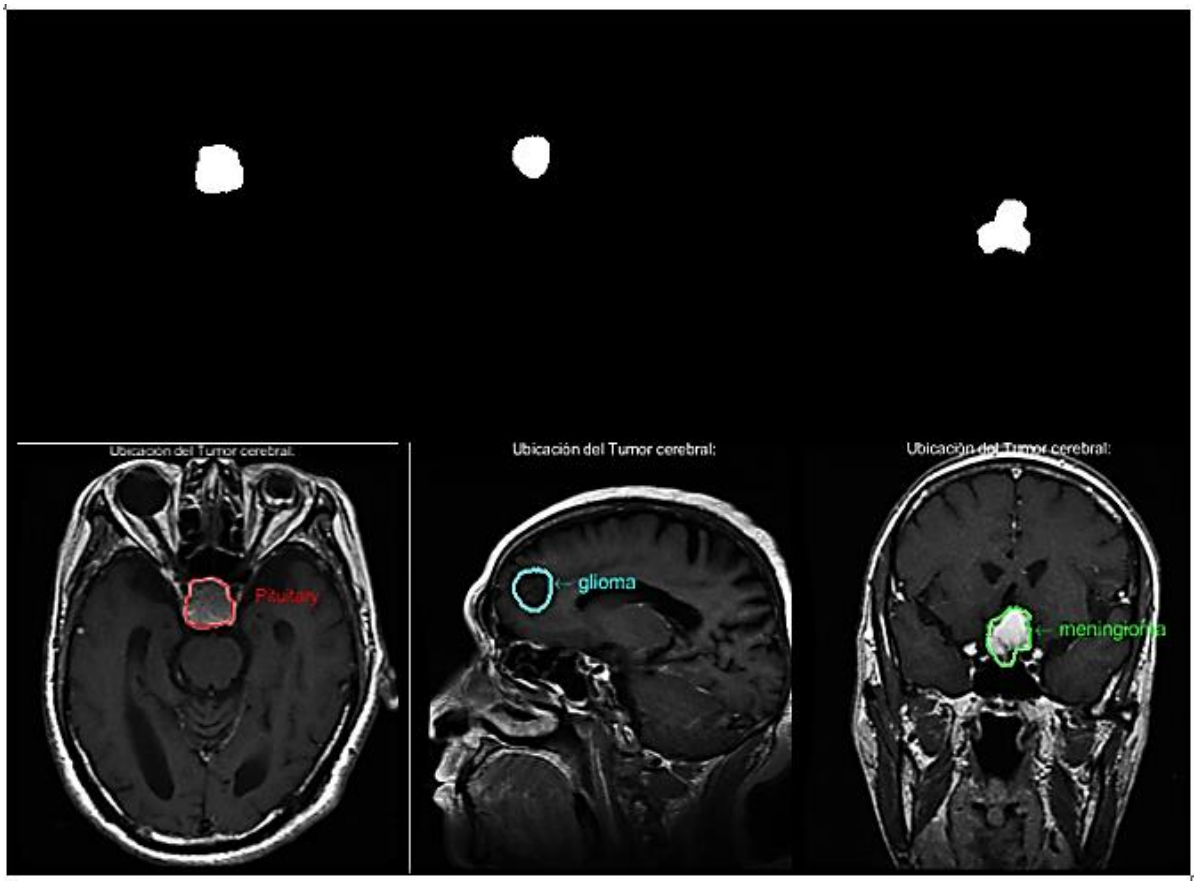


Tabla 11 máscara de la zona del tumor cerebral y Región demarcada donde se aloja el tumor cerebral.

4.8 IMAGEN ALMACENADA EN FIREBASE

En la figura 76 se aprecia el almacenamiento de imágenes en la base de datos en tiempo real, estas imágenes se pueden adquirir desde la raspberry pi4 para la evaluación con el modelo implementado de red neuronal convolucional para predecir una nueva imagen.

figura 76 Almacenamiento de imágenes en Firebase subidas desde Raspberry pi4.

En la figura 77 se aprecia el almacenamiento de imágenes en la base de datos en tiempo real, estas imágenes se pueden adquirir desde la raspberry pi3 para la evaluación con el modelo implementado de red neuronal convolucional para predecir una nueva imagen.

figura 77 Almacenamiento de imágenes en Firebase subidas desde Raspberry pi3.

4.9 INTERFAZ GRÁFICA CREADA EN PYTHON

En la figura 78 se muestra la interfaz gráfica dinámica con botones de acceso a internet, centros especializados en tumores cerebrales como clínicas, hospitales, y un botón para acceder a la página principal de la universidad, programada en Python con el paquete de creación de interfaz gráfica de Tkinter, luego de escrita la programación con el módulo de PyInstaller se crea el ejecutable para iniciarse desde la raspberry pi4, sin necesidad de que el médico ejecute el código desde Python sino inicie el ejecutable, es de anotar que el ejecutable puede ser distribuido desde Windows, Linux, mac y está pensado seguirse trabajando para distribución en aplicaciones móviles con sistema operativo android.

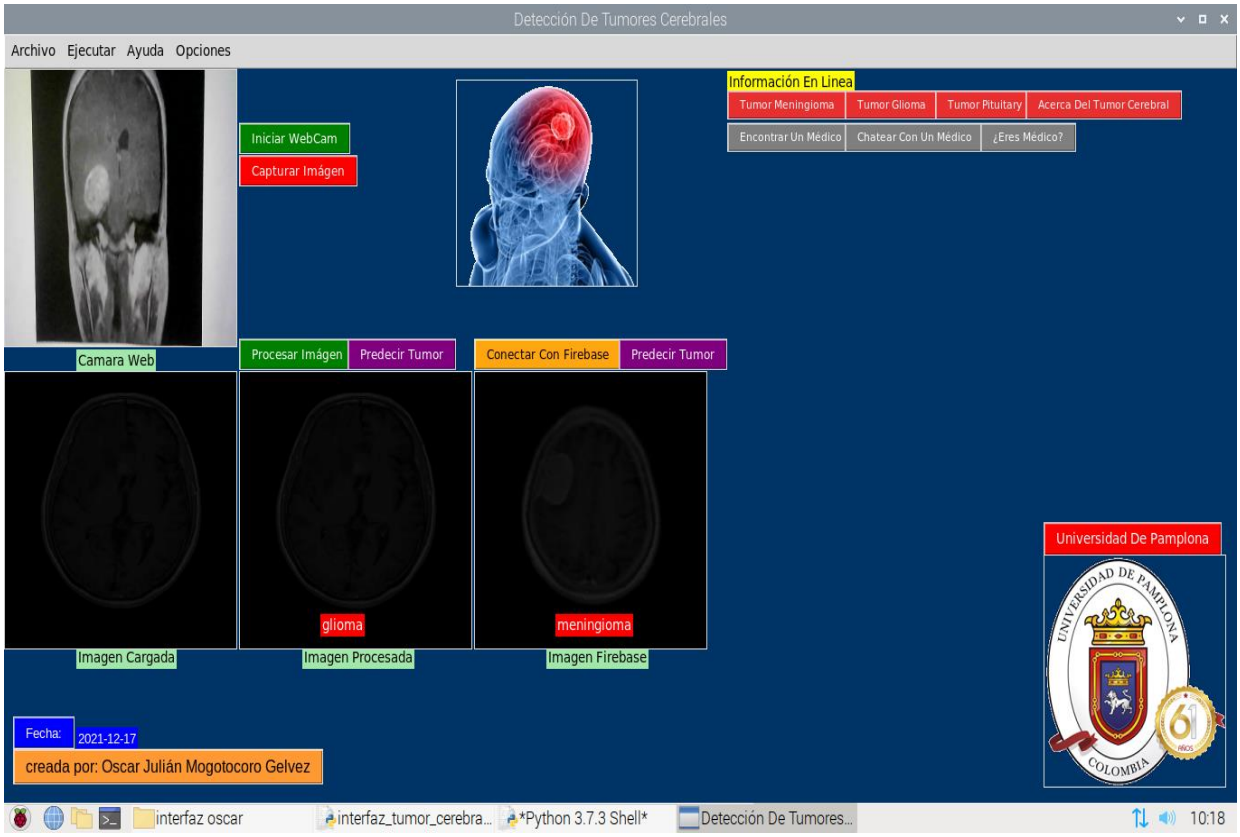


figura 78 interfaz gráfica dinámica distribuida con opción de predecir nuevas imágenes con modelo de red neuronal convolucional.

En la figura 79 se muestra la interfaz gráfica publicada y ejecutada desde Raspberry pi 4 en Linux dinámica distribuida con botones de acceso a internet, centros especializados en tumores cerebrales como clínicas, hospitales, y un botón para acceder a la página principal de la universidad, programada en Python con el paquete de creación de interfaz gráfica de Tkinter,

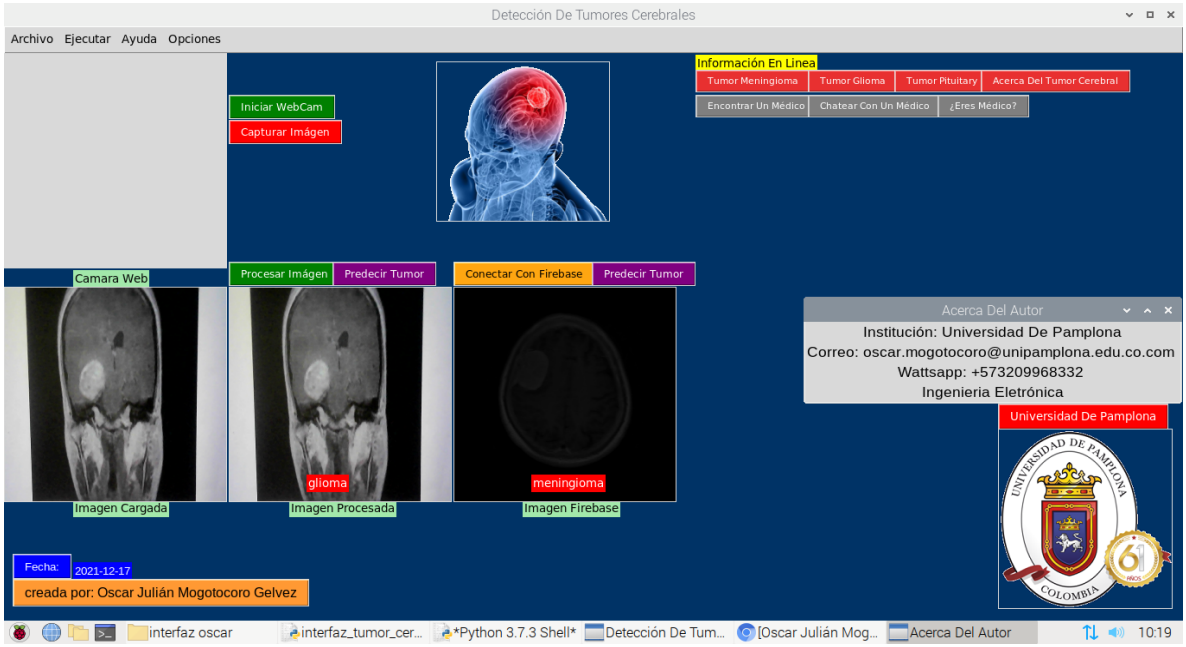


figura 79 interfaz gráfica dinámica publicada con opción de predecir nuevas imágenes con modelo de red neuronal convolucional.

5. CONCLUSIONES

En la literatura se encontró la usabilidad de algoritmos de inteligencia artificial tales como clasificadores bayesianos, máquinas de aprendizaje profundo, máquinas de soporte vectorial, transferencia de aprendizaje para la clasificación de tumores cerebrales, según la OMS un diagnóstico de tumor cerebral certero y efectivo prolonga la vida del paciente es por ello que cada vez más la medicina se acompaña de procesos inteligentes respaldados por uso de tecnologías modernas, un diagnóstico temprano preserva el sistema nervioso del paciente, y prolongan su esperanza de vida, además se encontró en la literatura que el tipo de tumor más frecuente en un 50% de casos diagnosticados es el tumor cerebral de tipo metástasis.

La aplicación de la transformada discreta de wavelet fue una técnica que mejoró la clasificación de los tumores cerebrales en un 98,3% gracias a la red neuronal convolucional creada, además se apreció como esta técnica matemática resalta las zonas de interés, y deja pasar a través del banco de filtros los coeficientes de aproximación, Wavelet en la actualidad es muy utilizada para el filtrado de imágenes. La ondita madre tipo Haar fue la que mejores resultados arrojó al momento del entrenamiento de la red, aunque no es donde mejor se alcanzan a apreciar las zonas de un posible tumor Cerebral, en cambio las imágenes a las que se les aplicó la wavelet Daubechies (Db1) con complemento y reducción atmosférica son las que mejor se aprecian las regiones donde se alojan los tumores cerebrales, meningioma, glioma y pituitary

Se implementó un sistema de inteligencia artificial basado en redes neuronales convolucionales, para clasificar tres tipos de tumor cerebral en la raspberry pi4, este sistema presentó un tiempo de predicción de tumor cerebral con nuevas imágenes menor a 10 segundos una alternativa a equipos de cómputo tradicionales, y estuvo fundamentado en el uso de máquinas virtuales para entrenamiento de la red neuronal, y la usabilidad en un sistema portátil de bajo costo sin necesidad de requerir equipos de cómputo muy robustos, y muy caros. En los países en vía de desarrollo este sistema presenta una alternativa ya que aliviana los costos de adquisición de equipos médicos.

Se diseñó una interfaz gráfica dinámica con botones de acceso a información acerca del tumor cerebral, donde se pueden cargar nuevas imágenes desde el directorio raíz del sistema operativo basado en Raspbian en el sistema embebido, imágenes capturadas por la cámara, e imágenes gestionadas desde Firebase que fueron cargadas desde una raspberry pi3, se escribió la programación en el software Python acudiendo a la librería de Tkinter que permitió la gestión de un servicio de interfaz gráfica con la posibilidad de comparar nuevas imágenes desde la interfaz con el modelo de red neuronal convolucional entrenado y exportado en la raspberry pi4.

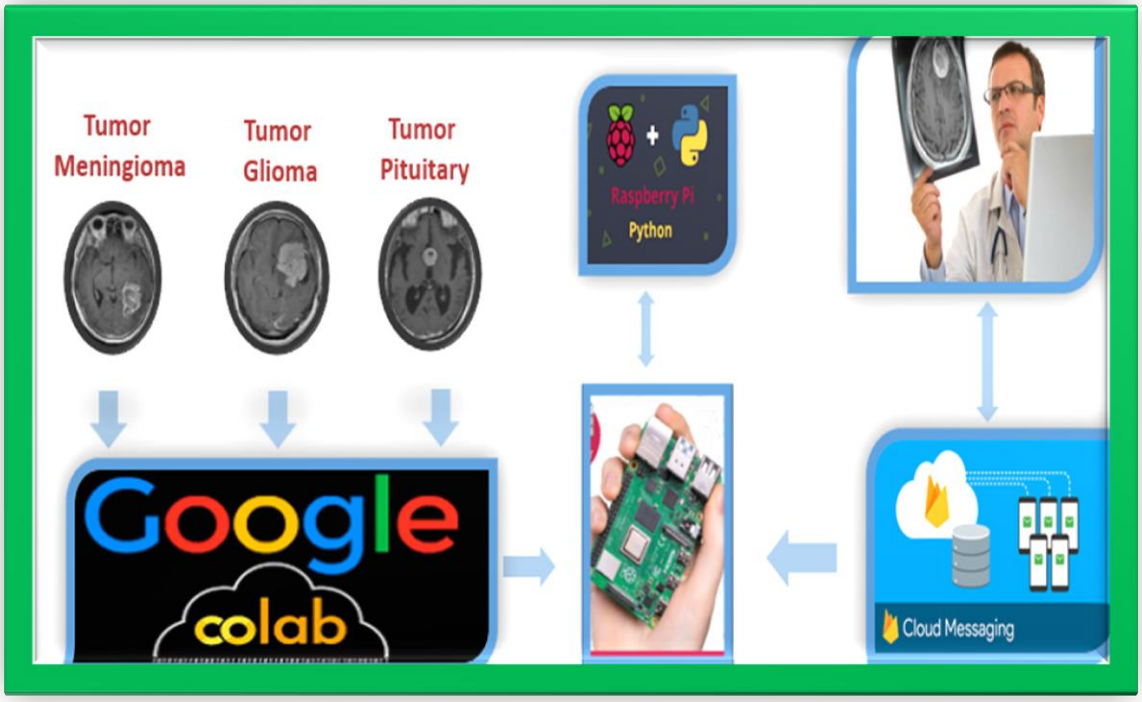
El aumento de datos en la red neuronal presentó una ventaja debido a que la red neuronal establece múltiples conexiones, y aprende varios caminos para diferenciar entre un tipo de tumor u otro. El sobreajuste fue controlado gracias a la determinación de un tamaño de lote y alivianó los procesos de ejecución en la memoria RAM, además optimizó el uso de la tarjeta gráfica proporcionada por Google GPU en el entrenamiento.

Se gestionó la conectividad con servicios de almacenamiento en la nube, en la actualidad el internet de las cosas permite la interacción y control desde aplicaciones móviles, y la conectividad en todo el mundo en pocos segundos.

6. TRABAJOS FUTUROS

Al lograrse la gestión de bases de datos en tiempo real con Firebase y raspberry se puede pensar en aplicaciones de telemedicina tales como la operación a distancia, el diagnóstico de enfermedades, aplicaciones en control como la adquisición de variables físicas en entornos donde el ser humano no puede acceder, estaciones Metereológicas con la posibilidad de adquirir señales desde cualquier parte del mundo y lograr la sintonización de plantas, la simulación de modelos, el modelado de reactores nucleares, control de equipos autónomos no tripulados, equipos para desactivación de explosivos, aplicaciones de domótica, la monitorización de cultivos en tiempo real, ofrecer diagnósticos médicos sin necesidad de que la persona asista a hospitales.

Un sistema de inteligencia artificial que asista el diagnostico de tumores cerebrales debe contar con todos los tipos de tumor cerebral, en trabajos futuros se piensa en utilizar más imágenes de resonancia magnética con todos los tipos de tumor cerebral y con una estructura de imágenes en 3D.



7. REFERENCIAS

[1]. Badža, M.M.; Barjaktarović, M.Č. Classification of Brain Tumors from MRI Images Using a Convolutional Neural Network. Appl. Sci. 2020, 10, 1999. https://doi.org/10.3390/app10061999.

[2]. Organización Mundial de la Salud — Cáncer. [en línea]. https://www.who.int/news-room/fact-sheets/detail/cancer. (fecha de Consulta 5 de noviembre de 2019).

[3]. Priya, VV Un enfoque de segmentación eficiente para la detección de tumores cerebrales en resonancia magnética. Indian J. Sci. Technol. 2016, 9, 1–6. [Google Académico].

[4]. Benet, M, Resonancia magnética de tumores cerebrales para la caracterización y clasificación de distintas regiones de interés: [en línea]. https://riunet.upv.es/bitstream/handle/10251/67519/35593872_TFG_1467737576566518063880143030119 2.pdf?sequence=2&isAllowed=y (fecha de consulta el 12 de noviembre de 2020).

[5]. Afshar, P. Plataniotis, KN. Mohammadi, A. Redes de cápsulas para la clasificación de tumores cerebrales basada en imágenes de resonancia magnética y límites de tumores gruesos. En Actas de la Conferencia Internacional de IEEE sobre Acústica, Habla y Procesamiento de Señales (ICASSP) de ICASSP 2019-2019, Brighton, Reino Unido, 12-17 de mayo de 2019; págs. 1368-1372. [Google Académico].

[6]. Portable Ultrasound Enables Anytime, Anywhere Imaging. Available online: **https://healthtechmagazine.net/article/2018/07/portable-ultrasound-enables-anytime anywhere-imaging** (accessed on 3 February 2020).

[7]. Gil, C. M. (s.f.). Mapfre salud. Pruebas diagnósticas por imágenes: [en línea]. http://www.mapfre.es/salud/es/cinformativo/resonanciasmagneticas (fecha de consulta el 12 de noviembre de 2020).

[8]. https://www.mayoclinic.org/es-es/diseases-conditions/glioma/symptoms-causes/syc-20350251 (fecha de consulta el 12 de noviembre de 2021).

[9]. Domínguez Torres, Alejandro Procesamiento digital de imágenes. Perfiles Educativos [en línea]. 1996, (72), [fecha de Consulta 15 de marzo de 2021]. ISSN: 0185-2698. Disponible en: https://www.redalyc.org/articulo.oa?id=13207206 .

[10]. Hardy, Thomas (IA: Inteligencia Artificial). POLIS, Revista Latinoamericana [en línea]. 2001, 1(2), 0ISSN: 0717-6554. Disponible en: https://www.redalyc.org/articulo.oa?id=30500219. [fecha de Consulta 12 de marzo de 2021].

[11]. McCulloch, W. S., y Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics, 5, 115-133.

[12]. Raspberry Pi, [en línea]. Available: https://www.raspberrypi.org/education/. [acceso:10/03/2021].

[13]. Sajjad, M. Khan, S. Muhammad, K. Wu, W. Ullah, A. Baik, SW Clasificación de tumores cerebrales multigrado utilizando CNN profundo con amplio aumento de datos. J. Comput. Sci. 2019, 30, 174–182. [Google Académico] [CrossRef].

[14] M. Sinning, "Clasificación De Los Tumores Cerebrales," *Rev. Médica Clínica Las Condes*, vol. 28, no. 3, pp. 339–342, 2017, doi: 10.1016/j.rmclc.2017.05.002.

[15] H. Mohsen, E.-S. A. El-Dahshan, E.-S. M. El-Horbaty, and A.-B. M. Salem, "Classification using deep learning neural networks for brain tumors," *Futur. Comput. Informatics J.*, vol. 3, no. 1, pp. 68–71, 2018, doi: 10.1016/j.fcij.2017.12.001.

[16] A. Çinar and M. Yildirim, "Detection of tumors on brain MRI images using the hybrid convolutional neural network architecture," *Med. Hypotheses*, vol. 139, no. February, p. 109684, 2020, doi: 10.1016/j.mehy.2020.109684.

[17] "American Cancer Society." https://www.cancer.org/es/cancer/tumores-de-encefalo-o-de-medula-espinal/deteccion-diagnostico-clasificacion-por-etapas/como-se-diagnostica.html (accessed Oct. 19, 2021).

[18] "National Institutes Of Health." https://salud.nih.gov/articulo/detectar-tumores-cerebrales/(accessed Oct. 19, 2021).

[19] "biblioteca nacional de medicina de los Estados Unidos." https://medlineplus.gov/spanish/braintumors.html (accessed Oct. 19, 2021).

[20] D. Torres, "Procesamiento digital de imágenes," Perfiles Educ., no. 72, 1996.

[21] Haar A. Zur Theory orthogonal Function Systems', Mathematische Annalen, 69, pp 331–371, 1910.

[22] "mathwords página oficial de matlab." https://la.mathworks.com/products/matlab.html (accessed Nov. 11, 2021).

[23] "Aprende Machine Learning Teoria + Practica" disponible en: https://leanpub.com/aprendeml/

[24] “Información de Google colab disponible en su página oficial”: https://colab.research.google.com/notebooks/intro.ipynb

[25] “Información sobre la GPU disponible en la página oficial de nvidia”: https://www.nvidia.com/es-la/drivers/what-is-gpu-computing/

[26] “Definición de Python disponible en su página oficial”: https://docs.python.org/3/tutorial/index.html

[27] “Información sobre opencv disponible en su página oficial”: https://opencv.org/about/

[28] “Información de TensorFlow disponible en su página oficial”: https://www.tensorflow.org/about?hl=es-419

[29] “Información de Keras disponible en su página oficial”: https://keras.io/about/

[30] “interfaces gráficas de usuario con tk.” https://docs.python.org/es/3/library/tk.html (accessed Nov. 11, 2021).

[31] “Publicación y distribución de ejecutables con pyinstaller en python.” https://www.pyinstaller.org/ (accessed Nov. 12, 2021).

[32] “Rohde & Schwarz España, S.A.” https://www.rohde-schwarz.com/es/productos/test-y-medida/osciloscopios/educational-content/que-es-uart_254524.html (accessed nov. 25, 2021).