GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

Clasificación de estados de desarrollo de Alzheimer mediante algoritmos de machine learning

Autor: Juan Ortiz de Zúñiga Faustmann

Director: Miguel Ángel Sanz Bobi

Madrid

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Clasificación de estados de desarrollo de Alzheimer mediante algoritmos de machine learning

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico 2023/24 es de mi autoría, original e inédito y

no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Fdo.: Juan Ortiz de Zúñiga Faustmann Fecha: 16/ 06/ 2024

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Firmado por SANZ BOBI MIGUEL ANGEL - ***6599** el día 17/06/2024 con un certificado emitido por AC FNMT Usuarios

Fdo.: Miguel Ángel Sanz Bobi Fecha: 17/ 06/ 2024

GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

Clasificación de estados de desarrollo de Alzheimer mediante algoritmos de machine learning

Autor: Juan Ortiz De Zúñiga Faustmann

Director: Miguel Ángel Sanz Bobi

Madrid

Agradecimientos

En primer lugar, me gustaría agradecer a mi director Miguel Ángel Sanz Bobi por ayudarme en este proyecto, al coordinador Francisco Javier Herraiz Martínez por supervisar el trabajo y asegurarse de que se cumple lo esperado, a todos los profesores que me han formado tanto a nivel profesional como en lo personal durante esta etapa de mi vida. También quiero agradecer a tanto a la universidad como a mi familia por los valores que me han inculcado.

CLASIFICACIÓN DE ESTADOS DE DESARROLLO DE ALZHEIMER MEDIANTE ALGORITMOS DE MACHINE LEARNING

Autor: Ortiz de Zúñiga Faustmann, Juan
Director: Sanz Bobi, Miguel Ángel
Entidad Colaboradora: ICAI – Universidad Pontificia Comillas

RESUMEN DEL PROYECTO

Una red neuronal es un modelo computacional que busca imitar el funcionamiento del cerebro humano. Es un sistema de procesamiento de información que aprende a realizar tareas a través de ejemplos, en lugar de ser programado explícitamente para ello.

Las redes neuronales están compuestas por unidades simples llamadas "neuronas artificiales", que están organizadas en capas y conectadas entre sí mediante enlaces ponderados. Estas conexiones ponderadas permiten que la red "aprenda" a partir de los datos de entrenamiento, ajustando los pesos de las conexiones para minimizar el error en la salida deseada.

Esta tecnología es cada vez más usada y en sectores como la medicina se puede ver su uso en diferentes sectores como la robótica quirúrgica, el análisis de datos genómicos o incluso el pronóstico de enfermedades. Dentro de la medicina hay un sector muy específico en el cual las soluciones actuales no son óptimas por diferentes razones, ese sector es la detección de Alzheimer.

Las distintas soluciones tienden a tener los mismos problemas, por una parte, están las que son muy tediosas como la evaluación clínica, que pese a sus buenos resultados es muy lenta. Por otro lado, están las que son menos fiables como las pruebas neuropsicológicas, que encima están sujetas a errores del examinador.

En general es difícil encontrar una solución que sea rápida y al mismo tiempo muy fiable, y por eso nace la necesidad de este proyecto que busca aprovechar el enorme crecimiento de las redes neuronales en la última década para aliviar la carga de un sector tan cargado como el sector médico.

En este proyecto se va a empezar por introducir lo que son las redes neuronales, como funcionan y los tipos en base a las arquitecturas y al tipo de aprendizaje. Luego se entrará en más detalle en las redes neuronales convolucionales, para ver cómo funcionan y como se han aplicado en este proyecto. Posteriormente se comentará los resultados obtenidos y finalmente se habrá una conclusión del proyecto.

Como ya se ha explicado anteriormente las redes neuronales se basan en el comportamiento de nuestras neuronas y en cómo se relacionan. En una primera instancia todas las redes están formadas por 3 capas: la capa de entrada, las capas ocultas, y la capa de salida.

Las capas de entrada y salida no tienen mucho misterio, sin embargo, las capas ocultas al ser opcionales y no tener ninguna restricción nos permiten formar diferentes arquitecturas que formaran diferentes tipos de redes neuronales como las convolucionales (CNN), las recurrentes (RNN) o incluso las radiales (RBF). Cada una con sus ventajas y desventajas que las permiten ser usadas en diferentes entornos. Un ejemplo seria las redes neuronales generativas adversarias (GAN), que mediante un sistema de dos redes neuronales enfrentadas permite generar contenido que se asemeja a los datos de entrenamiento.

Una red neuronal convolucional es un tipo de red neuronal artificial donde las neuronas artificiales, corresponden a campos receptivos simulando las neuronas en la corteza visual, lo que las hace muy eficaces para la clasificación y segmentación de imágenes. Gracias a esto es posible usar las redes convolucionales para automatizar diferentes tareas en las cuales sea necesario clasificar. Estas redes son cada vez más usadas en la actualidad, y en áreas como la medicina son usadas a modo de apoyo para los profesionales.

Lo siguiente que se explica en el trabajo son las diferentes decisiones que se han tomado en el proyecto. Los primeros serian los problemas provocados por el data set inicial, debido a que las clases no están balanceadas es necesario resolverlo y para ello se planteó usar SMOTE, pero al final se acabó quitando una clase por los problemas que conllevaba usar SMOTE.

Lo siguiente de lo que hablaremos es de las métricas usadas y por que se han elegido, en nuestro caso ha sido AUC de tres clases mediante el metodo one to all junto con otras 3 métricas de acompañamiento como son precisión, recall, y el rmse.

Lo siguiente que se trata es de las funciones de activación que en nuestro caso hemos usado SeLu, tambien el numero de bloques de convolución y de densidad que añadimos, estos son los principales problemasa del proyecto y en esta memoria se detalla como se han solucionado

También hacemos una prueba para elegir el mejor optimizador para nuestro modelo, en la cual probamos SGD, RMSprop, Adam y AdamW. Para la prueba usamos nuestro modelo y en un principio se iba a usar Keras Tuner para la hiperparametrización, pero finalmente como solo eran 4 opciones se decidió hacer una comprobación por separado de cada uno.

Después de entrenar el modelo queda analizar los resultados, para ello usaremos una matriz de confusión en la que encontramos un AUC del 98% y una precisión del 80%. Luego tenemos una matriz de confusión con la que podemos contrastar estos datos y ver como se distribuyen esos casos que están mal diagnosticados.

Por último, se describe el código usado para hacer la conexión desde una página web y el código para procesar las llamadas, la herramienta usada ha sido Flask y la página web permite añadir una imagen y enviarla mediante dos botones.

CLASSIFICATION OF ALZHEIMER'S DISEASE DEVELOPMENT STAGES USING MACHINE LEARNING ALGORITHMS

Author: Ortiz de Zúñiga Faustmann, Juan
Supervisor: Sanz Bobi, Miguel Ángel
Collaborating Entity: ICAI – Universidad Pontificia Comillas

SUMMARY

A neural network is a computational model that seeks to imitate the functioning of the human brain. It is an information processing system that learns to perform tasks through examples, rather than being explicitly programmed for them.

Neural networks are composed of simple units called "artificial neurons", which are organized into layers and interconnected through weighted links. These weighted connections allow the network to "learn" from the training data, adjusting the weights of the connections to minimize the error in the desired output.

This technology is increasingly being used and in sectors such as medicine, its use can be seen in different areas such as surgical robotics, genomic data analysis, or even disease prognosis. Within medicine, there is a very specific sector in which current solutions are not optimal for different reasons, that sector is Alzheimer's detection. The different solutions tend to have the same problems. On one hand, there are those that are very tedious, such as clinical evaluation, which despite its good results, is very slow. On the other hand, there are those that are less dependable, such as neuropsychological tests, which are also subject to examiner errors.

In general, it is difficult to find a solution that is both fast and very dependable, and that is why the need for this project arises. It seeks to take advantage of the enormous growth of neural networks in the last decade to alleviate the burden on such a demanding sector as the medical sector. In this project, we will start by introducing what neural networks are, how they work, and the types based on architectures and type of learning. Then we will go into more detail about the project. Subsequently, the results obtained will be discussed, and finally, there will be a conclusion of the project.

As explained earlier, neural networks are based on the behaviour of our neurons and how they relate to each other. Initially, all networks are formed by 3 layers: the input layer, the hidden layers, and the output layer. The input and output layers don't have much mystery, however, the hidden layers, being optional and having no restrictions, allow us to form different architectures that will form different types of neural networks such as convolutional (CNN), recurrent (RNN) or even radial basis function (RBF) networks. Each with its own advantages and disadvantages that allow them to be used in different environments. An example would be generative adversarial networks (GANs), which use a system of two opposing neural networks to generate content that resembles the training data.

A convolutional neural network is a type of artificial neural network where the artificial neurons correspond to receptive fields simulating neurons in the visual

cortex, which makes them remarkably effective for image classification and segmentation. Thanks to this, it is possible to use convolutional networks to automate different tasks in which classification is necessary. These networks are increasingly being used today, and in areas such as medicine they are used to support professionals.

The following explains the different decisions that have been made in the project. The first would be the problems caused by the initial dataset, since the classes are not balanced, it is necessary to resolve it and for this, using SMOTE was proposed, but in the end, one class was removed due to the problems involved in using SMOTE.The next thing we will discuss is the metrics used and why they were chosen. In our case, it was the three-class AUC using the one-to-all method along with three other accompanying metrics: precision, recall, and RMSE.

The next thing discussed is the activation functions we used, which in our case was Selu, as well as the number of convolution and density blocks, we added. These are the main problems of the project, and this report details how they have been solved.

We also test to choose the best optimizer for our model, in which we tried SGD, RMSprop, Adam, and AdamW. For the test, we used our model and initially, we were going to use Keras Tuner for hyperparameter tuning, but finally, since there were only 4 options, it was decided to do a separate check of each one.

After training the model, we need to analyse the results, with AUC and precision being the most important to define if the results are good, in which we find an AUC of 98% and a precision of 80%. Then we have a confusion matrix with which we can contrast this data and see how those misdiagnosed cases are distributed.

Finally, the code used to make the connection from a web page and the code to process the calls is described. The tool used has been Flask and the web page allows adding an image and sending it using two buttons.

Índice de la memoria

Índice de figuras

1. INTRODUCCIÓN

1.1 MOTIVACIÓN DEL PROYECTO

La motivación para este proyecto es ayudar a la sociedad en un área tan relevante como la medicina, desde la perspectiva de un ingeniero de telecomunicaciones. Usando las nuevas herramientas que nos aporta la tecnología como son las redes convolucionales, con el propósito de aliviar la carga del médico.

Este proyecto tiene como objetivo principal minimizar la dependencia de un análisis manual, ya que busca automatizar una parte esencial del proceso de detección de la enfermedad del Alzheimer. Al hacerlo, se pretende no solo agilizar el diagnóstico, sino también simplificar y mejorar significativamente la atención al paciente.

Estos sistemas pueden analizar varios parámetros médicos y detectar patrones que de otra manera los pasaríamos por alto. Además, este enfoque automatizado podría lograr un diagnóstico temprano, lo que es crucial para el tratamiento efectivo del Alzheimer.

2. DESCRIPCIÓN DE LAS TECNOLOGÍAS

En este proyecto se han usado Python con sus librerías de tensor Flow y Keras, además para conectar el modelo con una página web se ha usado flask, también se ha usado HTML, css y JavaScript para la página web.

- **Python:** Para el lenguaje de programación debido a su extensa disponibilidad de bibliotecas y recursos relacionados con el aprendizaje profundo y la inteligencia artificial.
- **Keras:** Actúa como una interfaz de alto nivel que simplifica la creación y el entrenamiento de redes neuronales. Su enfoque intuitivo y amigable permite a los desarrolladores definir modelos de redes neuronales de manera más accesible.
- **Tensor Flow**: Desarrollado por Google, es uno de los marcos de trabajo más poderosos y ampliamente utilizados en el campo del aprendizaje profundo. Ofrece una estructura sólida para la construcción y el entrenamiento de redes neuronales y es especialmente eficiente cuando se trabaja con datos a gran escala.
- **Flask:** Flask es un popular framework de desarrollo web en Python. Es minimalista y ligero, lo que lo hace ideal para crear aplicaciones web simples y rápidas. Flask está diseñado para ser fácil de usar, lo que lo convierte en una excelente opción para este trabajo en el que la mayor parte de la complejidad no está en la conexión con el servidor.

3. ESTADO DE LA CUESTIÓN

La inteligencia artificial ha tenido grandes avances debido a la interconexión y el aprovechamiento de diversas ramas dentro del campo de la IA. Un ejemplo de ello es el uso de Vision Transformers**¡Error! No se encuentra el origen de la referencia.**, los transformadores miden la relación entre pares de entradas, los cuales pueden ser palabras o pixeles, lo cual en una imagen puede ser muy costoso y por eso los tranformadores calculan las relaciones en secciones de las imágenes.

Este enfoque ha demostrado su eficacia en el reconocimiento de imágenes medianas y pequeñas, los Vision Transformers, entrenados con volúmenes masivos de datos, han mostrado una destreza notable en la interpretación y comprensión de información visual, lo que les permite realizar tareas de reconocimiento con una precisión en constante mejora.

En la actualidad hay muchas maneras de detectar el Alzheimer en pacientes, las cuales tienen sus ventajas y sus inconvenientes, las más usadas son:

1. **Evaluación clínica:** Se revisan los síntomas, historia médica y antecedentes familiares del paciente. Las fallas pueden ocurrir si los síntomas son inespecíficos, también es un método que consume mucho tiempo y por lo tanto puede ser muy lento.
2. **Pruebas cognitivas:** Mini-Mental State Examination[27] (MMSE) y Montreal Cognitive Assessment (MoCA) son pruebas comunes para evaluar la función cognitiva, pero tienen problemas para detectarlo en las fases tempranas de la enfermedad [27].
3. **Pruebas neuropsicológicas**: Estas pruebas evalúan habilidades cognitivas específicas, pero los resultados son muy variables y está sujeto a errores del examinador provocando que estas puedan ser menos fiables que otros métodos, y se suelen usar como punto de partida.[25]

4. **Análisis de sangre y líquido cefalorraquídeo:** Los biomarcadores, pueden indicar la presencia de la enfermedad[26]. Este método puede llegar a predecir la aparición del Alzheimer incluso mucho antes que otros métodos, pero no es una prueba concluyente. El método de análisis de sangre es muy reciente, y diferentes estudios en los últimos años tienen conclusiones muy diferente sobre su eficacia [26].
5. **Imágenes cerebrales:** La resonancia magnética (MRI) puede revelar cambios en el cerebro, en muchos casos tampoco es concluyente, pero a menudo se usa para descartar otras enfermedades[3].
6. **Genética:** Las pruebas genéticas pueden identificar mutaciones asociadas con un alto riesgo de Alzheimer, pero solo en casos hereditarios, y no para el resto de los casos lo cual lo convierte en un método poco fiable [28].
7. **Evaluación del deterioro funcional:** Evaluar la capacidad del paciente de manera directa[25]. No siempre se relaciona con el diagnóstico de Alzheimer y al igual que otras pruebas es muy lenta.
8. **Evaluación de cambios conductuales y psicológicos**: Identificar cambios en el comportamiento y el estado de ánimo del paciente. Al igual que en el método anterior no siempre está relacionado con el Alzheimer y puede ser inespecífico [29].

Aparte de los problemas que tienen cada una también hay una serie de problemas comunes a la hora de detectar esta enfermedad

- **Similitud con otras enfermedades:** Los síntomas del Alzheimer, como la pérdida de memoria, pueden confundirse con otras condiciones médicas, como la depresión.
- **Dificultad para detectar en etapas tempranas:** En las etapas iniciales, los síntomas del Alzheimer son difíciles de notar.
- **Limitaciones de las pruebas:** Muchas veces las pruebas de detección pueden tener limitaciones en cuanto a sensibilidad

Como se puede ver hay muchas maneras de detectar e identificar la enfermedad del Alzheimer cada una con sus ventajas y sus desventajas, pero en la mayoría de casos se usan varios métodos usando sobre todo los métodos de evaluación, el proceso muchas veces es un proceso largo y muy tedioso, por eso nace la idea de este proyecto que busca aliviar la carga humana, sin remplazarla, dicho de otra manera busca ser una herramienta a la disposición de las instituciones médicas que podrán usarlo de manera orientativa.

En el caso especifico de las imágenes cerebrales se lleva probando mucho tiempo el uso de redes convolucionales para aplacar este problema [8], con resultados muy buenos de hasta un 95.5% de Accurancy [30]. Hoy en dia, ya hay muchos modelos que sean capaces de resolver este problema, ya que es un campo que se ha estado estudiando desde el nacimiento de la tecnología, lo que por ahora no se ha encontrado es una herramienta a la disposición de todo el mundo que sirva de apoyo para el profesional, y que se centre en la detección temprana de la enfermedad para poder tratar la enfermedad.

4. DEFINICIÓN DEL TRABAJO

4.1 JUSTIFICACIÓN

Como se ha visto en el apartado anterior, ya existen modelos capaces de predecir con cierta exactitud la enfermedad del Alzheimer, pero en muchos casos el proyecto no está abierto al público y tampoco es de fácil acceso, esto es algo vital en la justificación del proyecto, ya que con este se busca que pueda ser usado de manera simple, accediendo a la página web.

También, en muchos casos las redes desarrolladas son muy cargadas y con mucho trabajo interno, en este trabajo se intenta simplificar el modelo, sin arriesgar el porcentaje de aciertos.

4.2 OBJETIVOS

El objetivo principal de este proyecto es diseñar, entrenar y desplegar una red convolucional altamente eficiente y precisa que pueda procesar imágenes de resonancia magnética (MRI) y clasificarlas en las diferentes categorías especificadas anteriormente.

También se busca crear un herramienta útil para las instituciones médicas y que pueda aligerar la carga del personal médico. Lo cual no es fácil, ya que, aunque el trabajo funcione correctamente es difícil conseguir que un médico decida usar la web cuando puede usar sus propios conocimientos, pero al menos puede usar este proyecto a modo de segunda opinión, o como método de apoyo.

5. REDES CONVOLUCIONALES

En 1943 en un artículo de Warren McCullough y Walter Pitts, presentaron un modelo matemático para crear una red neuronal artificial, este articulo sentó las bases de la inteligencia artificial, pero no fue hasta 1956 en la conferencia de Dartmouth que se acuño el término "inteligencia artificial" por primera vez por parte de John McCarthy.

Esta conferencia fue un evento clave que marcó el comienzo de la IA como un campo de investigación formal. Los investigadores exploraron áreas como el procesamiento de lenguaje natural, la resolución de problemas, la visión por computadora y el aprendizaje automático.

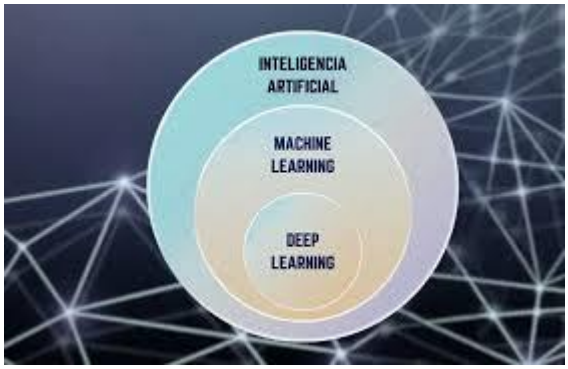

Figura 1 : Esquema de la inteligencia artificial (EvolMIND)[13]

Machine learning o el aprendizaje automatizado es la ciencia que estudia diferentes algoritmos y modelos estadísticos para llevar a cabo una serie de instrucciones en las que no tiene una serie de pautas específicas sobre lo que debe de hacer. En machine learning los algoritmos se entrenan para hacer predicciones o clasificaciones basándose en la información que ha obtenido previamente, esta información es muy importante ya que es la que en gran parte definirá si el modelo aprende realmente algo del entrenamiento.

La estructura de estos modelos presenta unos datos en los que el modelo se basara para hacer una predicción, luego el modelo comprobara el resultado y aprende si la predicción es buena o no, de esta manera mejorara sus predicciones.

El Deep learning, el cual es un subcampo del machine learning, el cual se basa en un conjunto de algoritmos de machine learning que tienen varias etapas de procesamiento y con estructura compleja. Estas estructuras se inspiran en la del cerebro humano y busca asemejar su funcionamiento para reconocer datos complejos como imágenes, audio y texto.

5.1 REDES NEURONALES

En este proyecto se ha estudiado el uso de redes convolucionales para la detección de imágenes, pero primero es necesario comprender el comportamiento de las neuronas.

En nuestras neuronas las dendritas reciben señales químicas (neurotransmisores) de otras neuronas a través de las sinapsis, luego las señales se propagan al cuerpo celular y si la suma de excitaciones supera un umbral generara un impulso eléctrico que se propaga a lo largo del axón. Este impulso eléctrico genera una serie de procesos en la neurona que libera neurotransmisores, los cuales permiten que el proceso se repita en otras neuronas cercanas

En una red neuronal artificial se busca imitar este proceso, con una entrada y una serie de neuronas artificiales que imitan el proceso de las neuronas, se envían señales mediante unos enlaces que tienen unos pesos que afectaran al estado de activación de las neuronas adyacentes, estos pesos tienen como objetivo establecer la importancia entre la conexión de las neuronas. También tenemos una función de activación que al igual que el umbral limita la salida antes de enviarlo a otras neuronas artificiales.

Una neurona artificial[23] recibe una serie de entradas con un peso, a las cuales luego suma entre ellas y al resultado se transmite de la función de activación. De esta manera imita el comportamiento previamente explicado de las neuronas.

Donde el vector de entradas seria X y el de pesos W quedaría que la suma ponderada seria $W^T * X$, el cual pasaria por la funcion de activacion, u cuya salida llamaremos Y que sera lo que se envie a las neuronas adyacentes.

En la siguiente imagen se puede ver el funcionamiento de la neurona artificial

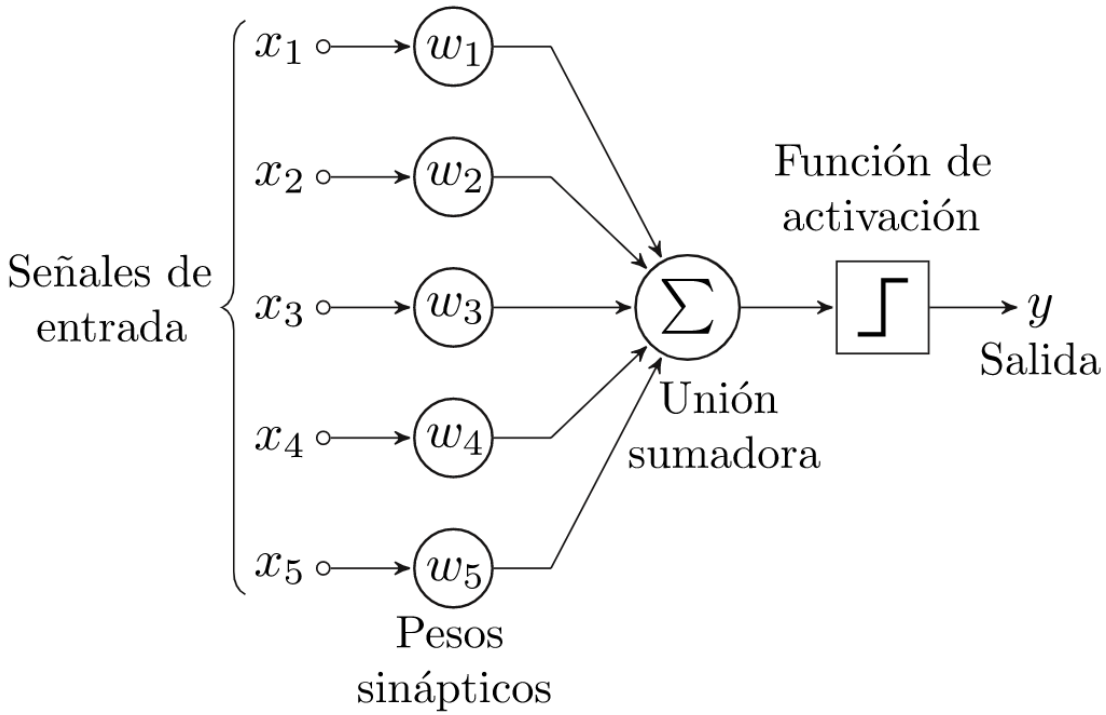

Figura 2 : Esquema de la neurona artificial (Telefonica Tech) [14]

El número de capas[6] en una red neuronal artificial puede variar dependiendo del tipo de red y la tarea para la que se está entrenando. Sin embargo, generalmente se pueden distinguir tres tipos principales de capas:

1. **Capa de entrada:** Esta capa recibe los datos de entrada, como imágenes, texto o señales. El número de nodos en esta capa corresponde al número de características o dimensiones de los datos de entrada.
2. **Capas ocultas:** Estas son las capas intermedias entre la capa de entrada y la capa de salida. Las neuronas de estas capas realizan cálculos y transformaciones en los datos de entrada para extraer características más

complejas y relevantes. Una red neuronal puede tener una o varias capas ocultas, dependiendo de la complejidad del problema.

3. **Capa de salida:** Esta capa produce la salida final de la red neuronal, como una clasificación, una predicción o una puntuación. El número de nodos en esta capa depende de la tarea específica, como el número de clases en un problema de clasificación.

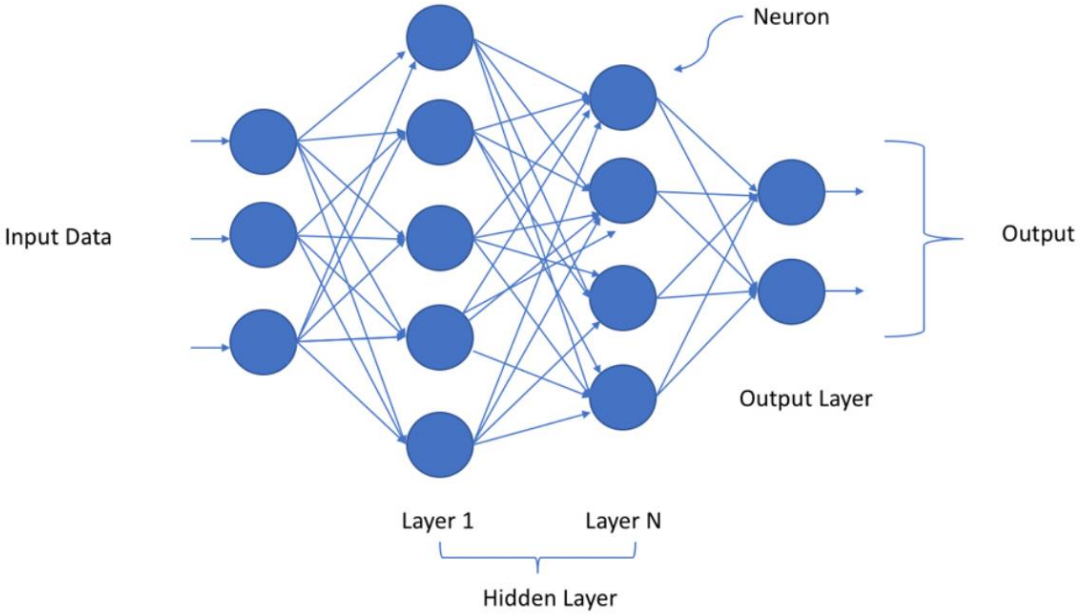

Figura 3 : Esquema de una Deep Neural Network (Towards Data Science)[15]

5.2 TIPOS DE REDES NEURONALES SEGÚN LA TOPOLOGÍA

Existen varios tipos de topologías [2] o arquitecturas de redes neuronales artificiales (RNA), cada una diseñada para tareas y tipos de datos específicos, los más comunes son:

- Perceptrón multicapa (MLP): Esta RNA es de tipo forwarding y tiene múltiples capas que le permiten resolver problemas no linealmente separables

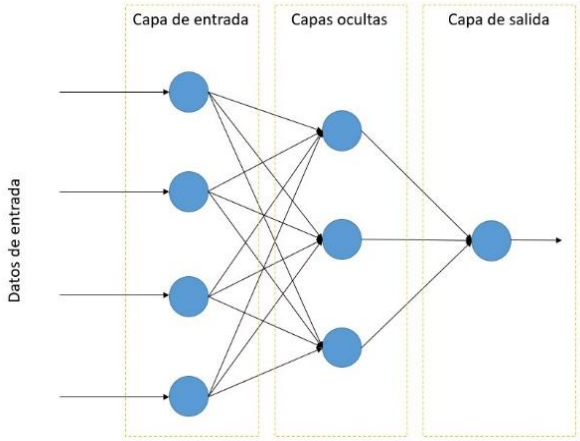

Figura 4 : Perceptrón multicapa (Interactive chaos)[16]

1. Redes convolucionales (CNN): Al igual que las MLP, las CNN también son forwarding y destacan por su gran capacidad en el área de visión computacional y en el reconocimiento de imágenes, ya que imitan la corteza visual del cerebro y son muy buenas reconociendo patrones.

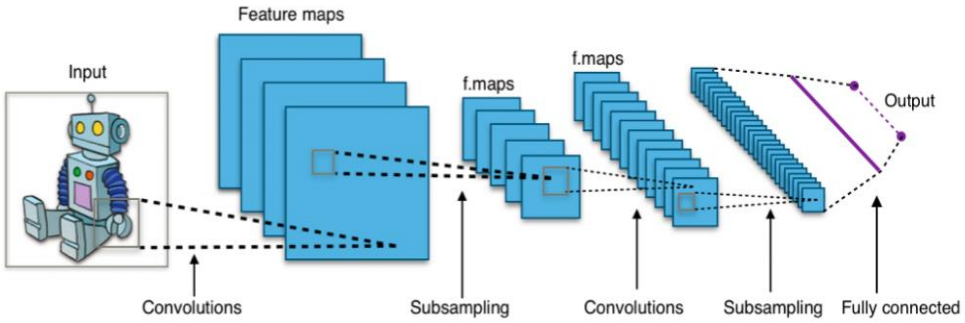

Figura 5 : Esquema de una CNN (Wikipedia)[17]

2. Redes neuronales recurrentes[5] (RNN): Se caracteriza por usar datos secuenciales, lo cual les hace especialmente buenos para la traducción de idiomas o para el reconocimiento de voz. También destaca por tener memoria, ya que, a diferencia de otras redes neuronales, esta arquitectura considera que hay una relación entre las distintas entradas.

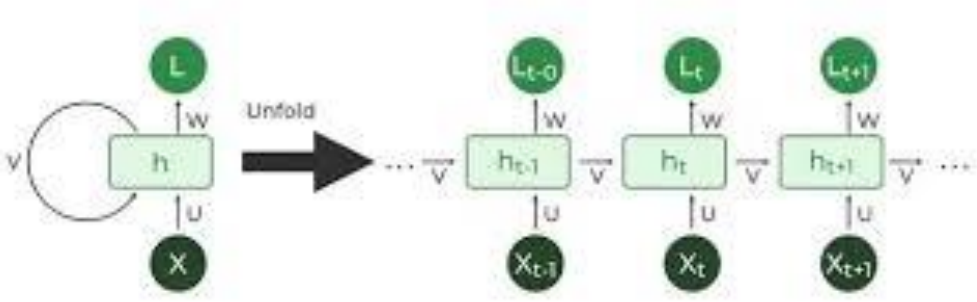

Figura 6 : Neurona recurrente (Geeks for Geeks)[18]

3. Redes de base radial (RBF): Las funciones de base radial [2][4] son funciones simétricas que calculan la distancia euclídea entre el vector de entrada y un vector de centros predefinidos, y producen un valor máximo cuando la entrada coincide exactamente con el centro. Son usadas especialmente para resolución de problemas de aproximación o en análisis de series temporales.

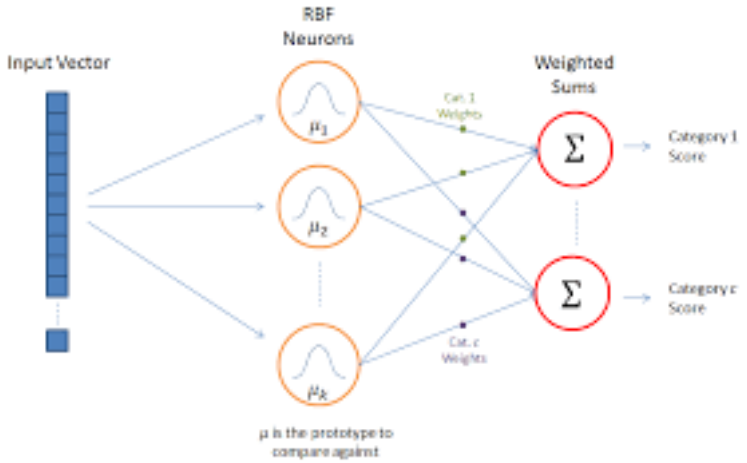

Figura 7 : Arquitectura de una RBF (Chris McCormick)[19]

4. Redes neuronales Transformers: Las redes neuronales Transformers permiten relacionar diferentes entradas entre sí, ya que aplica un mecanismo

de atención para aportar un contexto, esto permite al modelo entender las relaciones entre las entradas en grandes cantidades de datos como por ejemplo en una frase, donde la relación puede ser entre palabras en ambos extremos y donde el modelo no sería capaz de encontrar la relación, como por ejemplo en la frase “El chico al otro lado del pasillo está corriendo”.

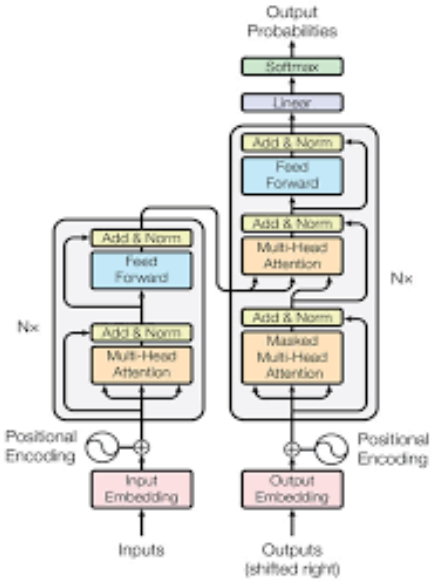

Figura 8 : Funcionamiento de los Transformers (IIC)[20]

5. Redes Generativas Adversarias (GAN): Consiste en un sistema de dos redes neuronales en las que una genera contenido (normalmente imágenes) y la otra red evalúa, la primera red busca engañar a la evaluadora para que acepte el contenido generado como auténtico. Se usan para producir imágenes fotorrealistas o para reconstruir modelos 3D a partir de imágenes 2D

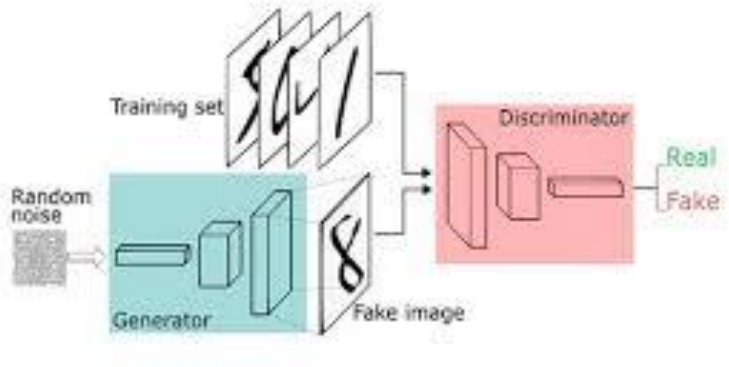

Figura 9 : Esquema de una red GAN (Xataka)[21]

5.3 REDES NEURONALES SEGÚN EL APRENDIZAJE

También se pueden diferenciar por su método de aprendizaje, en donde todos los métodos se pueden englobar en dos tipos:

1. **Aprendizaje supervisado**
   Usa un conjunto de datos de entrenamiento con valores etiquetados, en base a cuál el modelo aprenderá con el tiempo.
2. **Aprendizaje no supervisado**
   En este caso los datos no tienen etiquetas, por lo que los algoritmos descubren patrones ocultos, a cambio la complejidad computacional aumenta, y es necesario mayor cantidad de datos respecto al supervisado.

Tipos de aprendizajes:

1. **Aprendizaje por corrección de error**
   Este tipo de aprendizaje ajusta los pesos entre neuronas en base a la diferencia entre los valores obtenidos y los deseados.

2. **Aprendizaje estocástico**

   Como su nombre indica, usa procesos aleatorios para seleccionar los pesos a cambiar.

3. **Aprendizaje hebbiano**

   Se basa en el aprendizaje por repetición, ya que busca reforzar las conexiones más usadas entre neuronas, dicho de otra manera, aumenta los pesos entre neuronas que estén activas.

4. **Aprendizaje competitivo**

   En el aprendizaje competitivo las neuronas de salida de una red neuronal compiten entre sí para activarse, en el aprendizaje competitivo no puede haber más de una neurona activa al mismo tiempo. Es necesario un mecanismo que determine la fuerza de cada neurona.

5. **Aprendizaje por refuerzo**

   Los pesos aumentan cuando acierta y disminuyen cuando fallan, a diferencia del aprendizaje por error aquí solo se tiene en cuenta el valor de salida para toda la capa, mientras que en el de error tenemos un vector de valores de error.

6. MODELO

6.1 DATA SET

Para hablar del proyecto es necesario hablar primero de la base del proyecto, que en este caso serían los datos que nos permiten entrenar el modelo, en este caso se descargó un data set de Kaggle que es una página muy conocida para compartir data sets, modelos o códigos y que es una herramienta muy útil para comparar o aprender.

El data set tenía 2 carpetas, una para el entrenamiento y otra para el test, a la hora de entrenar un modelo es muy importante definir bien el porcentaje de datos que se usan para entrenar y cuales, para comprobar, para evitar overfitting y optimizar el proyecto, en este caso, la carpeta de entrenamiento tenía 5000 archivos y la de test tenía 1250 archivos, lo cual deja con una división del 80% para el entrenamiento y un 20% para el test, lo cual está dentro del rango aceptable.

Dentro de cada carpeta, los archivos estaban divididos en 4 categorías que eran: Sin demencia, demencia leve, demencia muy leve y demencia moderada. La división de los datos en las diferentes clases queda tal y como se muestra en la siguiente figura.

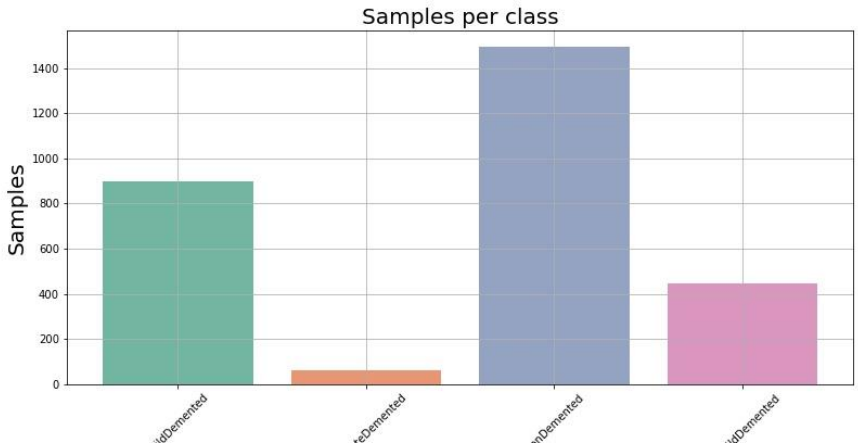

Figura 10 : Grafico del data set inicial

En la figura se puede apreciar que el número de ejemplos de las 4 categorías o clases de demencia están muy desbalanceados, normalmente se usarían herramientas para balancear los datos como SMOTE o redes neuronales adversarias (GAN).

SMOTE (Synthetic Minority Oversampling Technique) es una herramienta muy útil para balancear clases, es un algoritmo de muestreo sintético que permite generar nuevas muestras a partir de la interpolación de las clases minoritarias.

SMOTE funciona de la siguiente manera: Primero determina la cantidad de datos que ha de generar, luego selecciona instancias de una clase minoritaria al azar y elige sus vecinos más cercanos basándose en la distancia euclídea y toma la diferencia de valores entre estos dos, lo multiplica por un valor mayor que cero y menor que 1 y se lo suma al valor original, así es capaz de generar nuevas muestras sintéticas.

Pero SMOTE tiene varios problemas, entre ellos puede aumentar el ruido del sistema, además con tanta diferencia de datos entre las clases al balancear los datos daría muchos problemas para generar el resto de las muestras y con una cantidad tan baja de datos provocaría un falso aumento en los aciertos.

Por estas razones se descartó la idea de usar SMOTE [9], sin embargo, el problema seguía estando ahí, por eso se decidió cambiar el número de categorías a 3 y eliminar la categoría de Demencia moderada que estaba causando tantos problemas.

Tras esto nos quedó un data set más balanceado y con el cual era más fácil trabajar y los datos se quedaron de la siguiente manera.

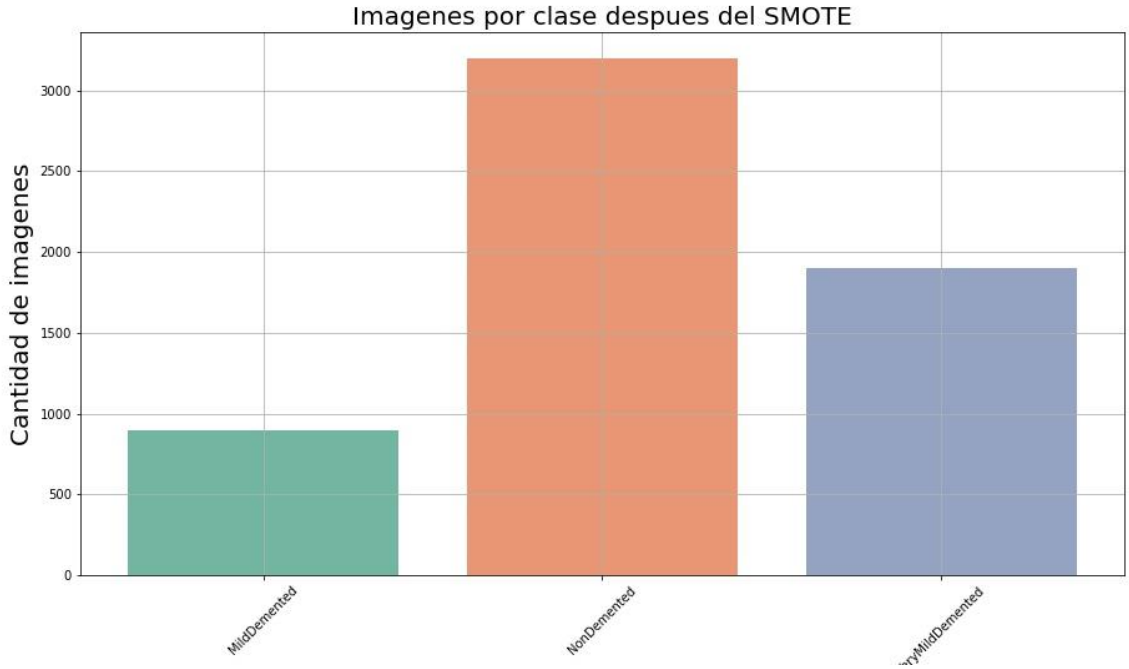

Figura 11 : Grafico del data set al eliminar una clase

Como se puede apreciar en la imagen, aún hay cierto desequilibrio, aunque es menor que en el caso anterior, se planteó usar SMOTE con 3 clases, pero aparte de lo ya mencionado se descartó por otras 2 razones: la primera es la cantidad de memoria que usa, SMOTE crea muchos datos que se guardan en la memoria y si hay muchos datos puede dar errores de memoria esto se debe a que estamos manejando grandes cantidades de datos y además SMOTE tiene que generar muchas por el desbalance.

La segunda razón es menos importante para el producto final pero afecta a la creación de este, y es el tiempo de entrenamiento. Para ello se mostrará 2 capturas del mismo modelo siendo entrenado en las mismas condiciones con la única diferencia del uso de SMOTE en uno de ellos.

```
Epoch 1/100
100/100 [==============================] - 372s 3s/step
C: 0.6589 - val_mse: 0.2872 - lr: 0.0100
Epoch 2/100
100/100 [==============================] - 571s 6s/step
C: 0.6191 - val_mse: 0.2909 - lr: 0.0079
Epoch 3/100
100/100 [==============================] - 547s 5s/step
C: 0.3914 - val_mse: 0.2930 - lr: 0.0063
Epoch 4/100
100/100 [==============================] - 563s 6s/step
C: 0.5342 - val_mse: 0.2852 - lr: 0.0050
Epoch 5/100
100/100 [==============================] - 557s 6s/step
C: 0.7650 - val_mse: 0.1800 - lr: 0.0040
Epoch 6/100
```

Figura 12 : Tiempo de entrenamiento del modelo con SMOTE

```
Epoch 1/100
100/100 [==============================] - 189s 2s/step
C: 0.3039 - val_mse: 0.5052 - lr: 0.0100
Epoch 2/100
100/100 [==============================] - 236s 2s/step
C: 0.6554 - val_mse: 0.3093 - lr: 0.0079
Epoch 3/100
100/100 [==============================] - 247s 2s/step
C: 0.7529 - val_mse: 0.2438 - lr: 0.0063
```

Figura 13 : Tiempo de entrenamiento sin SMOTE

Como se puede apreciar en las figuras el modelo tarda mucho más tiempo en entrenar cuando se usa SMOTE, pasa de tardar 2 segundos por cada muestra a tardar 5 segundos, lo cual es mucho mayor y es importante para el proyecto porque al tener que probar varios algoritmos distintos el tiempo es importante.

6.2 *MÉTRICAS*

Para este proyecto usaremos las métricas AUC[11], Precisión y Recall, y usaremos una matriz de confusión para validar los resultados del test

Es por esto por lo que no se puede usar Accuracy en data sets desbalanceados, a cambio podemos utilizar otras métricas como AUC (Area Under the Curve), recall o el error cuadrático medio.

Para AUC es necesario tener en cuenta que solo se puede hacer con dos clases, lo que hace el programa es que de manera automatica divide las clases en 2 para poder calcular el AUC entre estas dos.

Precisión es una métrica que permite calcular el número total de aciertos respecto el total de casos, es la métrica principal que usaremos para evaluar los resultados y si los valores son buenos

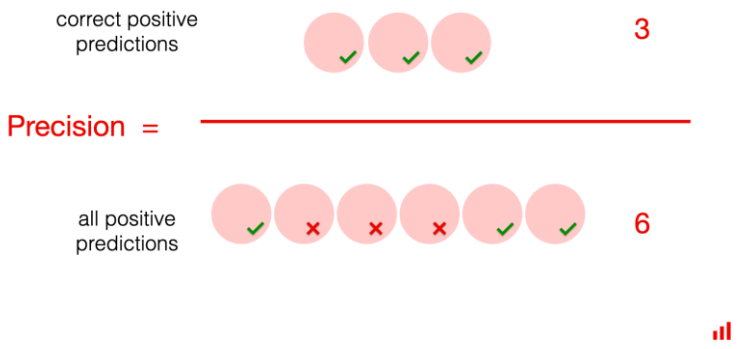

Figura 14: Precisión (evidentlyai) [22]

Por otro lado, Recall[10] permite calcular la cantidad de positivos verdaderos respecto la cantidad de falsos negativos lo cual nos permite saber el número de veces que acierta la clase.

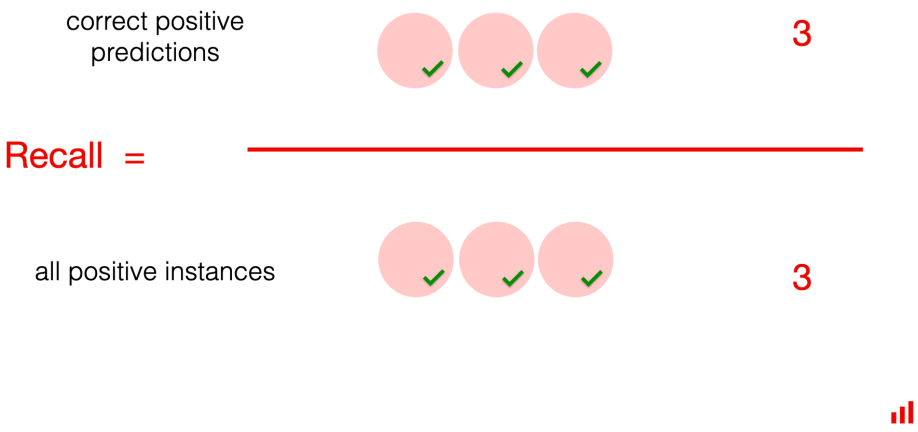

Figura 15 : Recall (evidentlyai)

Como estamos usando un data set con varias clases, recall y precisión harán una media estimada entre las distintas clases, pero esto no es lo que queremos por eso usamos uno de los argumentos que tiene en tensor Flow que es class_id, este argumento nos permite introducir un valor entre 0 y el número de clases -1.

Es decir, en nuestro caso entre 0 y 2, y lo que permite es que solo tenga en cuenta esta clase a la hora de hacer el cálculo, así podemos solo mirar los falsos positivos en aquellos en los que nos interesa.

En nuestro caso un falso positivo es peor cuando diagnostica Alzheimer a una persona que no lo tiene, y por lo tanto vamos a hacer énfasis en que recall este en la clase “NonDemented”, y por otro lado que recall este en una de las otras dos clases para ver cómo se reparten ese número de personas que son diagnosticadas erróneamente.

Por otro lado, está la métrica AUC, la AUC es el área debajo de la ROC (Receiver operating characteristic curve) que nos da una idea estimada de lo bien que funciona el modelo. En machine learning, la ROC nos dice lo bueno que es el

modelo clasificando. ROC es la representación gráfica de la proporción de verdaderos positivos respecto de los falsos positivos.

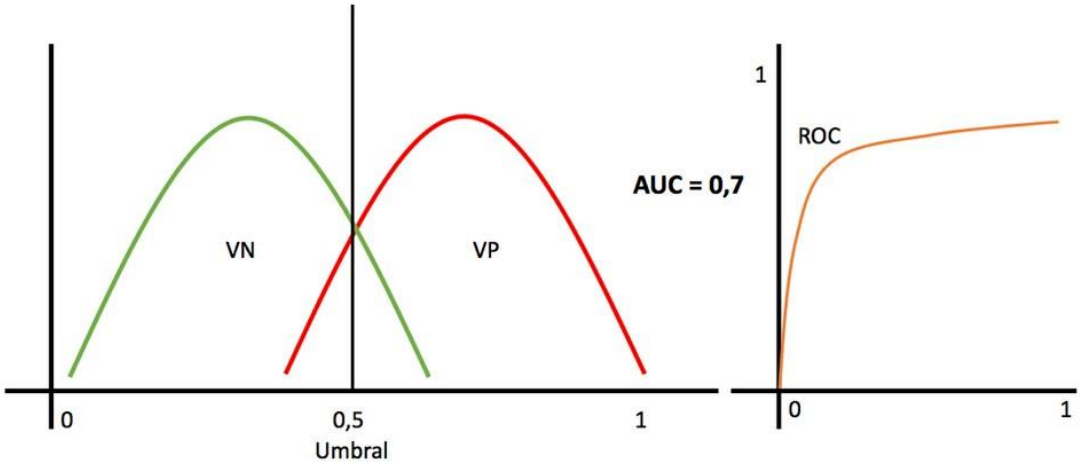

Figura 16: Esquema de la AUC y la curva ROC (Aprendeia)

Con esto ya tenemos las métricas que usaremos para evaluar el modelo, lo siguiente es la estructura del modelo, el número de bloques y el código usado.

6.3 *FUNCIONES*

Lo primero que hay que mencionar es la creación de los bloques mediante funciones predefinidas, en el siguiente código se explica la función para el bloque de convolución.

```
def convolutional_block(filters):
    block = tf.keras.Sequential([
        tf.keras.layers.SeparableConv2D(filters, kernel_size=3,
activation='selu', padding='same',strides=1),
        tf.keras.layers.SeparableConv2D(filters, 3, activation='selu',
padding='same',strides=1),
        tf.keras.layers.BatchNormalization(),
        tf.keras.layers.MaxPool2D()
    ]
    )
```

```
    return block
```

En esta función creamos un bloque de convolución, un bloque de convolución compuesta por 2 capas de convolución. Una capa de convolución es la pieza más importante de una CNN (Convolutional Neuronal Network), los filtros de la capa producen un mapa de activación, y a medida que el filtro se mueve genera un producto escalar entre el filtro y los valores de entrada.

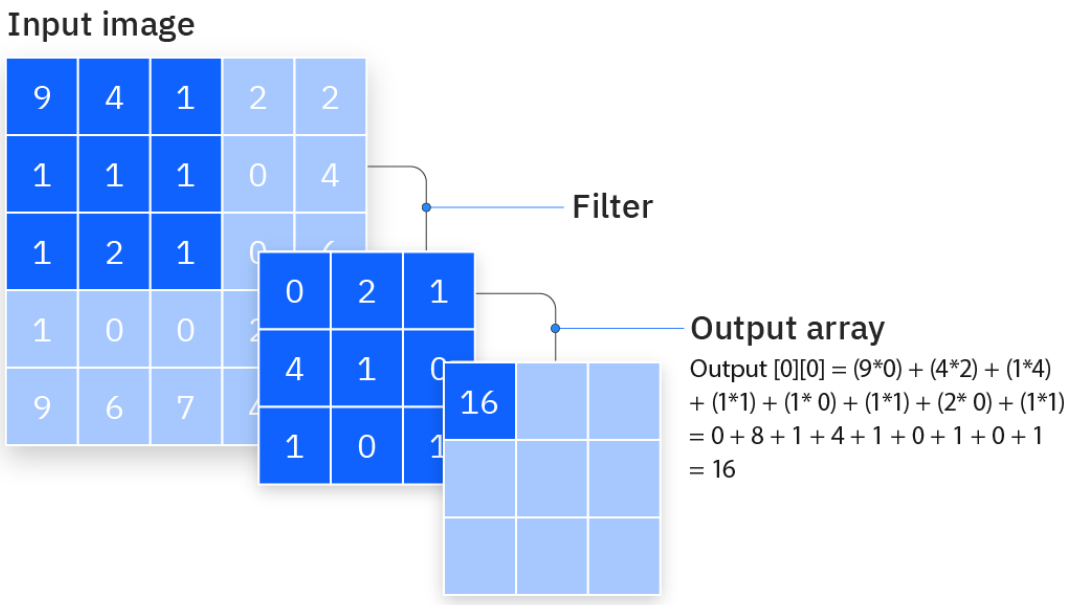

Figura 17 : Convolución de una matriz (IBM)

En nuestro caso tenemos puesto 2 capas separables de 2 dimensiones, las capas separables permiten utilizar los pixeles vecinos para obtener información, esto es muy útil cuando hay dependencia espacial, y podemos obtener información relevante de otra manera, esto nos permite ahorrar parámetros que a su vez nos permite ahorrar tiempo de entrenamiento.

En nuestra función tenemos una variable que son el número de filtros, los cuales irán cambiando entre los bloques, el número de filtros afecta a la profundidad del bloque, cuantos más filtros más mapas tendremos.

También tenemos el kernel_size que define el tamaño de la matriz que hace la convolución, normalmente para guiarnos en el tamaño debemos tener en cuenta si

los detalles de la imagen soy grandes o pequeños, si son pequeños preferiremos usar kernels más pequeños. En nuestro caso probamos varios tamaños y los que dieron mejor resultados fueron con kernel de 3x3 y 2x2, siendo entre los dos un poco mejor el de 3x3.

Stride es la distancia que se mueve el kernel sobre la matriz de entrada, cuanto mayor sea el stride menor será el tamaño de la salida. Luego esta padding, al poner el valor same-padding nos aseguramos de que el tamaño de entrada y el de salida sean el mismo.

Por otro lado también, está la función de activación en este caso hemos usado Selu [1], que es una versión que nace a partir de ReLu y tiene algunas ventajas sobre esta, aunque a cambio es más lenta. Al principio se implementó ReLu, pero con Selu se vieron algunas mejoras, no obstante, no parece al ser una función medianamente nueva no es muy utilizada y la gente en general prefiere usar ReLu por la comodidad y porque hace normalización interna lo cual puede afectar a la estructura del modelo.

Por último, está la capa de maxpool2D, la cual hace un submuestreo de la entrada a lo largo de su ancho y altura, y se queda con el valor mayor de cada ventana, estas ventanas tienen un tamaño que se puede cambiar, pero si no se especifica el tamaño es de 2x2.

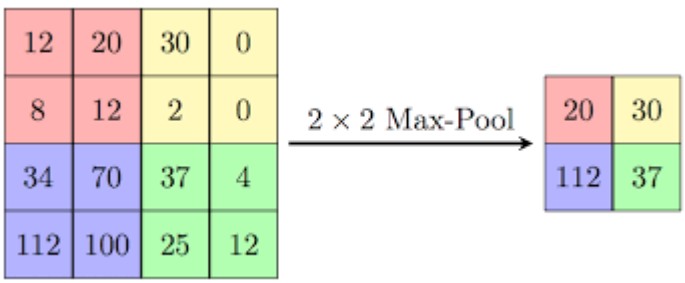

Figura 18: Funcionamiento de MaxPool2D (Papers with code)

Por otro lado, está el bloque denso, estos bloques hacen un cuello de botella conectados de manera forward, estos bloques funcionan como conectores y se originaron en la estructura DenseNet, de donde sale su nombre.

```
def dense_block(units, drop_rate):
    block = tf.keras.Sequential([
        tf.keras.layers.Dense(units, activation='selu'),
        tf.keras.layers.BatchNormalization(),
        tf.keras.layers.Dropout(drop_rate)
    ])
    return block
```

En nuestra función tenemos dos variables, units y drop_rate, las unidades dimensionan el tamaño del valor de salida[31], que será elegido en la estructura, y la segunda variable es el drop_rate. El drop_rate es la cantidad de parámetros que ponemos a 0, el resto de los parámetros que no ponemos a 0 se verán multiplicados por un factor de $\frac{1}{1-Drop\_rate}$, de tal manera que la suma de todos los inputs no se ve afectada.

En este bloque volvemos a usar la función de activación Selu, y luego aplicamos la normalización, al igual que en el bloque de convolución.

La siguiente función es una que nos permite hacer un ratio de aprendizaje dinámico

```
def exponential_decay(LearningRate,s):
    def exponential_decay_fn(epoch):
        return LearningRate * 0.1 **(epoch / s)
    return exponential_decay_fn
exponential_decay_fn = exponential_decay(0.01, 10)
LearningRate= tf.keras.callbacks.LearningRateScheduler(exponential_decay_fn)
```

Este bloque usa llamadas de retorno que nos permite usar las salidas del entrenamiento para cambiar valores previamente definidos, con

LearningRateScheduler[12] podemos adaptar los valores basándonos en la iteración.

Según la documentación de Keras, esta clase tiene dos argumentos, el primero es Schedule y es la función mediante la cual varia el LearningRate, el segundo argumento es verbose y si su valor es 1 hará mensajes con la información de la actualización.

Aparte de esta clase también tenemos otra función que nos permite aprovecharnos de la salida del entrenamiento, está clase es EarlyStopping, y según la documentación de TensorFlow esta clase permite parar el entrenamiento cuando el modelo deja de mejorar.

Esta clase tiene varios argumentos, entre ellos patience regula el número de repeticiones seguidas hasta que el modelo pare el entrenamiento, por defecto está a 0 pero en nuestro caso lo pusimos a 10 por que puede tener muchas variaciones y no queremos que pare el entrenamiento antes de tiempo por un error. También podemos usar el argumento start_from_epoch para elegir a partir de que repetición empieza a monitorizar si el modelo mejora o no.

El otro argumento que nos importa es restore_best_weights, como su nombre indica si es verdadero recuperara los pesos que han dado un mejor valor, por defecto está en falso, pero lo hemos puesto en verdadero.

```
stopping_callback=
tf.keras.callbacks.EarlyStopping(patience=10,start_from_epoch=10,
                                                    restore_best_weights=True)
```

Lo siguiente que se tuvo que hacer era elegir las perdidas del modelo, en TensorFlow hay muchos tipos de perdidas entre los que escoger, pero queremos tener en cuenta la entropía cruzada entre las distintas clases.

La entropía cruzada[24] se define como $H(p,q) = -\sum p(x)\log(q(x))$ y nos permite evaluar la varianza entre dos conjuntos de probabilidades y se usa sobre todo en problemas de clasificación como nuestro modelo, ya que permite calcular como de precisos son las predicciones respecto los resultados reales.

Hay varias clases de entropía cruzada en TensorFlow, pero hay que considerar que estamos usando un modelo donde las etiquetas tienen forma de one-hot donde one-hot se refiere a aquellas en las que las combinaciones de valores solo pueden tener un valor a 1, en nuestro caso una imagen no puede ser clasificado como 'NonDemented' y 'MildDemented' a la vez. Esto es importante porque en estos casos es muy eficaz usar CategoricalCrossentropy [7]. También podemos usar una variante de esta llamada CategoricalFocalCrossentropy, que nos permite tener en cuenta el desbalanceo de clases, pero en nuestro caso no es necesario porque ya lo estamos considerando con las métricas.

6.4 ESTRUCTURA

Lo siguiente que se hizo fue elegir la estructura, se creó una clase para ello en la cual se tiene una primera parte donde están todos los bloques de convolución y una segunda parte donde tenemos los bloques de densidad.

Se eligió usar 7 bloques para cada parte y se eligieron los filtros para cada bloque en potencias de 2 y en el caso de los bloques convolucionales que fuesen aumentando cada vez, también se añadieron capas de pérdidas para evitar el sobre entrenamiento. De tal forma esta parte quedo de la siguiente manera.

```
convolutional_block(16),
        convolutional_block(32),
        convolutional_block(64),
        convolutional_block(128),
        convolutional_block(256),
        tf.keras.layers.Dropout(0.1),
        convolutional_block(512),
        convolutional_block(1024),
        tf.keras.layers.Dropout(0.2),
```

En la siguiente parte, añadimos una capa de aplanamiento, la cual permite remodelar el tensor a uno de una única dimensión. Después añadimos 7 bloques

densos y distribuimos las unidades en potencias de 2 y que disminuye, para que cuando añadamos una última capa que convierta la salida del último bloque en una única salida no tener demasiadas variables.

Por último, añadimos la capa final en la cual usaremos una función de activación distinta, usaremos softmax la cual es una generalización de la función logística y nos permite aplastar los valores dentro de un rango, además tiene en cuenta si el estímulo que recibe es pequeño o grande. Otro beneficio que tiene es que debido a su definición donde

$Softmax(z) = \frac{\exp(z)}{\sum \exp(z)}$, se puede aproximar que la exponencial cancela el logaritmo en la entropía cruzada lo cual permite que las perdidas se asemejen a unas lineares, esto permite que el gradiente sea constante y que el modelo se corrija rápidamente.

```
dense_block(1024,0.1),
        dense_block(512,0.1),
        dense_block(256,0.1),
        dense_block(128, 0.1),
        dense_block(64, 0.1),
        dense_block(32, 0.1),
        dense_block(16, 0.1),


        tf.keras.layers.Dense(3, activation='softmax')
```

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 sequential_15 (Sequential)  (None, 88, 88, 16)        571

 sequential_16 (Sequential)  (None, 44, 44, 32)        2160

 sequential_17 (Sequential)  (None, 22, 22, 64)        7392

 sequential_18 (Sequential)  (None, 11, 11, 128)       27072

 sequential_19 (Sequential)  (None, 5, 5, 256)         103296

 dropout_9 (Dropout)         (None, 5, 5, 256)         0

 sequential_20 (Sequential)  (None, 2, 2, 512)         403200

 sequential_21 (Sequential)  (None, 1, 1, 1024)        1592832

 dropout_10 (Dropout)        (None, 1, 1, 1024)        0

 flatten_1 (Flatten)         (None, 1024)              0

 sequential_22 (Sequential)  (None, 1024)              1053696

 sequential_23 (Sequential)  (None, 512)               526848

 sequential_24 (Sequential)  (None, 256)               132352

 sequential_25 (Sequential)  (None, 128)               33408

 sequential_26 (Sequential)  (None, 64)                8512

 sequential_27 (Sequential)  (None, 32)                2208

 sequential_28 (Sequential)  (None, 16)                592

 dense_15 (Dense)            (None, 3)                 51

=================================================================
Total params: 3894190 (14.86 MB)
Trainable params: 3886062 (14.82 MB)
Non-trainable params: 8128 (31.75 KB)
_________________________________________________________________
```

Figura 19 : Estructura del modelo

Lo siguiente que había que hacer era elegir el optimizador a usar para el modelo y para ello comprobamos el rendimiento de los distintos optimizadores, entre los cuales se eligieron lo más importantes como SGD (Stochastic Gradient Descent), RMSprop o Adam (Adaptive Moment Estimation). Para la prueba se usó el modelo y en el entrenamiento se hicieron 10 iteraciones.

SGD es un optimizador simple y efectivo que aproxima clasificadores lineales es especialmente bueno para disminuir la carga computacional y sobre todo cuanto tenemos grandes conjuntos de datos, ya que usa subconjuntos para calcular el gradiente. Se basa en el Método de Robbins–Monro el cual es un algoritmo de optimización estocástica usado para estimar parámetros de un modelo matemático

en el que haya ruido, este método es particularmente bueno cuando los datos son incompletos.

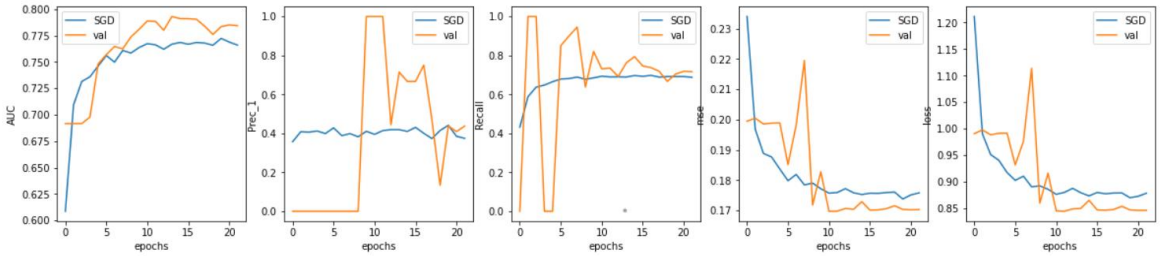

Figura 20 : Entrenamiento del SGD

```
29/29 [==============================] - 10s 343ms/step - loss: 0.8825 - AUC: 0.7652 - Prec_1: 0.0000e+00 - Recall: 0.6979 - ms
e: 0.1791
```

Figura 21 : Test del SGD

Por otro lado, esta RMSprop, este algoritmo de optimización es una variación del SGD propuesta por Geoffrey Hinton, en la que se busca evitar acumular los gradientes pasados, para eso usa ventanas de gradientes para considerar solo los más recientes, luego se retoca con una media exponencial ponderada con el fin de evitar cambios bruscos. También se caracteriza por usar tasas de aprendizaje variable.

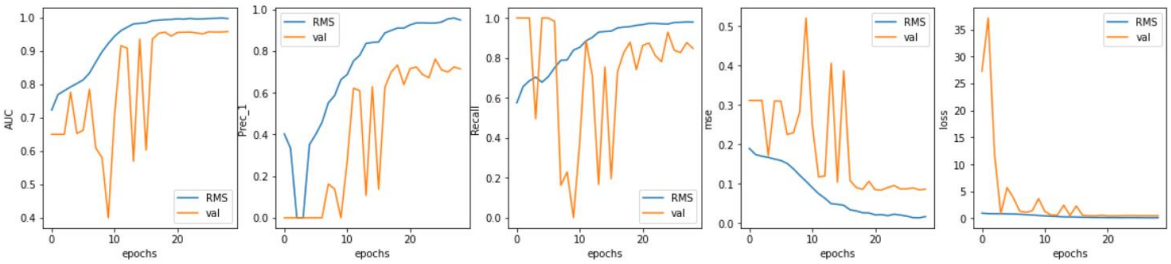

Figura 22: Entrenamiento del RMSprop

```
29/29 [==============================] - 9s 304ms/step - loss: 0.4277 - AUC: 0.9563 - Prec_1: 0.7782 - Recall: 0.9125 - mse: 0.
0760
```

Figura 23: Test del RMSprop

Adam es un algoritmo que aprovecha las características de RMSprop y de otro algoritmo llamado Momentum (el cual no está disponible en Tensorflow). Adam

hace una estimación del momento y de la magnitud de los gradientes previos para actualizar los parámetros en cada iteración. Además de adaptar la tasa de aprendizaje al igual que el RMSprop.

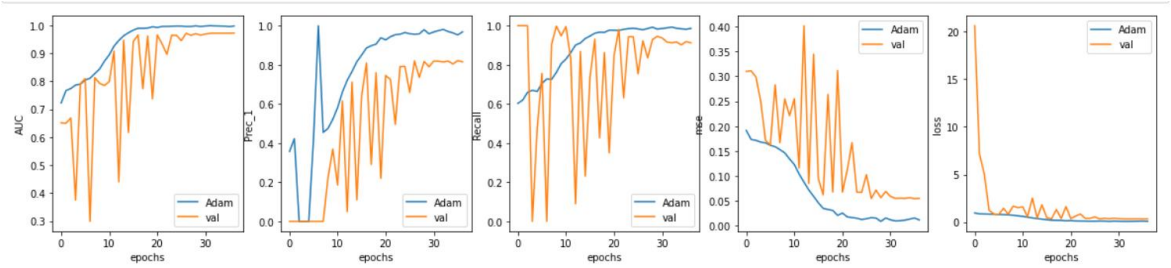

Figura 24 : Entrenamiento de Adam

```
modelo.evaluate(X_test,y_test)
```

```
29/29 [==============================] - 9s 321ms/step - loss: 0.3481 - AUC: 0.9709 - Prec_1: 0.8039 - Recall: 0.9312 - mse: 0.
0615
```

Figura 25: Test de Adam

El último algoritmo que se uso fue AdamW, el cual es una variación del Adam previamente mencionado. La principal diferencia es que la implementación de AdamW en Keras aplica el decaimiento de pesos después de la actualización de parámetros, mientras que la implementación de referencia aplica el decaimiento de pesos antes de la actualización de parámetros, lo cual demuestran que conduce a un mejor rendimiento en la práctica.

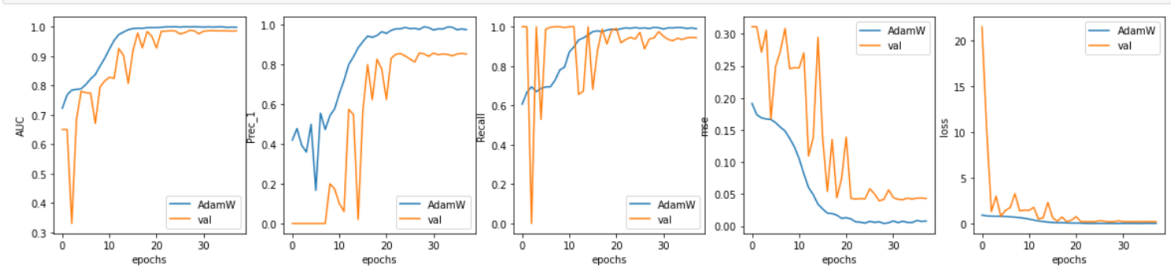

Figura 26 : Entrenamiento de AdamW

```
modell.evaluate(X_test,y_test)
```

```
29/29 [==============================] - 9s 314ms/step - loss: 0.2315 - AUC: 0.9863 - Prec_1: 0.8526 - Recall: 0.9458 - mse: 0.
0402
```

Figura 27 : Test de AdamW

Del entrenamiento se pueda sacar una cosa y es que el modelo SGD es el que peor rendimiento da, pero si nos fijamos en el resto no parecen dar un rendimiento muy distinto, por eso usaremos el test para decidir uno entre los tres, como se puede ver en la Figura 27, el optimizador AdamW consigue hasta un 5% más de precisión que el resto de los optimizadores en el test, por lo que podemos considerar que no es casuistica y que realmente funciona mejor.

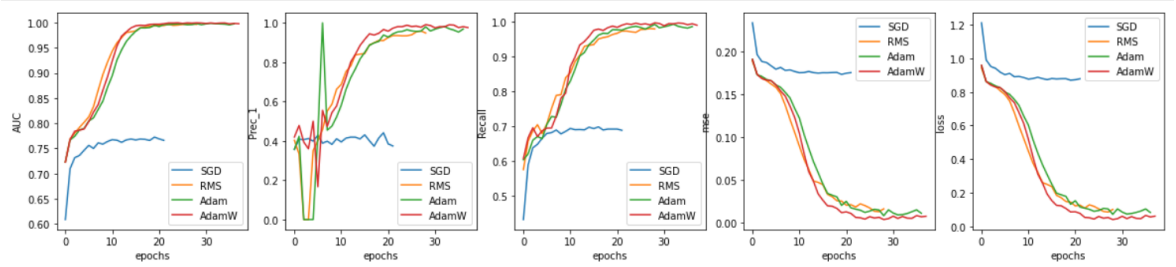

Figura 28 Prueba de los distintos optimizadores

Como se puede apreciar, el modelo da un AUC de aproximadamente un 98% lo cual esta muy bien y por encima de los resultados esperados, aunque por otro lado no se puede negar que la precisión es de un 0.85 lo cual significa que 3 de cada 20 casos de demencia muy leve no se detecta como tal, puede ser que el modelo considere que no tiene demencia o que lo considere leve.

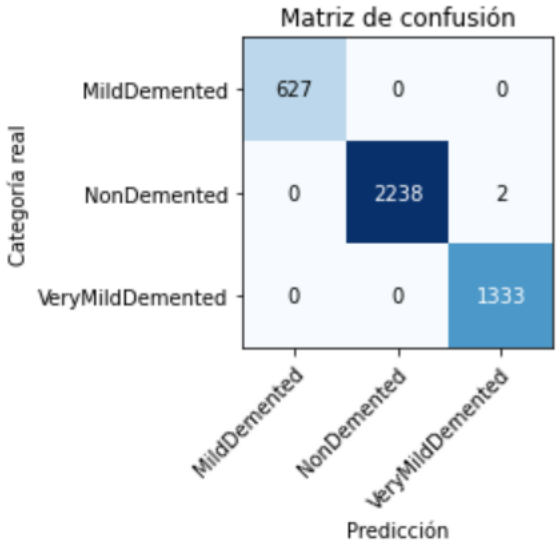

Figura 29: Matriz de confusión del entrenamiento de AdamW

Como se puede ver, la matriz de confusión del entrenamiento da una precisión del 99.85% para la clase 'VeryMildDemendted' y un 100% para la clase 'NonDemented' y 'MildDemented', lo cual es un resultado muy bueno.

Por último, se muestra l matriz de confusión del test del modelo, donde se puede ver que la distribución de los casos que se diagnostican mal para la demencia leve está distribuida entre ambos extremos.

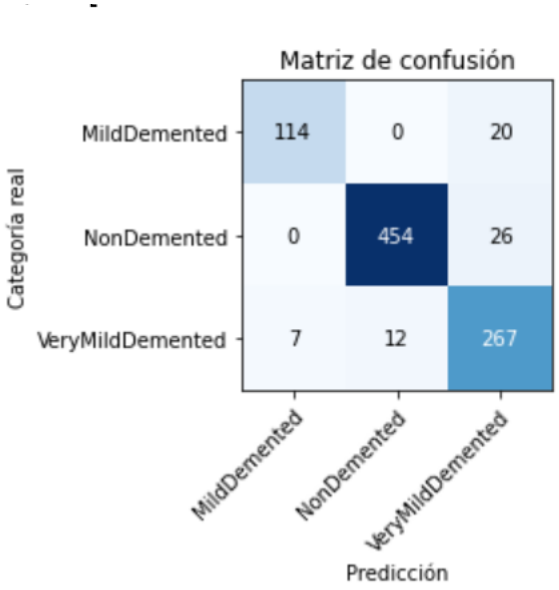

Figura 30 : Matriz de confusión del test

Una vez ya tenemos todo solo hay que guardar el modelo con la función model.save('Nombre_del_modelo.h5')

6.5 API

En nuestro proyecto queríamos que fuese accesible y de manera fácil, para eso usaremos una api desde la cual se pueda consultar al modelo.

Construimos una clase que haga la logica de consultar al modelo, para eso tendremos en cuenta que nos habran pasado la imagen

```
def pred_npy(img,modelo):
    class_names = [ 'MildDemented','NonDemented', 'VeryMildDemented']
    x_train=load_img(img)
```

```
    lista=modelo.predict(x_train)[0]#Se podria añadir una logica interna en caso
de que el modelo no este muy seguro entre opciones
    results=pd.Series(lista).idxmax()
    if(lista[results]>0.8):
        text=("The model is very certain of the result :" +class_names[results])
    elif(lista[results]>0.6):
         text=("The model is  certain of the result :" +class_names[results])
    else:
         text=("The model is not certain of the result :" +class_names[results])
    print(text)
    return text
```

En este código, se ha añadido diferentes niveles de seguridad, de manera que cuando el resultado no sea muy claro, la persona que lo esté viendo sea consciente de ello.

También se ha tenido que añadir un código para trabajar la imagen y dejarla en el formato que queremos, se fuerza el tamaño de la imagen para que sea el que usamos en el modelo y se convierte a numpy.

```
def load_img(img):
    Size=(176,176)
    images = []
    labels = []
    img.save('img.jpg')
    labels.append( 'None')
    images.append('img.jpg')
    df = pd.DataFrame({'image': images,'label':labels})
    work_dr = ImageDataGenerator(
    rescale = 1./255
    )
    train_data_gen = work_dr.flow_from_dataframe(df,x_col='image',
y_col='label',target_size=Size, batch_size=1, shuffle=False)
    train_data,train_labels= train_data_gen.next()
    os.remove('img.jpg')
    return train_data
```

Lo último que nos queda es crear la conexión con el servicio, para ello usaremos Flask y crearemos un HTML mediante el cual podamos enviarle las imágenes.

```
from flask import Flask, request,jsonify
from flask_cors import CORS
import json
import requests
import time
from PIL import Image as im
from keras.models import load_model
modelo = load_model('AD_model.h5')
app= Flask(__name__)
CORS(app)
@app.route('/upload',methods=['GET', 'POST'])
def im():
    image = request.files.get('image')
    predict=pred_npy(image,modelo)
    return predict
app.run(port=8000)
```

El código es muy sencillo, instanciamos el modelo y abrimos un puerto que espere llamadas, cuando llegue una llamada recogerá la imagen y ejecutara las funciones necesarias para poder hacer la predicción, CORS es una función que nos permite enviar consultas en local y enviar de vuelta una respuesta, esto era necesario para poder hacer las comprobaciones de Flask.

Para la parte front-end se ha creado un HTML sencillo, ya que no era la parte importante del proyecto y crear la lógica llevaría mucho tiempo, además de que habría que añadir medidas de seguridad y de control.

Figura 31 : Captura de la web HTML

Como se puede ver en la imagen, hay dos botones, el primero permite cargar la imagen desde el dispositivo, y el segundo bloque es el que envía la imagen al servidor, luego el servidor si está activo hará la predicción y devolverá un texto diciendo cual es la predicción y el grado de seguridad.

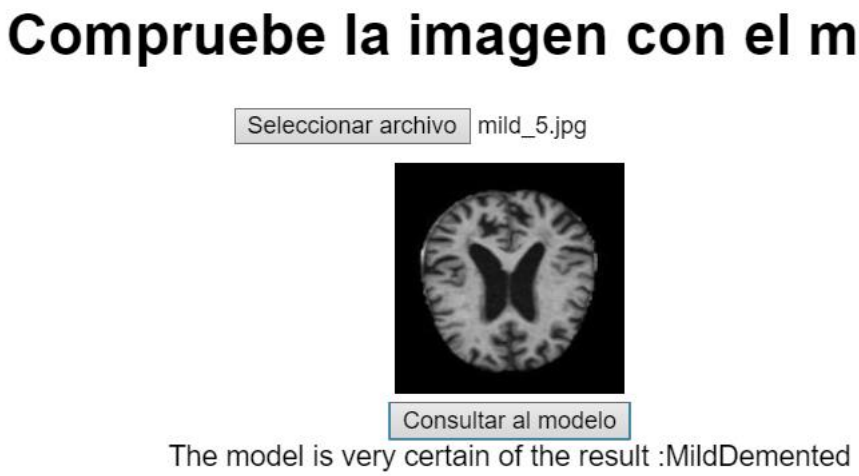

Figura 32 : Respuesta muy segura del modelo

Como se puede ver el modelo da una respuesta con la clase y con el grado de seguridad. En este caso el modelo predice la clase correctamente, cuando responde con un 'very certain' significa que le da un valor mayor de 0.9 para esa clase, es decir esta al menos un 90% seguro que es de esa clase.

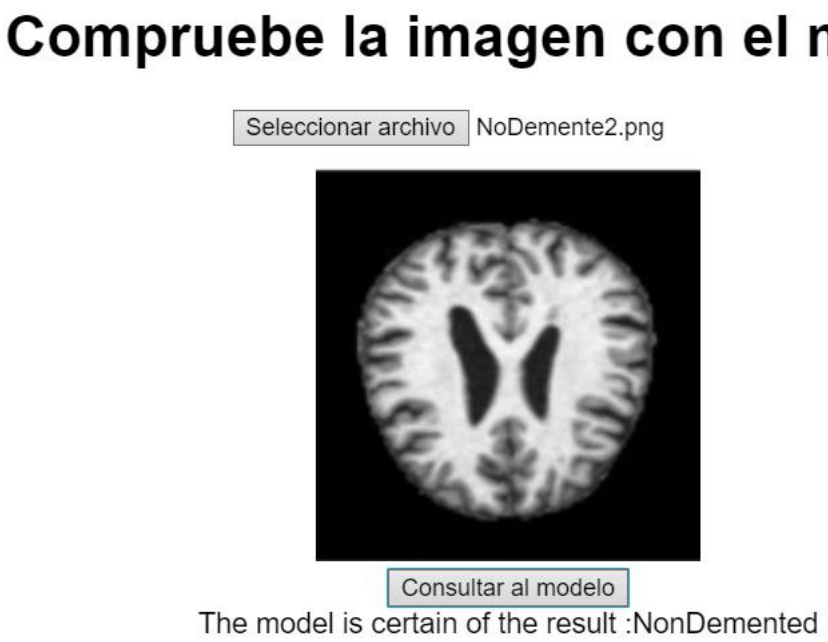

Figura 33: Respuesta segura del modelo

Como se puede apreciar en la imagen, en esta ejemplo el modelo está menos seguro que en el caso anterior, en este caso el modelo responde con un simple 'certain' cuando el valor esta entre 0.7 y 0.9.

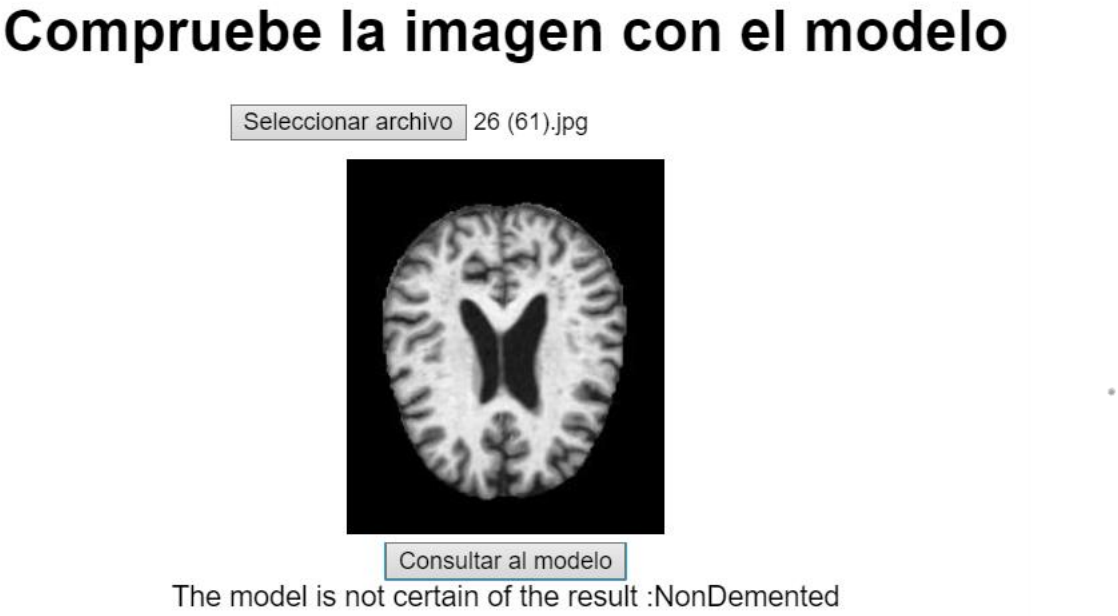

Figura 34 : Respuesta poco segura del modelo

Como se puede ver en la figura el modelo no está seguro, esto pasa cuando el mayor valor está por debajo de 0.7, ademas en este caso se equivoca ya que clasifica como ‘NoDemented’ una imagen que es de ‘VeryMildDemented’.

7. CONCLUSIONES Y TRABAJOS FUTUROS

En las conclusiones hay que destacar que se ha encontrado que la mejor función de activación s SeLu por encima de otros como ReLu o Sigmoid, a cambio de aumentar el tiempo de computación. Por otro lado, se ha encontrado que el mejor optimizador es AdamW con una gran diferencia respecto SGD, pero con menor respecto RMSprop y Adam, aun asi en el test se pudó notar una mejoria de hasta 5% respecto a este último.

Tambien hay que destacar los resultados obtenidos en la matriz de confusión de la Figura 30, un AUC del 98% es más del que se buscaba con este proyecto que era del 95% , pero no por eso se puede negar la parte negativa, según se puede ver en la: Matriz de confusión del test la clase VeryMildDemented tiene Recall del 93%, que está dentro de lo que se queria incluso un poco por encima, esto significa que habra menos falsos negativos para esta clase de los que esperabamos. Tambien hay que considerar que no es igual de importante que el modelo considere a un 'VeryMildDemented' como 'MildDemented' respecto el error que es que lo considere 'NonDemented', desde ese punto de vista tenemos que poner mayor enfasis en los errores entre estás dos clases ya que la diferencia es mucho mayor entre ellas que la que hay entre 'VeryMildDemented' y 'MildDemented'.

Para la clase NonDemented encontramos que tiene un Recall del 94.58%, en esta clase se esperaba que el Recall aumentase, pero aun asi está por debajo de lo esperable teniendo en que ha tenido muchos más datos que el resto de clases, eso significa que habra más falsos negativos, para ser especificos en la clase 'VeryMildDemented', lo cual era esperable por que deberia ser la clase más cercana.

Para la clase 'MildDemented' encontramos un Recall del 85%, lo cual esta por debajo de lo esperado y signifca que habra muchos más falsos negativos que lo esperado, por suerte todos los falsos negativos están en la clase 'VeryMildDemented', que como ya hemos mencionado, no es tan malo que el modelo se equivoque entre estas dos clases respecto el peor caso que es si se equivoc con la clase 'NonDemented' por que en ese caso puede provocar que una persona no llegue a tener el tratamiento lo suficientemente rapido, en ese sentido podemos pensar que los resultados no son tan malos.

Por otro lado, ha dado una precisión del 92,77% lo cuala es un valor muy bueno en terminos generales, esto significa que de cada 100 muestras acierta más o menos 93.

Para trabajos futuros habría dos cosas a trabajar, la primera sería el servicio para conectarse al modelo, podría hacerse mediante una aplicación o una página web más trabajada y que tuviera medidas para ayudar en caso de que el modelo acertase, con información sobre la enfermedad o con información sobre clínicas donde pueda recibir un tratamiento temprano.

Lo otro a trabajar es algo que en un principio era un objetivo del proyecto, pero debido a la falta de datos no era posible, y es que se quería extrapolar el modelo a varias enfermedades, se quería poder usar el modelo para saber si en general podría haber alguna enfermedad y no tener que crear un modelo para cada enfermedad.

8. BIBLIOGRAFÍA

[1] Murzone, F. (2022, 30 marzo). Funciones de activación para redes neuronales - EscuelaDeInteligenciaArtificial-Medium.*Medium*. https://medium.com/escueladeinteligenciaartificial/funciones-de-activación-para-redes-neuronales-de00fefb7150

[2] Finance, J. (2023, 23 agosto). *Tipos de redes neuronales (Clasificación)*. Inteligencia-Artificial.dev. https://inteligencia-artificial.dev/tipos-redes-neuronales/#Clasificacion_por_el_numero_de_capas

[3] Neira, P. (2023, 30 junio). Descubriendo el Futuro de la Salud Digital: Cómo las Redes Neuronales Convolucionales (CNN) Transforman Imágenes en Datos Significativos. https://www.linkedin.com/pulse/descubriendo-el-futuro-de-la-salud-digital-c%C3%B3mo-las-redes-paola-neira/

[4] Redes de función de base radial (RBF). (s. f.). https://www.ibiblio.org/pub/linux/docs/LuCaS/Presentaciones/200304curso-glisa/redes_neuronales/curso-glisa-redes_neuronales-html/x185.html

[5] *¿Qué son las redes neuronales recurrentes? | IBM*. (s. f.). https://www.ibm.com/es-es/topics/recurrent-neural-networks

[6] *¿Qué es una red neuronal? - Explicación de las redes neuronales artificiales - AWS*. (s. f.). Amazon Web Services, Inc. https://aws.amazon.com/es/what-is/neural-network/

[7] Ignacio G.R: Gavilán. (s. f.). Catálogo de componentes de redes neuronales (III): funciones de pérdida. http://bluechip.ignaciogavilan.com/2020/05/catalogo-de-componentes-de-redes_25.html

[8] Barreto, W. S. U., & Ygnacio, M. A. C. (2024). Sistema de Diagnóstico del Alzheimer basado en Imágenes de Resonancia Magnética mediante el Algoritmo VGG16. http://portal.amelica.org/ameli/journal/602/6024790007/html/

[9] pTorres-Vásquez, M., Hernández-Torruco, J., Hernández-Ocaña, B., & Chávez-Bosquez, O. (2021). Impacto de los algoritmos de sobre muestreo en la clasificación de subtipos principales del síndrome de Guillain-barré. https://www.redalyc.org/journal/5055/505565143002/html/

[10] Accuracy vs. precision vs. recall in machine learning: what's the difference? (s. f.). https://www.evidentlyai.com/classification-metrics/accuracy-precision-recall#:~:text=Accuracy%20is%20a%20metric%20that,often%20the%20model%20is%20right%3F

[11] Gonzalez, L. (2023, 2 mayo). *Curvas ROC y Área bajo la curva (AUC)*. Aprende IA. https://aprendeia.com/curvas-roc-y-area-bajo-la-curva-auc-machine-learning/

[12] Team, K. (s. f.). *Keras documentation: LearningRateScheduler*. https://keras.io/api/callbacks/learning_rate_scheduler/

[13] Medina, A. (2024, 8 mayo). Deep Learning: qué es, impacto y aplicación en e-learning. EvolMind. https://www.evolmind.com/blog/deep-learning-que-es-impacto-y-aplicacion-en-e-learning/

[14] Vicente, F. R. (2023, 29 junio). Las matemáticas del Machine Learning: Redes Neuronales (Parte I). Telefónica Tech. https://telefonicatech.com/blog/las-matematicas-del-machine-learning-redes-neuronales-parte-i

[15] Moolayil, J. J. (2021, 11 diciembre). A Layman's Guide to Deep Neural Networks - Towards Data Science. *Medium*. https://towardsdatascience.com/a-laymans-guide-to-deep-neural-networks-ddcea24847fb

[16] *El perceptrón multicapa | Interactive Chaos*. (s. f.). https://interactivechaos.com/es/manual/tutorial-de-deep-learning/el-perceptron-multicapa

[17] Wikipedia contributors. (2024, 3 junio). *Convolutional neural network*. Wikipedia. https://en.wikipedia.org/wiki/Convolutional_neural_network

[18] GeeksforGeeks. (2024, 5 Junio). *Introduction to Recurrent Neural Network*. GeeksforGeeks. https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/

[19] McCormick, C. (2013, 16 Agosto). *Radial Basis Function Network (RBFN) tutorial*. Chris McCormick. https://chrisjmccormick.wordpress.com/2013/08/15/radial-basis-function-network-rbfn-tutorial/

[20] De Ingeniería del Conocimiento, I. (2024, 20 marzo). *Transformers en Procesamiento del Lenguaje Natural*. Instituto de Ingeniería del Conocimiento. https://www.iic.uam.es/innovacion/transformers-en-procesamiento-del-lenguaje-natural/

[21] Merino, M. (2019, 31 marzo). *Conceptos de inteligencia artificial: qué son las GANs o redes generativas antagónicas*. Xataka. https://www.xataka.com/inteligencia-artificial/conceptos-inteligencia-artificial-que-gans-redes-generativas-antagonicas

[22] *Accuracy vs. precision vs. recall in machine learning: what's the difference?* (s. f.-c). https://www.evidentlyai.com/classification-metrics/accuracy-precision-recall

[23] *¿Qué son las redes neuronales convolucionales? | IBM*. (s. f.-b). https://www.ibm.com/es-es/topics/convolutional-neural-networks

[24] Artem. (2023, 15 septiembre). *Cruzentropía, loglosa y perplejidad: diferentes facetas de la probabilidad*. HackerNoon. https://hackernoon.com/es/entropia-cruzada-loglosa-y-perplejidad-diferentes-facetas-de-verosimilitud

[25] Enfermedad de Alzheimer - Diagnóstico y tratamiento - Mayo Clinic. (2024, 13 febrero). https://www.mayoclinic.org/es/diseases-conditions/alzheimers-disease/diagnosis-treatment/drc-20350453

[26] Thijssen, E. H., La Joie, R., Wolf, A., Strom, A., Wang, P., Iaccarino, L., Bourakova, V., Cobigo, Y., Heuer, H., Spina, S., VandeVrede, L., Chai, X., Proctor, N. K., Airey, D. C., Shcherbinin, S., Evans, C. D., Sims, J. R., Zetterberg, H., Blennow, K., . . . Boxer, A. L. (2020). Diagnostic value of plasma phosphorylated tau181 in Alzheimer's disease and frontotemporal lobar degeneration. *Nature Medicine*, *26*(3), 387-397. https://doi.org/10.1038/s41591-020-0762-2

[27] Maragall, F. P. (2024, 6 mayo). ¿En qué consiste y para qué se utiliza el test Mini-Mental? https://blog.fpmaragall.org/mini-mental-test

[28] ¿Puede una prueba genética directa para el consumidor decir si tendré Alzheimer?: MedlinePlus Genetics. (s. f.). https://medlineplus.gov/spanish/genetica/entender/pruebasdirectasalconsumidor/alzheimer/

[29] *Diagnóstico*. (s. f.). Alzheimer's Disease And Dementia. https://www.alz.org/alzheimer-demencia/diagnostico

[30] Poveda,P. (2023, junio). Aplicación de CNN al diagnóstico de la enfermedad del alzheimer a partir de MRI https://oa.upm.es/75167/1/TFM_PATRICIA_POVEDA_HERNANDEZ.pdf

[31] Team, K. (s. f.-a). *Keras documentation: Dense layer*. https://keras.io/api/layers/core_layers/dense/

9. ANEXO I: ALINEACIÓN DEL PROYECTO CON LOS ODS

Los ODS son un conjunto de objetivos que buscan lograr un futuro mejor, de todos ellos el ODS que se ve más afectado por este proyecto sería el de bienestar y salud.

Este objetivo se basa en promover una vida sana y bienestar en todas las edades, para ello se proponen diferentes ideas como una financiación más eficiente de los sistemas sanitarios, una mejora de la higiene, o un mayor acceso al personal médico. Además, debido a la reciente emergencia sanitaria del COVID-19, se ha puesto mucho énfasis en este objetivo y en asegurarse una mejor preparación para posibles emergencias futuras.

Dentro de este objetivo la meta a la que más afecta sería el 3.4, que busca **reducir en un tercio la mortalidad prematura por enfermedades no transmisibles mediante la prevención y el tratamiento y promover la salud mental y el bienestar.** Para lo cual es importante la detección de estas enfermedades para poder tratarlas.

También afectaría, aunque en menor medida, a la meta 3.d que busca **reforzar la capacidad de todos los países, en particular los países en desarrollo, en materia de alerta temprana, reducción de riesgos y gestión de los riesgos para la salud nacional y mundial**. Al igual que los objetivos de este proyecto también busca reforzar la capacidad en materia de alerta temprana para el tratamiento de la enfermedad.

También afecta al objetivo de ciudades y comunidades sostenibles, ya que como se ha indicado previamente, este proyecto buscar aliviar la carga del personal médico para que se pueda ayudar a más personas a un coste bajo, lo cual puede ser específicamente útil para la meta 11.1 **asegurar el acceso de todas las**

personas a viviendas y servicios básicos adecuados, seguros y asequibles y mejorar los barrios marginales.