TRABAJO FIN DE GRADO

APLICACIÓN DE REDES NEURONALES AL DIAGNÓSTICO, A PARTIR DE IMÁGENES, DE ENFERMEDADES NEURODEGENERATIVAS

TRABAJO FIN DE GRADO
PARA LA OBTENCIÓN DEL
TÍTULO DE **GRADUADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES**

JULIO **2022**

Álvaro Molina Calleja

TUTOR DEL TRABAJO FIN DE GRADO:

José Manuel Mira McWilliams

INDUSTRIALES
ETSII | UPM

POLITÉCNICA

AGRADECIMIENTOS

Me gustaría agradecer aquí a todas aquellas personas que, en menor o mayor medida, han ayudado a sacar adelante este trabajo durante los últimos meses.

En primer lugar, dar las gracias a José por acceder a ser mi tutor académico y haber depositado su confianza en mí durante estos meses para llevar a cabo este trabajo de fin de grado. Destacar la especial atención que ha tenido conmigo cuando más la necesitaba y su particular aporte de motivación para querer ir más allá con el tema a tratar.

Este trabajo no lo hubiera llevado a cabo sin ti.

A mi familia y amigos, por haberme dado los ánimos necesarios en los momentos difíciles y por vuestra compañía cuando la vida mostraba su lado más amargo. Sin vuestras palabras de apoyo y cariño, este trabajo no lo habría terminado.

CITA

La inteligencia no es una habilidad en sí misma, no es lo que puedes hacer, es cuan bien y cuan eficientemente puedes aprender cosas nuevas.

François Chollet

1. RESUMEN EJECUTIVO

Los tumores cerebrales son una masa de células anormales que se concentra en el cerebro. Existen tumores malignos y benignos. Según en donde se formen distinguimos primarios, originados en el cerebro, y secundarios, originados en otra parte del cuerpo, pero se extiende hasta el cerebro. Esta enfermedad representa el 85% de todos los tumores primarios del sistema nervioso central. Alrededor de 300000 personas fueron diagnosticadas con un tumor cerebral primario en 2020.

En la actualidad, la detección manual de tumores cerebrales en pacientes requiere de especialistas cualificados con cierta experiencia en el campo de la radiología. No todos los países del mundo cuentan con estos expertos, es más, la mayoría de países en desarrollo carecen de recursos para detectar tumores a través de un médico. Es por ello, que, en los años recientes, se han aprovechado las ventajas que proporcionan la Inteligencia Artificial y el Machine Learning para resolver problemas de este tipo en el sector médico. Una de estas soluciones consiste en la creación de redes neuronales convolucionales que consigan auto aprender patrones y algoritmos para detectar y clasificar de forma automática tumores cerebrales. Además, el punto diferencial de esta solución es que la red creada, se alimenta con una base de datos basada en imágenes de resonancia magnética (IRM) de pacientes de todo el mundo.

Hoy por hoy, existen modelos que realizan este tipo de clasificaciones con redes neuronales, no obstante, la red creada en este trabajo irá más lejos, consiguiendo clasificar las imágenes con tumor entre 3 clases distintas: glioma, meningioma y pituitario. Asimismo, existen diversos tipos de redes neuronales que operan con arquitecturas muy diferentes, como las redes neuronales recurrentes, no recurrentes, monocapa, multicapa, entre otras. Las más interesantes de estudiar y en las que se centrará este trabajo son las redes neuronales convolucionales. Este tipo de redes también llamadas CNN, se basan en el aprendizaje de patrones que la permitan autocorregir los errores cometidos en simulaciones anteriores con el objetivo de aumentar la precisión en la clasificación del tumor. Estas redes basan su estructura en una capa fundamental conocida como convolucional. La capa convolucional actúa como filtro de la información que pasa por ella, en este caso las imágenes procedentes de las resonancias magnéticas. A medida que la información va pasando por las capas que forman la red, esta va reteniendo características claves como formas, texturas, líneas, entre otras, para posteriormente, detectar y clasificar al tumor.

Para llevar a cabo la construcción de la red utilizada, se hará uso del lenguaje de programación de R, y RStudio como interfaz de trabajo. De entre las razones por las que se ha escogido R como lenguaje y no otros como Python, destacan su facilidad para visualizar y manipular los datos alimentados a la red, y su especialización en la estadística que nos permitirá analizar los resultados obtenidos tras la simulación de la red, de una manera más sencilla y profunda. Aun así, el lenguaje de Python suele ser la herramienta fundamental utilizada en programación de redes como la de este trabajo.

Las imágenes procesadas a través la red neuronal, previamente deberán ser preprocesadas con el fin de facilitar el aprendizaje que realiza la red y para obtener unos resultados finales más coherentes. Para ello, normalmente, se suelen aplicar modificaciones a las imágenes originales, como pueden ser rotaciones, volteos o zooms, creando nuevos datos artificiales que se alimentan a la red posteriormente. De esta forma, la red podrá trabajar con mayor cantidad de información y, por tanto, obtener mayor cantidad de características con las que clasificar de manera más precisa a las imágenes. Este proceso de aumento del conjunto de datos se conoce como data augmentation.

En un principio, y tras preparar las imágenes que se usaran en la red, se modela una red neuronal convolucional inicial con pocas capas convolucionales con el objetivo de entrenar a la red. Posteriormente, se ajusta la red, añadiendo muchas más capas convolucionales, con el fin de realizar una simulación final más real con los datos de prueba. De esta manera, la red, al llegar a la simulación final, ya conocerá todas las características necesarias para clasificar a los distintos tipos de tumores, ya que ha sido entrenada previamente.

Una vez se simula el modelo final, se analizan los resultados obtenidos y se encuentra que el modelo tiene dificultad para clasificar entre varias clases de tumores distintos, como puede ser entre gliomas y meningiomas. En otros casos, como es el de los pacientes sin tumor, el porcentaje de acierto en la clasificación es muy alto, de alrededor de 92%. Lo cual demuestra que el modelo es muy eficiente para detectar tumores, sin embargo, podría mejorar en la clasificación, una vez se detecta el tumor.

Finalmente, se comenta la necesidad de mejorar el modelo creado, aumentando su precisión para poder así en un futuro a corto plazo implementarlo a nivel médico en hospitales y clínicas medicas de todo el mundo. El objetivo sería comenzar con países en desarrollo, ya que al final son los que más lo necesitan por su falta de experiencia en la detección y clasificación de tumores cerebrales, y en general en la radiología. Posteriormente, en un futuro más lejano, se podría asentar la solución planteada en el resto de países que, aunque actualmente no la necesiten, proporcionarían mayor eficiencia y automatización en el proceso de detección de este tipo de enfermedades.

Palabras clave

Tumor cerebral, red neuronal convolucional, imágenes IRM, Deep Learning, países en desarrollo, detección y clasificación, simulación, autoaprendizaje.

Códigos UNESCO

1203.04 Inteligencia Artificial, 1203.20 Sistemas de Control Médico, 1203.23 Lenguajes de Programación, 1209.14 Técnicas de Predicción Estadística, 3205.07 Neurología, 3207.13 Oncología

ABSTRACT

Brain tumors are a mass of abnormal cells that are concentrated in the brain. There are malignant and benign tumors. According to where they are formed, we distinguish primary, originating in the brain, and secondary, originating in another part of the body but they extend to the brain. This disease represents 85% of all primary tumors of the central nervous system. Around 300,000 people were diagnosed with a primary brain tumor in 2020.

Currently, the manual detection of brain tumors in patients requires qualified specialists with experience in the field of radiology. Not all countries in the world have these experts, moreover, most developing countries lack the resources to detect tumors manually. That is why, in recent years, the advantages provided by Artificial Intelligence and Machine Learning have been used to solve problems of this type in the medical sector. One of these solutions consists of the creation of convolutional neural networks that manage to self-learn patterns and algorithms to automatically detect and classify brain tumors. In addition, the differential point of this solution is that the network created is fed with a database based on magnetic resonance imaging (MRI) of patients from all over the world.

Today, there are models that carry out this type of classification with neural networks, however, the network created in this project will go further, managing to classify images with tumors into 3 different classes: glioma, meningioma and pituitary. Likewise, there are various types of neural networks that operate with very different architectures, such as recurrent networks, non-recurrent networks, monolayer networks, multilayer networks, among others. The most interesting to study and on which this project will focus are convolutional neural networks. This type of network called CNN or convolutional neural networks is based on learning patterns that allow it to self-correct errors made in previous simulations with the aim of increasing accuracy in tumor classification. These networks base their structure on a fundamental layer known as convolutional. The convolutional layer acts as a filter for the information that passes through it, in this case the images from magnetic resonance imaging. As the information passes through the layers, it retains key characteristics such as shapes, textures, lines, among others, to subsequently detect and classify the tumor.

To carry out the construction of the network used, we will use R as our programming language, and RStudio as our work interface. Among the reasons why R has been chosen as our language and not others, such as Python, are its ease in visualizing and manipulating the data fed to the network, and its specialization in statistics that will allow us to analyze the results obtained after the simulation of the network, in a simpler and deeper way.

The images processed through the neural network must previously be preprocessed in order to facilitate the learning carried out by the network and to obtain more consistent final results. To do this, modifications are usually applied to the original images, such as rotations, flips or zooms, creating new artificial data that is subsequently fed to the network. In this way, the network will be able to work with a greater amount of information and, therefore, obtain a greater number of characteristics with which to classify the images more precisely. This process of augmenting the data set is known as data augmentation.

Initially, and after preparing the images to be used in the network, an initial convolutional neural network with few convolutional layers is modeled in order to train the network. Subsequently, the network is adjusted, adding many more convolutional layers, in order to perform a more realistic final simulation with the test data. In this way, when the network reaches the final simulation, it will already know all the characteristics necessary to classify the different types of tumors, since it has been previously trained.

Once the final model is simulated, the results obtained are analyzed and it is found that the model has difficulty classifying between several different classes of tumors, such as gliomas and meningiomas. In other cases, such as patients without tumors, the percentage of success in the classification is very high, around 92%. This shows that the model is very efficient in detecting tumors, however, it could improve the classification once the tumor is detected.

Finally, we discuss the need in improving the model used, increasing its precision in order to be able to implement it at a medical level in hospitals and medical clinics around the world in the short term. The objective would be to start with developing countries, since in the end they are the ones that need it most, due to their lack of experience in the detection and classification of brain tumors, and in radiology in general. Later, in a more distant future, the proposed solution could be established in the rest of the countries that, although they do not currently need it, would provide greater efficiency and automation in the detection process of this type of disease.

ÍNDICE

2. INTRODUCCIÓN

Los tumores cerebrales son una de las enfermedades más agresivas que pueden afectar tanto a adultos como a niños. Se forman debido a la agrupación de células anormales en torno a los tejidos del cerebro o medula espinal. Como se verá más adelante, no todos los tumores cerebrales son igual de dañinos, sino que existen tumores benignos y malignos. Asimismo, existen tumores cerebrales que no se originan como tales en el cerebro, sino que provienen de otras partes del cuerpo en forma de metástasis. Este tipo de tumores son conocidos como secundarios, mientras que los que se originan directamente en el cerebro se llaman primarios.

Actualmente, alrededor de 300000 tumores cerebral primarios se diagnostican en pacientes al año en todo el mundo, en España aproximadamente 3000. Aproximadamente, el 85% de los tumores del sistema nervioso se asocian a tumores cerebrales o de medula espinal. La edad del paciente es un factor fundamental a la hora de evaluar la tasa de supervivencia a uno de estos tumores. En niños de entre 0 y 5 años, los tumores cerebrales se encuentran entre las dos primeras causas de mortalidad [1].

Un tratamiento en condiciones y un diagnóstico preciso son imprescindibles para mejorar la esperanza de vida de un paciente con tumor cerebral. La principal técnica de diagnóstico desarrollada actualmente en estos pacientes es la resonancia magnética. España se encuentra entre las 10 primeras posiciones en el ranking europeo en resonancias magnéticas por habitante (1.3 máquinas por 100000 habitantes). Este tipo de máquinas genera una gran cantidad de imágenes, sin embargo, antes de diagnosticar un tumor, estas deben ser examinadas por radiólogos especializados [2].

Muchos países en desarrollo carecen de radiólogos capacitados para llevar a cabo el diagnostico de tumores cerebrales a partir de las imágenes obtenidas en los exámenes de resonancia magnética, y muchas veces se pueden dar errores en la clasificación o incluso en la detección de tumores cerebrales en pacientes.

Por ello, con este Trabajo de Fin de Grado se pretende aplicar una técnica de Deep Learning basada en la programación y uso de redes neuronales convolucionales para ayudar a este tipo de países a establecer diagnósticos de tumores cerebrales en pacientes de manera automática y precisa, sin tener que depender de doctores poco especializados y sin experiencia alguna en el campo de la radiología. No solo se busca ayudar a estos países, sino también desarrollar las técnicas de diagnóstico de este tipo de enfermedades en general. De esta forma, se mejora la precisión en la detección y clasificación de un tumor. Además, se reduce mucho el tiempo de examen en imágenes de resonancia magnética. Para lograr lo anterior, se va a utilizar una base de datos de imágenes de resonancias magnéticas del cerebro de pacientes de todo el mundo.

A lo largo de este trabajo, se empezará presentando la enfermedad, su diagnóstico y su situación actual en el mundo. Mas adelante, se desarrollará la teoría del Machine Learning y Deep Learning, y su aplicabilidad en el sector médico. Además, se plasmarán las bases de las redes neuronales convolucionales, así como su funcionamiento aplicado a las imágenes de resonancia magnética que se introducirán en la red. Para continuar, se desarrollarán las metodologías usadas para programar dichas redes neuronales y monitorizarlas. Se plasmarán los resultados obtenidos tras simular la red y se presentarán unas conclusiones finales. Por último, se mencionarán las líneas futuras del proyecto y los beneficios que podría presentar la idea en un futuro, y se presentarán los puntos claves relacionados con la gestión y dirección del proyecto, como son la EDP, Diagrama de Gantt y un análisis del presupuesto.

3. OBJETIVOS

El objetivo principal de este trabajo, es la creación de una red neuronal convolucional (CNN) para detectar y clasificar distintos tipos de tumores cerebrales a partir de imágenes de resonancia magnética (IRM).

Para llevar un mejor seguimiento del trabajo expuesto, la siguiente lista representa el conjunto de subobjetivos a cumplir:

- Se adquirirán altos conocimientos sobre tumores cerebrales y sus diferentes clasificaciones, establecidas según sus propiedades, síntomas o causas. Además, el trabajo se centrará en 3 tipos diferentes de tumor cerebral: glioma, meningioma y pituitario, que serán los que la red neuronal convolucional se encargue de detectar y clasificar.

- Se plasmarán los diferentes tratamientos para cada tipo de tumor cerebral.

- Se detallarán las diferentes técnicas de diagnóstico de la enfermedad y como se podrán sustituir en un futuro por sistemas de redes neuronales convolucionales totalmente prácticos que mejorarán la eficiencia en la detección de tumores cerebrales respecto a la detección manual humana.

- Se plasmarán las nociones básicas del Machine Learning y las técnicas fundamentales de Deep Learning.

- Se adquirirán altos conocimientos en redes neuronales convolucionales. Se explicará el funcionamiento fundamental de cada una de las capas que forman la red.

- Se darán ventajas de las redes neuronales convolucionales frente al resto de redes neuronales existentes. Así como, se adquirirán conocimientos de porque las redes neuronales convolucionales son las más óptimas para la tarea de detección de tumores cerebrales a partir de imágenes IRM.

- Se obtendrán y estandarizarán los datos necesarios para alimentar a la red.

- Se adquirirá un nivel avanzado en programación en RStudio para desarrollar la arquitectura principal de la red neuronal, y básico en Python. El lenguaje de programación elegido para establecer las redes es RStudio ya que la mayoría de proyectos ya desarrollados de redes neuronales convolucionales están programados en Python.

- Se adquirirán las características fundamentales de las bibliotecas de redes neuronales de código abierto de Keras y Tensorflow, aplicadas al lenguaje de programación de RStudio mediante MiniConda.

- Se analizará en detalle el método escogido para construir y programar la red encargada de clasificar los tumores cerebrales.

- Se comentarán los resultados obtenidos tras simular la red con los datos usados.

- Se establecerán unas conclusiones sobre el futuro a corto plazo de las redes neuronales como sistemas para detectar y clasificar enfermedades neurodegenerativas.

4. LA ENFERMEDAD Y SU DIAGNÓSTICO

4.1. Tumores cerebrales

De forma atípica, el ADN de un ser humano muta y hace que las células en el cerebro se dupliquen de forma anormal consiguiendo que estas se dividan de forma rápida y vivan mucho más tiempo que una célula normal. Estas células anormales se agrupan en una masa que puede ocupar zonas diversas del cerebro. Dicha concentración de células anormales es lo que conocemos como tumor cerebral. Los tumores cerebrales pueden destruir las células sanas o no anormales del cerebro [3].

Los tumores cerebrales pueden ser benignos o malignos. Los benignos son aquellos que son no cancerígenos y los malignos aquellos que se expanden e invaden otros tejidos del cuerpo humano, o también conocidos como cancerígenos.

Asimismo, se pueden clasificar en función de donde se origine el tumor: primarios si se originan en el propio cerebro y secundarios o metastásicos si se originan en otra parte del cuerpo y se expanden hacia el cerebro (metástasis). Estos últimos son siempre malignos, mientras que los primarios pueden ser tanto malignos como benignos.

En la actualidad existe una gran variedad de tumores cerebrales, cada uno con sus causas, síntomas y tratamientos distintos. En este trabajo nos centraremos en detectar y clasificar 3 de ellos: glioma, meningioma y pituitario.

- Glioma: se trata de uno de los tumores cerebrales primarios más comunes en el ser humano (65% de ellos). Se puede desarrollar no solo en el cerebro, sino también en la medula. Surgen en el soporte viscoso, que rodea a las células nerviosas. Este tipo de tumor está formado por células gliales y según este tipo de células tenemos varios tumores distintos (astrocitoma, ependimoma, gliobascoma). Alguno de los síntomas más comunes del glioma son dolores de cabeza, disminución de la función cerebral, perdida de la memoria o problemas de visión. Los tratamientos que más se suelen aplicar para eliminar los gliomas son la cirugía, la radioterapia y la quimioterapia [3].

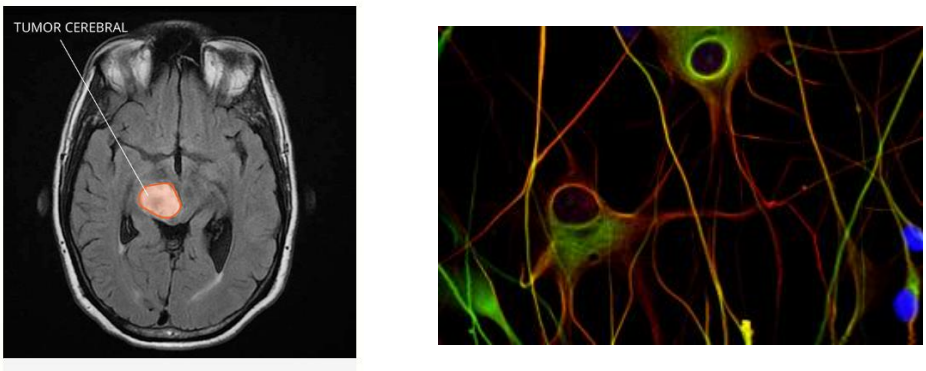

Figura 1: Glioma en una imagen IRM (izquierda) y células gliales (derecha)

- Meningioma: es un tumor cerebral primario que surge en las membranas que rodean al cerebro y la médula. Suele ser un tumor benigno que puede reaparecer tras su extirpación pasado un tiempo. Además, puede crecer muy lentamente sin presentar síntoma alguno en el paciente. Son característicos de las mujeres y normalmente en edad avanzada. Es un tumor cerebral que en algunos casos no requiere de

tratamiento alguno siempre y cuando se mantenga controlado. Dentro de los principales síntomas de estos tumores destacan los cambios de visión, dolores de cabeza o pérdida de memoria. Los tratamientos más comunes, en caso de que el meningioma se complique de más, son la cirugía y la radioterapia [3].

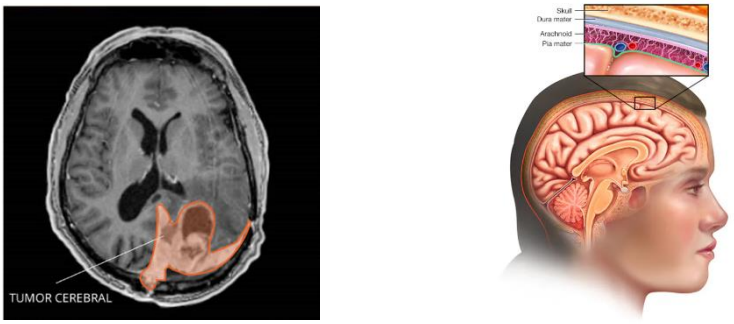

Figura 2: Meningioma en una imagen IRM (izquierda) y las meninges (derecha)

- Pituitario: son tumores cerebrales que se dan como consecuencia de un crecimiento anormal de células en la glándula pituitaria. Pueden dar lugar a un descenso o aumento, según el tipo de tumor, de las hormonas generadas por la glándula pituitaria y, por tanto, afectar a funciones importantes del cuerpo humano. Son tumores cerebrales generalmente benignos (adenomas) que, salvo en casos poco habituales, no invadirán otras partes del cuerpo. Es por esto, que son tumores cerebrales primarios, al generarse en el cerebro. En cuanto a los tratamientos más utilizados para combatir estos tumores destacan la extirpación y el uso de medicamentos para controlar el nivel de hormonas generadas. Los principales síntomas son la pérdida de visión y el dolor de cabeza. También destacar las náuseas, vómitos y debilidad en el cuerpo [3].

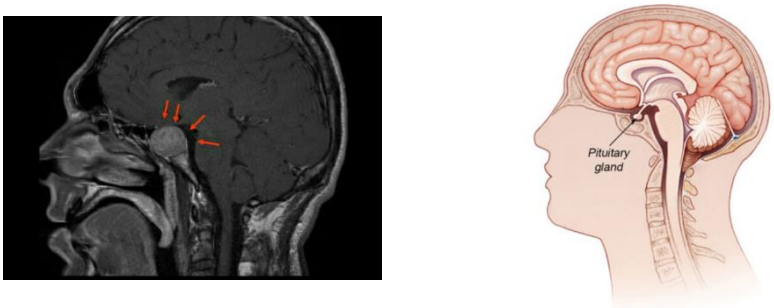

Figura 3: Tumor pituitario observado en resonancia magnética (izquierda) y glándula pituitaria (derecha)

Los tumores cerebrales representan el 2% de todos los tumores existentes en adultos y el 15% de los tumores que afectan a niños menores de 15 años. La edad media de los tumores cerebrales en adultos esta entre 40 y 70 años. Aun así, el 90% de los tumores cerebrales diagnosticados son benignos.

Actualmente, en España, se detectan cada año alrededor de 3500 casos de tumores cerebrales según la ASATE (Asociación de Afectados por Tumores Cerebrales en España) y

unos 300000 tumores cerebrales primarios en todo el mundo, siendo una de las causas de muerte más comunes en niños de entre 0 y 5 años [4].

4.2. Diagnóstico

La principal técnica de diagnóstico actual es la resonancia magnética o IRM (Imagen de Resonancia Magnética). Existen otro método de diagnóstico muy utilizado también en la detección de tumores cerebrales que es la tomografía axial computarizada o TAC. Aun así, la resonancia y el TAC únicamente proporcionan información sobre la localización y tamaño del tumor. Para saber el tipo de tumor es necesario practicar una biopsia. A continuación, se explicarán más a fondo cada uno de estos 3 diagnósticos:

- Resonancia Magnética o IRM
  Una resonancia magnética se encarga de crear imágenes transversales del interior del cuerpo, para ello no hace uso de radiación, sino de magnetismo. Además, muchas veces crea imágenes de tejidos blandos que suelen ser complicadas de ver con otras técnicas. Suele ser una técnica orientada a detectar tumores en el cuerpo. Para los tumores cerebrales y de medula se suelen usar resonancias de tinte de contraste. Mediante esta técnica se puede analizar perfectamente si el tumor es maligno o benigno, además de ayudar a los médicos a establecer el tiempo de tratamiento necesario.

  El funcionamiento de un examen de resonancia magnética es el siguiente: se tumba al paciente en una camilla que se introduce en un gran tubo, ese tubo contiene un gran imán que genera un campo magnético alrededor del paciente. La máquina usa la fuerza magnética generada para mandar una serie de ondas de radiofrecuencia que recogen las señales de los núcleos de los átomos de hidrogeno que hay en el cuerpo humano. A partir de dichas señales, un ordenador genera una imagen en blanco y negro.

  Asimismo, es interesante mencionar que la técnica de Resonancia Magnética está basada en cuadrupolo magnético. El cuadrupolo magnético es un tipo de fuente de campo magnético que consigue cancelar el momento dipolar y generar un momento cuadrupolar. De esta manera, el campo generado disminuye a grandes distancias de forma más rápida que un dipolo.

  Esta prueba suele durar entre 40 y 60 minutos, en casos poco habituales hasta 2 horas. Se pueden usar materiales de contraste para mejorar la calidad de las imágenes generadas. Sin embargo, este material puede llegar a provocar en algunos pacientes nauseas, dolores de cabeza y disminución de la presión arterial [3].

Figura 4: Tubo que contiene el imán de la prueba (izquierda) e imágenes resultantes de la prueba IRM (derecha)

- Tomografía Axial Computarizada o TAC
  Un TAC genera imágenes de secciones transversales del cuerpo en la que se pueden observar tejidos blandos, huesos y órganos con una buena claridad. Es una técnica que permite determinar el tamaño y forma de un tumor con mucha precisión.

  El funcionamiento de un TAC es el siguiente: se dirige un haz de radiación en diferentes ángulos para crear imágenes. Luego se manda la información de cada ángulo tomado a un ordenador para generar una imagen en blanco y negro que mostrará el corte de una determinada parte del cuerpo.
  Como ocurría en el IRM, también se pueden usar materiales de contraste para aclarar la imagen. En algunas personas, el uso de estos materiales puede producir en el paciente nauseas, dificultad para respirar y erupciones. La prueba suele durar entre 15 a 30 minutos [3].

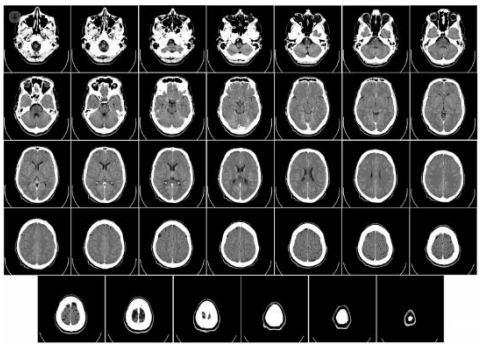

Figura 5: TAC craneal visto en diferentes cortes

- Biopsia
  Una biopsia consiste en extraer una muestra de tejido del cuerpo para que la examine un médico experto y por tanto poder establecer un diagnóstico. Es una técnica capaz de establecer el tipo de tumor, sus características y grado de peligrosidad, es decir, si es maligno o benigno.

  Es habitual aplicar una biopsia, tras haber realizado una resonancia magnética y no haber podido determinar el grado de peligrosidad del tumor. Esta prueba se suele realizar con aguja, es decir, de manera percutánea. Una vez mandada la muestra obtenida en la biopsia a diagnosticar, suelen tardar entre 1 o 2 semanas en evaluarse por un profesional [3].

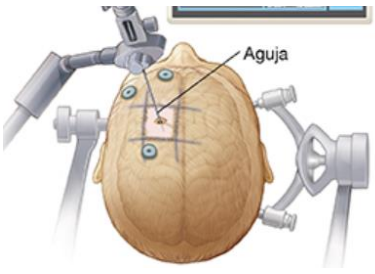

Figura 6: Representación gráfica de una biopsia cerebral

4.3. Tratamientos

En cuanto a los tratamientos más comunes para combatir un tumor cerebral destacaremos 3:

- Cirugía
  Se suele aplicar cuando el tumor se encuentra en una zona accesible para operar. Asimismo, es necesario que el tumor se pueda despegar del tejido cerebral al que esta adherido. Cuando el tumor se encuentra localizado en una zona sensible del cerebro, la cirugía se convierte en una técnica arriesgada [3].

- Radioterapia:
  Se usan haces de alta energía como rayos X para eliminar las células anormales que forman el tumor. Recientemente, se ha introducido una nueva forma de radiación para la radioterapia y estos son los rayos de protones. Esta nueva forma permite controlar la radiación entrante en el cerebro de forma más precisa. Esta técnica es la más usada cuando el tumor se encuentra en zonas sensibles del cerebro, y más aun usando la radiación en forma de protones por su precisión [3].

- Quimioterapia:
  Consiste en el uso de medicamentos para eliminar a las células que forman el tumor. Dichos medicamentos se pueden tomar vía oral o por vena. Según el tipo de tumor, se utilizan unos medicamentos u otros. Se van realizando pruebas en el paciente para analizar si la quimioterapia está siendo efectiva o no [3].

Como hemos visto, una de las mejores técnicas para detectar tumores cerebrales es la resonancia magnética o IRM. Tras la aplicación de esta técnica en pacientes, se requiere de personal de radiología que lleven a cabo una inspección manual de las imágenes obtenidas para poder diagnosticar la existencia de tumor cerebral. Asimismo, después de obtener las imágenes IRM es necesario personal médico especializado en radiología que sea capaz de realizar una detección manual del tumor cerebral. Esta tarea suele ser compleja por las distintas propiedades que presentan los tumores cerebrales, por lo que el radiólogo encargado de la detección tendrá que tener experiencia y una formación adecuada en el campo.

En países como España o Qatar se dispone de personal cualificada para llevar a cabo dicho diagnóstico manual del tumor cerebral tras la resonancia magnética, sin embargo, en países en desarrollo de Asia como Tailandia o de África como Ghana, no se dispone de los suficientes recursos para llevar a cabo estas tareas. En estos países en desarrollo, la falta de doctores habilidosos y la falta de conocimiento sobre tumores cerebrales hace que sea muy complejo establecer informes médicos del IRM obtenido y se pueden llegar a dar errores en la detección de la enfermedad.

Como consecuencia de lo anterior, el uso del Machine Learning y la Inteligencia Artificial han tomado recientemente protagonismo en la detección y clasificación de tumores cerebrales. Mientras que un médico es capaz de retener un centenar de imágenes en su cabeza, estas nuevas tecnologías, aparte de sustituir en cierta medida al médico, proporcionan el análisis de una muestra con millones de imágenes.

Con el objetivo de ayudar a países en desarrollo sin recursos y llevar a cabo la tarea de detección con un nivel de precisión mucho mayor que el humano, se han desarrollado sistemas de redes neuronales muy potentes alimentados con bases de datos de imágenes

IRM de todo el mundo. De esta forma, se podrá detectar de forma automática y precisa si un paciente tiene un tumor cerebral o no. En caso de tenerlo, existen redes neuronales, como es el caso de este trabajo, que consiguen clasificar el tumor cerebral.

5. TÉCNICAS DE MACHINE LEARNING

5.1. Machine Learning

El Machine Learning o aprendizaje automático es una de las ramas más importantes y desarrolladas de la Inteligencia Artificial. A lo largo de los años, no solo ha visto novedosas evoluciones, sino también numerosos estancamientos. Desde los test de Turing en 1950 hasta la fundación de OpenAI de la mano de Elon Musk y Sam Altman en 2015, el Machine Learning ha supuesto un cambio radical en como vemos y utilizamos la computación y las maquinas actuales. El Machine Learning permite a los sistemas aprender y mejorar de forma automática a partir de la experiencia, sin ser programados ellos mismos. Además, permite a los algoritmos reconocer patrones complicados a partir de una base de datos, todo ello de forma automática. A diferencia del Deep Learning, del cual hablaremos más abajo, el Machine Learning suele requerir habitualmente de participación humana, ya que permite revisar los resultados de un algoritmo y hacer ajustes según el nivel de precisión.

Dentro de las técnicas de Machine Learning que se pueden usar actualmente se destacan 3:

- Aprendizaje supervisado: la maquina se entrena con datos etiquetados. Es el caso de la red que se usará para este trabajo, ya que las imágenes de los cerebros a estudiar disponen de etiquetas según sea no tumor, glioma, meningioma o pituitario [5].
- Aprendizaje no supervisado: los algoritmos que desarrollan las maquinas carecen de un conocimiento previo a diferencia del supervisado. Es el propio sistema el que reconocerá patrones, similitudes o características por sí solo. Es por tanto que la maquina deberá ser autónoma. Si se conocen todas las variables de entrada relevantes para una respuesta, entonces, aunque no se disponga de muestras etiquetadas, se puede suponer que puntos con valores próximos en variables de entrada, tendrán valores próximos también en las de salida [5].
- Aprendizaje de refuerzo: la maquina aprende a partir de un proceso de prueba y error, es decir, a partir de su experiencia. De esta forma el algoritmo conseguirá optimizar los procesos ya que sabe dónde se equivocó con anterioridad. Se trata de un aprendizaje intermedio al supervisado y no supervisado, en donde se dan etiquetas con información parcial [5].

Algunas de las técnicas más usadas en cada aprendizaje son las siguientes:

- En aprendizaje supervisado distinguimos dos problemas: Regresión y Clasificación. En regresión, el objetivo es predecir un resultado que varía en un rango numérico. Por ejemplo, a partir de una foto de una persona, predecir su edad. En clasificación, lo que se busca es predecir resultados, pero estos siendo categóricos. Un ejemplo claro es el de este trabajo, detectar y clasificar tumores cerebrales en pacientes a partir de imágenes IRM.
- En aprendizaje no supervisado destacamos el Clustering y Detección de Anomalías. El clustering, por un lado, se encarga de agrupar datos de entrada según criterios en concreto como puede ser en la venta de productos, mientras que, por el otro lado, la detección de anomalías se encarga de agrupar datos de entrada en función de unos parámetros normales prefijados detectando cuales no encajan. Algunos ejemplos de problemas que resuelve la técnica detección de anomalías son el fraude de tarjetas de crédito, videovigilancia o intrusiones en la red.
- La principal técnica del aprendizaje de refuerzo es el de resolver juegos y conseguir rendimientos sobrehumanos.

5.2. Deep Learning

Si se profundiza más en el Machine Learning y en la Inteligencia Artificial, es necesario hablar del Deep Learning, que se considera una rama del Machine Learning. El Deep Learning se encarga de analizar información que se le presenta a un sistema sin ningún tipo de programación previa, después establece una serie de predicciones que se comprueban con información de contraste para ver el nivel precisión de dicha predicción y, por último, se le informa a la nueva predicción que realizara el sistema con el nivel de precisión de la predicción anterior, de tal forma que evite caer en los mismos errores y mejore los resultados.

A diferencia del Machine Learning, el Deep Learning se basa en el uso de redes neuronales, es decir, en un conjunto de algoritmos organizado en capas que simula la estructura del cerebro humano. Se le llama “Deep” ya que se profundiza en una red de capas. También, a diferencia del Machine Learning, no hace uso de participación humana para ajustar los algoritmos, sino que la red desarrolla un autoaprendizaje para mejorar en la tarea que desempeñe. Es importante destacar el carácter secuencial que presenta el Deep Learning, en donde primero se identifican aspectos más bastos de la imagen, como puede ser la silueta de un objeto en una imagen, y después se focalizan aspectos más concretos, como puede ser las partes que forman dicha silueta en la imagen.

Las principales ventajas del Deep Learning frente al Machine Learning es que trabaja mejor con datos sin estructurar, aplica algoritmos con personal menos cualificado y realiza técnicas más complejas que ML. Sin embargo, también tiene alguna desventaja como que requiere mejor hardware para trabajar, perjudica más al medioambiente debido a su alto consumo eléctrico y sus consecuentes emisiones contaminantes, y requiere de mayor cantidad de datos que ML [6].

Dentro del Deep Learning distinguiremos entre los siguientes tipos de redes neuronales según su clasificación:

- Según el número de capas: redes neuronales monocapa (capa de entrada y de salida) y multicapas (capa de entrada, ocultas y de salida).
- Según el tipo de conexión: redes neuronales no recurrentes (conexiones de único sentido, no hay realimentación y sin memoria) y recurrentes (conexiones con realimentación y memoria)
- Según el grado de conexiones: redes neuronales totalmente conectadas (todas las neuronas conectadas entre ellas) y conectadas (no todas las neuronas están conectadas entre ellas) [7].

5.3. Redes Neuronales Convolucionales

Las redes neuronales convolucionales o convolutional neural network (CNN) son redes multicapa cuyas entradas son explícitamente imágenes. En el caso de este trabajo, dichas imágenes serán las obtenidas en los diagnósticos de IRM a pacientes con o sin tumor cerebral y que serán proporcionadas por una base de datos del portal de Kaggle en Internet.

5.3.1. Capa convolucional

El aspecto fundamental de estas redes es el uso de la operación de convolución. Mientras que en matemáticas dicha operación consiste en transformar dos señales o funciones en una tercera, en el caso de las redes neuronales se usa para aplicar una serie de operaciones matemáticas (productos escalares) entre una matriz formada por algunos de los

pixeles de la imagen de entrada y un filtro o kernel, también en forma de matriz. Como resultado de esta operación se obtiene una nueva matriz de salida también conocida como mapa de características o feature map, con información más útil y significativa que la que había entrado en un principio en la red. La tarea principal de la red CNN es ir aprendiendo valores para la matriz de filtro o kernel, que den mejores resultados a los anteriores [9]. A continuación, se muestra una figura que representa lo anterior comentado:

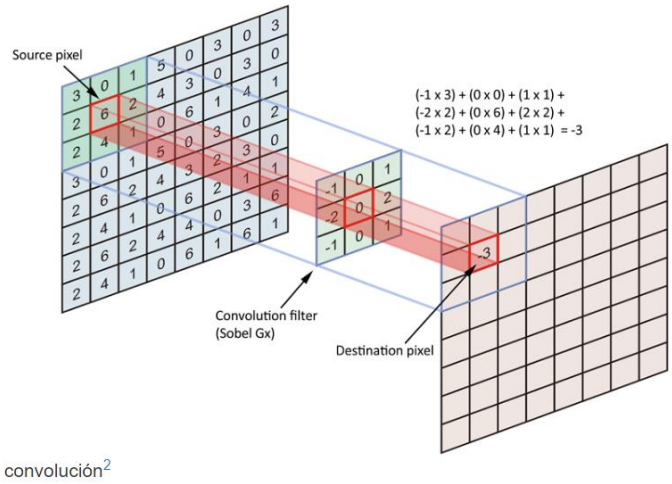

Figura 7: Aplicación de la convolución para crear un mapa de características.

El bloque de construcción central de una red neuronal es la capa o layer, y al trabajar con redes neuronales convolucionales dichas capas pasaran a llamarse capas convolucionales. Estas, son módulos de procesamiento de datos que actúan como filtros. Una red CNN está formada por una gran cantidad de capas y a medida que nos adentramos en la red, la información que se obtendrá como salida de cada capa convolucional será cada vez más útil y precisa [8].

Cada capa convolucional, a su vez, dispone de múltiples canales de salida. Cada canal de salida representa una característica o feature que obtendrá de la entrada a la capa convolucional. Por ejemplo, algunas características comunes que se suelen asociar a los canales de salida son: detección de líneas verticales, detección de líneas horizontales, detección de líneas diagonales, etc. De esta forma, a medida que una imagen avanza por la red y debido a las operaciones de convolución de cada capa convolucional, se podrán detectar cambios de contraste, texturas y demás características de la imagen. Dentro de una misma capa convolucional, todos sus canales de salida tendrán la misma resolución o número de neuronas por canal. Esta medida se representa con el ancho y alto de los canales en cada capa convolucional. A continuación, se plasma en una figura:

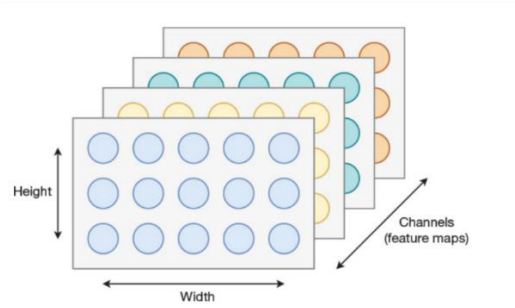

Figura 8: Topología de una capa convolucional

Cada punto de la figura 8 representa una neurona y la resolución de la capa convolucional por tanto será el número de neuronas dentro de cada canal. Como se puede ver cada canal de salida tiene el mismo número de neuronas (6x3), sin embargo, aunque cada neurona de un mismo canal tenga el mismo peso que el resto, en el resto de canales de la capa dicho peso puede cambiar.

Como la propia red va aprendiendo patrones a medida que pasan imágenes por ella, no es necesario predefinir las características que van a representar los canales de salida de cada capa convolucional. Además, es importante destacar que el número de canales de salida en una capa es independiente del número de canales de entrada. Por un lado, el número de canales de entrada determinará los pesos de cada neurona en los distintos canales de salida, mientras que, por el otro lado, el número de canales de salida será función del número de neuronas que queramos tener en nuestra capa convolucional [8].

Se pueden superponer capas convolucionales unas sobre otras, de tal manera que los canales de salida de una de las capas serán los canales de entrada de la siguiente. Así, los mapas de características creados tras las operaciones de convolución de una de las capas serán una combinación de los mapas de las capas anteriores. Es por esto que se le da el nombre al Deep Learning.

Para ver esto anterior en mayor detalle usaremos un ejemplo más sencillo que el de detección y clasificación de tumores cerebrales, como es el de clasificación de animales. A continuación, se observa una figura que representa una red neuronal convolucional con 2 capas convolucionales y una capa densamente conectada que se encarga de clasificar imágenes de distintos tipos de animales:

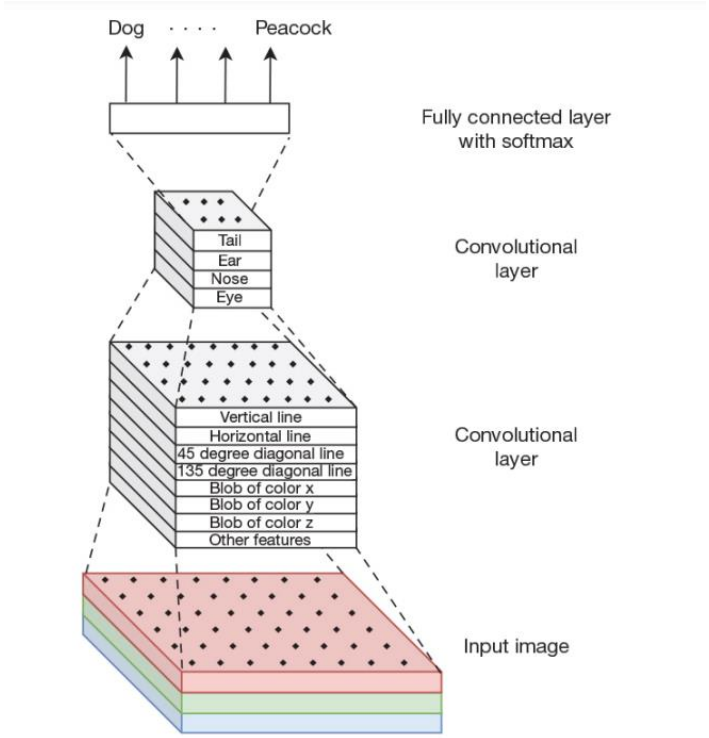

Figura 9: CNN construida con dos capas convolucionales y una capa densamente conectada

En la figura se aprecia lo anteriormente comentado, en donde la imagen de entrada de resolución 8x6, que tiene 3 canales de salida al ser una imagen en color (canal rojo, canal verde y canal azul), se introduce en la primera capa convolucional de resolución menor (8x4) pero con muchos más canales de salida, cada uno con una característica distinta (línea vertical, línea horizontal, …). Asimismo, los pesos que tendrán las neuronas en cada canal de salida de esa primera capa de convolucional no tienen por qué ser iguales, sino que vendrán dados por el número de canales de entrada de la capa. Por consiguiente, se puede ver como los mapas de características o canales de salida de la segunda capa convolucional son muchos más complejos (cola, oreja, nariz y ojo) que los de la primera, ya que esta segunda capa ha utilizado los mapas de la capa anterior para mejorar la clasificación. Esto demuestra que las primeras capas de una red CNN identifican características de bajo nivel como esquinas, líneas o colores, mientras que en las capas más profundas se detectan características de mayor nivel como formas o texturas.

5.3.2. Capa reductora o pooling

A medida que se avanza por la red, la resolución de las capas disminuye hasta tal punto que la última capa, en este caso la capa densamente conectada (fully connected layer en la figura 3), tiene el mismo número de neuronas o resolución (1x4) que el número de animales a clasificar. Para conseguir que la resolución de las capas vaya disminuyendo a medida que nos adentramos en la red, se hace uso del stride, o también conocido como paso, y del max pooling. El stride es un parámetro de la capa convolucional que representa el número de posiciones que el kernel o filtro se moverán por la entrada de la capa. La teoría nos dice que para disminuir la resolución de las capas a medida que avanzamos por la red, es necesario siempre un stride mayor que 1. Por otro lado, el max pooling consiste en hacer pasar un

kernel o filtro vacío por la imagen entrante, pero, en vez de aplicar la convolución, se toman los valores más grandes de la sección de la imagen. El max pooling se suele representar como una capa a continuación de la capa convolucional. También es conocida como capa reductora o pooling. La principal ventaja de esta capa es que reduce el fenómeno de overfitting o sobreajuste. El overffiting ocurre cuando se tienen en el entrenamiento de la red características muy comunes por lo que nuestra red se aprenderá casos muy particulares y será incapaz de clasificar casos nuevos con los datos de test, por lo que los rendimientos en el entrenamiento y en el test serán muy dispares. Por tanto, esta capa reductora permite reducir la carga de trabajo de la red [8]. En la siguiente figura se representa lo anterior:

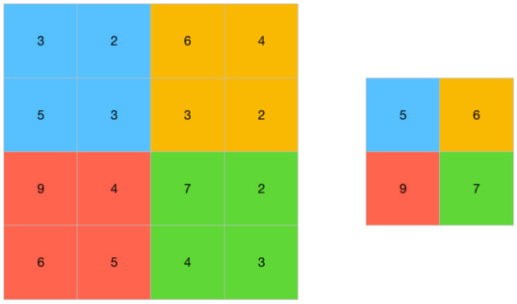

Figura 10: Aplicación del max pooling para reducir la resolución

En la figura 10, se aprecia como el kernel pasa 4 veces por la imagen, cada una representada por un color, recopilando el máximo valor de cada caja de color. Con este método, se consigue reducir el tamaño de la matriz resultante o mapa de características, y, por tanto, la resolución de la capa.

Aunque la resolución de las capas convolucionales disminuya al avanzar a través de la red, el grosor de dichas capas va aumentando. Esto, quiere decir que, cuantas más capas convolucionales tengamos, más mapas de características obtendremos, por lo que se podrá clasificar de una manera más precisa las imágenes entrantes a la red [10].

En la siguiente imagen, en la que se clasifica un automóvil, se usan dos capas convolucionales y cada una de ellas con una capa de max pooling:

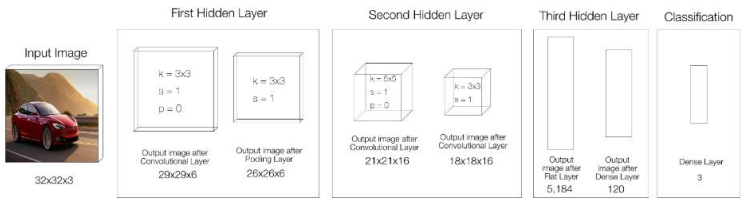

Figura 11: Red CNN para clasificar un automóvil

Se aprecia como la resolución o número de neuronas por canal de salida va disminuyendo (disminución del largo y altura del paralelepípedo en la Figura 11), es decir, pasa de tener

26x26 neuronas por canal de salida en la primera capa convolucional a tener 18x18 neuronas por canal de salida en la segunda capa convolucional. Aun así, como ya se ha comentado, el grosor de dichas capas va aumentando (el ancho del paralelepípedo va aumentando), ya que en la primera capa tenemos 6 mapas de características o canales de salida, mientras que en la segunda capa se tienen 16. Esto nos indica que al avanzar por la red van aumentando el número de características que podremos detectar en la imagen inicial de entrada. En otras palabras, la clasificación al avanzar por la red se vuelve más útil y precisa al disponer de mayores recursos para detectar y clasificar [13].

Por tanto, a modo de recopilación de todo lo comentado anteriormente, una red neuronal convolucional CNN funciona de la siguiente forma:

En primer lugar, entra una imagen a la red, ya sea en color (3 canales de salida) o en tono de grises (blanco y negro, 1 solo canal de salida). Se le aplica una serie de convoluciones, en función del número de capas convolucionales que disponga la red (cuantas más clases haya para clasificar, mayor número de capas tendrá la red). Tras aplicar todos los filtros o kernels necesarios se generan un conjunto de mapas de características. Como ya hemos visto, la resolución (número de neuronas por canal de salida en cada capa convolucional) irá disminuyendo a medida que la imagen avanza por la red, mientras que el número de características (grosor de las capas convolucionales) para clasificar irá aumentando. Los outputs de cada capa se convertirán en los inputs de las siguientes capas.

Cuando la imagen pasa por todo el “embudo” convolucional, se llegará a un punto en el que la red conocerá todos los patrones necesarios para clasificar la imagen, es decir, la red tendrá un gran número de mapas de características que ahora, podrá introducir como entradas en una red neuronal multicapa que acabará tomando la decisión de lo que es la imagen inicial de entrada.

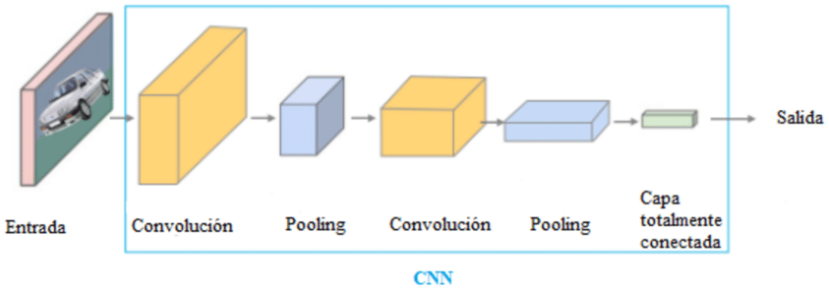

Figura 12: Red CNN con forma de embudo

5.3.3. Capa clasificadora o capa completamente conectada

Una vez hemos hablado de cómo funcionan las capas convolucionales, pasaremos a hablar de la capa totalmente conectada, siendo esta la parte final de red, encargada de clasificar a la imagen. Esta última capa simula una neurona por cada pixel a analizar y actúa como una red neuronal multicapa. En esta capa se tienen tantas neuronas como clases a predecir haya. Normalmente, también se suele llamar capa clasificadora, ya que, con todos los recursos obtenidos atrás en la red, como son los mapas de características y patrones aprendidos, los usará para clasificar la imagen [8].

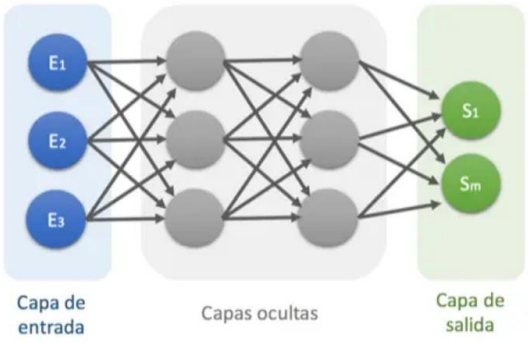

Figura 13: Arquitectura de una red neuronal multicapa

Cada neurona de la red neuronal multicapa tiene asociada una función de activación. Esta vera si la suma de los valores recibidos que han sido multiplicados por el peso correspondiente a dicha neurona supera un cierto valor, la propia función actuará como "trigger" o interruptor para activar la neurona y enviar el valor a la próxima capa. Dentro de las funciones más comunes destaca la función ReLu (Rectifier Linear Unit), definida por la siguiente expresión:

$$ReLu(x) = \begin{cases} 0 & si\ x \leq 0 \\ x & si\ x > 0 \end{cases}$$

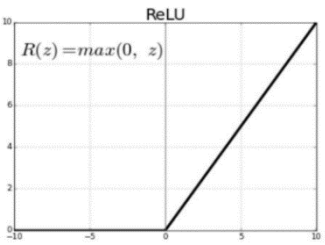

Figura 14: Expresión matemática y grafica de la función ReLu

La principal ventaja de esta función es que tiene una gran velocidad de cálculos al ser estos lo suficientemente simples. Aun así, como desventaja puede que la conversión a cero de la función lleve a que los pesos de las neuronas tiendan a cero. Asimismo, a partir de dicha función de activación se puede conocer como aprenden los filtros o kernels de las capas de convolución, explicadas anteriormente. Variando la activación del mapa de características respecto a los pixeles de la imagen, se podría optimizar al filtro buscando su maximización de tal forma que mejore los resultados posteriores de la red en la clasificación de las imágenes [8] [9].

Por consiguiente, en esta red multicapa, cada neurona tiene unos parámetros asociados, aparte de la función de activación asociada, en los que se incluyen los pesos asociados, y deberán estar perfectamente ajustados para que todas las operaciones internas acaben procesando las entradas a dicha red con la salida que le corresponde.

La combinación idónea de dichos parámetros para cada neurona es una tarea compleja que se lleva a cabo con el algoritmo de descenso del gradiente. Este algoritmo consiste en obtener el error mínimo de los parámetros seleccionados de tal forma que la red vaya aprendiendo y mejorando los resultados de clasificación anteriores.

Cuando se entrena una red CNN lo que se hace es optimizar una función de error a partir del ajuste de los parámetros y así, obtener un punto mínimo satisfactorio. Esta función de

error se conoce como función de pérdida o loss function y mide cuanto de buenos son los resultados obtenidos por la red. El caso ideal se da cuando la función de costes es igual a cero, siendo esto muy poco probable. Aun así, a medida que la red va cogiendo experiencia, los resultados mejoraran y, por tanto, la función de coste disminuirá [10]. A continuación, se ejemplifica gráficamente como se da dicha optimización de la función de error:

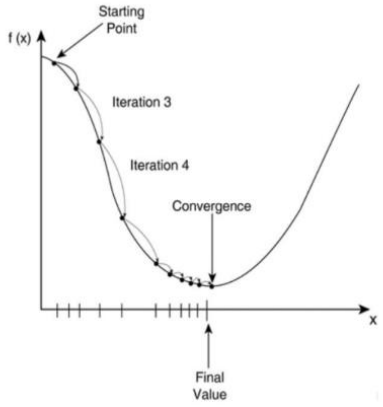

Figura 15: Grafica que explica el gradiente de descenso

Otro algoritmo que destaca por mejorar resultados en la red y reducir la función de coste es el algoritmo de back propagation, que se encarga de calcular estimaciones que fluyen hacia atrás en la red para calcular un gradiente que mejore los resultados futuros. En términos simples, después de cada paso adelante a través de la red, el algoritmo realiza un paso hacia atrás mientras ajusta los parámetros del modelo.

5.3.4. Ventajas de las redes neuronales convolucionales

- La principal ventaja de las redes CNN frente a otro tipo de redes neuronales es que se pueden detectar automáticamente características de la imagen sin ser necesario la presencia de supervisión humana.
- Es una red que destaca por su autoaprendizaje.
- La red CNN hace uso de un menor número de parámetros ya que las características necesarias para clasificar a la imagen ya han sido extraídas con antelación por las capas convolucionales.
- Son mucho más eficientes que las redes neuronales convencionales, incluso haciendo uso de un menor número de operaciones por neurona.
- Al hacer uso de menos operaciones, se consigue un ahorro de memoria neural alto.
- Tienen una mejor capacidad de análisis de los parámetros necesarios.
- Menor perdida de información relevante que otras redes, y también menor coste computacional.
- Necesitan menor espacio para almacenar los pesos
- Necesitan menor búsqueda de espacio de tal forma que el algoritmo aprende más rápido

[11]

5.3.5. Aplicación de las redes neuronales convolucionales a la detección de tumores cerebrales

Una vez visto cómo funcionan las redes neuronales convolucionales, veremos porque estas son las más indicadas para detectar y clasificar tumores cerebrales a partir de imágenes IRM.

En primer lugar, como se ha visto, las redes neuronales convolucionales trabajan de forma automática sin ningún tipo de participación humana. Es por ello que, al ser los tumores cerebrales complicados de detectar de forma manual, un sistema que, mediante el autoaprendizaje y la automejora, permitirá facilitar esta tarea.

En segundo lugar, a diferencia de las redes neuronales recurrentes que se utilizan para tareas de habla y lenguaje, se ha demostrado con resultados que las redes CNN son las más adecuadas para el sector de la medicina, más concretamente los tumores cerebrales a partir de IRM.

Algunos de los motivos de lo anterior son los siguientes:

- La invarianza traslacional, la cual permite reconocer el mismo elemento de una imagen en una ubicación distinta de la misma. Por esto, aunque el tumor en distintas imágenes se encuentre en distintas ubicaciones del cerebro, al ser un tipo de tumor diferente, la red, aun así, conseguirá detectarlo [9].

- Dado que las características importantes de la imagen IRM las aprenderá automáticamente la red, no es necesaria la extracción de información de las imágenes antes del proceso de aprendizaje. Por lo tanto, las redes CNN son relativamente fáciles de aplicar en la práctica clínica [8].

- Las imágenes IRM que se utilizan para detectar los tumores requieren de un preprocesamiento previo a la introducción en la red, de tal forma, que se consiga mejorar el rendimiento de los resultados obtenidos [8] [9]. Algunas de las tareas de este preprocesamiento son:

    - Separar las imágenes en conjuntos de entrenamiento (training, 70% aprox de los datos) y de prueba (test, 30% aprox de los datos). Estos conjuntos de datos suelen ser muestreados de forma aleatoria para asegurar que los conjuntos sean representativos de todo el conjunto de datos. Además, se realiza esto para eliminar el sesgo de selección. Muchas de las bases de datos actuales que te proporcionan con las imágenes IRM ya realizan de antemano dicha separación, lo cual ahorra mucho tiempo previo.

    - Cuando el número de datos pertenecientes a una clase es mucho mayor o menor que el de resto de clase, se suele aplicar una técnica de data augmentation, muy característica en problemas del sector médico, como el de este trabajo, ya que normalmente hay muchos más datos de pacientes sin tumor que con tumor. El data augmentation consiste coger una imagen cualquiera de nuestro conjunto de datos y aplicarle una serie de modificaciones, como pueden ser rotaciones, reflejos o volteos de la imagen original para así, tener muchas más muestras. Esta técnica se suele aplicar a la clase que tenga menor número de imágenes que el resto hasta conseguir igualarla

6. LENGUAJE DE PROGRAMACIÓN Y TENSORFLOW

6.1. Lenguaje de programación: R y RStudio

La programación de la red neuronal convolucional que se usará en este trabajo se basará en el lenguaje de programación de R. R es un entorno de software libre que ejecuta instrucciones de forma directa sin una compilación previa del programa a instrucciones del lenguaje máquina, como pasaría con otros lenguajes como C+.

R es un lenguaje de programación totalmente diferente al resto, principalmente dedicado al campo de la estadística. Aunque se puede usar directamente R para programar, es recomendable instalar un entorno integrado de desarrollo que nos permita escribir y revisar el código necesario. También es necesario para poder trabajar de una forma más eficiente con los archivos correspondientes que se necesiten. De entre todos los entornos integrados que existen, en este trabajo se usará RStudio, principalmente por su enfoque de colaboración y accesibilidad. Asimismo, RStudio destaca por permitir crear códigos de forma limpia y ordenada, por su compatibilidad con otros lenguajes de programación, por su incorporación de gráficos y su acceso gratuito. En este trabajo se usará RStudio en un equipo Windows.

6.2. Uso de RStudio para programar redes neuronales convolucionales

Para poder programar con RStudio las redes neuronales convolucionales, se necesita hacer uso de distintos frameworks e interfaces de trabajo, como son Tensorflow y Keras, que nos permitirán acceder a algoritmos y modelos típicos de problemas de clasificación de imágenes, como es el caso de este proyecto, y de regresión, como el procesamiento de lenguaje natural.

De forma general, Tensorflow es un framework creado por Google que se suele utilizar para el lenguaje de programación de Python, aun así, se puede usar en otros lenguajes distintos. Para su uso en RStudio es necesario la instalación de paquetes de Keras, que en este caso actuará como interfaz de programación de Tensorflow, desarrollada en Python. Para ello, en RStudio se instala el paquete {keras} lo cual nos permitirá tener los beneficios de programación de R, pero además aprovechando la capacidad de Python.

Además del paquete de Keras correspondiente se necesitan los siguientes paquetes y funciones: paquete {Tensorflow}, función Tensorflow::install_tensorflow(). Para la función anterior es necesario la instalación de Miniconda, una herramienta que se encarga de ordenar los paquetes que se usaran de forma sencilla [12].

6.3. Ventajas del uso de Tensorflow para redes neuronales frente a otros frameworks

Los principales factores por los que se escoge Tensorflow como framework de trabajo en redes neuronales frente a otros como Pytorch, o Microsoft Cognitive Toolkit son los siguientes:

- Esta principalmente orientado a la clasificación de imágenes, además de a muchas otras aplicaciones
- Es un software que sigue mejorando y está teniendo un gran crecimiento en los últimos años. Es considerado el futuro de la modelización de Machine Learning.

- Es una herramienta que usan multitud de empresas de todo el mundo para realizar proyectos de investigación, como el de este TFG.
- Al tener modelos gráficos hace que sea muy útil para el desarrollo de redes neuronales.
- Las bibliotecas auxiliares de Tensorflow ayudan a depurar la red neuronal de forma sencilla.

[15]

7. METOLOGÍA Y RESULTADOS

7.1. Introducción

Una vez presentado el marco teórico del proyecto, como son la enfermedad y su diagnóstico, y los fundamentos del funcionamiento de las redes neuronales convolucionales, se pasará a presentar las etapas seguidas en la obtención de datos y la programación de la red neuronal. Por último, en este mismo capitulo, se analizarán los resultados obtenidos tras simular la red, una vez programada la misma.

A continuación, se presentan las etapas seguidas en este capítulo:

1. Inicio del proyecto
2. Instalación de paquetes y funciones necesarias para trabajar con RStudio, Keras y Tensorflow. Además, se realiza una comprobación de la compatibilidad de estas interfaces y frameworks con el equipo utilizado (Windows).
3. Obtención de datos
4. Gestión e importación de los datos en el equipo utilizado
5. Preprocesamiento de los datos (data augmentation)
6. Construcción de la arquitectura del modelo
7. Evaluación del modelo
8. Ajuste del modelo
9. Predicción en la base de datos de prueba (Testing)
10. Análisis de los resultados obtenidos

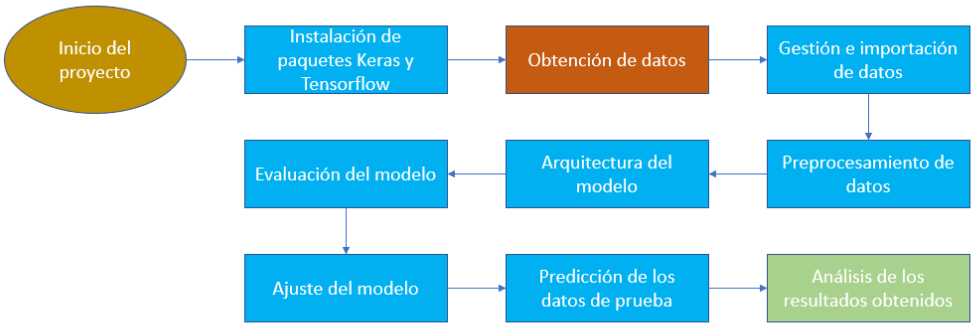

Figura 16: Diagrama secuencial del desarrollo de la metodología en el proyecto

En el diagrama anterior, se han utilizado colores para distinguir entre tareas realizadas en Internet, en la propia interfaz de RStudio o en las carpetas del ordenador utilizado. Para ello el azul representa todas aquellas operaciones realizadas en RStudio, como pueden ser la introducción de las carpetas con los datos, la instalación de los paquetes y bibliotecas necesarias para programar la red o la escritura del código. El color salmón, representa la única tarea que requiere del uso de Internet, que es la búsqueda y obtención de los datos. Por último, en verde, se muestra la tarea de análisis de resultados.

7.2. Fase de obtención y gestión de datos

Actualmente, existe una plataforma web de data science que proporciona al usuario con multitud de publicaciones sobre temas muy variados e interesantes. Además, estas publicaciones contienen todas las herramientas y recursos disponibles para llevar a cabo proyectos de análisis de datos. El aspecto fundamental de esta web es que los datos necesarios para cada proyecto, vienen incorporados de forma pública en cada publicación de la web y su descarga correspondiente es sencilla y gratuita. Asimismo, en Kaggle se puede interactuar con otros usuarios vía foro, para profundizar más a fondo en temas en concreto, compartir códigos y datos, y realizar competencias sobre temas interesantes.

Los datos utilizados en este Trabajo de Fin de Grado han sido obtenidos en Kaggle. Al descargarlo, se tienen dos carpetas etiquetadas: Testing que representan los datos de prueba y Training que representan los datos de entrenamiento. Normalmente, en problemas de redes neuronales se tiene un 30% de datos de prueba y un 70% de datos de entrenamiento. Sin embargo, en este modelo en concreto, se trabajará con 394 datos de prueba y 2870 datos de entrenamiento, por lo que habrá, un 12.07% de datos de Testing y 87.93% de datos de Training.

Tanto en la carpeta de Testing como en la de Training, se encontrarán los datos divididos en 4 conjuntos diferentes: glioma_tumor, meningioma_tumor, no_tumor y pituitary_tumor. Estos conjuntos representan los 3 tipos de tumores cerebrales que detectara la red y un conjunto en caso de que el paciente no tenga tumor.

7.3. Fase de programación de la red neuronal convolucional

7.3.1. Librerías utilizadas

Lo primero que habrá que hacer será cargar en RStudio los paquetes necesarios para poder trabajar con las imágenes MRI y por tanto poder crear la red neuronal convolucional correspondiente [14]. Para ello se usarán las siguientes librerías:

- Tidyverse: permite manipular, importar, explorar y visualizar las imágenes IRM que se alimentan a la red.
- Imager: contiene un conjunto de funciones que permiten trabajar con imágenes y procesarlas. No solo permite cargar y guardar imágenes en distintos formatos, sino también visualizarlas, esquematizarlas y filtrarlas.
- Keras: permite crear los bloques de construcción para desarrollar la red neuronal convolucional. Además, se utiliza para entrenar a modelos de Deep Learning.
- Caret: contiene funciones para entrenar y esquematizar modelos de clasificación y regresión.

7.3.2. Importación y análisis exploratorio de los datos

Una vez se cargan los paquetes correspondientes en la interfaz de RStudio, se importan los datos, proporcionados por el portal de internet de Kaggle, en el equipo que se va a usar. Estos datos vienen organizados en 2 carpetas: Training y Testing. Dentro de cada carpeta se tienen las imágenes ordenadas en 4 carpetas distintas: 3 de ellas según el tipo de tumor que tenga el paciente, glioma_tumor, meningioma_tumor o pituitary_tumor, o, en caso de que no hubiera tumor, en una cuarta carpeta llamada no_tumor.

Si, por ejemplo, dentro de la carpeta de training se abriera la carpeta de glioma_tumor, no se encontrarían datos estructurados, sino que solo encontramos imágenes IRM con gliomas. Por lo tanto, se necesitan extraer dichas imágenes antes de procesarlas.

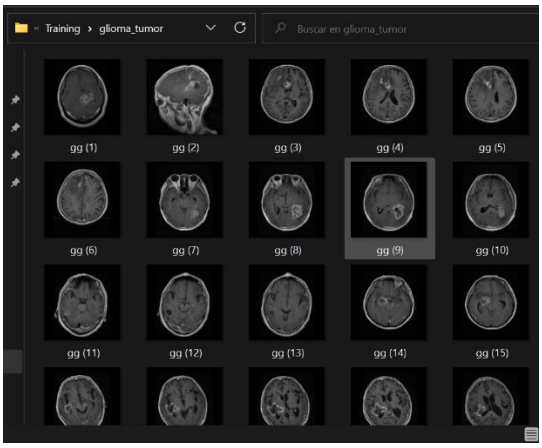

Figura 17: Visualización de la carpeta Training > glioma_tumor

Para extraer las carpetas que están dentro de Training se usa la función *list.files.* Después, se combinan las 4 carpetas en el directorio Training, para ello se usa la función *paste0* de R [14].

Para recopilar los nombres de los archivos de cada carpeta (glioma_tumor, meningioma_tumor, no_tumor y pituitary_tumor) se utiliza la función *map()* de R. Esta función se usará de forma repetitiva de tal forma que se forme una lista con todas las imágenes de Training [14]. A continuación, se muestran las 6 primeras filas de esta lista para hacerse a una idea:

```
[1] "dataset/training/glioma_tumor/gg (1).jpg"   "dataset/training/glioma_tumor/gg (10).jpg"
[3] "dataset/training/glioma_tumor/gg (100).jpg" "dataset/training/glioma_tumor/gg (101).jpg"
[5] "dataset/training/glioma_tumor/gg (102).jpg" "dataset/training/glioma_tumor/gg (103).jpg"
```

Figura 18: 6 primeras filas de la lista formada por los datos de Training

Ahora, se comprobará la dimensión de la lista, para ver cuantas imágenes se usarán para entrenar a la red neuronal convolucional. Tras aplicar la función *length()* se concluye que se tienen 2870 imágenes en la carpeta de Training. Si quisiéramos ver alguna de las imágenes en concreto, se usa la función *load.image* que pertenece a la librería imager [14]. A continuación, se encuentran 6 muestras de la carpeta Training:

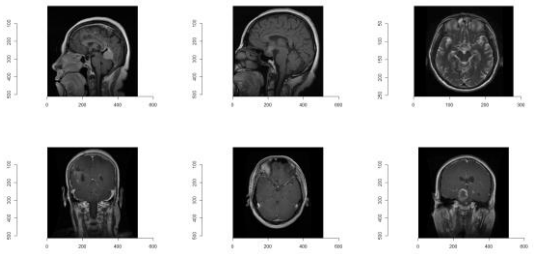

Figura 19: 6 imágenes IRM de pacientes con o sin tumor

Para crear un buen modelo de Deep Learning como el que se busca en este trabajo, es necesario que la distribución de dimensiones de las imágenes que estamos usando sean iguales. Para ello, antes de fijar dichas dimensiones, comprobaremos los baremos de cada imagen y así, ver como de alejadas están unas de otras. Mediante la función *dim()* de R se pueden obtener con facilidad 4 parámetros fundamentales de cada imagen: el ancho, la altura, la profundidad y el canal de colores. Si el canal de colores fuera 3 significa que tiene 3 canales: rojo, verde y azul. Si el canal de colores fuera 1 significa que únicamente tiene 1 canal: escala de grises [14].

En este modelo, nos centraremos únicamente en el ancho y altura de la imagen para obtener estadísticas de las imágenes en general. Para ello, se crearán 1000 muestras de la lista creada anteriormente con *map()* y se convertirán en dataframes que nos proporcionarán información del ancho, altura y nombre del archivo [14]. A continuación, se muestran las 10 primeras filas de dataframes:

	height	width	filename
1	512	512	dataset/training/pituitary_tumor/p (477).jpg
2	512	512	dataset/training/pituitary_tumor/p (52).jpg
3	512	512	dataset/training/pituitary_tumor/p (264).jpg
4	512	512	dataset/training/glioma_tumor/gg (572).jpg
5	512	512	dataset/training/glioma_tumor/gg (274).jpg
6	824	755	dataset/training/no_tumor/image(211).jpg
7	512	512	dataset/training/meningioma_tumor/m1(23).jpg
8	512	512	dataset/training/meningioma_tumor/m2 (123).jpg
9	512	512	dataset/training/meningioma_tumor/m2 (137).jpg
10	212	226	dataset/training/meningioma_tumor/m1(11).jpg

Figura 20: 10 primeras filas del dataframe creado, con la altura, ancho y nombre del archivo

A partir de las 1000 muestras y las funciones *map_df* y *summary()* se obtiene una tabla con distintos parámetros que servirán para fijar la dimensión que tendrán las imágenes al ser alimentadas a la red [14]. Dicha tabla se muestra abajo:

```
     height         width          filename
Min.   : 198   Min.   : 201.0   Length:1000
1st Qu.: 512   1st Qu.: 512.0   Class :character
Median : 512   Median : 512.0   Mode  :character
Mean   : 483   Mean   : 481.9
3rd Qu.: 512   3rd Qu.: 512.0
Max.   :1446   Max.   :1375.0
```

Figura 21: Tabla con estadísticas de las dimensiones del dataframe

Se puede apreciar que, de las 1000 muestras en la lista, existe variación en las dimensiones, tanto en la altura como en el ancho. Por ejemplo, en la altura hay imágenes que están alrededor de 198 píxeles, mientras que otras alrededor de 1446 píxeles. Aun así, la media se encuentra en 483 píxeles. Entender esta variación de dimensiones nos puede ser útil para cuando estemos en la etapa de preprocesamiento de las imágenes, ya que cada imagen deberá tener el mismo alto y ancho antes de ser alimentada a la red y así se podrá entrenar de manera más eficiente al modelo.

7.3.3. Preprocesamiento de las imágenes

Como se comentó en apartados anteriores, antes de alimentar a la red con las imágenes para que las detecte y clasifique, es necesario aplicar un preprocesamiento previo a estas, como puede ser el ajuste de dimensiones de altura y ancho, o un data augmentation [17], que consiste en aumentar el número de muestras que hay en alguno de los conjuntos de datos para así poder entrenar mejor al modelo. En este caso, se aumentarán las muestras del conjunto de entrenamiento, y para ello se creará un generador de imágenes que modificará las imágenes originales creando nuevas que permitan entrenar al modelo.

Para la programación de la red neuronal, se van a fijar unas dimensiones para todas las imágenes IRM que se alimenten a la red. Estas dimensiones serán de 200 píxeles de alto por 200 píxeles de ancho (200x200). Es importante destacar, que cuanto mayores dimensiones se escojan para las imágenes, mayor cantidad de información se retendrá en la red, pero mayor tiempo se tardará en entrenar al modelo. Por otro lado, cuanto menores dimensiones tengan las imágenes, mayor cantidad de información se perderá por la red, pero se tardará menos en entrenar al modelo.

Además, es importante establecer previamente el tamaño del lote de imágenes que se alimentaran a la red para entrenarla. De esta forma, el modelo se actualizará cada vez que un lote pase por la red. Está actualización es inmediata. Cuanto más pequeño sea el tamaño del lote, más largo será el proceso de entrenamiento, ya que habrá mayor número de procesos de optimización que realizar en el modelo. Aun así, si el lote es más pequeño se podrán evitar mayor número de cálculos complicados a la vez. Para la red neuronal convolucional que se estudia en este trabajo se establece un numero de lote de 100.

Una vez fijados los parámetros de dimensión de imágenes y tamaño de lote, se pasa a aplicar el data augmentation. Al no tener las imágenes suficientes para trabajar en la carpeta de entrenamiento, se crearán datos artificiales a partir de las imágenes originales. El objetivo de usar data augmentation es enseñarle al modelo a no solo utilizar las imágenes originales, sino también a estudiar a imágenes modificadas. De esta manera, si se usara una base de datos distinta a la que se va a usar en este trabajo, la red podría seguir detectando y clasificando imágenes como tumores, siendo estas imágenes nuevas para la red.

Los resultados de aplicar el data augmentation en las imágenes pueden ser en forma de imagen volteada, imagen rotada, ampliación/reducción de imagen, imagen recortada, etc. Así, cuando el modelo estudia las imágenes originales y modificadas con el data augmentation, se consigue tener una red eficiente y mejorada.

Para aplicar esta técnica, se usa el generador de datos de imagen, que proporciona Keras y sus paquetes correspondientes, y en este trabajo se aplicaran los siguientes atributos a las imágenes originales [14]:

- Escalar el valor de píxel dividiéndolo por 255, usando la función *rescale* de R
- Voltear la imagen horizontalmente, usando la función *horizontal_flip* de R
- Voltear la imagen verticalmente, usando la función *vertical_flip* de R
- Rotación de la imagen desde 0 a 45 grados, usando la función *rotation_range* de R
- Zoom in o zoom out en un 25% de la imagen original, usando la función *zoom_range* de R
- Usar el 20% de los datos como conjunto de datos de validación, usando la función *validation_split* de R

Después de haber configurado el generador de datos de imagen con los atributos expuestos arriba, se pueden insertar los datos en forma de imagen en el generador con la función *flow_images_from_directory* de R [14]. Debido a que la ubicación de guardado de datos se encuentra dentro la carpeta de Training, que a su vez se encuentra en la carpeta dataset, el directorio deberá configurarse como: dataset/training.

La red estará configurada para detectar los 3 canales de color o RGB, el tamaño de lote y las dimensiones de imagen serán las comentadas anteriormente, es decir, 100 imágenes de lote y 200x200 pixeles, respectivamente. Asimismo, se usará una semilla aleatoria igual a 123, usando *random seed* en R. Por lo tanto, en este proceso, se obtendrá el data augmentation tanto de los datos de entrenamiento como los de validación [14].

Tras aplicar todas estas características a las imágenes, se obtiene la siguiente información:

```
Found 573 images belonging to 4 classes.
```

Esto quiere decir que, dentro de la carpeta de entrenamiento, se han conseguido introducir 573 imágenes nuevas, a partir de las originales, a las que se les ha aplicado una serie de modificaciones, de tal forma que aumente el número de muestras a alimentar a la red.

A continuación, se busca encontrar la siguiente información de utilidad en la base de datos [14]:

- El número de muestras a entrenar, usando la función *train_image_array_gen$n* en R
- El número de muestras de validación, usando la función *val_image_array_gen$n* en R
- El número de clases a clasificar por la red, usando *n_distinct* en R
- Una tabla que muestre la proporción entre clases (frecuencia), usando *table()* en R. Esta tabla se muestra a continuación.

```
Frequency
        0         1         2         3
0.2500000 0.2488654 0.2507564 0.2503782
```

Figura 22: Tabla con la frecuencia de clases

Se puede apreciar que ahora, tras aplicar la técnica de data augmentation, se ha conseguido igualar la proporción de las 4 clases que la red clasificará. El 0 representa la clase de glioma, el 1 la clase de meningioma, el 2 la clase de no tumor y el 3 la clase de pituitario.

7.3.4. Arquitectura del modelo

Como ya se ha comentado en múltiples ocasiones, para construir el modelo se usará una red neuronal convolucional (CNN). El principal beneficio de usar imágenes como matrices en dos dimensiones o capas convolucionales es que se podrán extraer características muy concretas de las imágenes, como la localización del tumor, su tamaño, la clase de tumor, etc.

Para empezar, se creará un modelo simple o red CNN con los siguientes atributos [14]:

- Una capa convolucional que extraiga características de imágenes en forma de matrices o vectores en 2D con una función de activación ReLu. Se usará el código *activation="relu"* de R.
- Una capa de max pooling para reducir la resolución de los mapas de características que vayan saliendo de las capas convolucionales. Se usará la función *layer_max_pooling_2d* de R.
- Una capa de aplanamiento o flattening para aplanar los datos de una matriz 2D a una 1D, así, las capas densas y de salida de la red puedan procesarlos. Se usará la función *layer_flatten()* de R.
- Una capa densa que capture la información de la capa de aplanamiento. Se usará la función *layer_dense()* de R.
- Una capa de salida densa con una función de activación *softmax.* Se usará la función *layer_dense()* de nuevo en R.

Es importante no olvidar fijar el tamaño de la entrada en la primera capa convolucional. Además, si las imágenes IRM tuvieran 3 canales de color, como es el caso de este trabajo, habría que establecerlo a través del código *c()* en R [14].

Tras aplicar en RStudio los anteriores atributos a la red, se obtiene la siguiente información:

```
Model: "simple_model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 200, 200, 16)      448
 max_pooling2d (MaxPooling2D) (None, 100, 100, 16)     0
 flatten (Flatten)           (None, 160000)            0
 dense (Dense)               (None, 16)                2560016
 Output (Dense)              (None, 4)                 68
=================================================================
Total params: 2,560,532
Trainable params: 2,560,532
Non-trainable params: 0
_________________________________________________________________
```

Figura 23: Tabla representativa de los parámetros que forman la red neuronal inicial

En la tabla superior, se aprecian en la parte izquierda las distintas capas creadas con sus correspondientes parámetros.

En primer lugar, a la capa convolucional se le alimentan imágenes de 200x200 pixeles, que es justo el ancho y altura que se fijó en un principio a las imágenes IRM en el preprocesamiento. Esta capa dispone de un kernel de 3x3 (matriz), que permitirá extraer los 16 mapas de características de las imágenes de entrada a la red. Después, se reduce la muestra que ha salido de la capa convolucional, mediante la capa de max pooling en la que se toma el valor máximo para cada área de agrupación de 2x2, parámetro que se fija en RStudio. Ahora, la muestra que sale de la capa de max pooling dispone de 100x100 píxeles, y se siguen teniendo 16 mapas de características. Tras esto, se aplana la muestra de salida de la capa de max pooling, que es un vector de 2D (matriz), mediante la capa de flattening, y pasa a ser un vector de 1D con 160,000 nodos (100x100 píxeles x 16 mapas de características). A continuación, se puede extraer más información de las imágenes con la capa densa que pasará a tener 16 nodos. Por último, se pasa a la capa de salida densa que, a partir de la función de activación softmax, se consiguen clasificar las 3 clases de tumores en caso de existencia de tumor y el no tumor en caso de ausencia de tumor. Es por eso que se tienen 4 nodos de salida en la última capa densa, como se aprecia en la tabla superior. La salida que se proporciona en la última capa viene en forma de probabilidad a ser clasificado en alguna de las 4 clases.

7.3.5. Ajuste del modelo

Después de establecer la arquitectura del modelo que se usará y antes de ajustar el modelo, se compilará este especificando la función de perdidas o costes y su optimización [8] [9].

Para clasificaciones como en las que se realizan en este trabajo en las que se tienen múltiples clases, se usa como función de costes la entropía cruzada categórica. Esta función se basa en que una muestra únicamente puede pertenecer a una clase de entre todas las que haya, y el modelo debe decidir cuál es. Además, la función de entropía cruzada categórica está diseñada para cuantificar la diferencia entre dos distribuciones de probabilidad.

Para la optimización, se utiliza un optimizador Adam con una tasa de aprendizaje del 0.005 [8]. Como se comentó en capítulos anteriores, un optimizador se usa en redes neuronales

para optimizar los valores de los parámetros de la red y así, reducir los errores cometidos por esta. El proceso para conseguir esto es el “backpropagation” [18]. Los optimizadores más básicos lo que hacen es ir actualizando los parámetros de forma equitativa basándose en una tasa de aprendizaje, fijada previamente. El optimizador Adam, que se usará en la red CNN de este trabajo, se puede utilizar en sustitución del algoritmo de descenso de gradiente para actualizar los pesos de red de forma iterativa en función de los datos de entrenamiento. De entre los principales factores por los que se ha escogido usar el optimizador Adam de entre otros destacan su fácil implementación, su eficiencia computacional y su necesidad de poca memoria.

A parte de la función de costes y el optimizador, se fijarán un numero de epochs igual a 20. Un epoch se define como el ciclo completo a través del conjunto de datos de entrenamiento. Nos indica la cantidad de pasadas que hace la red durante el entrenamiento. También se conocen como iteraciones del algoritmo [19].

Tras compilar el modelo se obtiene la siguiente gráfica:

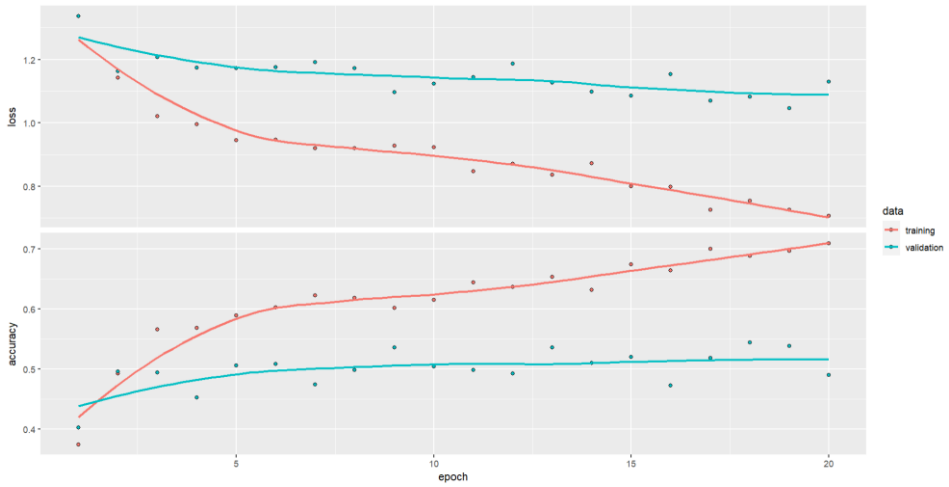

Figura 24: Representación de la función de costes y eficiencia del modelo frente al número de pasadas que realiza el modelo *model*

Se puede apreciar en la gráfica superior, que al aumentar el número de pasadas o epochs disminuyen las perdidas o costes del modelo y aumenta la eficiencia. Esto es debido a que la red a medida que se le alimentan imágenes consigue evitar los errores que había cometido antes y mejora los resultados obtenidos al detectar y clasificar los tumores. Esto es lo que se conoce como back propagation. En la gráfica, se ve que se analizan las pérdidas y eficiencia tanto en los datos de entrenamientos como en los de validación. Además, se puede apreciar en la figura superior que la precisión de los datos de entrenamiento es mucho mayor que la de los de validación, y las perdidas en los datos de entrenamiento son menores que las de los de validación.

7.3.6. Evaluación del modelo

Para ser evaluado el modelo creado, se obtendrá la matriz de confusión utilizando los datos de validación del generador.

En primer lugar, habrá que adquirir el nombre de archivo (filename) de las imágenes que se utilizan como datos de validación. Del nombre de archivo, se extrae la etiqueta categórica como la etiqueta o categoría real de la variable de destino. Para ello, se usarán las funciones *data.frame* y *mutate* de R [14]. A continuación, se muestran las 10 primeras etiquetas de la lista de etiquetas categóricas extraídas de los datos de validación.

	file_name	class
1	dataset/training/glioma_tumor\\gg (1).jpg	glioma_tumor
2	dataset/training/glioma_tumor\\gg (10).jpg	glioma_tumor
3	dataset/training/glioma_tumor\\gg (100).jpg	glioma_tumor
4	dataset/training/glioma_tumor\\gg (101).jpg	glioma_tumor
5	dataset/training/glioma_tumor\\gg (102).jpg	glioma_tumor
6	dataset/training/glioma_tumor\\gg (103).jpg	glioma_tumor
7	dataset/training/glioma_tumor\\gg (104).jpg	glioma_tumor
8	dataset/training/glioma_tumor\\gg (105).jpg	glioma_tumor
9	dataset/training/glioma_tumor\\gg (106).jpg	glioma_tumor
10	dataset/training/glioma_tumor\\gg (107).jpg	glioma_tumor

Figura 25: 10 primeras filas del dataframe creado con los datos de validación

Como se puede ver las 10 primeras etiquetas son de gliomas.

Ahora, mediante RStudio se convertirán las imágenes en matrices. Dado que la dimensión de entrada de las imágenes en el modelo CNN es de 200x200 píxeles con 3 canales de color (RGB), también habrá que convertir las imágenes dentro de los datos de validación a las dimensiones y canales de color correspondientes. La razón por la que se convierten las imágenes de forma manual a matrices mediante la función *image_to_array* de R en vez de usar el generador de imágenes es porque se busca realizar una predicción basada en los datos de validación originales de la carpeta de entrenamiento. Si se usará el generador de imágenes, los datos de validación se transformarían directamente por lo que no se reflejaría la imagen original, a diferencia de si realizamos predicciones de forma manual [14].

A continuación, RStudio nos proporciona información relevante de los datos de validación disponibles y de sus dimensiones:

```
[1] 573 200 200   3
```

Los datos de validación están formados por 573 imágenes con dimensiones 200x200 píxeles y 3 canales de color (RGB). Una vez preparados los datos de validación o de Testing, podremos pasar a predecir las etiquetas de cada imagen que entre en el modelo o red CNN que hemos construido.

Para una mejor interpretación de los resultados posteriores, asociaremos cada categoría o clase en forma de numero al nombre de la clase correspondiente, es decir, “1” a “glioma_tumor, “2” a “meningioma_tumor”, “3” a “no_tumor” y “4” a “pituitary_tumor”.

A continuación, se muestran las 10 primeras filas de datos de predicción:

```
[1] "glioma_tumor"     "meningioma_tumor" "meningioma_tumor" "meningioma_tumor"
[5] "meningioma_tumor" "glioma_tumor"     "meningioma_tumor" "meningioma_tumor"
[9] "meningioma_tumor" "glioma_tumor"
```

Figura 26: Primeras 10 predicciones realizadas en el modelo inicial en la clase de meningioma

Ahora, se evalúa el modelo a partir de la matriz de confusión que nos proporciona RStudio con la función *confusionMatrix()* [14]:

```
Confusion Matrix and Statistics

                 Reference
Prediction         glioma_tumor meningioma_tumor no_tumor pituitary_tumor
  glioma_tumor               52               56        3               3
  meningioma_tumor           80               82       23              36
  no_tumor                    4                0       43              12
  pituitary_tumor            29               26       10             114

Overall Statistics

               Accuracy : 0.5079
                 95% CI : (0.4661, 0.5495)
    No Information Rate : 0.288
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.3241

 Mcnemar's Test P-Value : 4.097e-09

Statistics by Class:

                     Class: glioma_tumor Class: meningioma_tumor Class: no_tumor
Sensitivity                      0.31515                 0.5000         0.54430
Specificity                      0.84804                 0.6601         0.96761
Pos Pred Value                   0.45614                 0.3710         0.72881
Neg Pred Value                   0.75381                 0.7670         0.92996
Prevalence                       0.28796                 0.2862         0.13787
Detection Rate                   0.09075                 0.1431         0.07504
Detection Prevalence             0.19895                 0.3857         0.10297
Balanced Accuracy                0.58160                 0.5801         0.75596
                     Class: pituitary_tumor
Sensitivity                          0.6909
Specificity                          0.8407
Pos Pred Value                       0.6369
Neg Pred Value                       0.8706
Prevalence                           0.2880
Detection Rate                       0.1990
Detection Prevalence                 0.3124
Balanced Accuracy                    0.7658
```

Figura 27: Matriz de confusión representativa del modelo inicial

Además, mediante la función *history$metrics$accuracy[]* de R [14], se obtiene la precisión de los datos de entrenamiento:

```
[1] 0.7091488
```

Según los resultados obtenidos en la matriz de confusión superior, encontramos que la precisión de los datos de validación asciende al 50.79%, lo cual demuestra el alto porcentaje de error en la clasificación (49.21%). Por otro lado, la precisión de los datos de entrenamiento a partir de la función *history$metrics$accuracy[]* de R es del 70.91%, lo que nos indica que el modelo está un poco sobre ajustado o también conocido como overfitting. El overffiting ocurre cuando se tienen en los datos de entrenamiento de la red características muy comunes, por lo que nuestra red se aprenderá casos muy particulares y será incapaz de clasificar casos nuevos con los datos de test, por lo que los rendimientos en el entrenamiento y en el test serán muy dispares. Como queremos que el modelo prediga con precisión, se evaluará el modelo con la métrica del valor pos pred (predicción). Este valor aparece en la figura superior como *Pos Pred Value*. En esta métrica, es obvio que la precisión para la clase no_tumor es demasiado alta en comparación con las otras clases. Este valor es de alrededor de 72.8%, mientras que las otras clases están entre 37 y 64%. Si se tienen en cuenta los valores de precisión y predicción, se deberá ajustar el modelo para obtener mejores resultados [14].

Asimismo, de la matriz de confusión podemos sacar conclusiones de interesar. Por ejemplo, se aprecia que el modelo tiene dificultad para clasificar los 3 tipos de tumores. En la fila de

glioma_tumor se clasifican 52 muestras como gliomas, pero 56 como meningiomas, lo que da un error elevado. Para ser más concretos, este error es del 54.4%. En meningioma_tumor clasifica 82 muestras como meningiomas, pero 80 como gliomas, siendo el error de alrededor de 62.9%. Se aprecia que al modelo le cuesta distinguir entre estos dos tipos de tumores, ya que a la hora de detectar y clasificar el no_tumor y el pituitary_tumor, el error de clasificación es mucho menor.

7.3.7. Refinamiento del modelo

El refinamiento del modelo se va dividir en 3 subcategorías: arquitectura del modelo, ajuste del modelo y evaluación del modelo

- **Arquitectura del modelo:** si nos fijamos en el primer modelo que hemos construido, se puede ver cómo podemos extraer una mayor información cuando los datos o imágenes se encuentran en forma de matrices o arrays en 2D. Antes, una única capa convolucional se encargaba de extraer los mapas de características de las imágenes entrantes en la red y posteriormente mediante la capa de max pooling se reducían dichos mapas. Incluso después de aplicarle la reducción, los mapas de características, que tenían unas dimensiones 100x100 píxeles, seguían aportando mucha información relevante de las imágenes antes de pasar por la capa de aplanamiento o flattening. Por lo tanto, el objetivo ahora es añadir al modelo una mayor cantidad de capas para así, extraer aún más características relevantes de las imágenes que se alimentan a la red. A continuación, se muestran todos los atributos que se añadirán al nuevo modelo. A modo de resumen se tendrán hasta 6 capas convolucionales, 6 capas de max pooling, 1 capa de aplanamiento, 3 capas densas y 1 capa densa de salida [14]:

    - 1ª capa convolucional que extraiga características de imágenes en forma de matriz con función de activación ReLu, filtro=mapa de características=16, kernel=5x5

    - Capa de max pooling para reducir la resolución de los mapas de características que vayan saliendo de las capas convolucionales. Área de agrupación 2x2.

    - 2ª capa convolucional que extraiga características de imágenes en forma de matriz con función de activación ReLu, filtro=mapa de características=32, kernel=5x5

    - Capa de max pooling para reducir la resolución de los mapas de características que vayan saliendo de las capas convolucionales. Área de agrupación 2x2.

    - 3ª capa convolucional que extraiga características de imágenes en forma de matriz con función de activación ReLu, filtro=mapa de características=32, kernel=5x5

- Capa de max pooling para reducir la resolución de los mapas de características que vayan saliendo de las capas convolucionales. Área de agrupación 2x2.

- 4ª capa convolucional que extraiga características de imágenes en forma de matriz con función de activación ReLu, filtro=mapa de características=64, kernel=5x5

- Capa de max pooling para reducir la resolución de los mapas de características que vayan saliendo de las capas convolucionales. Área de agrupación 2x2.

- 5ª capa convolucional que extraiga características de imágenes en forma de matriz con función de activación ReLu, filtro=mapa de características=128, kernel=3x3
- Capa de max pooling para reducir la resolución de los mapas de características que vayan saliendo de las capas convolucionales. Área de agrupación 2x2.

- 6ª capa convolucional que extraiga características de imágenes en forma de matriz con función de activación ReLu, filtro=mapa de características=256, kernel=3x3

- Capa de max pooling para reducir la resolución de los mapas de características que vayan saliendo de las capas convolucionales. Área de agrupación 2x2.

- Capa de aplanamiento o flattening para aplanar los datos desde arrays en 2D a arrays en 1D.

- 1ª capa densa que capture información de la capa de aplanamiento con una función de activación ReLu y 64 nodos.

- 2ª capa densa que capture información de la anterior capa densa con una función de activación ReLu y 128 nodos.

- 3ª capa densa que capture información de la anterior capa densa con una función de activación ReLu y 256 nodos.

- Capa densa de salida con función de activación softmax.

Como se vio con anterioridad, para programar las capas convolucionales en R se usa la función *layer_conv_2d*, para programar las capas de max pooling en R se usa la función *layer_max_pooling_2d*, para programar la capa de aplanamiento en R se

usa la función *layer_flatten()*, para programar las capas densa y la capa densa de salida en R se usa la función *layer_dense* [14].

El modelo que se construyó en un principio, con una única capa, se llamó model. Ahora, el nuevo modelo con las 6 capas convolucionales pasa a llamarse model_big.

A continuación, se muestra la estructura final del modelo programado:

```
Model: "sequential"
_________________________________________________________________________________________________
 Layer (type)                                Output Shape                            Param #
=================================================================================================
 conv2d_6 (Conv2D)                           (None, 200, 200, 16)                    1216
 max_pooling2d_6 (MaxPooling2D)              (None, 100, 100, 16)                    0
 conv2d_5 (Conv2D)                           (None, 100, 100, 32)                    12832
 max_pooling2d_5 (MaxPooling2D)              (None, 50, 50, 32)                      0
 conv2d_4 (Conv2D)                           (None, 50, 50, 32)                      25632
 max_pooling2d_4 (MaxPooling2D)              (None, 25, 25, 32)                      0
 conv2d_3 (Conv2D)                           (None, 25, 25, 64)                      51264
 max_pooling2d_3 (MaxPooling2D)              (None, 12, 12, 64)                      0
 conv2d_2 (Conv2D)                           (None, 12, 12, 128)                     73856
 max_pooling2d_2 (MaxPooling2D)              (None, 6, 6, 128)                       0
 conv2d_1 (Conv2D)                           (None, 6, 6, 256)                       295168
 max_pooling2d_1 (MaxPooling2D)              (None, 3, 3, 256)                       0
 flatten_1 (Flatten)                         (None, 2304)                            0
 dense_3 (Dense)                             (None, 64)                              147520
 dense_2 (Dense)                             (None, 128)                             8320
 dense_1 (Dense)                             (None, 256)                             33024
 Output (Dense)                              (None, 4)                               1028
=================================================================================================
Total params: 649,860
Trainable params: 649,860
Non-trainable params: 0
_________________________________________________________________________________________________
```

Figura 28: Tabla representativa de los parámetros que forman la red neuronal final

Como podemos observar en la tabla superior, a la 1ª capa convolucional entran imágenes de 200x200 píxeles, como se fijó en el preprocesamiento, y salen 16 mapas de características de 200x200. Tras esto, se pasa a la capa de max pooling que reducirá las dimensiones de los 16 mapas de características a 100x100 píxeles. Dichos mapas entraran sin modificaciones a la 2ª capa convolucional. Este proceso de convolución y reducción se repetirá 6 veces hasta que, tras pasar la 6ª capa convolucional, se obtengan 256 mapas de características de 3x3.

Después, dichos mapas pasarán a la capa de aplanamiento que lo que hará es transformar las matrices obtenidas en vectores de 1D de 2304 nodos (3x3 píxeles x 256 mapas de características).

Por último, mediante la 1ª capa densa se pasa de 2304 nodos a 64. Este proceso de reducción de nodos se repite hasta llegar a la capa densa de salida que estará formada por 4 nodos de salida. Como se comentó con anterioridad, estos 4 nodos de salida representan las 4 clases que clasificará el modelo. Esto representa muy bien como las redes neuronales trabajan a modo de “embudo”.

- **Ajuste del modelo:** después de la construcción del modelo, se puede compilar el modelo especificando la función de pérdidas y el optimizador a utilizar. Estos dos parámetros son necesarios configurarlos antes de ajustar el modelo. Como se usó en el primer modelo inicial con una única capa convolucional (model), se hará uso de la función de pérdida para clasificaciones multiclase llamada entropía cruzada categórica. Como se comentó con anterioridad, esta función se basa en que una muestra únicamente puede pertenecer a una clase de entre todas las que haya, y el modelo debe decidir cuál es. Además, la función de entropía cruzada categórica está diseñada para cuantificar la diferencia entre dos distribuciones de probabilidad [8] [9].

  Ahora en vez de usar un optimizador Adam con tasa de aprendizaje de 0.005, se usará para el nuevo modelo (model_big) uno Adam con tasa de 0.001. Como se comentó en capítulos anteriores, un optimizador se usa en redes neuronales para optimizar los valores de los parámetros de la red y así, reducir los errores cometidos por esta. El proceso para conseguir esto es el “backpropagation”. Los optimizadores más básicos lo que hacen es ir actualizando los parámetros de forma equitativa basándose en una tasa de aprendizaje, fijada previamente [18].

  El optimizador Adam, que se usará en la red CNN de este trabajo, se puede utilizar en sustitución del algoritmo de descenso de gradiente para actualizar los pesos de red de forma iterativa en función de los datos de entrenamiento. De entre los principales factores por los que se ha escogido usar el optimizador Adam de entre otros destacan su fácil implementación, su eficiencia computacional y su necesidad de poca memoria.

  A diferencia de antes que se usaban 20 epochs, ahora se aumentaran a 60. Como se comentó, un epoch representa el ciclo completo a través del conjunto de datos de entrenamiento y nos indica la cantidad de pasadas que hace la red durante el entrenamiento. También se conocen como las iteraciones del algoritmo [19].

  Además, el modelo será evaluado usando los datos de validación del generador.

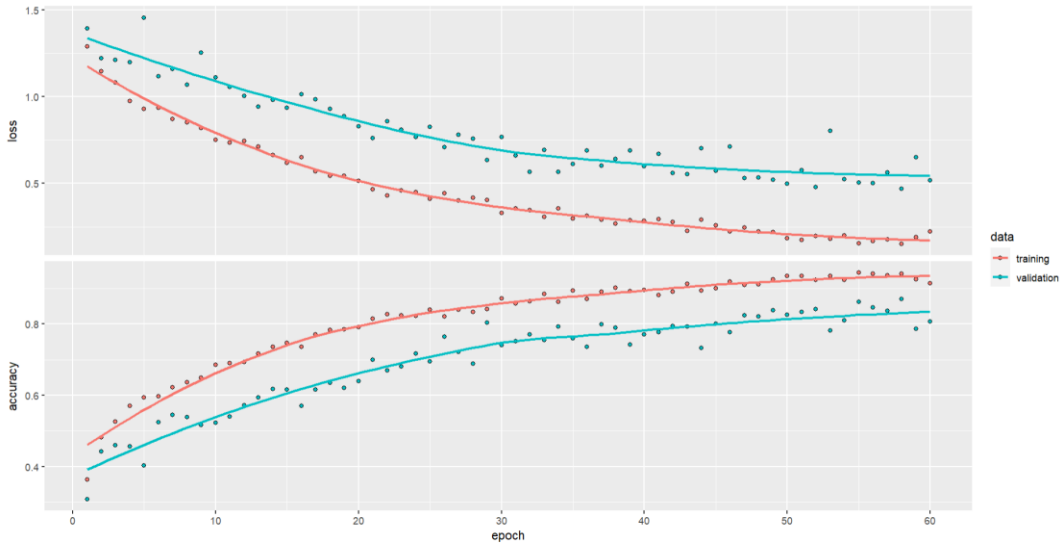

Figura 29: Representación de la función de costes y eficiencia del modelo frente al número de pasadas que realiza el modelo *model_big*

A partir de la figura superior, se puede ver como a mayor número de iteraciones del algoritmo o epochs (eje horizontal), la precisión del modelo aumenta. Esto es debido a que la red, a medida que se le alimentan imágenes, consigue evitar los errores que había cometido antes y mejora los resultados obtenidos al detectar y clasificar los tumores. Esto es lo que se conoce como back propagation. En la gráfica, se ve que se analizan las pérdidas y eficiencia tanto en los datos de entrenamientos como en los de validación. Además, se aprecia en la figura superior como ahora la diferencia precisión y perdidas entre datos de entrenamiento y de validación no es tan acusada como en el modelo inicial (model).

- **Evaluación del modelo:** ahora, mediante el nuevo modelo creado y ajustado, se evaluarán los datos, el modelo afinado y la matriz de confusión de los datos de validación. A continuación, se muestra la información relevante que se obtiene con R:

```
Confusion Matrix and Statistics

                  Reference
Prediction         glioma_tumor meningioma_tumor no_tumor pituitary_tumor
  glioma_tumor              130              12        1              0
  meningioma_tumor           16             128        2              9
  no_tumor                    0               0       72              2
  pituitary_tumor            19              24        4            154

Overall Statistics

               Accuracy : 0.8447
                 95% CI : (0.8124, 0.8734)
    No Information Rate : 0.288
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7878

 Mcnemar's Test P-Value : 3.835e-05

Statistics by Class:

                     Class: glioma_tumor Class: meningioma_tumor Class: no_tumor
Sensitivity                       0.7879                 0.7805          0.9114
Specificity                       0.9681                 0.9340          0.9960
Pos Pred Value                    0.9091                 0.8258          0.9730
Neg Pred Value                    0.9186                 0.9139          0.9860
Prevalence                        0.2880                 0.2862          0.1379
Detection Rate                    0.2269                 0.2234          0.1257
Detection Prevalence              0.2496                 0.2705          0.1291
Balanced Accuracy                 0.8780                 0.8572          0.9537
                     Class: pituitary_tumor
Sensitivity                          0.9333
Specificity                          0.8848
Pos Pred Value                       0.7662
Neg Pred Value                       0.9704
Prevalence                           0.2880
Detection Rate                       0.2688
Detection Prevalence                 0.3508
Balanced Accuracy                    0.9091
```

Figura 30: Matriz de confusión representativa del modelo final

Además, mediante la función *history_big$metrics$accuracy[]* de R, se obtiene la precisión de los datos de entrenamiento [14]:

```
[1] 0.9136364
```

Basándonos en los resultados que se muestran en las tablas superiores, se observa que la precisión de los datos de validación usados en el modelo es de alrededor de 85% (Accuracy), mientras que la precisión obtenida de los datos de entrenamiento del modelo a partir de la función *history_big$metrics$accuracy[]* de R es de 91.4%. Esto nos indica, que el modelo nuevo (model_big) con respecto al anterior de prueba (model) es muy sofisticado y fiable, ya que la discrepancia entre los datos de entrenamiento y de validación no son tan acusados [14].

Para ir más lejos, si se evalúa el modelo a partir de los valores de predicción obtenidos o pos pred value en las tablas superiores, se contempla que para todas las clases se supera el valor de 75%. Esto demuestra la buena precisión en la clasificación de todas las clases.

Como se vio con el modelo inicial de una única capa de convolución, este tenía dificultad para distinguir entre dos de los tumores: glioma y meningioma. Incluso los errores de clasificación superaban el 54%. Si ahora nos fijamos en la nueva matriz de confusión obtenida, los resultados en la clasificación de gliomas y meningiomas ha mejorado exponencialmente. En la fila de gliomas, se aprecia que, de las 143 muestras de gliomas, se clasifican correctamente 120. Esto quiere decir que el modelo clasifica con un acierto del 84%. Por otro lado, en la fila de los meningiomas, se contempla que de las 155 muestras de meningiomas, se clasifican correctamente 128. Esto quiere decir que el modelo clasifica con un acierto del 83%.

Además, se ha encontrado un descenso en el error de clasificación de las otras dos clases no mencionadas: no_tumor y pituitary_tumor. El error en la clasificación de no tumores es casi nulo y en la clasificación del tumor pituitario ha disminuido casi 13%.

7.3.8. Predicción de datos en el conjunto de datos de prueba

Después de haber entrenado al modelo y comprobado su correcto funcionamiento con los datos de validación, ahora, se podrá evaluar con los datos de prueba o testing que obtuvimos en un inicio.

Como ocurría con los datos de entrenamiento, en la carpeta de los datos de Testing en nuestro equipo, encontramos 4 subcarpetas conteniendo las imágenes IRM de cada uno de los tumores a clasificar (glioma_tumor, meningioma_tumor y pituitary_tumor) o de cerebros sin tumor (no_tumor).

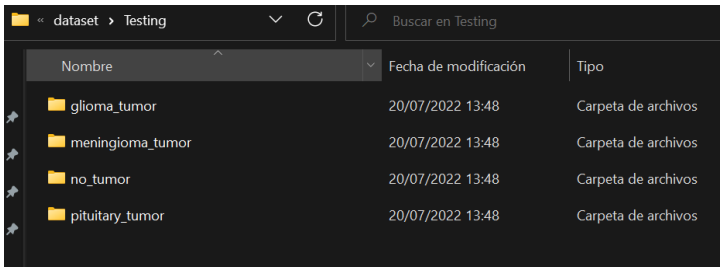

Figura 31: Carpeta de los datos de prueba en el equipo utilizado

Para extraer las imágenes de la carpeta de Testing, se hará lo mismo que con las imágenes de la carpeta de datos de entrenamiento o Training. Se usarán las funciones de R, *paste0(), map()* y *unlist()*. Inicialmente, se realizará la extracción de las imágenes pertenecientes a la carpeta de meningioma_tumor. Posteriormente, se aplicará el mismo proceso al resto de carpetas [14].

A continuación, se observan los nombres de archivo de las 6 primeras imágenes dentro de la carpeta de meningioma_tumor en los datos de prueba:

```
[1] "dataset/testing/meningioma_tumor/image(1).jpg"
[2] "dataset/testing/meningioma_tumor/image(10).jpg"
[3] "dataset/testing/meningioma_tumor/image(100).jpg"
[4] "dataset/testing/meningioma_tumor/image(102).jpg"
[5] "dataset/testing/meningioma_tumor/image(106).jpg"
[6] "dataset/testing/meningioma_tumor/image(107).jpg"
```

Figura 32: 6 primeros archivos de la carpeta de meningiomas en los datos de prueba

Utilizando la función de R, *length(meningioma_tumor_file_name)*, se proporciona el número de datos que hay en la carpeta correspondiente a meningioma_tumor. En este caso, ese número es 115 imágenes IRM de tumores con meningiomas.

El siguiente paso a realizar, es convertir los datos extraídos, en este caso los datos pertenecientes a imágenes de meningiomas, en imágenes en forma de matriz o vectores en 2D. Para ello, se hará uso de la función *image_prep()* de R. Mediante la función *dim()*, utilizada anteriormente, se podrá conocer las dimensiones de las matrices y sus canales de color [14].

```
[1] 115 200 200   3
```

La función de R, *dim()*, nos proporciona la información superior. El 115, representa el número de matrices o imágenes en forma de vector 2D que disponemos, el 200x200, es el número de pixeles de las matrices y el 3, el número de canales que tienen las imágenes, en este caso canal RGB (red-green-blue).

Dado que ya se han preparado los datos de prueba con los que trabajaremos en el modelo, ahora, haciendo uso del model_big construido, ajustado y evaluado en los apartados anteriores, predeciremos las etiquetas de cada imagen de los datos de prueba. Se usa el model_big y no el model inicial, ya que el modelo con 6 capas convolucionales dispone de mayor precisión de predicción en la clasificación de los tumores, como se vio en el apartado de refinamiento del modelo.

Posteriormente, se evaluará la frecuencia de cada etiqueta obtenida en la predicción realizada, para analizar la precisión de los resultados.

Para realizar la predicción de las etiquetas de cada imagen entrante al modelo, se usará la función de R, *model_big %>% predict()%>%k_argmax()*. Después, mediante la función de R, *data.frame()*, se creará un marco de datos con las imágenes de meningioma_tumor, extraídas con anterioridad [14].

A continuación, se muestran las 10 primeras etiquetas del data frame creado. Se puede ver como alguna imagen perteneciente en un inicio a la carpeta de meningioma_tumor, la clasifica o etiqueta como no_tumor.

	id	label
1	image(1).jpg	meningioma_tumor
2	image(10).jpg	meningioma_tumor
3	image(100).jpg	meningioma_tumor
4	image(102).jpg	meningioma_tumor
5	image(106).jpg	meningioma_tumor
6	image(107).jpg	meningioma_tumor
7	image(109).jpg	no_tumor
8	image(11).jpg	meningioma_tumor
9	image(112).jpg	meningioma_tumor
10	image(113).jpg	meningioma_tumor

Figura 33: 10 primeras predicciones realizadas del modelo final en la clase meningioma_tumor

Para finalizar con las predicciones de meningioma_tumor, se obtiene una tabla que muestra la frecuencia con las que se predice cada clase dentro del data frame de meningiomas.

	label	freq
	<chr>	<int>
1	glioma_tumor	1
2	meningioma_tumor	107
3	no_tumor	5
4	pituitary_tumor	2

Figura 34: Tabla de confusión del modelo final en la clase meningioma_tumor

La interpretación de esta tabla es la siguiente: de las 115 imágenes existentes en la subcarpeta meningioma_tumor en la carpeta de Testing, se clasifican correctamente 107 muestras, es decir, hay una clasificación con acierto del 93%. De estas 115 imágenes de meningiomas se clasifican incorrectamente 8 muestras, 1 como glioma, 5 como no tumor y 2 como pituitario, es decir, existe un error en la clasificación del 7% aproximadamente.

Una vez se conocen los resultados en la clasificación de meningiomas, se podría repetir el proceso anterior para las otras 3 clases restantes. Así, se podrá analizar como de bien clasifica el modelo creado para cada una de las clases. Se podrá ver también, cual es la clase con la que se suele equivocar más al clasificar.

Si ahora analizamos como de bien clasifica el modelo los tumores glioma de la carpeta de datos de prueba, glioma_tumor, obtenemos la siguiente tabla de clasificación:

	label	freq
	<chr>	<int>
1	glioma_tumor	21
2	meningioma_tumor	43
3	no_tumor	27
4	pituitary_tumor	9

Figura 35: Tabla de confusión del modelo final en la clase glioma_tumor

Se puede ver como el modelo ahora encuentra dificultad para detectar y clasificar a los gliomas. De las 100 muestras pertenecientes a glioma_tumor en los datos de prueba, el model_big solo es capaz de clasificar 21 gliomas. Esto representa un error de clasificación muy elevado, entorno al 79%. Además, se aprecia que el modelo suele confundir gliomas con meningiomas. De los 100 gliomas, clasifica como meningioma a 43 de ellos. También suele confundirse gliomas con muestras que no representan tumor. Esto es raro, ya que cuando hay existencia de tumor en las imágenes, claramente se ve en ellas, y el modelo tendría que ser capaz de distinguir una forma irregular, que no se vería si la muestra no presentará tumor.

Este análisis puede llegar a ser un problema, ya que, si en un futuro se implementara, a nivel real en hospitales y clínicas, un sistema de detección y clasificación de tumores a partir de redes neuronales convolucionales como él descrito en este trabajo, la baja precisión en la clasificación de alguno de los tumores como él que se aprecia en la tabla superior, podría llegar a confundir a médicos y pacientes.

Analizando ahora la clasificación que realiza el modelo de los datos de no_tumor en los datos de prueba, se observa la siguiente tabla de clasificación:

	label	freq
	<chr>	<int>
1	glioma_tumor	4
2	meningioma_tumor	4
3	no_tumor	97

Figura 36: Tabla de confusión del modelo final en la clase no_tumor

Ahora, si analizamos la tabla superior, se contempla la precisa clasificación que consigue el modelo con las muestras de no_tumor. De las 105 muestras de no tumores que se tienen, 97 son clasificadas correctamente, mientras que únicamente 8 incorrectamente. Esto demuestra que el modelo consigue clasificar de forma correcta el 92% de las veces, mientras que se equivoca el 8%. Asimismo, el modelo no confunde ninguna muestra como tumor pituitario, lo que da a pensar que los tumores pituitarios tienen características muy específicas comparados con los otros dos tipos de tumores, glioma y meningioma.

Por último, pasaremos a ver la clasificación que realiza el modelo del tumor pituitario:

	label	freq
	<chr>	<int>
1	meningioma_tumor	27
2	no_tumor	9
3	pituitary_tumor	38

Figura 37: Tabla de confusión del modelo final en la clase pituitary_tumor

Se puede ver que el modelo creado no se adapta del todo bien al clasificar tumores pituitarios. De los 74 tumores pituitarios que encontramos en la carpeta pituitary_tumor en los datos de prueba, clasifica correctamente 38 de ellos. Esto representa una clasificación con acierto del 51%. Como se comentó con anterioridad, esta precisión no es suficiente como para implementar estos sistemas de redes CNN en hospitales y clínicas médicas, ya

que, en la mayoría de países desarrollados la precisión con la que se detectan estos tumores de forma manual es mucho mayor.

7.4. Resultados, análisis y discusión

Tras la programación de la red neuronal convolucional y su posterior evaluación, se puede decir que el objetivo de crear un buen modelo CNN se ha cumplido. A pesar de ello, y tras analizar con detalle las tablas de clasificación de cada una de las clases con respecto a los datos de prueba, el modelo creado podría mejorarse con el objetivo de obtener mayor precisión en la clasificación.

Como se ha visto, tras entrenar a la red con los datos de entrenamiento y de validación, se pasa a predecir las etiquetas de las imágenes pertenecientes a los datos de prueba. En la clase de no_tumor se aprecia que la clasificación es muy satisfactoria, con un error de clasificación de alrededor del 8%. Lo cual demuestra que el modelo es muy seguro en la detección de tumores. No obstante, en la clasificación del tumor, una vez detectado, en algunas clases, como la del glioma o del pituitario, el modelo suele confundirse con un elevado porcentaje de error. En el caso del glioma, el error era de alrededor del 79%, y en el pituitario del 51%. Estos valores son excesivamente elevados, lo cual impediría que el modelo construido pudiera implementarse a nivel laboral en hospitales o centros médicos, ya que, a pesar, de que ahorran mucho tiempo a los doctores y automatizan mucho el proceso de detección y clasificación, no clasifican con la precisión suficiente. En la mayoría de países desarrollados con médicos especializados en radiología, capaces de detectar y clasificar tumores, la precisión a la hora de decidir el tipo de tumor de los pacientes es mayor que la que se ha obtenido en algunas clases con el modelo creado. Es por ello que actualmente, el modelo creado no estaría listo para implementarse y debería ser mejorado.

Una de las formas más típicas con las que se podría mejorar el modelo, sería añadir un mayor número de capas convolucionales que permitan filtrar de forma más sofisticada las imágenes que se alimentan a la red. De esta forma, añadiendo más capas, se crearían un mayor número de mapas de características que permitirían obtener características más complejas que las de las capas anteriores, llevando a una clasificación más profunda y, por tanto, más precisa. De esta manera, los porcentajes de error obtenidos se verían reducidos notablemente.

Otras formas que también se usan habitualmente para perfeccionar modelos basados en redes neuronales convolucionales son las siguientes:

- Modificación de los kernels o filtros usados entre capas convolucionales. Los kernels son fundamentales para obtener los mejores mapas de características posibles para que nuestro modelo clasifique con mayor precisión. El kernel más comúnmente utilizado es el 3x3, ya que, tras aplicarlo a una entrada en forma de matriz, obtendrá una salida también de 3x3. Otros kernels como el 2x2 o el 4x4, no generan salidas con las mismas dimensiones (por ejemplo, el kernel 2x2 genera una salida de 4x4), por lo que aumentaran el tiempo de procesamiento del modelo. Usando 3x3 como kernel se establece un tiempo de procesamiento constante. Es por ello, que recomendaría que el modelo se basará en usar kernels entre capas de estas dimensiones (3x3) [13].
- Reducción del overfitting. Muchas veces la existencia de altos porcentajes de error en la clasificación en redes neuronales convolucionales viene dada por el fenómeno de overfitting o sobreajuste. Esto ocurre cuando entrenamos al modelo con

características específicas de los datos de entrenamiento, y por tanto cuando el modelo trabaja con los datos de prueba o testing no es capaz de detectar características nuevas diferentes a las de entrenamiento. Para solucionar esto debemos encontrar un punto medio en el aprendizaje de nuestro modelo sin incurrir en el overfitting. Para ello, como hemos realizado en los primeros apartados de la programación del modelo en R, debemos subdividir el conjunto de datos de entrenamiento en dos: entrenamiento y validación. Normalmente, esta división suele ser del 20% en validación y 80% en entrenamiento. El subconjunto de validación deberá tener muestras diversas en lo posible y una cantidad de muestras suficiente para poder comprobar los resultados una vez entrenado el modelo. Para lograr que el modelo mejore los resultados anteriores, se irá revisando y contrastando el entrenamiento con el conjunto de validación y su tasa de errores, hasta dar buenas predicciones y evitar el problema de overfitting [16].

- Mejorar el preprocesamiento que se le realiza a las imágenes a utilizar. Si se aumentará el tamaño de las imágenes que se alimentan a la red, se conseguiría mejorar la retención de información por parte de esta. Si se consigue retener más información, la red obtendría mayor número de características de las imágenes y por tanto clasificaría mejor, reduciendo los elevados errores obtenidos en los apartados anteriores. La única desventaja de aumentar las dimensiones de las imágenes sería el aumento de tiempo de procesamiento del modelo. En un inicio, con las imágenes de 200x200 píxeles el modelo tardaba en procesar 53 minutos. Si aumentamos las dimensiones no cambiaría mucho el tiempo, y los resultados mejorarían drásticamente, por lo que saldría beneficioso.

8. CONCLUSIONES

El objetivo de este trabajo era construir una red neuronal convolucional encargada de detectar y clasificar tumores cerebrales a partir de imágenes de resonancias magnéticas. Por consiguiente, se ha conseguido crear una red que no solo clasifica entre pacientes sanos y no sanos, sino que también es capaz de clasificar entre 3 tipos diferentes de tumores en pacientes con existencia de tumor. Para ello se ha programado un modelo basado en Deep Learning a partir de la herramienta de programación de R, y en la mayoría de los casos se ha obtenido unos resultados satisfactorios.

Como conclusión principal, cabe destacar que, la red neuronal convolucional que se ha creado en este trabajo consigue satisfactoriamente distinguir entre imágenes de pacientes sanos y no sanos, sin embargo, una vez se detecta la presencia del tumor, el modelo encuentra cierta dificultad a la hora de clasificar entre las 3 clases de tumor distintas. Es por ello, que como el principal objetivo del trabajo era proporcionar a países en desarrollo con una solución para detectar y clasificar distintos tipos de tumores, el modelo creado no llegaría a ser apto en su totalidad. Sería necesario aplicar una serie de mejoras, como las que se describen en el apartado 7.4, que aumentarán los rendimientos en la distinción y clasificación de alguno de los tumores, cómo, por ejemplo, entre el glioma y el meningioma.

Asimismo, como consecuencia de lo anterior comentado, la implementación en el ámbito medico de una red neuronal convolucional como la que se ha creado en este trabajo, aún no es del todo viable, debido a su alto porcentaje de error en la clasificación de algunas clases. A pesar de que la detección de tumores consigue llevarse a cabo con gran exactitud y por tanto podría beneficiar a muchos países en desarrollo, se considera que el principal aspecto diferencial del modelo es su capacidad de clasificación entre tumores, y al no ser aun del todo óptima en esta parte, no se aconseja su puesta en servicio.

Tras haber trabajado con datos de 3 clases de tumores cerebrales, considero que, para que el modelo hubiera tenido mejor rendimiento en la clasificación, hubiera sido conveniente buscar una mayor base de datos con muchos otros tipos de tumores cerebrales, como pueden ser astrocitomas (tipo de glioma), glioblastomas (tipo de glioma) o linfomas, entre otros. De esta manera, y junto con una mayor cantidad de capas convolucionales, capas de max pooling, capas de aplanamiento y capas densas, la red neuronal podría obtener una mayor cantidad de características específicas típicas de cada clase con las que poder distinguir mejor.

En mi opinión, la necesidad de aumentar el número de clases que el modelo puede clasificar es fundamental, ya que, aunque la base de datos utilizada únicamente tenga imágenes de las 4 clases mencionadas, si se buscará implantar la red, con el entrenamiento de esas 4 clases, a nivel real en hospitales y centros médicos, se produciría el fenómeno de underfitting, lo cual impediría que funcionará correctamente. El underfitting, ocurre cuando se entrena al modelo con una serie de clases de tumor en concreto y se alimenta a este con una imagen nueva de una clase de tumor distinta a con las que se ha entrenado al modelo. Por tanto, la red fallará al clasificar la nueva imagen, por falta de muestras [16].

En el caso, de que se quiera mantener la base de datos con las 4 clases a detectar, considero que, las principales mejoras que se deberían aplicar al modelo son las comentadas en el apartado anterior, y son la reducción del fenómeno del overfitting debido a la diferencia de valores entre datos de entrenamiento y de prueba, la modificación de los kernels o filtros empleados entre capas convolucionales y la mejora del preprocesamiento de las imágenes previamente a ser alimentadas a la red. Estas mejoras no solo mejorarán el rendimiento de nuestro modelo en concreto, sino también en cualquier otro modelo, distinto

al desarrollado en este trabajo, basado en redes neuronales. El problema del overfitting suele ser muy común en el campo del Deep Learning, por lo que siempre deberá buscarse su reducción.

Por último, considero que las redes neuronales convolucionales y el Deep Learning en general, tienen un amplio rango de aplicación en el mundo de la medicina, y en un futuro a corto plazo, pueden llegar a proporcionar soluciones muy ventajosas en todo el mundo. Mencionar además que, modelos basados en redes neuronales convolucionales como el creado en este trabajo podrían adaptarse para detectar y clasificar otro tipo de enfermedades neurodegenerativas, como el Alzheimer o Parkinson, entre otras.

9. LINEAS FUTURAS

- Aplicación de la red neuronal convolucional desarrollada en este proyecto al diagnóstico de enfermedades distintas al tumor cerebral, como puede ser el Alzheimer o el Parkinson. Si se obtuviera una base de datos con imágenes de cerebros de pacientes con posibles indicios de padecer alguna de estas enfermedades neurodegenerativas, la red creada podría detectarlas al igual que ha hecho con los tumores.

- Implementación del sistema construido en el proyecto en hospitales y centros médicos de países en desarrollo con médicos con falta de experiencia en la detección y clasificación de tumores cerebrales. De esta manera, los médicos con pocos conocimientos en radiología, podrán de forma automática y precisa, localizar y clasificar el tumor cerebral en cuestión.

- Mejora futura de la precisión del modelo creado. Para ello, habrá que construir una red neuronal convolucional más compleja que la de este proyecto, siendo necesario un programador con mayor experiencia y conocimientos en el lenguaje de programación de R.

- Uso de otros lenguajes de programación aparte de R, como Python. De esta manera, se podrán aprovechar las ventajas que presente Python, y otros lenguajes más avanzados, para mejorar la red y su porcentaje de acierto en la clasificación.

- Posibilidad de contar con un mejor hardware que permita mejorar la memoria GPU del equipo usado y optimice el tiempo de procesamiento en la simulación de la red neuronal.

- Aumento de datos que mejore el entrenamiento del modelo y por consiguiente mejore los resultados obtenidos tras la simulación. Asimismo, este aumento de datos convendría que fuera de clases nuevas de tumores, distintas a las 3 que ya se han utilizado para entrenar al modelo actual, de esta forma se podría evitar problemas que han ido surgiendo en la programación de la red, como el overfitting o el underfitting [16].

10. GESTIÓN DEL PROYECTO

10.1. EDP

La EDP (Estructura de Descomposición del Proyecto) tiene como objetivo recoger todas las operaciones necesarias para llevar a cabo el proyecto, desde la idea inicial hasta la finalización del mismo.

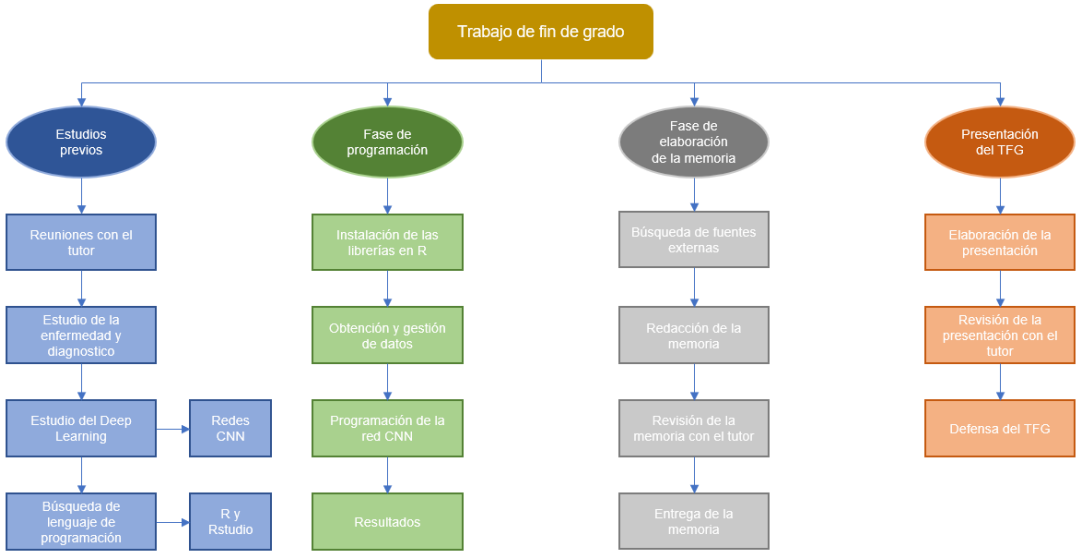

Figura 38: Estructura de Descomposición del Proyecto (EDP)

Dentro de la planificación, destacamos 4 fases importantes:

- En la fase de estudios previos, se recogen las actividades para la determinación del proyecto en sí mismo; que se pretende hacer, si se disponen de los recursos necesarios para llevarlo a cabo, cuanto tiempo va a durar, cuanto va a costar, alcance del proyecto, etc. Esta fase marca si el proyecto es viable o no. Es una parte fundamental del proyecto. En esta fase, asimismo, se realizan reuniones con el tutor y se desarrollan investigaciones previas del tema a tratar, es decir, se investiga sobre los tumores cerebrales y sus posibles diagnósticos, se estudia el campo del Deep Learning y las posibles redes neuronales a usar en el proyecto, y se elige entre distintas posibilidades de lenguajes de programación.

- En la fase de programación, y una vez se decide seguir adelante con el proyecto tras la fase de estudios previos, nos centramos en la construcción de la red neuronal convolucional del proyecto. Para ello, previamente se instalaron los paquetes necesarios para poder trabajar con R y RStudio, y además se buscaron los datos que se usaron para evaluar el problema planteado en el proyecto. Finalmente, tras la simulación de la red, se analizan los resultados obtenidos, los cuales nos permiten obtener unas conclusiones finales del proyecto.

- En la fase de elaboración de la memoria, en un principio, se realiza una investigación de fuentes externas para poder contrastar los resultados obtenidos en la fase anterior. Por consiguiente, se pasa a escribir la memoria en base a todo lo anterior. Entre medias, se realizan revisiones con el tutor.

- En la fase de presentación del TFG, al tratarse de un trabajo de fin de grado, una vez finalizada y entregada la memoria, se desarrolla una presentación oral que se expone delante de un tribunal.

10.2. Diagrama de Gantt

El diagrama de Gantt tiene por objeto conocer las relaciones y dependencias entre las distintas actividades del proyecto. Junto con al EDP, el diagrama de Gantt es una herramienta muy extendida en la elaboración de proyectos. Este se suele realizar después de haber establecido la EDP.

En el diagrama de Gantt, se muestran las duraciones y dependencias de las actividades definidas previamente en la EDP. Las actividades están representadas en el eje vertical del diagrama. Además, se ha continuado con el código de colores establecido en el apartado de la EDP, para distinguir las diferentes fases del proyecto.

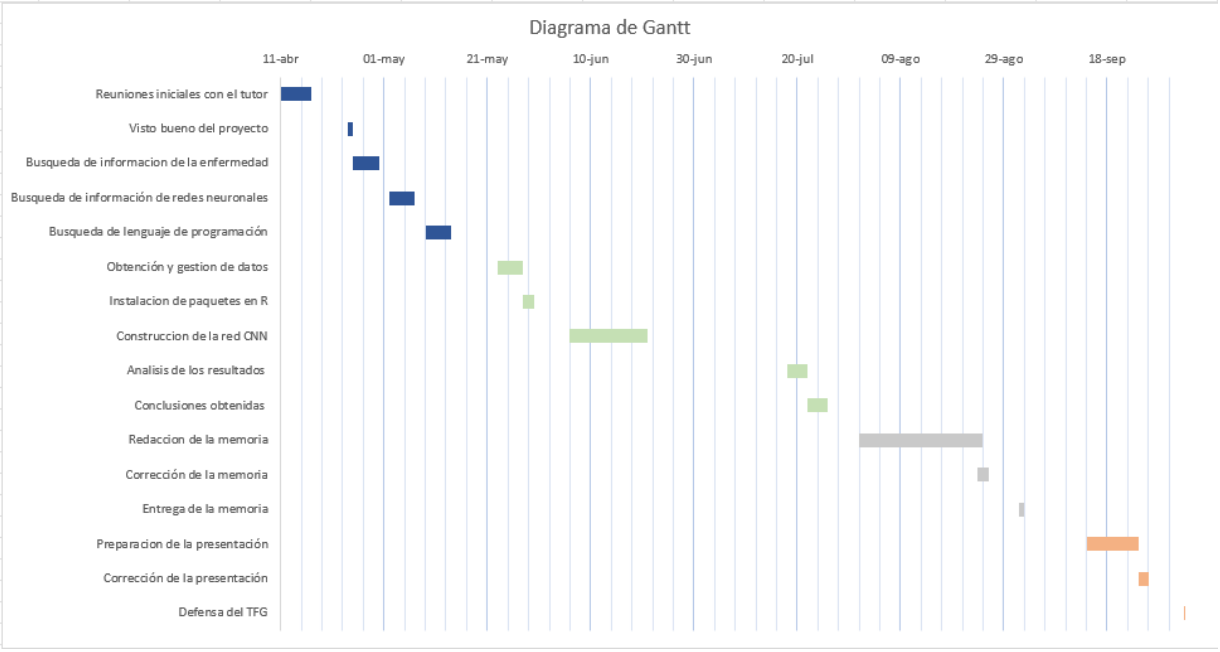

Figura 39: Diagrama de Gantt del TFG

Como se puede observar, el proyecto se comenzó con las reuniones iniciales con el tutor el 11 de abril de 2022 y se finalizó con la defensa del TFG el 3 de octubre de 2022.

El diagrama de Gantt se ha elaborado con Microsoft Excel. Las barras de color azul representan los estudios previos, las de color verde representan la fase de programación, las de color gris representan la fase de elaboración de la memoria y las de color naranja representan la presentación del TFG.

10.3. Presupuesto

Presupuesto			
Concepto	Unidades	Coste Unitario	Coste Total
Dedicación del alumno	250 horas	15,00 €	3.750,00 €
Dedicación del tutor	20 horas	35,00 €	700,00 €
Amortizacion ordenador personal	1	25% p.inicial/año	250,00 €
Matriculacion TFG	1	152,00 €	152,00 €
Precio total			**4.852,00 €**

Figura 40: Presupuesto total del TFG

El coste del proyecto es el que se muestra arriba, el cual se ha obtenido de forma aproximada a partir del desglose realizado en la EDP. Como se observa en la tabla, el coste total asciende a 4.852,00€.

Los costes principales del proyecto son las horas de trabajo dedicadas por el tutor y sobre todo por el alumno en cuestión. Como se aprecia en la tabla, se ha realizado una estimación del precio de trabajo por horas, ya que no ha sido un trabajo continuo a lo largo del tiempo.

Asimismo, se han considerado 2 costes adicionales:

- Amortización del ordenador utilizado para el trabajo. Se ha considerado un periodo de 5 años para la amortización completa del mismo.
- Matriculación del TFG.

Al tratarse de un proyecto de enseñanza no se le ha añadido el valor del IVA

11. ÍNDICE DE FIGURAS

12. REFERENCIAS

[1] *Tumor cerebral: Estadísticas*, (2022). Obtenido de:

https://www.cancer.net/es/tipos-de-c%C3%A1ncer/tumor-cerebral/estad%C3%ADsticas

[2] J. Vera, *España, octavo país de la UE en resonancias magnéticas por habitante,* (2021). Obtenido de:

https://www.plantadoce.com/entorno/espana-octavo-pais-de-la-ue-en-resonancias-magneticas-por-habitante.html

[3] Personal de Mayo Clinic, *Tumor Cerebral*, (s.f.). Obtenido de:

https://www.mayoclinic.org/es-es/diseases-conditions/brain-tumor/symptoms-causes/syc-20350084

[4] *Guía sobre los tumores cerebrales*, (s.f.). Obtenido de:

http://www.asate.es/

[5] *Machine Learning & Deep Learning: Los sistemas de IA aprenden de tus datos*, (s.f.). Obtenido de:

https://www.iic.uam.es/inteligencia-artificial/machine-learning-deep-learning/

EALDE, *Las 4 técnicas principales de Machine Learning y su aplicación en ciberseguridad*, (2021). Obtenido de:

https://www.ealde.es/tecnicas-machine-learning-ciberseguridad/

[6] *IA, Deep Learning e impacto medioambiental*, (s.f.). Obtenido de:

https://www.controlp.es/deep-learning-ia/

[7] Javier Finance, *Tipos de redes neuronales (clasificación)*, (2021). Obtenido de:

https://inteligencia-artificial.dev/tipos-redes-neuronales/

[8] Magnus Ekman, *Learning Deep Learning: Theory and Practice of Neural Networks, Computer Vision , Natural Language Processing, and Transformers Using Tensorflow*, páginas 235-276, (2021).

[9] Santanu Pattanayak, *Pro Deep Learning With Tensorflow: A Mathematical Approach to Advanced Artificial Intelligence in Python*, páginas 169-237, (2017).

[10] DotCSV, *¡Redes Neuronales CONVOLUCIONALES! ¿Cómo funcionan?*, (2020), Video didáctico. Obtenido de:

https://www.youtube.com/watch?v=V8j1oENVz00&t=3s

[11] *Ventajas de las redes neuronales convolucionales*, (s.f.). Obtenido de:

https://programmerclick.com/article/7174589545/

[12] Johan Rosa, *Tensorflow y keras con R*, (2020). Obtenido de:

https://www.johan-rosa.com/post/tensorflow-y-keras-con-r/

[13] *¿Cómo funcionan las Convolutional Neural Netwoks? Visión por Ordenador*, (2018). Obtenido de:

https://www.aprendemachinelearning.com/como-funcionan-las-convolutional-neural-networks-vision-por-ordenador/

[14] Margareth Devina, *Image Classification on Brain Tumor MRI Data*, (2021). Obtenido de:

https://rstudio-pubs-static.s3.amazonaws.com/734070_e19cdf2c9d5049caa55ab8aa5208d2f7.html

[15] Ligdi Gonzalez, *¿Qué es Tensorflow?¿Cómo funciona?*, (2021). Obtenido de:

https://aprendeia.com/que-es-tensorflow-como-funciona/

[16] Jason Brownlee, *How to Avoid Overfitting in Deep Learning Neural Networks*, (2018). Obtenido de:

https://machinelearningmastery.com/introduction-to-regularization-to-reduce-overfitting-and-improve-generalization-error/

[17] Aysegul Takimoglu, *What is Data Augmentation? Techniques & Examples in 2022*, (2022). Obtenido de:

https://research.aimultiple.com/data-augmentation/

[18] Simeon Kostadinov, *Understanding Backpropagation Algorithm*, (2019). Obtenido de: https://towardsdatascience.com/understanding-backpropagation-algorithm-7bb3aa2f95fd

[19] *¿Qué es epoch en Machine Learning?*, (s.f). Obtenido de: https://ciberseguridad.com/guias/nuevas-tecnologias/machine-learning/epoch/

13. ANEXOS

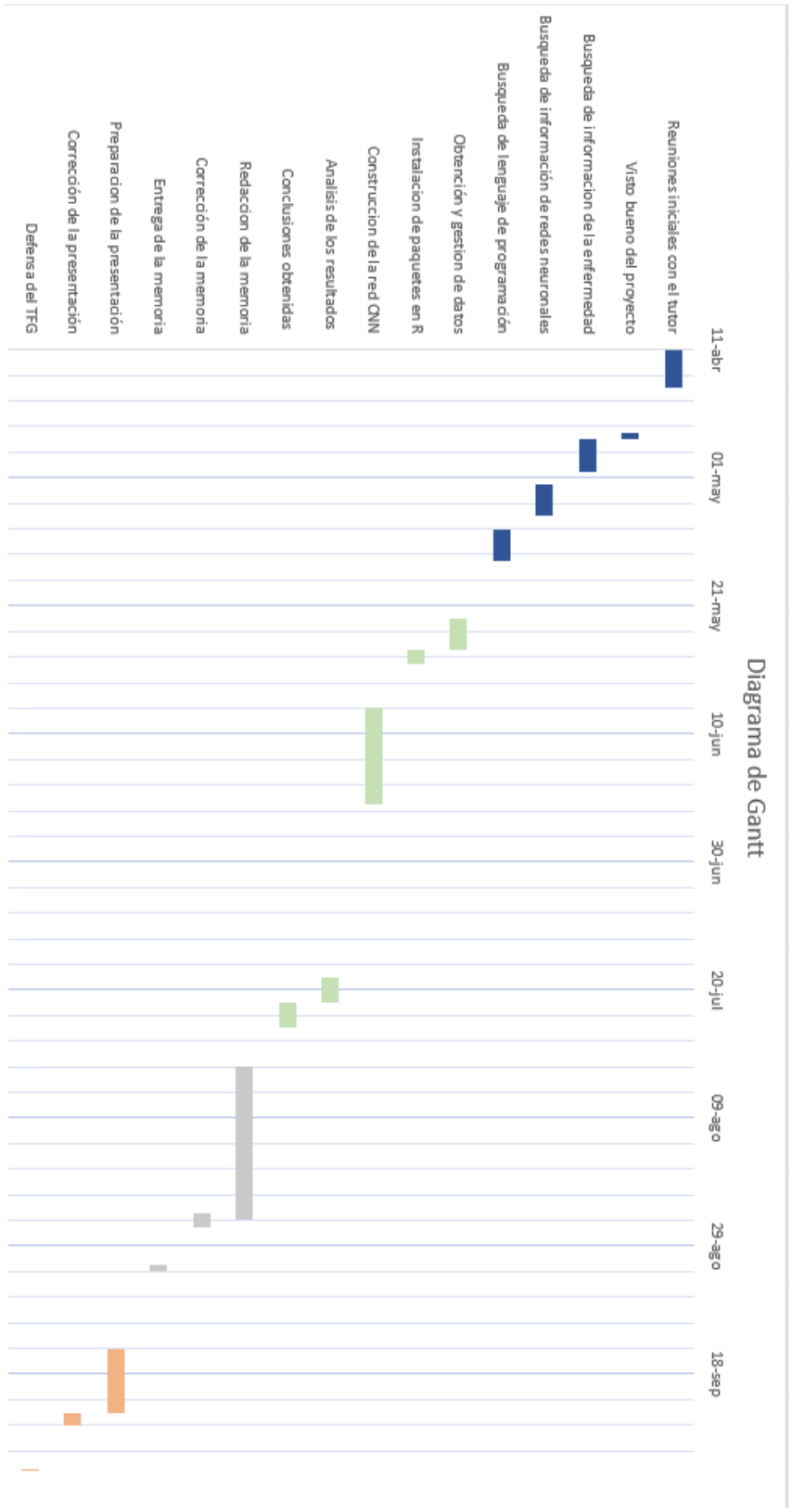

Figura 41: Detalle del Diagrama de Gantt del TFG

A continuación, se muestran algunos de los códigos de RStudio más relevantes que se han usado en el proyecto para construir la red neuronal convolucional:

```
model <- keras_model_sequential(name = "simple_model") %>%

    # Convolution Layer
    layer_conv_2d(filters = 16,
                  kernel_size = c(3,3),
                  padding = "same",
                  activation = "relu",
                  kernel_initializer = initializer,
                  bias_initializer = initializer,
                  input_shape = c(target_size, 3)
    ) %>%

    # Max Pooling Layer
    layer_max_pooling_2d(pool_size = c(2,2)) %>%

    # Flattening Layer
    layer_flatten() %>%

    # Dense Layer
    layer_dense(units = 16,
                activation = "relu",
                kernel_initializer = initializer,
                bias_initializer = initializer) %>%

    # Output Layer
    layer_dense(units = output_n,
                activation = "softmax",
                name = "Output",
                kernel_initializer = initializer,
                bias_initializer = initializer)
```

Figura 42: Código usado en R para construir el modelo inicial, model

El texto con # representan comentarios, y en este caso cada comentario representa una de las capas que forman la red.