



ESTIMACIÓN DE LA GRAVEDAD DEL ALZHEIMER SEGÚN SÍNTOMAS, USANDO MACHINE LEARNING EN IMÁGENES DE RESONANCIA MAGNÉTICA (MRI).

Ing. Jhonatan Avellaneda Gutierrez*, MSc. Luis Muñoz Bedoya*

*Universidad de Pamplona, Facultad de Ingenierías y Arquitectura, Ingeniería Electrónica, Grupo de investigación GIBUP.

Km 1 Vía Bucaramanga Ciudad Universitaria, Pamplona, Norte de Santander, Colombia.
(57+7)5685303

Email: {jhonatan.avellaneda, luismb}@unipamplona.edu.co

RESUMEN:

El objetivo principal de esta investigación fue desarrollar una herramienta web interactiva para estimar la gravedad de la enfermedad del Alzheimer, usando aprendizaje automático en imágenes de resonancia magnética que corresponden a cuatro clases que son: demencia leve, demencia moderada, no demencia, demencia muy leve. Las imágenes contenidas en el dataset fueron preprocesadas con dos métodos que son: zero padding para redimensionar las imágenes sin modificarlas y la corrección del campo de sesgo. En el planteamiento inicial se tenía pensando comparar distintos modelos de Machine Learning, los modelos que entrenamos fueron: CNN, VGG 19, Inception V3, MobileNet V3 y EfficientNet. Para el entrenamiento de estos modelos se utilizó el lenguaje Python y la librería TensorFlow, como se requerían recursos más potentes se empleó Google Colab. Para la página web utilizamos HTML 5 para diseñarla, JavaScript para algunas funcionalidades y CSS para definir los estilos. Con respecto a los resultados, algunos modelos tenían overfitting y otros poseían problemas en el comportamiento de las métricas, los modelos con mejores resultados fueron CNN con aumento de datos, VGG 19 e Inception V3 con una precisión de 97.9%, 99.8% y 90% respectivamente. Con el desarrollo de este trabajo se concluyó que utilizar transferencia de aprendizaje no garantiza que se obtenga un buen resultado, sino que esto depende del dataset que estemos utilizando para entrenar nuestros modelos. Adicionalmente, se concluye que al entrenar modelos de Machine Learning se puede presentar sobreajuste, pero es posible solucionarlo con técnicas que pueden contribuir a obtener un mejor rendimiento.



1. PALABRAS CLAVES

Alzheimer, Machine Learning, interfaz web, Python, MRI, CNN, VGG 19, Inception V3, MobileNet V3 y EfficientNet.

2. ABSTRACT

The main objective of this research was to develop an interactive web tool to estimate the severity of Alzheimer's disease, using Machine Learning in MRI corresponding to four groups: mild demented, moderate demented, non demented, very mild demented. The images of the dataset were preprocessed, two methods were used: zero padding to resize the images without modifying them and the correction of bias field. The initial approach was to compare different Machine Learning models, the models we trained were: CNN, VGG 19, Inception V3, MobileNet V3 and EfficientNet. For the training of these models, we used the Python language and the TensorFlow library; as more powerful resources were required, we used Google Colab. For the web page we use HTML5 to design it, JavaScript for some features and CSS to define the styles. Regarding the results, some models had overfitting and others had problems in the behavior of the metrics, the models with the best results were CNN with data augmentation, VGG 19 and Inception V3 with an accuracy of 97.9%, 99.8% and 90% respectively. With the development of this work, it was concluded that using transfer learning doesn't guarantee that a good result will be obtained, but this depends on the dataset that we are using to train our models. Additionally, it is concluded that when training Machine Learning models overfitting can occur, but it is possible to solve it with techniques that can contribute to obtain a better performance.

3. KEYWORDS

Alzheimer, Machine Learning, web interface, Python, MRI, CNN, VGG 19, Inception V3, MobileNet V3 and EfficientNet.

4. INTRODUCCIÓN

Este trabajo se propuso desarrollar porque el Alzheimer se ha posicionado como una de las enfermedades de más rápido crecimiento y se tiene previsto que los casos de esta aumenten cada vez con el pasar de los años, debido al incremento de la población de adultos mayores. Por otro lado, este trabajo surgió como una posible ayuda para los



neurólogos o médicos especialistas para que con sus conocimientos y el prediagnóstico que entrega esta herramienta se les haga más fácil y rápido dar un diagnóstico a los pacientes.

El estado del arte, del cual fue basado el presente trabajo es el siguiente. José Ortega propuso el uso de redes neuronales convolucionales para el desarrollo de un algoritmo que sea capaz de clasificar diferentes pacientes con distintos grados de severidad de la enfermedad del Alzheimer. Para entrenamiento de los modelos utilizó Python 3 y su librería TensorFlow incluidos en la herramienta Google Colab. Desde el punto de vista del autor su trabajo tendrá un gran impacto en el ámbito médico y de la inteligencia artificial debido a que las imágenes de resonancia magnética no tienen diferencias visibles a los ojos de los seres humanos e incluso a los de un profesional de la salud. También tiene como debilidades el tiempo para desarrollar la investigación de su trabajo y desde mi punto de vista creer que su herramienta puede realizar un diagnóstico sin la necesidad de un neurólogo lo cual no es correcto de alguna manera. [1] Valentina Paniza propuso el diseño e implementación de una herramienta que sea capaz de extraer patrones de imágenes de resonancia magnética de pacientes diagnosticados con Alzheimer, algo llamativo de este trabajo fue que uno de ellos objetivos principales de este fue clasificar pacientes sanos con respecto de los que tienen demencia, pero que en sus imágenes tengan patrones cerebrales similares. Se utilizó una unidad de procesamiento de gráficos de alto rendimiento para ejecutar los modelos, de igual manera que estos se entrenaron con Python 3 y su librería TensorFlow. Según la autora su trabajo podría servir como una herramienta para que los neurólogos puedan constituir un diagnóstico de manera más rápida. [2]

5. MARCO TEÓRICO

La enfermedad del Alzheimer y su pronóstico de aumento de casos a futuro es la problemática y la justificación de este trabajo, se puede definir como un trastorno neurodegenerativo progresivo con propiedades clínicas y aspectos patológicos. Esta patología puede ser producida por mutaciones en ciertos cromosomas o por factores casuales que aún a la fecha son desconocidos. [3]



Secciones Transversales Del Cerebro

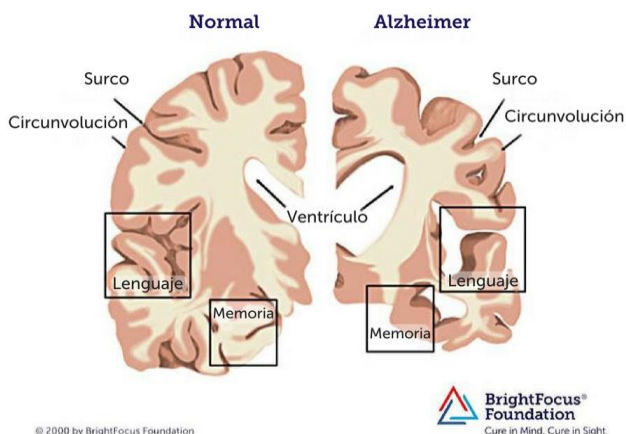


Figura 1. Comparación de un cerebro normal y uno con Alzheimer. [4] Fuente: Afaga Alzheimer.

Machine Learning es una disciplina de la ciencia de datos que aplica procedimientos estadísticos es el cerebro principal de la herramienta que se creó. Se utiliza un aprendizaje supervisado y se hace en el entrenamiento de 5 modelos pertenecientes al aprendizaje automático. Un aspecto relevante del aprendizaje automático es el uso de algoritmos de mejora automática. De la misma forma como los humanos aprenden de una vivencia previa con el método de prueba y error para tomar algunas decisiones, también lo hacen los algoritmos de automejora del aprendizaje automático. [5]

Se utilizó HTML que es un lenguaje semántico de marcado para diseñar la interfaz web. Este lenguaje se complementó con otras tecnologías como lo son CSS para darle estilo a la página web y JavaScript para definir algunas funcionalidades y también para hacer la migración del modelo a la interfaz web se utilizó TensorFlow js para hacer la predicción directamente con JavaScript. [6]

Como se contaba con un computador con recursos óptimos para hacer el entrenamiento de los modelos se utilizó un servicio brindado por la empresa Google, llamado Google Colab. Este ofrece un espacio de trabajo completamente dedicado para la ciencia de datos basado en el lenguaje de programación Python, que comparte semejanzas con la aplicación Jupyter Notebook. [7]



Se ha utilizado algunos modelos con los que se debe aplicar una técnica llama transferencia de aprendizaje, esta técnica utiliza el conocimiento conseguido previamente por una red neuronal para solucionar problemas nuevos de manera más veloz y eficaz. Esta técnica se utiliza para conseguir mejores resultados al momento de entrenar modelos de Machine Learning. [8]

En el entrenamiento de la red neuronal convolucional se presentó un problema llamado sobreajuste, por esta razón se hizo necesario aplicar aumento de datos. Este consiste en incrementar las muestras de un conjunto de datos con transformaciones al azar, de esta manera los datos aumentados representaran un grupo más completo y variado de los datos originales. [9]

En este caso se presentó sobreajuste en uno de los de los modelos entrenados, se puede definir como un problema que se presenta en el momento de trabajar con algoritmos de Machine Learning, este se muestra como el impedimento de generalizar los modelos para que se ajusten bien a los datos vistos en los datos de entrenamiento. [10]

Una de las mejores arquitecturas para clasificar imágenes se trata de las redes neuronales convolucionales, esta es un tipo de red neuronal artificial, está diseñada para procesar datos y para aprender de manera automática jerarquías de características. Básicamente una CNN está constituida por tres tipos de capas: capa convolucional, capa de agrupación (estas dos primeras se dedican de la extracción de características) y capa completamente conectada (esta capa es la encargada de la clasificación). [11]

En las capas convolucionales tenemos una fórmula general que es la siguiente:

$$y_j = g \left(b_j + \sum_i K_{ij} \otimes y_i \right) \quad (1)$$

En la capa de clasificación también tenemos una formula general, que es idéntica a la fórmula de una red neuronal perceptrón multicapa. Esta es:

$$y_j = g \left(b_j + \sum_i \omega_{ij} \cdot y_i \right) \quad (2)$$

Esta segunda ecuación es bastante similar a la ecuación uno, sin embargo, en esta fórmula no tenemos la influencia del operador de convolución.



Se quería buscar los mejores modelos para esta aplicación, la primera se trata de una arquitectura llamada VGG. Es un tipo de arquitectura de red neuronal convolucional, esta arquitectura fue considerable porque ilustro numerosos principios de diseño que se convirtieron en un estándar para arquitecturas futuras. Algo novedoso en VGG es que se redujo el volumen o tamaño de los filtros, pero se aumentó la profundidad. [12]

También se hizo el entrenamiento del modelo Inception V3, este se basa en una red neuronal convolucional, este modelo en su versión 3 tiene un total de cuarenta y ocho capas de profundidad. Inception v3 es una edición de la red ya entrenada con más de un millón de imágenes pertenecientes al banco de datos ImageNet. [13]

De igual manera se entrenó un modelo de redes neuronales convolucionales, llamado MobileNet V3 que fue creado por la compañía Google. Esta arquitectura esta fundamentalmente diseñada con el propósito de poder ser ejecutados en móviles o dispositivos de baja potencia. La estructura de MobileNet V3 integra algunas propiedades que mejoran la eficacia computacional con respecto a MobileNet V2. [14]

Por último, el modelo EfficientNet también se entrenó con la base de datos, este es un modelo de red neuronal convolucional para ser utilizada primordialmente para el reconocimiento de imágenes. A grandes rasgos, este modelo se creó usando bloques residuales invertidos del modelo MobileNet V2 como un tipo de arquitectura combinada con la estrategia de búsqueda de MnasNet. [15]

Para hacer todos los entrenamientos de los modelos se utilizó la librería TensorFlow desarrollada por la empresa Google, que es una librería de software de código abierto para el cálculo numérico. Esta biblioteca hace posible de manera simple y eficaz entrenar y ejecutar redes neuronales de tamaños considerables. [16]

6. METODOLOGÍA

Antes de empezar debemos conocer el dataset que vamos a clasificar, así como la repartición de las clases de las imágenes. El dataset con el que se trabajó contiene un total de 6400 imágenes de resonancia magnética donde se utilizó 5121 para el entrenamiento y 1279 para hacer la validación.

Tabla 1. Información de la cantidad de imágenes las clases. Fuente: Autores

No	Clase	Entrenamiento	Validación
1	Mild Demented	717	179
2	Moderate Demented	52	12
3	Non Demented	2560	640
4	Very Mild Demented	1792	448
Total		5121	1279

En la tabla 1 encontramos la información de cuantas imágenes tenemos de cada una de las clases para los datos de entrenamiento y validación.

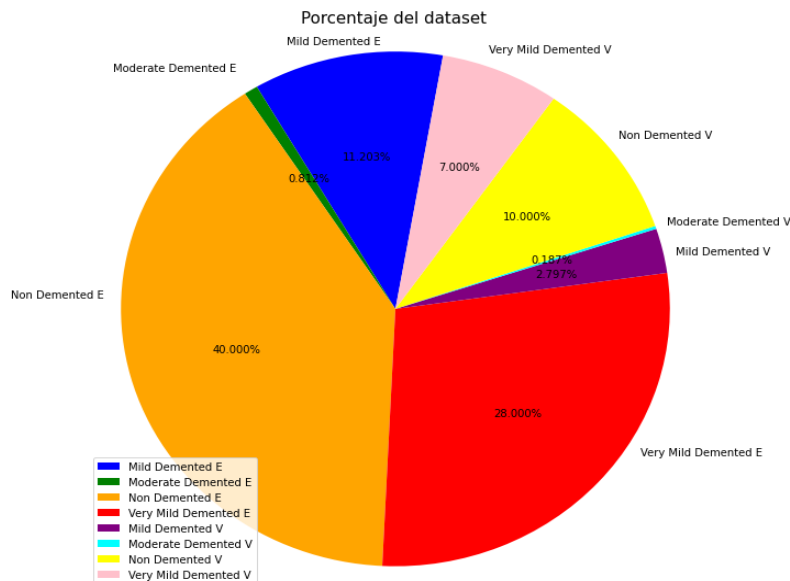


Figura 2. Gráficas circulares de la cantidad de clases. Fuente: Autores

En la figura 2 podemos observar las dos graficas circulares correspondientes a los porcentajes de cada una de las clases tanto en los datos de entrenamiento como en los de validación.

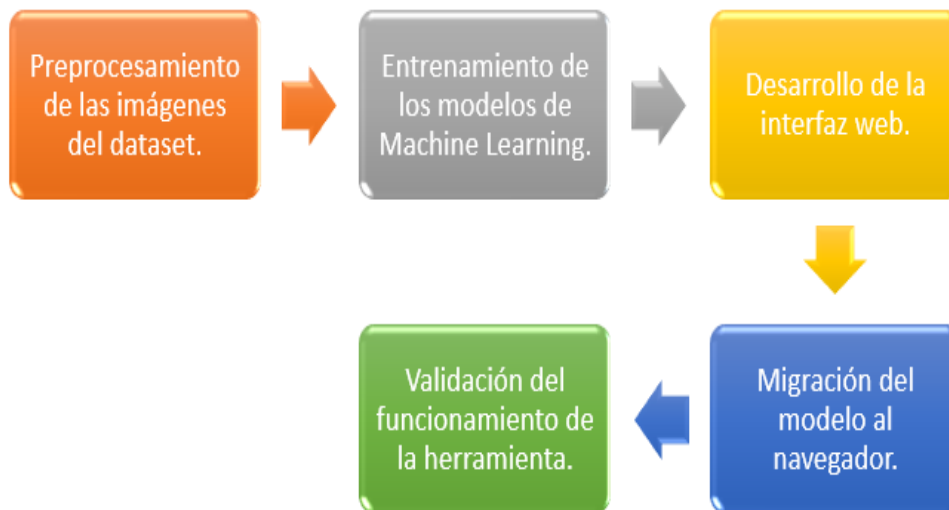


Figura 3. Diagrama de bloques de la metodología. Fuente: Autores

6.1. Preprocesamiento de las imágenes del dataset

Lo primero que se hizo fue el preprocesamiento de las imágenes, se tuvo que preparar las imágenes que íbamos a utilizar para entrenar los modelos de Machine Learning seleccionados con el objetivo de lograr un mejor resultado y teniendo en cuenta que las imágenes del dataset no tiene ningún tipo de preprocesamiento. Se plantearon dos técnicas para preparar las imágenes, la primera se trata de zero padding y la segunda consiste en corregir el campo de sesgo de las imágenes.

6.1.1. Zero padding

Se aplicó zero padding con el objetivo de cambiar el tamaño de las imágenes (176x208) a uno más adecuado (en este caso 208x208), sin alterar las estructuras atómicas de estas, ya que puede se pueden afectar características de interés. Esta técnica consiste en agregar ceros alrededor de la imagen de esta manera se redimensionan sin hacerle modificaciones a los pixeles directamente.

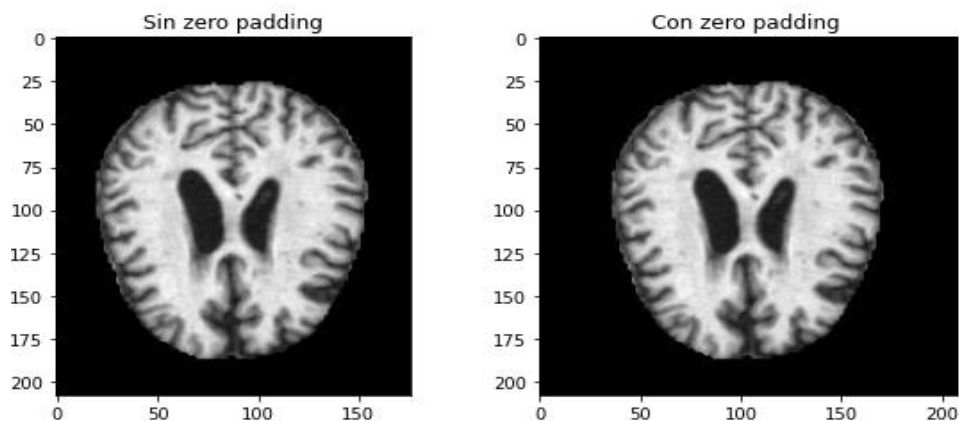


Figura 4. Preprocesamiento imágenes zero padding. Fuente: Autores

6.1.2. Corrección del campo de sesgo

Puede ser por varios motivos, pero las imágenes de resonancia magnética generalmente suelen tener un problema de intensidades en el mismo tejido del cerebro, para solucionar esto se realizó una corrección del BIAS o de intensidades a todas las imágenes. Hay algunas imágenes que el algoritmo considera que el BIAS es correcto.

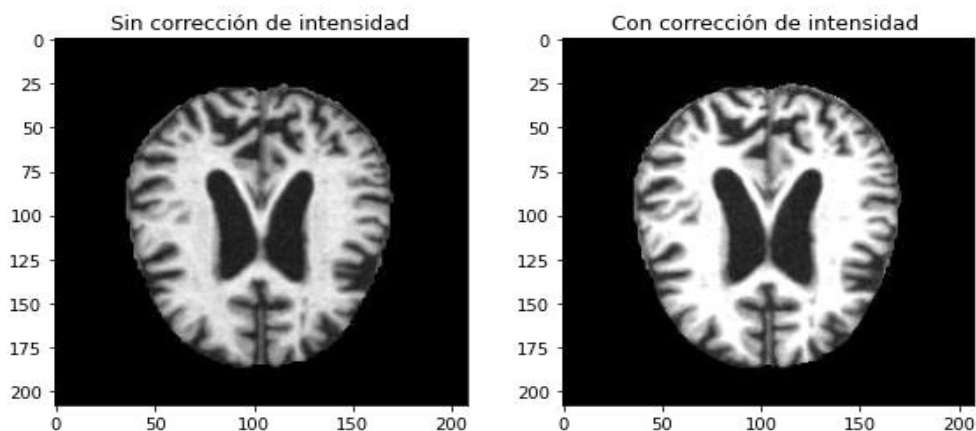


Figura 5. Preprocesamiento imágenes corrección de intensidades. Fuente: Autores

6.2. Buscando el modelo que mejor funcione para la aplicación

Actualmente existen varios modelos que pueden funcionar bastante bien con nuestra aplicación, pero para el cumplimiento de uno de los objetivos de esta investigación se



debía buscar el modelo que de los mejores resultados. Se intentó con un total de 5 modelos estos son: CNN, VGG 19, Inception V3, MobileNet V3 y EfficientNet. En esta sección se plantea el entrenamiento de estos modelos con la base de datos y validación de cada uno con objetivo de seleccionar los modelos con el mejor desempeño.

CNN

Lo primero que se intento fue el entrenamiento de una Red Neuronal Convolutacional que consistente en una arquitectura con 3 capas convolucionales de 16, 32 y 64 neuronas, una capa flatten y dos capas densas la primera con 128 neuronas y la segunda con 4 que son el número de clases a clasificar.

VGG 19

Ahora seguimos con el entrenamiento del modelo VGG 19, para el cual utilizamos transfer Learning para cargar este modelo y congelamos las capas y parámetros de esta arquitectura y a su salida le agregamos una capa densa con el número de clases a clasificar.

Inception V3

Con Inception V3 también se tuvo que aplicar transferencia de aprendizaje con TensorFlow ya que se hace de manera sencilla, después congelamos todas las capas que se pueden entrenar del modelo, agregamos un dropout y por último una capa densa con cuatro neuronas.

EfficientNet

De igual manera para EfficientNet se utilizó transferencia de aprendizaje, cuando cargarnos el modelo pre entrenado con TensorFlow lo primero que se hizo fue congelar las capas propias de este modelo, se le agregó un dropout de 0.20 y por último agregamos una capa densa con cuatro neuronas.

MobileNet V3

Para este último modelo y de la misma manera que los modelos anteriores, se tuvo que aplicar transferencia de aprendizaje para cargar el modelo desde la librería TensorFlow, también debimos congelar todos sus parámetros y capas ocultas. Finalmente, agregamos una capa densa con cuatro neuronas que son el número de clases a clasificar.

6.3. Arreglo del problema del sobreajuste del modelo CNN

Después lo que hizo fue aplicarle aumento de datos a las imágenes del dataset, esta técnica consiste en aplicarle transformaciones aleatorias a las imágenes para así crear nuevas imágenes y hacer más robusto el dataset.

En la figura 6 se muestran algunas de las transformaciones que se hicieron, se intentó de que las transformaciones no afectaran directamente las características de las imágenes. Algunas de estas transformaciones que se aplicaron fueron: rotation range, width shift range, height shift range, de igual manera en estas transformaciones se debe incluir una normalización para tener los valores de las imágenes de 0 a 1 en vez de 0 a 255.

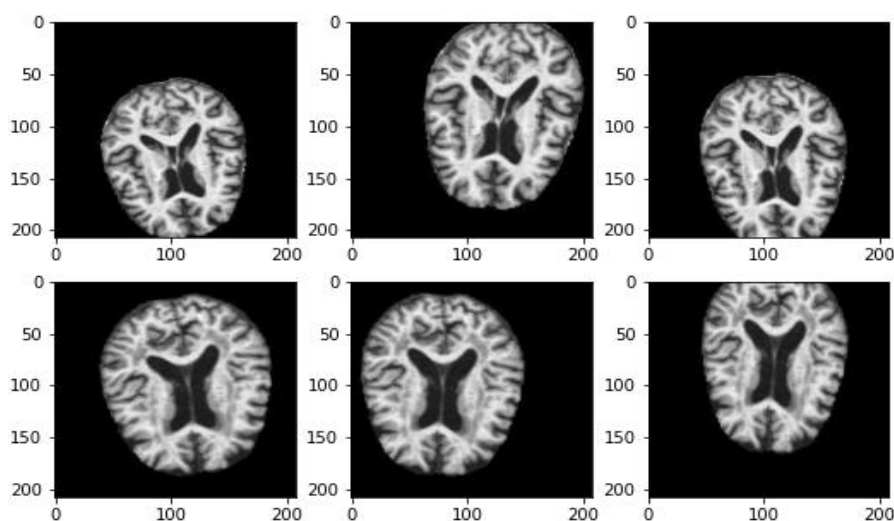


Figura 6. Aumento de datos al dataset. Fuente: Autores

Con el objetivo de mejorar el resultado que se tenía en un momento se optó por utilizar esta técnica, con respecto a los modelos de Inception V3 y VGG 19 se dejaron así porque en términos generales dan buen resultado.

6.3.1. Entrenamiento de CNN con aumento de datos y dropout

Se realizó el entrenamiento de nuevo el modelo de Red Neuronal Convolutiva ahora con el dataset aumentado y también le agregamos un dropout de 0.20.



6.4. Guardamos el modelo entrenado

Después de haber entrenado todos los modelos y seleccionar el mejor, en este caso fue el modelo VGG 19. Se utilizó el comando save de Python para guardar el modelo en formato h5 y se hizo uso de TensorFlow js para convertir este de formato h5 a un archivo json y varios archivos bin que contienen los pesos de la red neuronal.

6.5. Diseño de la página web

Después de entrenar los modelos y seleccionar el mejor, la tarea siguiente fue hacer el diseño de la página web que en este caso fue la interfaz que utiliza la herramienta para hacer las predicciones. Esta página web se dividió en varias partes que se explican a continuación:

Página principal

Esta página funciona como página de inicio, aquí se verá información básica de la herramienta como: ¿qué es? y ¿cómo funciona? esta. También se encuentra un botón que llamado identificate, el cual nos lleva directamente a la página de registro e ingreso. Se intento que con la información básica de esta herramienta se explique de manera muy general pero que al usuario le quede bastante claro estos aspectos que son bastante importantes para comprender el funcionamiento de esta.



Figura 7. Página de inicio. Fuente: Autores



Figura 8. Página de inicio. Fuente: Autores

Página de registro e ingreso

Como esta herramienta fue pensada para que fuera utilizada por los neurólogos, se agregó un formulario para que estos puedan registrarse y posteriormente ingresar. Para este formulario se utilizó una base de datos creada en MySQL que es donde se guardan los datos del usuario al momento de registrarse y se consultan estos al momento de ingresar, para esta última acción se utilizó PHP para enlazar la base datos con la página web. Una vez el usuario ingrese los datos correctamente en el formulario de login, lo lleva directamente a la página del clasificador.

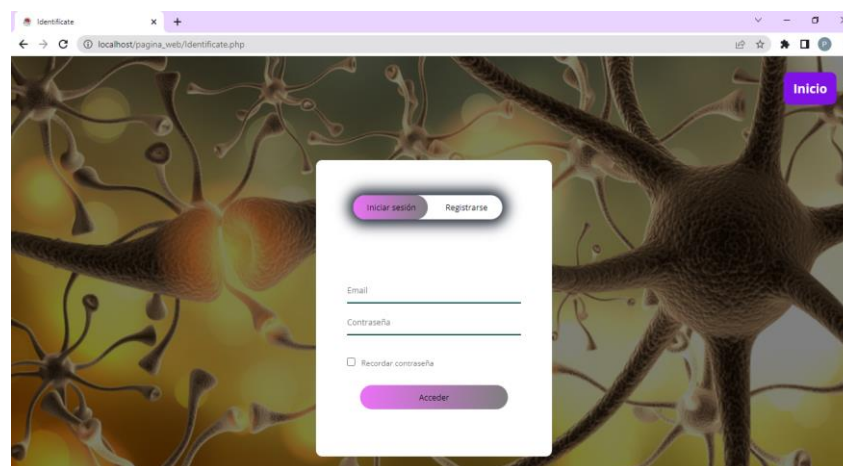


Figura 9. Página de registro y login. Fuente: Autores



Página donde se encuentra el formulario del clasificador

En esta parte podemos encontrar un formulario donde se puede seleccionar o arrastrar la imagen de resonancia magnética a clasificar. Para este formulario se utilizó JavaScript para poder definir la funcionalidad de arrastrar y soltar las imágenes, estas imágenes se guardan en una carpeta de Google drive. También hay dos botones, el primero llamado inicio este direcciona al usuario directamente a la página principal y el segundo se llama funcionamiento este lleva al usuario a un pequeño tutorial de cómo funciona el clasificador.



Figura 10. Página del clasificador. Fuente: Autores



Figura 11. Página del clasificador. Fuente: Autores



Página obtención del resultado

En esta página se muestra el resultado de la predicción, además se expone una previsualización de la imagen subida por el usuario, También tenemos tres botones, el primero es para volver a la página principal, el segundo se utiliza para hacer una predicción con una imagen diferente y el ultimo redirecciona al usuario a la página donde se explica el funcionamiento de esta herramienta.

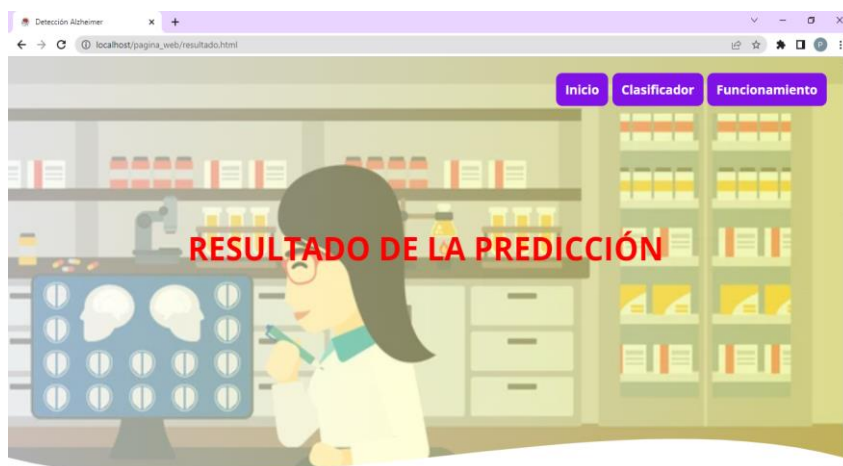


Figura 12. Página del resultado. Fuente: Autores

Página funcionamiento

En esta se le da al usuario un panorama general pero bastante completo de la herramienta, explicando cómo funciona y cómo se debe utilizar esta. De igual manera se especifica cada parte de este clasificador y así poder dar al usuario una mejor experiencia.

Se decidió que en esta página un video tutorial porque es más sencillo para el usuario entender un video que algo escrito al tener la posibilidad de hacer una explicación más didáctica en este y también porque desde la perspectiva de los autores se iba a mejorar mas estético el video que una explicación escrita.



Figura 13. Página funcionamiento. Fuente: Autores



Figura 14. Página funcionamiento. Fuente: Autores

6.7. Migración del modelo de Machine Learning al navegador web

Para poder utilizar el modelo entrenado en el navegador, lo que se hizo fue utilizar TensorFlow.js para poder hacer la predicción con JavaScript para luego enviarla a la página de resultado y hacer su visualización. Y para guardar la predicción y la imagen se utilizó un complemento de JavaScript que poseen todos los navegadores actuales, este es el local storage donde se pueden guardar elementos de tipo clave valor.



7. RESULTADOS

7.1. Resultados del entrenamiento de los modelos propuestos

A continuación, se muestran los resultados obtenidos para los modelos que se formularon en un momento inicial.

CNN

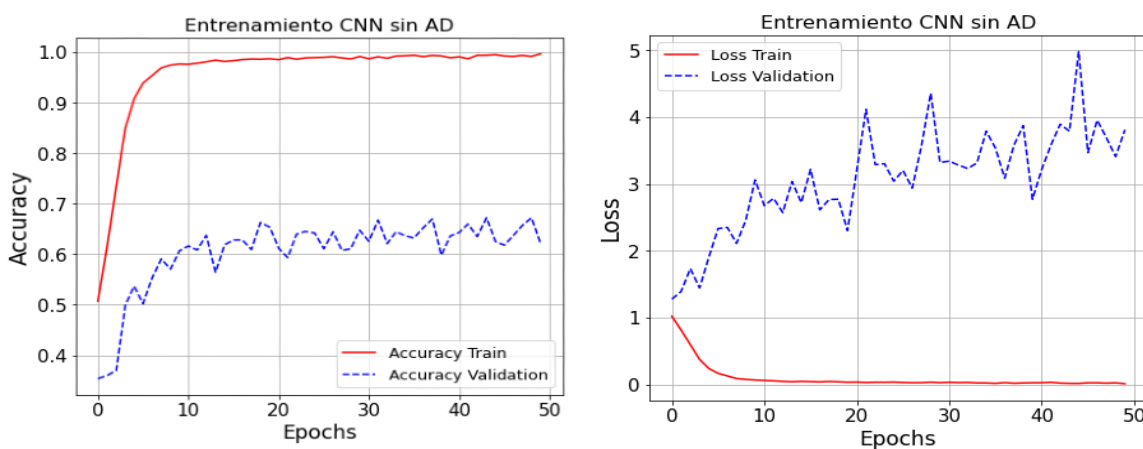


Figura 15. Gráfica de las métricas modelo CNN. Fuente: Autores

Como podemos observar en la figura 15 el modelo tiene overfitting, con los datos de entrenamiento le va muy bien, tanto que se aprende las características de estos datos, pero cuando se le presentan nuevos datos (datos de validación) a la red le va bastante regular y la precisión no llega a un porcentaje bueno y la pérdida empieza a subir en vez de bajar.

VGG 19

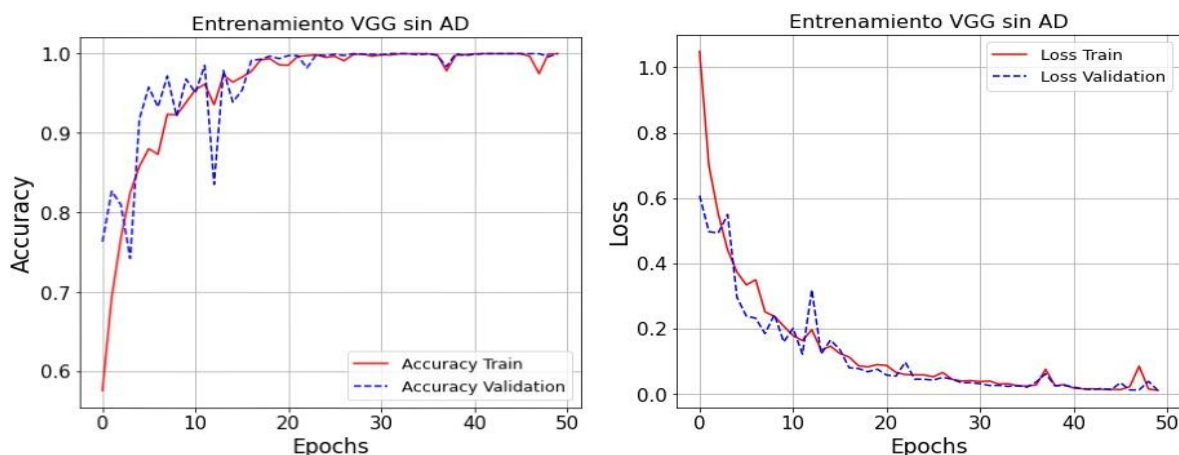


Figura 16. Gráfica de las métricas modelo VGG 19. Fuente: Autores

En la figura 16 se puede notar que este modelo tiene un buen comportamiento de la precisión y la pérdida, en los datos de entrenamiento y validación la precisión empieza a subir y la pérdida empieza a bajar ambos al mismo tiempo.

Inception V3

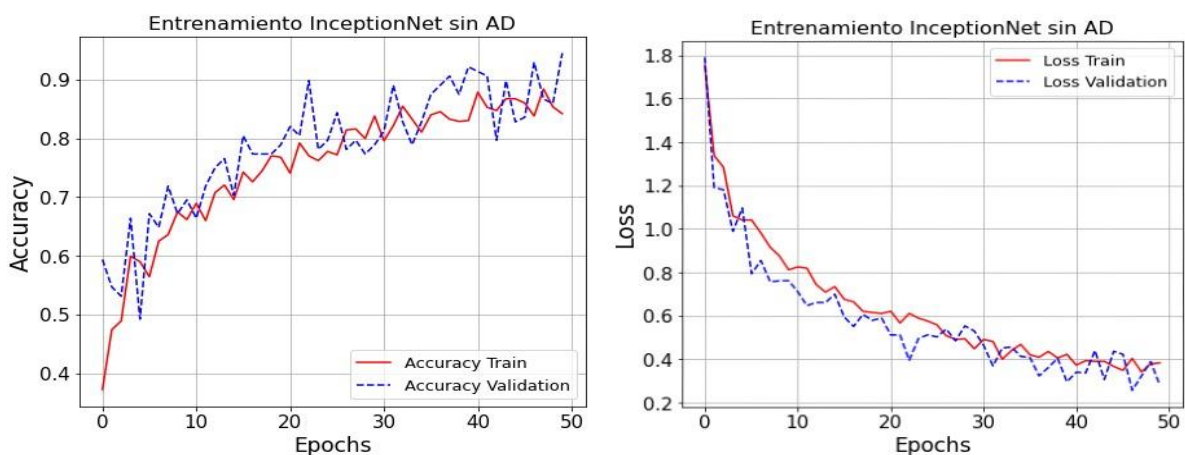


Figura 17. Gráfica de las métricas modelo InceptionNet. Fuente: Autores

En la figura 17 podemos observar el comportamiento del modelo Inception V3, este modelo no tiene sobreajuste, en términos generales se comporta bastante bien, pero las



métricas son muy variantes, la precisión es de 0,90 y puede ser que mejore con más iteraciones, pero esto último no es seguro.

EfficientNet

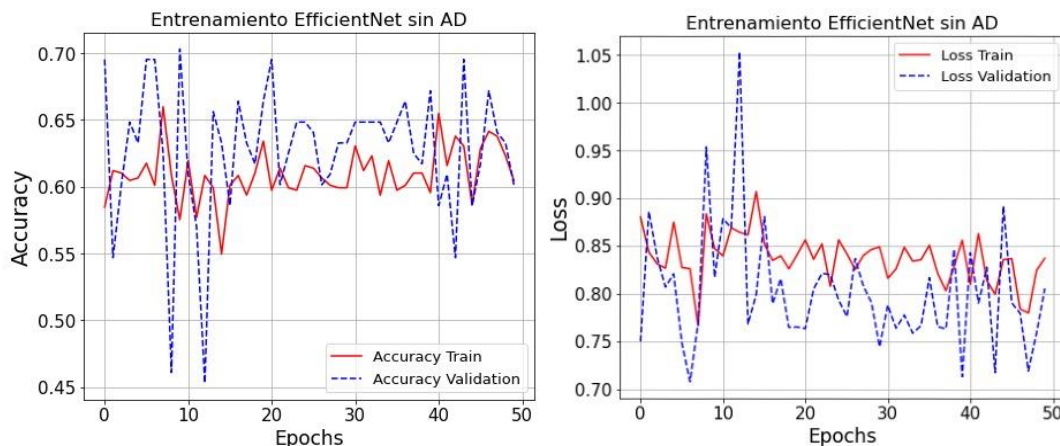


Figura 18. Gráfica de las métricas modelo EfficientNet. Fuente: Autores

Según la figura 18 con este modelo tenemos un comportamiento bastante errático debido a que varían demasiado las métricas, además la precisión no sube del 0.70 y es bastante claro que no mejorará con más iteraciones.

MobileNet V3

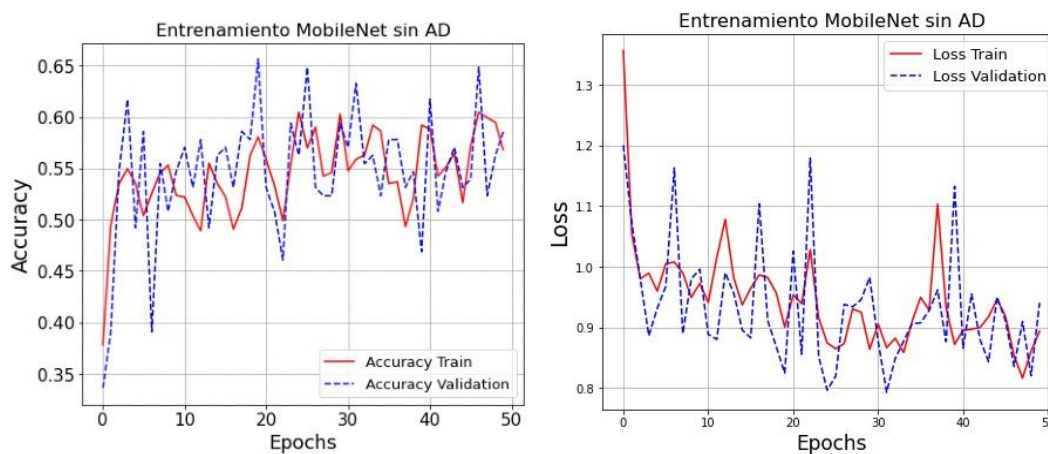


Figura 19. Gráfica de las métricas modelo MobileNet. Fuente: Autores



Con respecto a este último modelo en la figura 18 podemos observar que no tiene un problema tan presente del sobreajuste, pero si es verdad que ni en los datos de entrenamiento ni en los de validación, la precisión y la pérdida no pueden llegar a valores buenos por más de que pasen las iteraciones.

Comparación Métricas de cada uno de los modelos

En cuanto a la evaluación de los modelos se hizo un análisis para escoger tres de estos, se seleccionó los que mejor resultado daban, para esto se utilizó como métrica principal la precisión. En la tabla 2 se resumen los resultados de las métricas de cada uno de los modelos entrenados con respecto a los datos de validación.

Tabla 2. Resumen de las métricas de los modelos iniciales. Fuente: Autores

No	Modelo	Accuracy	Loss
1	CNN	0.647	3.808
2	VGG 19	0.998	0.019
3	Inception V3	0.90	0.302
4	MobileNet V3	0.609	0.854
5	EfficientNet	0.641	0.776

Según la tabla 2 se puede deducir que los tres modelos que mejor funcionan para este caso son Redes Neuronales Convolucionales, VGG 19 e Inception V3. Por esto se trabajó con estos tres para escoger uno solo. De igual manera, aunque se está utilizando transfer Learning para de algunos modelos pre entrenados esto no asegura que con el dataset que se está clasificando se obtengan buenos resultados y es algo que se creería que sí, pero no es cierto.

7.2. Mejoramiento del modelo CNN

CNN con aumento de datos y dropout

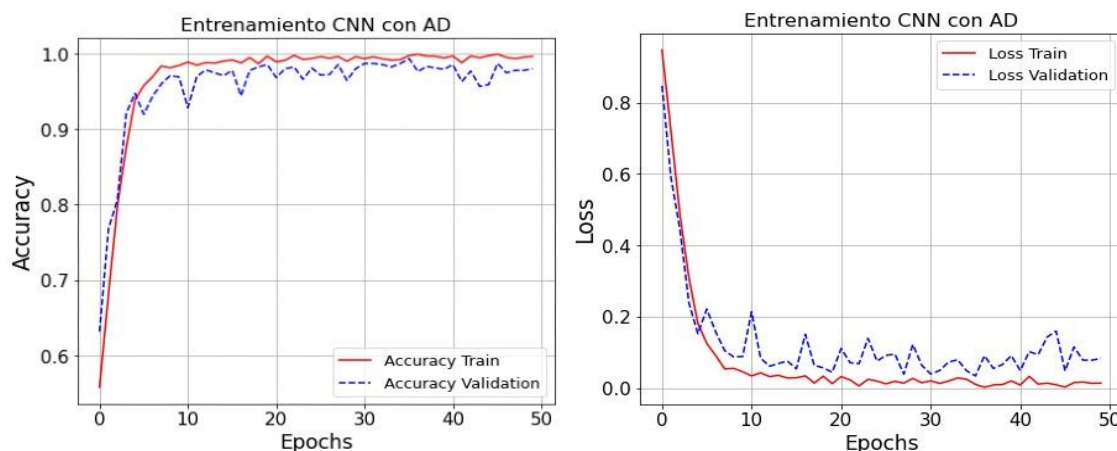


Figura 20. Gráfica de las métricas modelo CNN con aumento de datos. Fuente: Autores

En la figura 20 podemos observar que se elimina el overfitting totalmente de este modelo, tiene un comportamiento bastante bueno y la precisión y la pérdida alcanzan valores bastante decentes.

Comparación de las métricas de los mejores más adecuados

Tabla 3. Resumen de las métricas de los mejores modelos. Fuente: Autores

No	Modelo	Accuracy	Loss
1	CNN (con AD)	0.979	0.083
2	VGG 19 (sin AD)	0.998	0.019
3	Inception V3 (sin AD)	0.90	0.302

De la información contenida en la tabla 3 podemos concluir que el mejor modelo para nuestra aplicación fue VGG 19, por lo cual fue con este modelo con el que decidimos trabajar.

7.3. Matriz de confusión del modelo escogido

Tabla 4. Tabla de la matriz de confusión. Fuente: Autores

	precision	recall	f1-score	Support
0	1.0000	0.9988	0.9944	179
1	1.0000	1.0000	1.0000	12
2	1.0000	1.0000	1.0000	640
3	0.9956	1.0000	0.9978	448
Accuracy			0.9984	1279
macro avg	0.9989	0.9972	0.9980	1279
weighted avg	0.9984	0.9984	0.9984	1279

En la tabla 4 se puede observar una recreación de la tabla de la matriz de confusión que contiene las métricas del modelo.

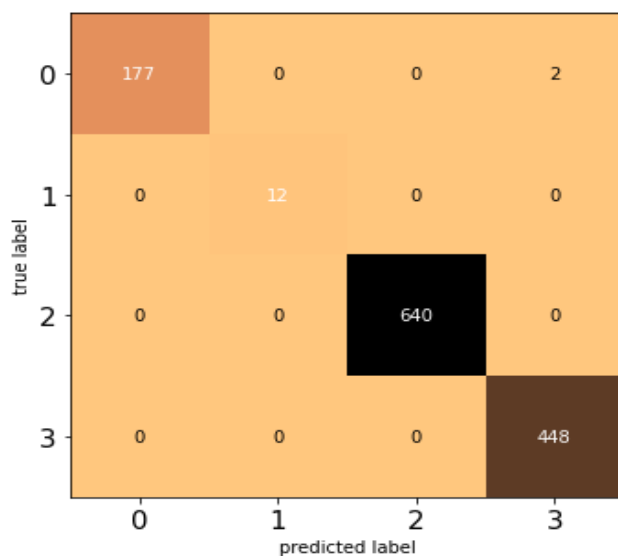


Figura 21. Matriz de confusión modelo VGG 19 entrenado. Fuente: Autores

En la figura 21 se puede observar la matriz de confusión con respecto a evaluar el modelo entrenado con los datos de validación, podemos verificar que tiene unas métricas



bastante buenas llegando a tener unas clases con el 100% de precisión. Algo importante es que la clase 0 es mild demented, la clase 1 es moderate demented, la clase 2 es non demented y la clase 3 es very mild demented.

7.4. La página web entregando el resultado de la predicción

A continuación, mostrara el funcionamiento de la herramienta web cuando un usuario le ingresa una imagen de resonancia magnética.



Figura 22. Página entregando el resultado. Fuente: Autores

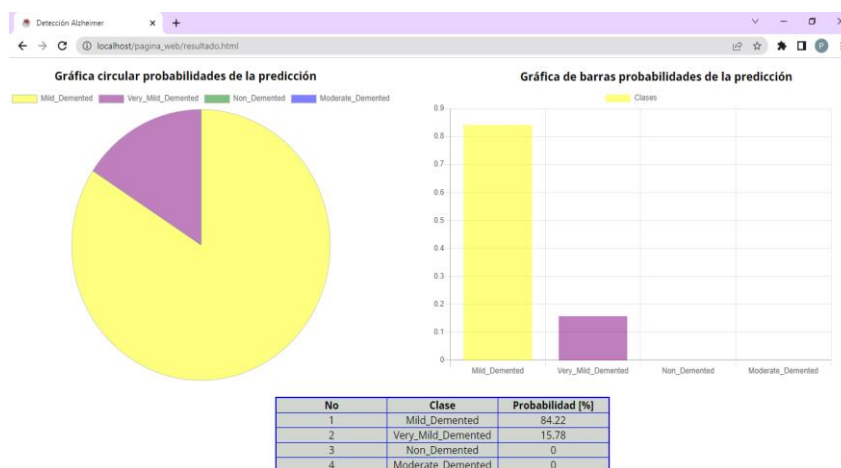


Figura 23. Página entregando el resultado. Fuente: Autores



Para hacer estas gráficas que muestran en la figura 23 se utilizó una librería o complemento de JavaScript llamado chart.js. Estas gráficas representan las posibilidades de que la imagen pertenezca a cada clase, tomando en cuenta que al momento de hacer la predicción con Tensorflow.js esta es un tensor de 4 posiciones con las posibilidades de cada una de las clases y los valores de este tensor son los que se grafican.

8. CONCLUSIONES

En la clasificación de imágenes se tiene la posibilidad de utilizar diferentes modelos propuestos que están incluidos en el Machine Learning y los resultados de cada uno depende de manera directa de la aplicación que se requiere desarrollar. Como bien hemos visto para este caso algunos modelos funcionaron de mejor manera que otros, en este orden de ideas lo que se puede concluir es que utilizar transferencia de aprendizaje no garantiza obtener buenos resultados en nuestros clasificadores, sino que depende del tipo de imágenes que estemos clasificando.

Al momento de entrenar algunos modelos de Machine Learning se puede presentar un inconveniente bastante común que es el sobreajuste o overfitting esto se observó en los resultados del entrenamiento de la Red Neuronal Convolucional, este problema se puede dar por numerosas razones como que el dataset tiene muy pocos datos o que el dataset tenga un desbalance en la cantidad de imágenes de cada una de las clases, también se puede dar por la complejidad de la arquitectura del modelo es decir que tenga muchas capas ocultas o muchas neuronas en estas. Existen diferentes soluciones para este problema, en este caso utilizamos dos que funcionaron bastante bien, que fueron llevar a cabo aumento de datos al dataset y la utilización de un dropout.

9. REFERENCIAS

- [1]. J. L. Ortega, "Diagnosis de pacientes de Alzheimer a partir de visión por computador" Trabajo Final de Grado, Facultad de Informática de Barcelona, Departamento de Ingeniería de Sistemas, UPC, Barcelona, 2021.
- [2]. V. Paniza, "Deep Learning en la detección de Alzheimer utilizando Imágenes de Resonancia Magnética", Trabajo de grado, ITBA, Buenos Aires, 2020.
- [3]. S. Gauthier, "*Clinical Diagnosis and Management of Alzheimer's Disease*". 3ª ed. Londres: Informa UK Ltd, 2006.
- [4] AFAGA. "Alzheimer - AFAGA". [Online]. Disponible en: <https://afaga.com/es/las-demencias/alzheimer/> (accedido el 10 de diciembre 2022).



- [5]. O. Theobald, *"Machine Learning for Absolute Beginners"*. 2ª ed. Todos los derechos reservados por Oliver Theobald, 2017.
- [6]. L. Aronson, *"HTML Manual of Style: A Clear, Concise Reference for Hypertext Markup Language (Including HTML5)"*. 4ª ed. Addison-Wesley Professional, 2010.
- [7]. D. Paper, *"TensorFlow 2.x in the Colaboratory Cloud: An Introduction to Deep Learning on Google's Cloud Service"*. 1ª ed. New York: Apress, 2021.
- [8]. J. Lu *et al.*, "Transfer learning using computational intelligence: A survey", *Knowl Based Syst*, vol. 802, pp. 14-23, Enero 2015.
- [9]. C. Shorten, "A survey on Image Data Augmentation for Deep Learning", *Big Data*, vol. 6, Julio 2019.
- [10]. X. Ying, "An Overview of Overfitting and its Solutions", *J Phys: Conf Ser*, vol. 1168, no. 2, 2019.
- [11]. R. Yamashita *et al.*, "Convolutional neuronal networks: an overview and application in radiology", *Insights into Imaging*, vol. 9, pp. 611-629, Junio 2018.
- [12]. C. Aggarwal, *"Neural Networks and Deep Learning"*. 1ª ed. New York: Springer International Publishing AG, 2018.
- [13]. D. Paper, *"State-of-the-Art Deep Learning Models in TensorFlow Modern Machine Learning in the Google Colab Ecosystem"*. 1ª ed. New York: Apress, 2021.
- [14]. A. Howard *et al.*, "Searching for MobileNetV3", *IEEE/CVF ICCV*, pp. 1314-1324, 2019.
- [15]. B. Koonce, *"Convolutional Neural Networks with Swift for Tensorflow: Image Recognition and Dataset Categorization"*. 1ª ed. New York: Apress, 2021.
- [16]. A. Géron, *"Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems"*. 1ª ed. Sebastopol, California: O'Reilly Media. 2017.