Desarrollo de un Modelo de Predicción para la Detección Temprana del Cáncer de Piel Utilizando Redes Neuronales

Trabajo de Fin de Grado
GRADO EN ESTADÍSTICA

VNiVERSiDAD
D SALAMANCA

Julio 2024

Autor:

ARTURO SÁNCHEZ SÁNCHEZ

Tutor:

PEDRO IGNACIO DORADO DÍAZ

Desarrollo de un Modelo de Predicción para la Detección Temprana del Cáncer de Piel Utilizando Redes Neuronales

Trabajo de Fin de Grado
GRADO EN ESTADÍSTICA

VNiVERSiDAD D SALAMANCA

Julio 2024

Autor:

SANCHEZ SANCHEZ ARTURO - 44743678F

Firmado digitalmente por SANCHEZ SANCHEZ ARTURO - 44743678F
Fecha: 2024.06.29 01:41:19 +02'00'

Tutor:

PEDRO IGNACIO DORADO DÍAZ

Certificado de los tutores TFG Grado en Estadística

D. PEDRO IGNACIO DORADO DÍAZ, profesor/a del Departamento de Estadística de la Universidad de Salamanca.

HACEN CONSTAR

Que el trabajo titulado "Desarrollo de un Modelo de Predicción para la Detección Temprana del Cáncer de Piel Utilizando Redes Neuronales", que se presenta, ha sido realizado por D. ARTURO SÁNCHEZ SÁNCHEZ, con DNI 44743678F y constituye la memoria del trabajo realizado para la superación de la asignatura Trabajo de Fin de Grado en Estadística en esta Universidad.

Salamanca, a fecha de firma electrónica.

DORADO DIAZ, PEDRO IGNACIO (FIRMA)

Firmado digitalmente por DORADO DIAZ, PEDRO IGNACIO (FIRMA)
Fecha: 2024.06.29 14:55:33 +02'00'

Fdo.: PEDRO IGNACIO DORADO DÍAZ

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1. INTRODUCCIÓN

1.1. Resumen

La inteligencia artificial es un campo del conocimiento que en la actualidad está en el punto de mira, sufriendo una revolución en términos tecnológicos. Desde los primeros sistemas inteligentes, pasando por los primeros prototipos de neurona artificial, hasta las arquitecturas más complejas de redes neuronales han buscado recrear la capacidad de razonamiento de un ser humano. En el desarrollo se va a aplicar esta tecnología en el campo de la medicina, concretamente en la dermatología.

Este proyecto se enfocará en el desarrollo de un modelo predictivo basado en técnicas de minería de datos y aprendizaje automático supervisado para la detección temprana del cáncer de piel.

Utilizando datos del ISIC[1], se aplicarán diferentes metodologías y se implementará una red neuronal eficiente y ligera para clasificar las diferentes lesiones dermatológicas y por lo tanto la probabilidad de que una lesión en la piel sea cancerígena.

El objetivo central es crear una herramienta precisa y veloz que pueda ser implementada en un servidor web o en dispositivos móviles para el diagnóstico automático de imágenes de lesiones cutáneas. Para este hito es imprescindible desarrollar un modelo que no exija demasiado en términos computacionales, y teniendo en cuenta la complejidad del problema el tratamiento de los datos en el pipeline será vital. Es por esto que se desarrollará una metodología minuciosa con cada apartado del desarrollo para conseguir potenciar los resultados.

Esta herramienta contribuirá significativamente a una detección temprana y precisa del cáncer de piel, al tiempo que aliviará la carga en los sistemas sanitarios, pues servirá como un cribado en primera instancia. Esto también conlleva una reducción de costes asociados con diagnósticos tardíos y tratamientos avanzados.

[1] International Skin Imaging Collaboration

1.2. Abstract

Artificial intelligence is a field of knowledge that is currently in the spotlight, undergoing a revolution in technological terms. From the first intelligent systems, through the first prototypes of artificial neurons, to the most complex architectures of neural networks have sought to recreate the reasoning capacity of a human being. The development will apply this technology in the field of medicine, specifically in dermatology.

This project will focus on the development of a predictive model based on data mining and supervised machine learning techniques for the early detection of skin cancer.

Using ISIC data, different methodologies will be applied and an efficient and lightweight neural network will be implemented to classify different dermatological lesions and therefore the probability that a skin lesion is carcinogenic.

The central objective is to create an accurate and fast tool that can be implemented on a web server or mobile devices for the automatic diagnosis of skin lesion images. For this milestone it is essential to develop a model that does not require too much in computational terms, and considering the complexity of the problem the treatment of the data in the pipeline will be vital. This is why a thorough methodology will be developed for each section of the development in order to enhance the results.

This tool will significantly contribute to early and accurate detection of skin cancer, while easing the burden on healthcare systems by serving as a first-line screening. This also leads to a reduction in costs associated with late diagnosis and advanced treatment.

1.3. Objetivos

El objetivo principal del proyecto es el desarrollo de un modelo predictivo de clasificación y optimizado computacionalmente, que mediante el análisis de imagen, ayude a la detección temprana del cáncer de piel.

Para llevar a cabo este hito, debemos especificar una serie de objetivos específicos que nos lleven hasta el resultado que buscamos. Estos objetivos son los siguientes:

- Comprender profundamente los fundamentos teóricos de la inteligencia artificial y el ***Deep Learning***.
- Aplicar técnicas de minería de datos y de preprocesamiento para optimizar la base de datos con la que se alimentarán los modelos.
- Desarrollar un modelo predictivo de clasificación para la detección temprana del cáncer de piel.
- Obtener un modelo optimizado y no muy pesado para su posterior aplicación en un servidor web, de forma que se consigan los análisis con relativa fluidez y sin mucho coste computacional.
- Tener la capacidad de reducir la carga de trabajo del sistema de salud al optimizar el proceso de diagnóstico y remisión de pacientes.

Las metodologías aplicadas van a estar inspiradas en los mejores desarrollos que han conseguido data miners de todo el mundo en este campo. Es de suma importancia en la implementación del ***pipeline*** que el tratamiento de los datos sea óptimo, pues es seguro que en un tema como es la salud, la confidencialidad de los datos hace que sean de difícil acceso.

Es importante evaluar y comparar el rendimiento de diferentes modelos de ***Machine Learning*** para determinar su rendimiento sobre el conjunto de datos y el hardware. Para ello, se van a utilizar métricas de evaluación sobre los modelos que se vayan construyendo a través del proceso.

Se usarán arquitecturas pensadas para la implementación de esta tecnología en entornos con capacidad computacional limitada de forma que podamos cumplir nuestro objetivo de explotar el modelo en una aplicación web.

1.4. Organización memoria

La memoria de este proyecto se va a organizar según se describe a continuación y por las razones que se especifican. Primero se describe el estudio del arte, previo al desarrollo de la práctica, que servirá para poner en contexto el problema y profundizar en el dominio de la solución.

Se va a describir el contexto de las afecciones que se estudiarán, desde los diferentes tipos hasta la epidemiología del cáncer de piel. Además se profundizará en las metodologías de detección de los profesionales sanitarios y su efectividad.

Se desarrollará y explicarán las bases teóricas de la inteligencia artificial, su contexto histórico y la evolución del paradigma, desde la representación del conocimiento del problema hasta las primeras neuronas artificiales. Además se explicarán los diferentes paradigmas de aprendizaje del ***Machine Learning***.

Posteriormente se profundizará de manera teórica en el funcionamiento de la neurona artificial, explicando su evolución, funcionamiento matemático y algoritmos de entrenamiento.

Cuando se hayan asentado las bases del ***Machine Learning*** se va a contextualizar la evolución de la profundidad de las redes, o dicho de otro modo ***Deep Learning.*** Además se desarrollará la introducción del problema a resolver y su relación con este paradigma.

Se van a desarrollar las múltiples arquitecturas que hay en el campo del ***Deep Learning***, su funcionamiento y el objetivo para el que están destinadas. Es importante además explicar todas las métricas con las que se han de valorar los resultados de este campo, se hará un desarrollo de las mismas. Así como se contextualizan teóricamente los hiperparámetros a ajustar en el desarrollo práctico y la relación del ***Deep Learning*** y la capacidad computacional.

En cuanto al desarrollo de la práctica, está dividida en tres apartados. El primero es material y métodos, donde se expone y desarrolla toda la metodología. El segundo son los resultados donde se exponen las evaluaciones realizadas y los resultados. Para finalizar la práctica se exponen las conclusiones y las posibles líneas de mejora.

Parte I: Marco teórico

2. CÁNCER DE PIEL

2.1. Introducción

El cáncer se denomina como el conjunto de afecciones que se caracteriza por la transformación de las células, que proliferan de manera anormal e incontrolada, en algunos casos invadiendo otros tejidos y con la capacidad de derivar en metástasis. Según la ***American Cancer Society***, a una de cada tres personas les será diagnosticado algún tipo de cáncer a lo largo de su vida.[1] Particularmente, el cáncer de piel consiste en el desarrollo de células cutáneas cancerígenas. A día de hoy, el cáncer de piel representa un problema a valorar en todo el mundo, debido al impacto en el estado del bienestar de las personas y a su incidencia. En España el cáncer de piel se encuentra entre los cinco[2] cánceres más comunes.[3] Además se trata de una afección con bastante capacidad de prevención, puesto que según la IARC[2], la mayoría de casos están directamente relacionados con la exposición solar.[4]

En la detección del cáncer de piel debemos tener en cuenta la dermatoscopia, técnica de diagnóstico de imagen donde se logra eliminar el reflejo superficial de la piel y ampliar la lesión, mejorando y profundizando la visualización de esta. Esto proporciona una precisión diagnóstica mejorada para los médicos que la van a inspeccionar visualmente.[5]

2.2. Tipos

Existen múltiples afecciones de piel, nos centraremos en los tipos de cáncer de piel. Cada afección tiene unas características diferenciales en cuanto a los factores de riesgo, comportamiento clínico y pronóstico.[6]

Primero, respecto a los melanomas:

- Melanoma maligno.

  El melanoma es un cáncer menos frecuente que los otros tipos, pero su comportamiento es el más agresivo, tiene características más invasivas y por ende mayor capacidad de derivar en metástasis, lo que lo hace un tipo de cáncer grave y potencialmente mortal. Respecto a su origen, se deriva de los melanocitos, y puede aparecer en cualquier parte del cuerpo, incluso en áreas no expuestas al sol.[6]

Segundo, respecto a los NO melanomas:

- Carcinoma basocelular.

  El carcinoma basocelular es el tumor más frecuente dentro de los cánceres de piel, siendo alrededor del 80-90% de los cánceres de piel. Respecto al comportamiento, tiene un crecimiento que suele avanzar lento, se desarrolla en áreas de la piel expuestas al sol de forma crónica. En contraposición del melanoma, este cáncer no es invasivo, por lo tanto no produce metástasis.[7]

[2] International Agency for Research on Cancer

- Carcinoma espinocelular.

  El carcinoma de células escamosas es el segundo cáncer de piel más frecuente. Hablando de su comportamiento se origina en los queratinocitos de la epidermis, y se representa como una lesión o herida, con costra y comúnmente sangrados.[6][7]

- Adenocarcinoma de glándulas sebáceas.

  El adenocarcinoma de glándulas sebáceas es un tipo muy raro de cáncer de la piel, se trata de una afección habitualmente situada en los párpados. Tiene capacidad invasiva para derivar en metástasis a través de los ganglios linfáticos y se trata de una afección de difícil detección y tratamiento.[6]

Hay que destacar que el la práctica el adenocarcinoma no se estudiará por tener un diagnóstico tan específico, además el carcinoma de células escamosas se agrupará con otras lesiones precancerosas.

2.3. Epidemiología

Como se ha mencionado anteriormente, el cancer de piel es uno de los más comunes, dependiendo siempre del lugar y de la raza, pues la cantidad de melanina en la piel, será crucial para la protección de la misma frente a los rayos UV[3], y principal factor de riesgo en los carcinomas basocelulares.[8]

Esta correlación directa entre la raza y la incidencia del cáncer se puede observar en la figura 1 donde los países con más incidencia tienden a tener más población blanca.[9]

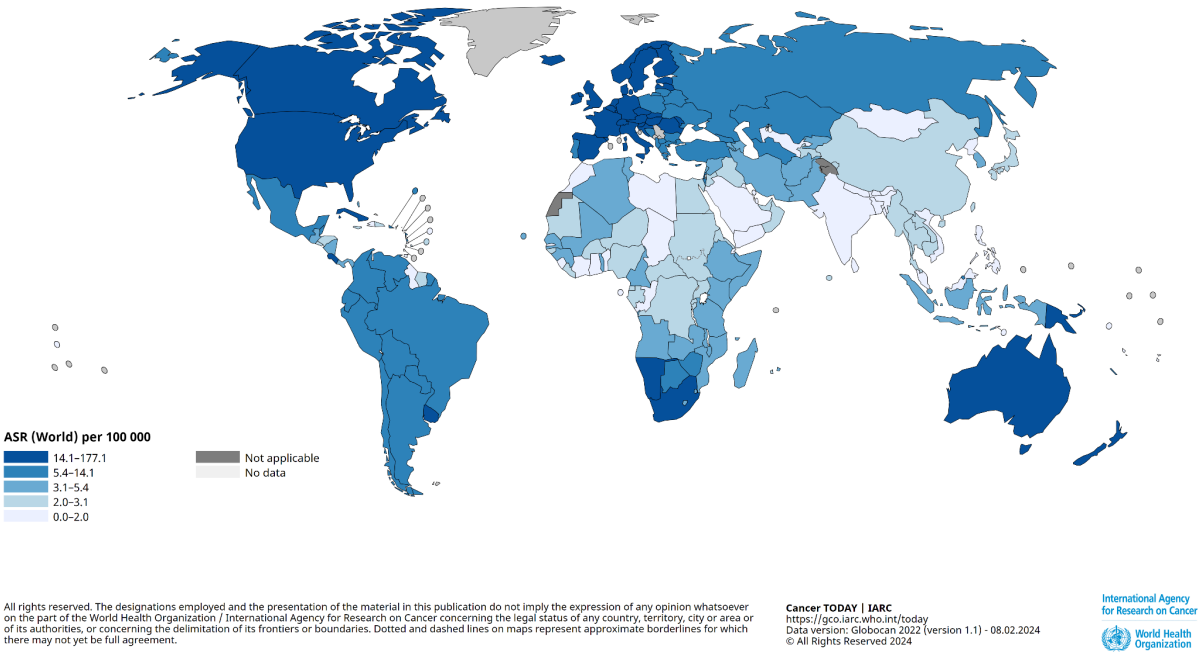

Figura 1: Incidencia del cáncer de piel (melanoma + no-melanoma) estandarizada por edad por 100.000 habitantes. Ambos sexos, para todas las edades, en 2022 [10]

[3] Ultravioleta

También hay que destacar que en los países desarrollados como España, la pandemia del COVID-19 afectó positivamente a la incidencia de los cánceres de piel de tipo no melanoma, pues las personas tuvieron que cambiar su forma de vida y se popularizó el teletrabajo, lo que redujo drásticamente la exposición al sol. Aunque esto pueda conllevar otros problemas, en nuestro caso supuso una disminución de 45 a 30 casos cada 100.000 personas-año para los no melanomas. Los cánceres de tipo melanoma no se vieron afectados puesto que la exposición al sol no les influye tanto, se mantuvieron en 12 casos cada 100.000 personas-año.[2]

En total en España se detectaron 21900 casos de cáncer no melanoma el 2020, así como 6000 cáncer de tipo melanoma. Cabe destacar que según el artículo "Incidencia y mortalidad del cáncer cutáneo en España: revisión sistemática y metaanálisis" de la Academia Española de dermatología y venereología estas tasas sufren de infraestimación. Teniendo en cuenta su investigación en el año 2016 la tasa de incidencia del carcinoma basocelular fue 113,05 (IC 95%: 89,03-137,08)/100.000 personas-año, de 38,16 (IC 95%: 31,72-39,97)/100.000 personas-año para el carcinoma espinocelular y de 8,76 (IC 95%: 7,50-10,02)/100.000 personas-año para el melanoma.[11]

2.4. Diagnóstico

Existen varias metodologías usadas en la medicina para la detección de cáncer de piel, entre ellas el método Menzies, el método ABCD y el Análisis de patrones.[12]

El Análisis de patrones es la mejor metodología de todas las mencionadas, esta es la técnica más usada por los médicos con experiencia. Las lesiones están caracterizadas por tener unos patrones globales que permiten una clasificación general, además existen patrones o estructuras más localizadas dentro de la propia lesión de las que se extraen características más profundas, que proporciona la información suficiente para la clasificación de la lesión.[12]

El Análisis de patrones se complementa con la regla ABCD, la cual es un acrónimo de asimetría, borde, color y estructuras dermatoscópicas. Al valorar la asimetría se divide la lesión pigmentada en dos ejes de 90º y se valora la asimetría con respecto al color, la forma y estructuras en ambos lados del eje. Respecto al borde, la lesión es dividida en 8 segmentos y se observa cada porción que presente una finalización abrupta o brusca del borde. Se valora la presencia de estos 6 colores como rasgo de lesión cancerosa: blanco, marrón claro, marrón oscuro, azul-gris, rojo y negro. En cuanto a las estructuras dermatoscópicas, se valora la presencia de cinco tipos de subestructuras diferentes en la lesión. Cada una de las valoraciones hechas por los criterios anteriores reflejarán un valor, que debe ser multiplicado por un peso ponderado para calcular el valor TDS[4].[12]

$$\mathbf{TDS = 1{,}3A + 0{,}1B + 0{,}5C + 0{,}5D.}$$

TDS	Diagnóstico
Menor a 4.75	Benigno
Entre 4.75 y 5.45	Sospechoso
Mayor a 5.45	Maligno

Tabla 1: Interpretación del índice dermatoscópico total para la regla del ABCD.

[4] Total Dermoscopy Score

3. DESARROLLO TEÓRICO DE LA INTELIGENCIA ARTIFICIAL

3.1. Breve contextualización de la inteligencia artificial.

La inteligencia artificial se compone de varias disciplinas, que buscan la creación de sistemas con capacidad de resolver problemas, que normalmente requieren de inteligencia humana, problemas que en el paradigma de la computación clásica, serían a priori computacionalmente muy complejos o imposibles de resolver. Podemos incluir en esta disciplina campos como el reconocimiento de patrones, el aprendizaje, la planificación y la comprensión del lenguaje natural, entre otras.[13]

La definición formal de inteligencia artificial fue dada por John McCarthy en 1956, describiendola como una máquina con un comportamiento que sería considerado inteligente en un ser humano, aunque en las memorias de Alan Turing ya se menciona la posibilidad de que las máquinas se asemejen al comportamiento humano. Es de hecho conocido el Test de Turing, propuesto por el matemático homónimo, un "juego de imitación" en el cual un evaluador humano tiene que interactuar con dos entidades de manera anónima. Una entidad será la inteligencia artificial y la otra un ser humano, mediante una serie de preguntas y respuestas escritas, el evaluador debe determinar qué entidad es la inteligencia artificial y cuál es el humano. Si la inteligencia artificial consiguiera engañar al evaluador humano habría pasado la prueba.[13][14]

La computación clásica aplica métodos directos, para la resolución de problemas, busca la implementación de los razonamientos realizados por un experto en un formato establecido por el lenguaje de programación, derivando en un algoritmo. El proceso de solución será rígido y eficiente, así como la implementación será sencilla. En cambio, la metodología de la inteligencia artificial se basa en implementar el propio proceso de razonar, el proceso de la búsqueda de la solución. Esta implementación será poco eficiente y compleja de implementar pero se podrá aplicar a procedimientos de solución desconocida.[13]

Dentro del paradigma de la inteligencia artificial se puede diferenciar entre el planteamiento no conectivista y el planteamiento conectivista.[15]

El planteamiento no conectivista, apodado clásico, utiliza algoritmos de búsqueda de la solución apoyados de la representación del conocimiento del problema, así como algoritmos simbólicos y el aprendizaje basado en reglas. En este ámbito entran por ejemplo los agentes inteligentes basados en reglas o metas.[15]

El planteamiento conectivista, es el ámbito donde se encuentran los modelos inspirados en la estructura y funcionamiento del cerebro humano, simula la forma en que las neuronas están interconectadas así como su funcionamiento.[15] Dentro del planteamiento conectivista tenemos a las redes neuronales artificiales, en las cuales nos enfocaremos durante este trabajo. Estos sistemas de inteligencia artificial se enfocan en la emulación del cerebro a través de unidades de procesamiento (neuronas artificiales) en capas interconectadas. Varias neuronas artificiales forman una RNA[5], al añadir las interfaces de entrada y salida tendremos un sistema de proceso neuronal.[15][13]

Los tres conceptos clave que una RNA intenta emular son el procesamiento paralelo de datos, la memoria distribuida que es la información almacenada en cada sinapsis, y la adaptabilidad al entorno.[13]

[5] Red Neuronal Artificial

3.2. Aprendizaje automático y la toma de decisiones basada en datos.

El aprendizaje automático, también conocido como ***Machine Learning***, es un campo de la inteligencia artificial, basado en desarrollar algoritmos, modelos y técnicas que permiten a las computadoras aprender a realizar tareas específicas a partir de unos datos, sin que tengan la implementación explícita para resolver cada tarea. En lugar de seguir reglas estáticas como los sistemas expertos del paradigma no conectivista, los sistemas de aprendizaje automático pueden mejorar su rendimiento a medida que se les proporciona más datos. Las máquinas son dotadas de la capacidad de identificar relaciones o patrones dentro de una gran cantidad de datos, y de esta forma poder tomar decisiones, así como de realizar predicciones sin necesidad de un algoritmo concreto para cada tarea.[16]

La toma de decisiones basada en datos es un componente fundamental del aprendizaje automático. El aprendizaje es el proceso de actualizar los pesos para llevar a cabo una tarea determinada. A través del aprendizaje automático los modelos deben extraer información subyacente en los datos, mediante patrones y relaciones para así poder después hacer predicciones sobre nuevos datos que el modelo no conoce. Esta perspectiva ha demostrado ser especialmente efectiva en áreas como la clasificación de imágenes, el reconocimiento de voz o el procesamiento del lenguaje.[13]

Existen cuatro paradigmas de aprendizaje:

- **Aprendizaje supervisado:** Se emplea una función del error o coste, que depende de los pesos sinápticos. A partir de pares de ejemplo (dato - resultado), se busca minimizar la función de error entre las salidas observadas y las salidas objetivo.

- **Aprendizaje no supervisado:** Este paradigma también llamado auto-organizado, busca modelar la distribución de los datos en el espacio de entrada, sin la presencia de etiquetas. Busca patrones, estructuras o agrupaciones en los datos sin intervención humana, con el objetivo de descubrir información oculta o segmentar los datos. Un conocido enfoque de esta técnica sería el PCA[6].

- **Aprendizaje híbrido:** Se trata de una combinación del aprendizaje supervisado y no supervisado. Trabajamos con conjuntos de datos que contienen tanto ejemplos etiquetados como no etiquetados, los datos etiquetados son usados para guiar el proceso de aprendizaje, mientras que los datos no etiquetados sirven para extraer información oculta y descubrir patrones adicionales.

- **Aprendizaje por refuerzo:** Este paradigma implica que un agente interactúe con un entorno, recibiendo retroalimentación en forma de recompensas, las cuales intentará maximizar. El objetivo es que a largo plazo el agente aprenda una política de acciones que maximicen las recompensas.

[6] Análisis de componentes principales

4. FUNDAMENTOS DE LAS REDES NEURONALES

4.1. Definición y estructura básica

Las redes neuronales artificiales son estructuras de proceso de información, paralela y distribuida, formada por Elementos de Proceso (EP). Uno de los primeros modelos importantes en este campo fue desarrollado por Rumelhart y McClelland en 1986, donde definen un Elemento de Proceso (EP), o neurona artificial.[13]

El modelo de Rumelhart, McClelland se define como:

- Un dispositivo que a partir de un vector de entradas $\mathbf{x_j}$ de $\mathbf{n}$ componentes, y genera una salida única $\mathbf{y_i}$.
- Un vector $\mathbf{w_{ij}}$ que representa la intensidad de la interacción entre la neurona presináptica $\mathbf{j}$ y la postsináptica $\mathbf{i}$, es decir, el vector de pesos sinápticos.
- Una regla de propagación tal que, dada la función $\mathbf{\sigma(w_{ij}\ ,\ x_j\ (t))}$ se define el potencial postsináptico $\mathbf{h_i(t)}$.
- Una función de activación tal que, $\mathbf{a_i(t) = f_i\ (a_i(t\text{-}1),\ h_i(t))}$, proporciona el estado de activación de la neurona $\mathbf{i}$ en función del estado anterior $\mathbf{a_i(t\text{-}1)}$, y del valor postsináptico en ese momento $\mathbf{h_i(t)}$.
- Una función de salida definida por $\mathbf{F(t)}$.

Esta definición nos dejaría con la salida $\mathbf{y_i}$ en un momento determinado $\mathbf{t}$ como:

$$\mathbf{y_i\ (t) = F_i(a_i(t)) = F_i(f_i\ (a_i(t\text{-}1),\ \sigma(w_{ij}\ ,\ x_j\ (t))))}$$

Esta estructura básica de una neurona artificial, según el modelo de Rumelhart y McClelland, es la base para comprender el funcionamiento de las redes neuronales artificiales, la figura 2 representa un diagrama de la misma. Cada neurona artificial toma múltiples entradas, las pondera y las combina para producir una salida, que será activada mediante una función que introduce normalmente no linealidades en el modelo.[13]

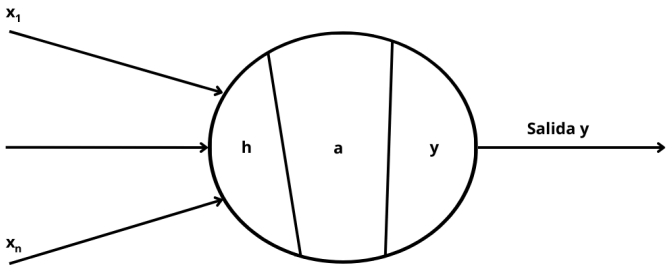

Figura 2: Neurona artificial desarrollada por Rumelhart y McClelland en 1986.

Profundizando más en el modelo de Rumelhart y McClelland, tenemos que tener en cuenta ciertos aspectos sobre la regla de propagación, la función de activación y la función de salida. Estos componentes del elemento de proceso han tenido una evolución en base a los preceptos establecidos por el modelo base que veremos a continuación.

La regla de propagación $\mathbf{h_i(t)}$, normalmente y en los modelos más simplificados que derivaron, es de tipo lineal, se trataría de una simple suma ponderada por los pesos sinápticos de forma $\sum \mathbf{w_{ij}}\ \mathbf{x_j}\ \mathbf{(t)}$, comúnmente llamada regla euclidiana ya que se trata de un producto escalar de **w** y **x**. Existen otras reglas de propagación menos comunes como las Distancias de Voronoi o la regla de Mahalanobis. [13]

En cuanto a la función de activación $\mathbf{a_i(t) = f_i\ (a_i(t\text{-}1),\ h_i(t))}$, normalmente, se contempla que el estado de activación de una neurona no depende del estado de activación anterior por lo tanto la función se simplificará de esta manera $\mathbf{f_i\ (\ h_i(t)\ )}$, dependiendo únicamente del valor postsináptico. La función de activación suele ser determinista, continua y monótona creciente.

Finalmente, la función de salida ha derivado a que normalmente se "obvie", usando casi siempre la identidad, es decir $\mathbf{y_i\ (t) = F_i(a_i(t))}$, quedaría como $\mathbf{y_i\ (t) = a_i(t)}$. También puede usarse una función umbral, es decir, la función de salida no propaga la carga sináptica si no supera cierto valor.[13]

Teniendo en cuenta lo anteriormente descrito, el modelo de la neurona artificial se ha simplificado bastante como podemos ver en la figura 3, de forma que llegamos a la tipología más usada a dia de hoy, descrita matemáticamente de forma:

$$\mathbf{y_i = f_i\ (\ \sum w_{ij}\ x_i\ -\ \theta_j\ )}$$

Cabe destacar que también se omite la dependencia temporal, variable **t**, al no tener en cuenta el estado de activación anterior, de esta forma, la salida $\mathbf{y_i}$ solo dependerá de la función de activación $\mathbf{f_i\ (\ \sum\ w_{ij}\ x_i\ -\ \theta_j\ )}$, donde $\mathbf{\theta_j}$ es en caso de que sea necesario el umbral o polarización.

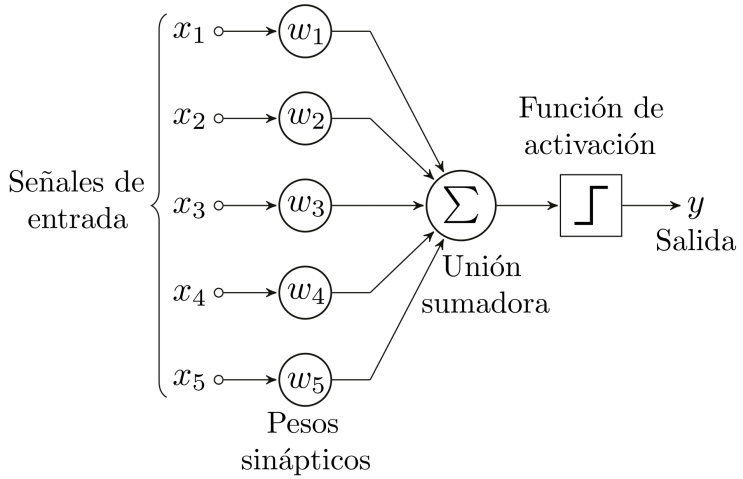

Figura 3: Neurona artificial, modelo simplificado. [17]

Sobre las función de activación, introduce la capacidad de modelar relaciones no lineales entre las entradas y las salidas de la red. Introducir no linealidad es muy importante pues gran parte de los problemas no pueden ser resueltos linealmente debido a sus complejos patrones.

Podemos encontrar varios tipos de funciones de activación, desde las simples como las funciones lineales o las funciones a signo hasta las más usadas hoy en dia:[18]

- **Función Lineal:** Es muy simple y no introduce linealidad, se usa en problemas de regresión.
- **Función Sigmoide:** Se usa en problemas de clasificación binaria, donde la salida deseada es una probabilidad entre 0 y 1.
- **Función Tangente Hiperbólica**: Usada en problemas de clasificación binaria y en problemas de regresión. La salida está entre -1 y 1.
- **Rectified Linear Unit (ReLU):** Esta función devuelve 0 para valores negativos y lineal para valores positivos. Tiene una variante llamada Leaky ReLU que devuelve valores muy pequeños en vez de 0 para valores negativos.
- **Softmax**: Se utiliza en la capa de salida de una red neuronal de clasificación. Transforma las salidas de la red en una distribución de probabilidad sobre las clases.

4.2. Entrenamiento

En la práctica las redes neuronales están formadas por varios Elementos de Proceso (EP), configurando diferentes arquitecturas que veremos más adelante. Cuando se forma una estructura con varios EP se denomina perceptrón multicapa, el proceso de entrenamiento de este implica ir ajustando los pesos de las conexiones entre los diferentes Elementos de Proceso o neuronas para minimizar la función de error, normalmente llamada función de coste.

El entrenamiento se lleva a cabo mediante un algoritmo de optimización, como el que veremos a continuación, llamado algoritmo de retropropagación o Backpropagation, se trata de el algoritmo más usado en el paradigma supervisado para el entrenamiento de redes neuronales. También existen otros algoritmos en el paradigma supervisado como Gradient descent, RMSProp o Adagrad.[19]

4.3. Backpropagation

Este es el algoritmo de entrenamiento más común para redes neuronales feedforward, como los perceptrones multicapa. Su funcionamiento se basa en propagar el error del modelo hacia atrás, calculando el gradiente de la función de coste respecto a los pesos para minimizar la función de coste.[13]

El algoritmo se inicia estableciendo de manera aleatoria los pesos de toda la red neuronal, aunque a veces se puede utilizar una red pre entrenada para el paradigma de trabajo que necesitemos, de forma que se necesitaran en principio menos cambios en los pesos de las neuronas artificiales. Posteriormente se establece un vector $\mathbf{V(\ m_i\ ,\ o_i\ )}$ de entradas $\mathbf{m_i}$ y su objetivo de salida $\mathbf{o_i}$, y cada entrada se propagara hacia delante en la red generando una predicción $\mathbf{s_i}$.[13][20]

Cuando el modelo ha generado la salida $\mathbf{s_i}$ correspondiente a la entrada $\mathbf{o_i}$, se procede a calcular el error $\mathbf{e_i}$ con la función de coste, esto cuantifica lo bien o mal que predice el modelo, comparando el par objetivo y salida $(\mathbf{o_i}, \mathbf{s_i})$.

Una vez llegados a este punto es cuando se hace la retropropagación del error calculado $\mathbf{e_i}$, desde la salida hasta la entrada, cada neurona propaga hacia atrás el error, pero teniendo en cuenta que cantidad de error es inducido por cada uno de los pesos de sus conexiones de entrada. Para saber cuánto error comete cada conexión de entrada se utiliza el cálculo diferencial. Dicho de otra manera, en cada capa va calculando los gradientes locales para disminuir el error final.[20][21]

El proceso descrito se hace para cada entrada del vector $\mathbf{V(\ m_i\ ,\ o_i\ )}$, y una vez se han procesado las $\mathbf{i}$ entradas del vector los pesos de la red son actualizados, lo que completa una iteración. El tamaño de este vector es lo que se llama tamaño de lote de entrenamiento (***batch training***), y el número de iteraciones que se realiza durante todo el proceso, con vectores distintos, se llama número de épocas de entrenamiento (***epochs***).

4.4. Funciones de Coste

La función de coste o función de error, como se acaba de explicar es la medida que determina el error entre el valor estimado y el valor real de las predicciones, cuantificando el rendimiento de los modelos de predicción. Existen varios tipos de funciones de coste, las cuales tienen diferentes características:

- **RMSE (Raíz cuadrada media):** Se calcula como la raíz cuadrada de la media de los residuos $\mathbf{r_i} = (\mathbf{o_i} - \mathbf{s_i})$ al cuadrado.[22]

$$\mathbf{RMSE} = \sqrt{\sum_{i=1}^{n} \frac{(y_i - \widehat{y_i})^2}{n}}$$

- **MAE (Error absoluto medio):** Se calcula como la suma media de los valores absolutos de los residuos.[22]

$$\mathbf{MAE} = \sum_{i=1}^{n} \frac{|y_i - \widehat{y_i}|}{n}$$

- **MASE (Error absoluto medio escalado):** Se calcula como la medida MAE, pero escalado, es decir, se normaliza la métrica en relación a una referencia, en este caso a un modelo de referencia.[22]

- **Categorical Cross-Entropy (Entropía cruzada categórica):** Se calcula la discrepancia entre dos distribuciones de probabilidad. Esta función se utiliza para problemas de clasificación con variables categóricas o discretas.[22]

- **Binary Cross-Entropy (Entropía cruzada binaria):** Funciona igual que la función ***Categorical Cross-Entropy***, pero a diferencia de esta, se usa para modelos con variables binarias.[22]

5. DEEP LEARNING: CONCEPTOS BÁSICOS

5.1. Definición y diferencia con machine learning tradicional

El ***Deep Learning*** o aprendizaje profundo es un paradigma dentro del aprendizaje automático, subconjunto del ***Machine Learning***. El ***Deep Learning*** está basado en la utilización de múltiples capas de procesamiento de neuronas artificiales. Esta arquitectura de gran profundidad es capaz de abstraer y comprender patrones y representaciones complejas de los datos.[23]

Este paradigma difiere del ***Machine Learning*** tradicional en la forma de abstraer patrones y representar información, el ***Machine Learning*** usa datos estructurados para hacer las predicciones, donde se requiere un experto humano para la selección y extracción manual de las características importantes de los datos.

El concepto de ***Deep Learning***, elimina parte del preprocesamiento, este tipo de algoritmo puede procesar datos no estructurados, por ejemplo imágenes o texto, y gracias a su profundidad es capaz de seleccionar y extraer las características más importantes, sin expertos humanos que hagan ese trabajo. Por ejemplo, si tenemos un conjunto de imágenes de tres tipos de flores y se requiere la clasificación de las mismas, el ***Deep Learning*** determina por sí mismo las características (forma de los pétalos, color) más importantes para la clasificación. En cambio el ***Machine Learning*** requiere que el experto humano establezca la jerarquía de características.[23][24]

5.2. Deep Learning en la medicina

El Deep Learning nos aporta avances en el campo de la medicina, gracias a la capacidad de abstracción de características se está comenzando a implementar por ejemplo para realizar diagnósticos de imagen, cada vez mejores a los de los propios profesionales.

Analizando el articulo ***"The Development of a Skin Cancer Classification System for Pigmented Skin Lesions Using Deep Learning"***[25], podemos ver como realizan un estudio sobre la viabilidad de introducir el ***Deep Learning*** en el diagnóstico del cáncer de piel. En el estudio se realiza una comparación de la capacidad de predicción de cáncer de piel entre un modelo de ***Deep Learning*** y dermatólogos profesionales.

En los resultados del estudio se puede ver como, para una clasificación multiclase, el modelo obtuvo un ***accuracy*** del 86,2%, por otra parte los especialistas dermatólogos con experiencia consiguieron un ***accuracy*** del 79,5% y los dermatólogos sin experiencia un 75,1%. El estudio concluye que el modelo de inteligencia artificial tiene mejor capacidad para la detección de cáncer en la piel que los dermatólogos.

Podemos concluir que los modelos de ***Deep Learning*** tienen como mínimo un gran potencial en el campo médico y según el estudio en la dermatología. Sería muy conveniente que se avanzara en este campo, para poder implementar esta tecnología en la detección temprana de múltiples afecciones y diagnósticos.

6. TIPOLOGÍAS DE REDES NEURONALES

En este apartado procedo al análisis y explicación de las diferentes topologías y arquitecturas dadas en el campo del ***Deep Learning***, concretamente en el análisis de imágenes.

Partimos de la arquitectura del perceptrón simple explicada en el apartado **4.1 Definición y estructura básica**, para extrapolar este diseño hacia las arquitecturas más complejas.

6.1. Redes Neuronales Feedforward

Se trata de la topología más simple y común de las redes neuronales, también conocidas como redes neuronales de propagación hacia adelante, en la cual la información fluye solo hacia delante, de la entrada a la salida pasando por las posibles capas ocultas.

Esta arquitectura se conoce comúnmente como red neuronal multicapa o perceptrón multicapa, en la cual no hay bucles ni realimentación. Una red neuronal multicapa se compone de una capa de entrada con la misma dimensión de los datos analizados, una capa de salida y una o más capas intermedias u ocultas entre la capa de entrada, como se puede ver en la figura 4. Las capas ocultas pueden estar total o parcialmente conectadas.[26]

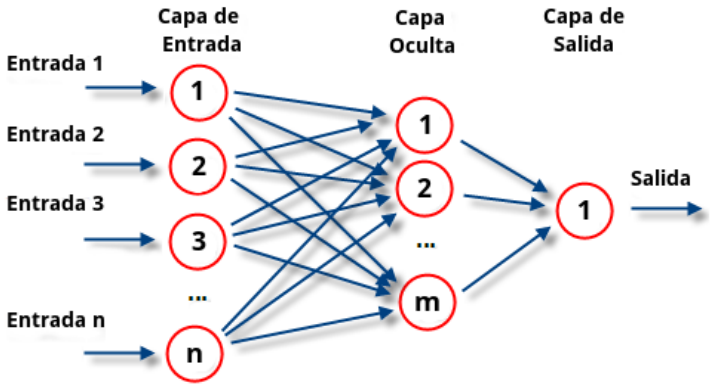

Figura 4: Diagrama de un perceptrón multicapa o feedforward. [27]

6.2. Redes Neuronales Convolucionales (CNN[7])

Las redes neuronales convolucionales son utilizadas especialmente en tareas de análisis de imagen. Esta arquitectura también llamada CNN, imita el córtex visual, aprovechando la estructura espacial de las imágenes. Esta arquitectura se basa en la extracción de características y el análisis de las mismas mediante una red neuronal.[28]

La arquitectura CNN está compuesta de dos fases, la primera se encarga de la transformación de la información y de la extracción de mapas de características, y una segunda fase que se ocupa de analizar estos mapas y características para razonar un análisis. Esta arquitectura es así porque si intentamos analizar una imagen con una red neuronal multicapa común, esta "vería" un simple vector de píxeles en la capa de entrada, sin tener en cuenta la estructura espacial de las imágenes, en cambio la CNN tiene en cuenta la ubicación de cada píxel para clasificar la imagen.[29]

[7] Convolutional Neural Networks

La primera fase está compuesta por capas de convolución, que son las encargadas de crear los mapas de características de los datos de entrada, y de capas de ***pooling*** para reducir la dimensionalidad, y de esta forma resaltar las características más importantes. Estas capas funcionan de la siguiente manera:

- **Capa de convolución:** El proceso de convolución consiste en aplicar un filtro (***kernel***) a cada píxel de la imagen de entrada para transformarlo. El filtro o ***kernel***, es una matriz de (n x n) píxeles que da como resultado 1 pixel, aplicando sobre cada píxel de la imagen, la operación que realiza dicho filtro es la combinación lineal de los valores de los píxeles de la imagen con los valores del filtro.[29]

  $$\mathbf{y_j} = \sum \mathbf{K_{ij}} \oplus \mathbf{y_i}$$

  Formalmente, la salida $\mathbf{y_j}$ de una neurona convolucional $\mathbf{j}$ es un valor, dado por la matriz que se calcula por medio de la combinación lineal de los valores de entrada $\mathbf{y_i}$ de las neuronas en la capa anterior, o de la imagen si estamos en la primera capa, cada una de ellas operadas con el núcleo $\mathbf{K_{ij}}$. [30][31]

  Esta operación de convolución produce un "mapa de características" que resalta ciertas características de la imagen, como bordes o texturas. Los valores de cada uno de estos filtros o núcleos los determina la red neuronal durante el entrenamiento.

- **Convolución + ReLU[8]:** En la capa de convolución, generalmente se aplica al valor de la combinación lineal una función de activación $\mathbf{g}$ no-lineal o función de activación ReLU (unidad lineal rectificada). [29]

  $$\mathbf{y_j} = \mathbf{g}\left(\sum \mathbf{K_{ij}} \oplus \mathbf{y_i} \right)$$

  De esta forma se introduce no linealidad en la red y promueve la comprensión de representaciones más complejas.

- **Capa de pooling:** La capa de ***pooling*** se utiliza para reducir la dimensión de los mapas de características, es decir, su resolución. El ***pooling*** busca destacar la información más importante de los mapas de características con la reducción de resolución, la forma más común es el "***max pooling***".[29][31]

  El ***"max pooling"*** divide el mapa de características en matrices y se toma el valor máximo de cada una, reduciendo así la cantidad de información mientras se conservan las características importantes.

[8] Rectified Linear Unit

En la segunda fase, se usaría la gran cantidad de mapas de características conseguidos, ahora con una resolución viable para que la red neuronal multicapa pueda analizarlos. De esta forma el diagrama de esta arquitectura como podemos ver en la figura 5 forma un embudo, ya que disminuimos la resolución pero aumentamos el número de mapas de características.[29]

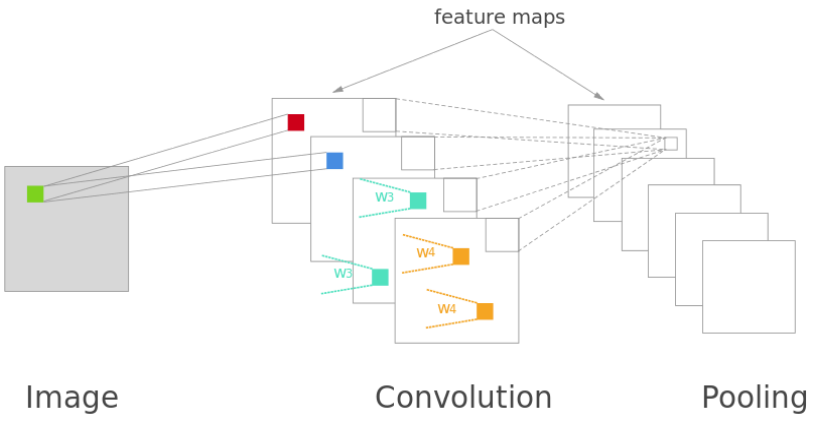

Figura 5: Diagrama de una red neuronal convolucional. [32]

Según Sebastian Raschka y Vahid Mirjalili, la red neuronal convolucional (CNN) utilizan una jerarquía de características combinando las de bajo nivel con una estructura de capas para formar características de alto nivel.[31]

6.3. Redes Neuronales Residuales (ResNet)

Las redes residuales son similares a los perceptrones multicapa, salvo porque estas introducen unas nuevas conexiones que van de las capas menos profundas a las capas más profundas. Las conexiones "atajo" de este tipo de arquitectura realizan mapeos de identidad, es decir, fusionan el valor de origen de la conexión con la salida de la capa destino por adición.

Esta arquitectura se originó para solucionar el problema del desvanecimiento del gradiente, este se origina al aplicar secuencialmente más capas a un perceptrón multicapa y provoca una rápida reducción de la precisión del entrenamiento. Cuando en las capas profundas el gradiente es cercano a cero en el entrenamiento, se hace imposible cambiar los pesos de dichas capas.[33]

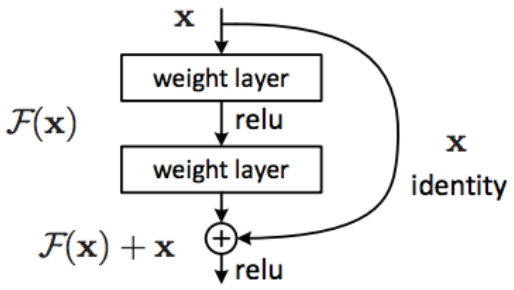

Figura 6: Diagrama de un bloque residual de una red neuronal residual.[34]

Esta red es muy efectiva en el entrenamiento de modelos que necesitan de una arquitectura muy compleja y profunda, donde al aumentar significativamente el tamaño de nuestra arquitectura se deriva en un aumento de la precisión.

Dentro de las redes residuales, tenemos una arquitectura llamada red densa o más comúnmente DenseNet, en la cual, cada capa se conecta directamente con todas las capas subsiguientes en la red. Esta arquitectura se fundamenta entonces en que cada capa obtiene entradas adicionales de todas las capas anteriores, dicho de otra forma, la salida de cada capa se concatena con la entrada de todas las capas subsiguientes en lugar de realizar la operación de agregación de la ResNet.[13] [31]

En esta tipología, la cual se dice que está densamente conectada, se provoca que la información fluya de forma más directa y la representación que esta arquitectura obtiene de los datos es más completa. Normalmente se reduce la cantidad de capas respecto a otras arquitecturas, lo que hace que esta red sea más eficiente computacionalmente, con ello también se facilita el flujo del gradiente durante el entrenamiento, lo que ayuda a evitar el desvanecimiento del gradiente.[35]

6.4. Redes Neuronales Recurrentes (RNN[9])

La arquitectura RNN se caracteriza en que las conexiones entre las neuronas forman ciclos, permitiendo que la información se retroalimente, usando estas para el tratamiento de series de datos secuenciales, a través del tiempo. Es decir, la salida en un momento de tiempo determinado, se tiene en cuenta en la entrada de un momento posterior.[13]

Además, las RNN son más eficientes que otros tipos de redes neuronales para ciertas tareas, como tiene la posibilidad de crear ciclos de recurrencia , son especialmente útiles para trabajos que implican datos secuenciales, debido a la capacidad de recordar parámetros a lo largo del tiempo y manejar entradas de longitud variable. Por ejemplo tenemos el tratamiento y predicción de secuencias temporales, vídeos, audio o generación de texto. Esta arquitectura, al ser recurrente es muy profunda, lo que puede derivar en la degradación del gradiente. Para solucionar este problema de las RNN, surgen variantes como las arquitecturas LSTM[10] y GRU[11]. Las redes LSTM fueron inventadas en 1997 y fueron una revolución en la eficiencia de tareas como el modelado del habla, superando a los modelos tradicionalmente usados en este campo.[36]

6.5. Redes Neuronales Generativas (GAN[12])

Las redes generativas antagónicas (GAN), son una arquitectura que se basa en el paradigma del aprendizaje no supervisado. La funcionalidad de esta red se basa en enfrentar dos modelos de redes neuronales, un generador y un discriminador, que se enfrentan en un juego de suma cero.[37]

El modelo generador, tomando datos aleatorios llamados semilla, genera resultados que intentan replicar la distribución de los datos de entrenamiento. El modelo discriminador en cambio, intentará discernir entre los datos generados por el modelo generador y las muestras del conjunto de datos de entrenamiento. En este proceso, el modelo generador se irá aproximando a la creación realista, mientras que el discriminador mejorará su capacidad para distinguir lo real de lo generado por inteligencia artificial. Este modelo de red es muy utilizado en la creación de imágenes realistas o texto artificial, ya que por ejemplo, una foto generada por una buena GAN, al haber sido entrenados tanto generador como discriminador, puede ser tan realista que pase desapercibida por el ser humano.[37]

[9] Recurrent Neural Networks
[10] Long Short-Term Memory
[11] Gated Recurrent Unit
[12] Generative Adversarial Networks

7. MÉTRICAS PARA EVALUAR LA VALIDEZ DE UNA RED NEURONAL

7.1. Matriz de Confusión

La matriz de confusión es la herramienta más básica para realizar la evaluación del desempeño de un modelo de clasificación. En la matriz se presenta el rendimiento que la red neuronal tiene con un conjunto de datos de prueba, mostrando la relación entre la clasificación real de los datos y la predicción del modelo. Como podemos ver en la figura 6, las filas representan las clases reales y las columnas representan las predicciones del modelo.[38]

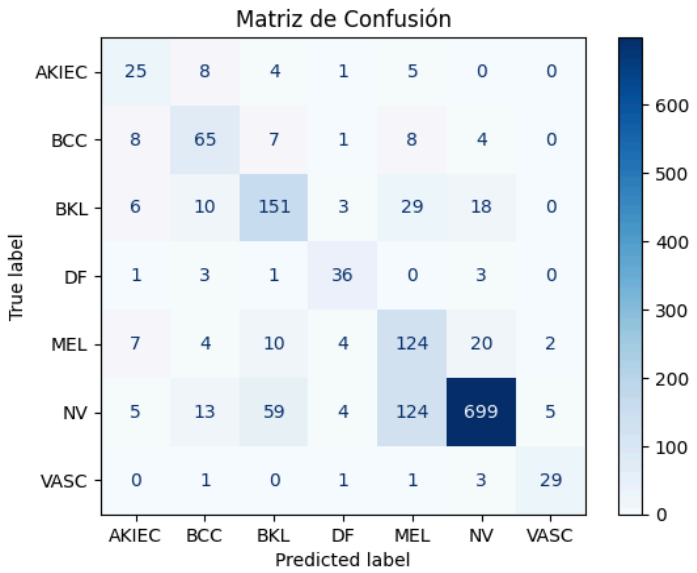

Figura 7: Matriz de confusión de un modelo de siete clases.

Estableciendo la base en una clasificación binaria, de esta clasificación podemos extrapolar a más clases, obtendremos cuatro categorías en la matriz de confusión:[38]

- Verdaderos Positivos (TP[13]): Predicción de verdad y verdad en realidad.
- Verdaderos Negativos (TN[14]): Predicción de falso y falso en realidad.
- Falsos Positivos (FP[15]): Predicción de verdad y falsas en realidad.
- Falsos Negativos (FN[16]): Predicción de falso y verdad en realidad.

Teniendo en cuenta estos valores, obtenemos dos tipos de errores que puede arrojar nuestro modelo al ponerlo a prueba:

- Error Tipo I (FP): Es el rechazo de una hipótesis nula verdadera, es decir, cuando rechazamos el hecho falso, prediciendo verdad, cuando en realidad es falso. En el contexto clasificatorio, nos referimos a asignar incorrectamente una instancia negativa como positiva.
- Error Tipo II (FN): Es la aceptación de una hipótesis nula falsa, es decir, cuando afirmamos que algo es falso cuando en realidad es verdadero. En el contexto clasificatorio, nos referimos a asignar incorrectamente una instancia positiva como negativa.

[13] True Positive
[14] True Negative
[15] False Positive
[16] False Negative

7.2. Exactitud y Valores Predictivos

Exactitud (***Accuracy***): Proporción de todas las predicciones del modelo que son correctas, positivas y negativas. Es decir, la exactitud mide la calidad global de las predicciones del modelo. La exactitud, al igual que la precisión se ve afectada por el desbalance de clases.[39][40]

$$\textbf{Accuracy} = (TP + TN) / (TP + TN + FP + FN)$$

Exhaustividad (***Precision***) o Valor Predictivo Positivo: Es la proporción de positivos predichos por el modelo que realmente se tratan de muestras positivas. La exhaustividad o PPV[17] mide la calidad de las predicciones positivas del modelo y es susceptible de verse afectada por el desbalanceo de clases, por ejemplo si tuviéramos 95 puertas positivas de la clase A y 5 de la clase B, la precisión será del 95% si predice todo como A. Es importante tener en cuenta el contexto del modelo, ya que en el ámbito médico el desbalanceo de clases es muy común.[39][40]

$$\textbf{Precision} = TP / (TP + FP)$$

Valor Predictivo Negativo (NPV[18]): Es la proporción de negativos predichos por el modelo que realmente se tratan de muestras negativas.[39]

$$\textbf{NPV} = TN / (TN + FN)$$

Tanto el PPV como el NPV, representan la probabilidad de que una predicción sea cierta, teniendo en cuenta una clasificación binaria. Estas métricas junto con ***accuracy*** se encuentran entre las más usadas en el análisis del rendimiento de un modelo sin embargo como se explica no son perfectas, a continuación se tratarán más métricas que dan un conocimiento más profundo del funcionamiento del modelo. En la figura 7 podemos ver la conexión que subyace entre todas las métricas.[39]

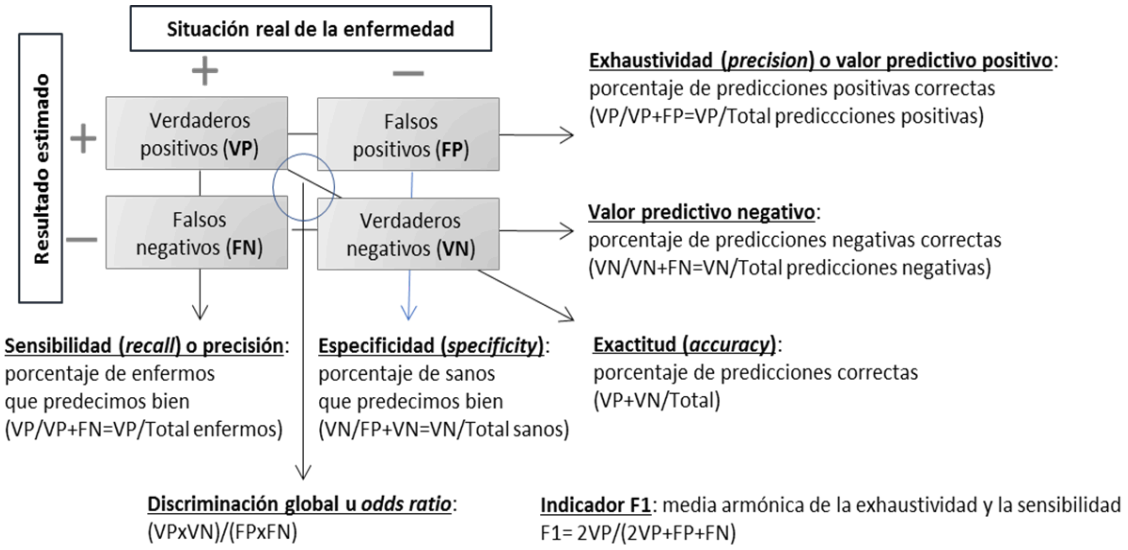

Figura 8: Métricas para la evaluación de modelos predictivos.[40]

[17] Positive Predictive Value
[18] Negative Predictive Value

7.3. Sensibilidad y Especificidad

Sensibilidad (***Recall***) y Especificidad (***Specificity***): Se refieren, en una clasificación binaria, a la tasa de aciertos de la clase A (cáncer) y a la tasa de aciertos de la clase B (no cáncer) respectivamente. El ***recall*** también se conoce como TPR[19], es decir, tasa de verdaderos positivos. A su vez la métrica FPR[20], la tasa de falsos positivos, será (1 - ***specificity***).[39][40]

$$\textit{Recall} = TP / (TP + FN) \qquad \textit{Specificity} = TN / (TN + FP)$$

En el contexto de la medicina, para las pruebas de diagnóstico, lo cual es objetivo de nuestro modelo, hay que tomar la decisión de qué tipo de prueba nos interesa más hacer.

- Prueba sensible (***recall***): Usadas en las primeras fases de un estudio médico, el objetivo de la prueba es descubrir o descartar alguna enfermedad, cuando para esta a priori la probabilidad es baja.

- Prueba específicas (***specificity)***: Usadas para confirmar una enfermedad concreta, normalmente en una fase más avanzada del estudio, cuando un diagnóstico positivo puede hacer daño física o emocionalmente al paciente, ya sea por la gravedad de la enfermedad o por un tratamiento muy agresivo.

En nuestro caso, como uno de los objetivos del modelo es que sirva de cribado y para el diagnóstico en etapa temprana y teniendo en cuenta que el diagnóstico no es muy grave, buscamos conseguir un modelo sensible, para detectar el mayor porcentaje posible de casos positivos y realizar pruebas de confirmación posteriormente. Si el modelo detectara otro tipo de cáncer habría que revalorar esta métrica, sin embargo como la opinión final de un diagnóstico recae en los profesionales sanitarios, para un diagnóstico ágil en principio siempre se necesitará un modelo sensible.[40]

Para concluir esta sección, definimos la métrica de puntuación ***F1-score***. En estadística, el valor ***F1-score*** (también conocido como ***F-score*** o métrica F1) es una medida de la precisión de una prueba, muy utilizado cuando las muestras son desbalanceadas. Se calcula como la media armónica de la exhaustividad ***(precision***) y la sensibilidad (***recall***).[40]

$$F_1 = 2 \cdot (\text{Precision} \cdot \text{Recall} / \text{Precision} + \text{Recall})$$

Los posibles valores de la prueba ***F1-score*** están entre 0 y 1. Una prueba perfecta, en caso de tener un modelo clasificador perfecto, arrojaría un ***F1-score*** igual a 1, ya que tanto su ***precision*** como su ***recall*** valdrían 1. En cambio si un clasificador fuera malo, el valor mínimo posible es 0, que se producirá cuando, ***precision*** y ***recall***, una o ambas sean igual a 0.[41]

[19] True Positive Rate
[20] False Positive Rate

7.4. Curva ROC y AUC-ROC

Para explicar el funcionamiento de la curva ROC[21] es necesario hablar del umbral de discriminación, considerando un modelo de clasificación binaria, es el valor que determina cómo se asigna una muestra a una de las dos categorías. En un modelo de diagnóstico médico como el nuestro, este umbral sería la probabilidad por encima del cual se clasifica una observación como positiva.

La curva ROC es una representación gráfica definida por la TPR (tasa de verdaderos positivos) y la FPR (tasa de falsos positivos) como ejes x e y, respectivamente. Esto representa el balance entre verdaderos positivos y falsos positivos a medida que se varía el umbral de discriminación, dicho de otra manera beneficios frente a costes.[42]

Dado que la TPR, como hemos visto antes en el apartado 7.3, se corresponde a la sensibilidad y la FPR equivale a (1 - especificidad), la curva ROC también se denomina gráfica de sensibilidad frente a (1 - especificidad). De cada resultado de predicción correspondiente a un umbral de discriminacion diferente, se obtiene una instancia de la matriz de confusión y así se representa un punto en el espacio ROC.[42]

La diagonal de un gráfico de curva ROC representa un modelo clasificador aleatorio, así que cuanto más lejos esté la curva ROC de esta diagonal, mejor será el rendimiento del modelo. La curva ROC ideal se desplaza hacia la esquina superior izquierda del gráfico, lo que indica un modelo con alta sensibilidad y alta especificidad, aunque en realidad, como lo que marca el poder predictivo es la distancia a la diagonal, si la curva más lejana estuviera desplazada hacia la esquina inferior derecha, solo habría que invertir las predicciones.[42]

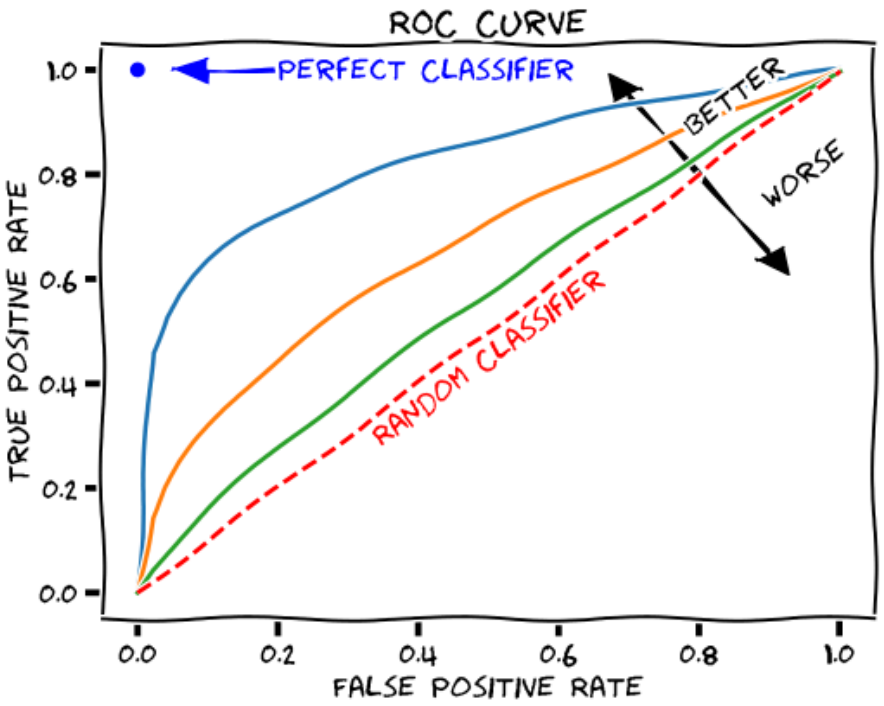

Figura 9: Gráfico de una curva ROC y métrica AUC.[43]

[21] Receiver Operating Characteristic

La información proporcionada sobre el rendimiento de un clasificador binario en la visualización de la curva ROC, se puede resumir con la métrica AUC-ROC. La métrica AUC[22], mide el área bajo la curva ROC, cuanto mayor sea la puntuación AUC mejor será el rendimiento del modelo de clasificación. La figura 8 muestra que para un clasificador aleatorio, representado por la diagonal, el AUC es 0.5, y para un clasificador perfecto, el AUC es 1.0. Esto representa el área bajo sus correspondientes curvas ROC.

7.5. Curva PR y AUC-PR

También tenemos que hablar de la curva PR (precision-recall) y de la métrica AUC-PR (Area Under Curve precision-recall). Estas son equivalentes a la curva ROC y la métrica AUC-ROC salvo que deben usarse para valorar el rendimiento de los modelos de clasificación cuando los conjuntos de datos son desequilibrados.[44]

La curva PR[23] Como se puede ver en la figura 9, está definida por el eje Y representando ***precision*** y el eje X representando ***recall***. Cada punto de la curva PR representa un par de valores de ***precision*** y ***recall*** obtenidos para un umbral de discriminación particular. Lo ideal sería una curva que se acerque lo máximo posible a la esquina superior derecha (alta ***precision*** y alto ***recall***).[44] Aplicando la métrica AUC a la curva PR obtenemos de forma resumida el rendimiento del modelo clasificador, una métrica AUC-PR más alta indica un mejor rendimiento, siendo la ideal igual a 1.0 y la peor, osea un clasificador aleatorio, igual a 0.5.[45]

Cuando se trata de conjuntos de datos desequilibrados, es decir, cuando la clase positiva ocurre pocas veces, AUC-ROC puede ser engañoso porque el desequilibrio de clases puede simplemente causar que TPR y FPR no sean representativos. En cambio, como AUC-PR refleja el desempeño del modelo en términos de precision y recall, en situaciones de desequilibrio de clases es más adecuado para evaluar el rendimiento de los modelos.[44]

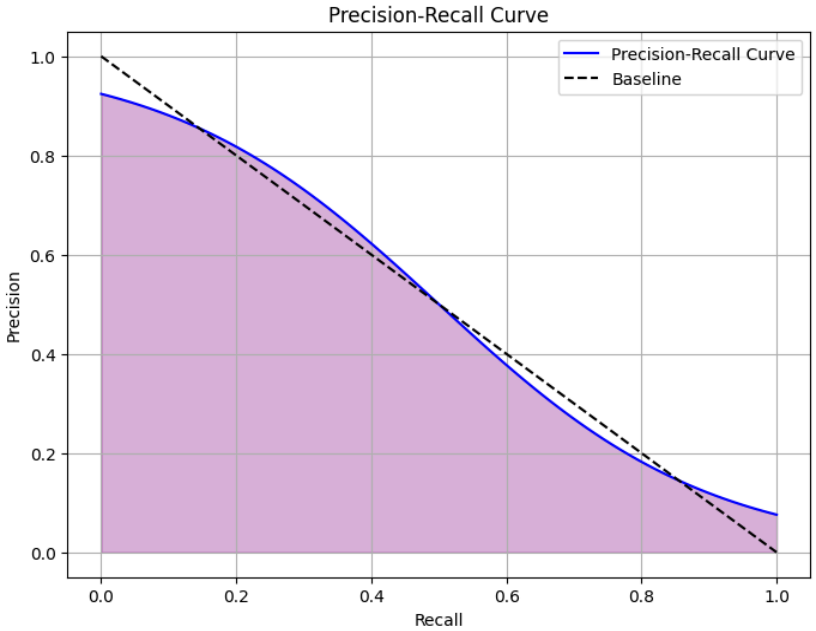

Figura 10: Gráfico de una curva PR y métrica AUC-PR.

[22] Area Under Curve
[23] precision-recall

7.6. Ajuste de la base de datos e hiperparámetros

Para el entrenamiento de un modelo de Deep Learning es necesario que el conjunto de datos sea dividido en tres subconjuntos: [37] [40]

- ***Training***: Conjunto de datos con el que se entrena el modelo, es decir, son los datos con en los que se basa el modelo para modificar los pesos. Este subconjunto suele estar entre el 70% y el 80% del volumen de la base de datos y ha de ser lo suficientemente grande y representativo. Cabe la posibilidad de que la red se aprenda de memoria este conjunto en vez de generalizar, consiguiendo un falso ***accuracy***, es lo que se llama ***overfitting***.[42]

- ***Validation***: Conjunto de datos con el que se valida el modelo, se utiliza para comprobar la capacidad de predicción de cada ***epoch***. Este conjunto es un subconjunto del conjunto ***training*** y requiere entre el 20% y el 30% del mismo. Este subconjunto no es visible durante el entrenamiento y es el que usa la red para optimizar las métricas.[42], [46]

- ***Test***: Conjunto de datos nuevos y desconocidos por nuestro modelo usados para probar la generalización del mismo. Se realiza esta comprobación una vez ha acabado el entrenamiento y nos da una visión más completa de la eficiencia del mismo, además con esta comprobación podemos prevenir el ***overfitting***. Este subconjunto es lo que queda del volumen completo de los datos cuando le quitas los usados para el ***training*** y ocupa entre el 20% y 30% del volumen total.

No existe ninguna regla que especifique el tamaño exacto que han de tener estos subconjuntos, pero se ha dado un aproximado bastante usado en la práctica.

Es vital mencionar los siguientes hiperparámetros que serán de gran importancia a la hora de configurar el entrenamiento del modelo:[42]

- Iteración: En cada iteración se procesa simultáneamente un lote de entrenamiento, se calculan los gradientes de los pesos respecto a la función de coste y se actualizan los pesos de la red.

- ***Epoch***: Una época es el proceso de entrenamiento que realiza la red con todo el conjunto de datos de ***training***, por lo tanto la red recorre todo el subconjunto de ***training*** tantas veces como ***epochs*** se especifiquen.

- ***Batch***: Tamaño del vector de datos analizado en cada iteración, es decir, el subconjunto de datos de ***training*** que se procesan simultáneamente en la red neuronal durante una iteración.

Si tenemos que el tamaño de ***batch*** es **X** y el tamaño de todo el conjunto de ***training*** es **N**, se necesitarán **N/X** iteraciones por cada época para completar el entrenamiento. Además el hiperparámetro ***epoch*** es bastante importante, según este aumente, más veces se actualizan los pesos, produciendo si es muy pequeño ***underfitting***, pasando por un ajuste óptimo, a ***overfitting*** si el valor es muy grande.[42]

8. CAPACIDAD COMPUTACIONAL EN EL DESARROLLO DE REDES NEURONALES

La capacidad computacional es otra métrica a tener en cuenta en el contexto de las redes neuronales complejas, pues esta es necesaria tanto para el entrenamiento de la red como para obtener una predicción. En este estudio se busca implementar una red lo suficientemente sencilla para que pueda ser almacenada y explotada como un servicio web, así como implementarla de cara al futuro en dispositivos móviles, lo que ayudaría al uso de la herramienta en el campo médico.

Con el fin de adaptar la capacidad computacional que podríamos tener en un móvil o en una web y para que funcione con relativa rapidez, se van a usar redes neuronales que tengan en torno a 5 millones de parámetros. Este número de parámetros se sustenta en las pruebas que se van a realizar en la parte práctica y los modelos probados, por ejemplo las eficientes arquitecturas de las redes ***EfficientNet***. Las arquitecturas ***EfficientNet***, desarrolladas por ***Google***, están diseñadas optimizar la capacidad computacional y tener un buen rendimiento en tareas como el reconocimiento de imágenes.[47] Las arquitecturas que usaremos en el proyecto serán de la familia ***EfficientNet***, ***DenseNet y NASNet***.

8.1. Paralelización y procesamiento masivo

La paralelización es un aspecto importante para optimizar la capacidad computacional, esto implica distribuir la carga de trabajo en diferentes núcleos de proceso para realizar cálculos simultáneos y de esta forma acelerar el entrenamiento y procesamiento de las redes neuronales.

La paralelización del procesamiento en las redes neuronales es muy importante, ya que contamos con millones de parámetros que son actualizados de manera conjunta. Analizando el hardware de procesamiento, se pueden usar dos dispositivos, la CPU (***Central Processing Unit***) y la GPU (***Graphics Processing Unit***), mientras que la primera tiene núcleos de gran potencia diseñados para muchos tipos de procesos que se realicen de forma serial, la GPU está diseñada para abordar de forma eficiente el paralelismo de procesos específicos y simples. Como el paradigma del ***Deep Learning*** se basa en realizar muchas operaciones muy específicas de manera simultánea, la GPU, por su diseño se adapta mejor.

Además de la utilización de la GPU, existe un tipo de hardware llamado unidad de procesamiento tensorial o TPU (***Tensor Processing Unit***), es un circuito integrado de aplicación específica diseñado para acelerar operaciones relacionadas con redes neuronales. Este hardware está diseñado por Google, y orientado a la biblioteca de código abierto TensorFlow. En comparación con el hardware de procesamiento gráfico las unidades TPU están diseñadas para operar con menor precisión un mayor volumen de cálculo y además no incluyen el hardware dedicado a tareas como la cartografía de texturas, esto las hace una unidad de procesamiento muy potente y eficiente para el ***Deep Learning***.[48]

Parte 2: Marco práctico

9. MATERIAL Y MÉTODOS

Concluido el estudio teórico voy a proceder a explicar el desarrollo de los experimentos que he llevado a cabo hasta concluir los diferentes modelos con sus respectivas métricas. Voy a describir tanto las bases de datos que he usado y probado en el proceso, como el tratamiento de las mismas y la implementación del pipeline desarrollado.

9.1. Hardware y software

En cuanto al hardware usado en el desarrollo, he contado con un ordenador que cuenta con una capacidad suficiente para desarrollar modelos ciertamente pesados con facilidad.

- Procesador (CPU): AMD Ryzen 5 2600X, se trata de un procesador de gama media pero con una relación de potencia muy interesante al prescindir de GPU integrada, así como dispone de Overclock de fábrica, junto con refrigeración líquida da un buen rendimiento.

- Memoria (RAM): El dispositivo cuenta con 16GB de memoria RAM a 3200MHz. Esta capacidad de memoria física nos capacita para entrenar redes de complejidad moderada.

Como la CPU no cuenta con una GPU integrada, el dispositivo cuenta con una gráfica Geforce 1660 TI OC[24], pese a esto el entrenamiento de las redes se ha realizado con la CPU. Esto se debe a que el sistema operativo utilizado para el desarrollo de la práctica es Windows 10, y el entrenamiento de redes neuronales en este sistema operativo no está completamente estandarizado, de hecho ya no tiene soporte. Las librerías que necesita Nvidia para entrenar las redes dependen de las versiones de Python, CUDA[25] y librerías CNN, lo que complica la configuración. Realice una prueba con la configuración de software correcta, pero este entorno de trabajo en vez de usar la memoria RAM, usa la memoria de la gráfica, que resulta insuficiente. Dicho esto procedo a enunciar el software utilizado:

- **Sistema operativo(OS):** Windows 10 Professional, aunque se recomiendan distribuciones de Linux para el desarrollo de modelos neuronales debido a su compatibilidad.

- **Frameworks de Machine Learning:** Se ha usado como lenguaje principal Python 3.11.8. He utilizado principalmente las librerías de ***TensorFlow***, que ahora también incluyen las de Keras, para el desarrollo de los modelos, a su vez me he apoyado en ***PyTorch*** para el aumento de las imágenes y en *Scikit-learn* y Pandas para el preprocesado.

- **Entorno de desarrollo:** El editor de código utilizado es ***Visual Studio Code***, complementado con ***Jupyter***. Esto permite una implementación fluida e interactiva del código, modulando sus partes para estructurar y hacer más configurable el ***pipeline*** de los datos.

[24] Overclock
[25] Compute Unified Device Architecture

9.2. Redes utilizadas

Para hablar de las arquitecturas que se han elegido para el desarrollo, primero tengo que exponer la existencia de unos concursos organizados por el ISIC que han sido de gran ayuda y la base de este trabajo, pues me apoyo en sus métodos y métricas para evaluar los resultados de los diferentes modelos desarrollados.

Durante varios años el ISIC propuso una serie de objetivos sobre diferentes bases de datos, con el fin de desarrollar modelos de clasificación binarios, multiclase o segmentación de características, en estos concursos han participado empresas y multinacionales de todo el mundo, así como universidades y ***data miners*** de renombre[49].

Siguiendo los preceptos que establecen los ganadores de las diferentes convocatorias, la única manera de llegar a unos resultados decentes, teniendo en cuenta la complejidad intrínseca del problema a resolver, es usando un modelo pre entrenado con ***Imagenet***, un proyecto de entrenamiento de modelos para el reconocimiento y clasificación de imágenes con 14 millones de imágenes de muestra.[50] A esta técnica se la apoda como ***transfer learning***, que consiste en usar un modelo entrenado previamente para una tarea similar, en este caso el reconocimiento de imágenes, y que así pueda compartir información con el modelo resultante.[51]

Dicho lo anterior se han seleccionado tres arquitecturas pre entrenadas diferentes para realizar los modelos, se tratan de arquitectura medianamente complejas, y todas han sido entrenadas con ***Imagenet***. Se ha descartado la implementación de redes más típicas como una CNN ad hoc porque al tener arquitecturas más básicas y no haber sido entrenadas con ***Imagenet*** no llegaríamos a buenos resultados.

- ***EfficientNet:*** Se trata de una familia de arquitecturas diseñadas con el fin de optimizar la capacidad computacional consiguiendo buenos resultados en términos de precisión. Se podría hablar de una arquitectura escalable dependiendo de la complejidad del problema y la capacidad computacional, podemos encontrar desde la ***EfficientNet-b0*** con 4 millones de parámetros hasta la ***EfficientNet-b7***, que cuenta con unos 70 millones de parámetros. La complejidad de esta familia crece exponencialmente hasta la arquitectura mencionada. Es una arquitectura muy usada en clasificación de imágenes cuando se trabaja con capacidad computacional limitada.

- ***DenseNet:*** Esta otra familia de arquitecturas, desarrollada en la teoría, nos provee de modelos pre entrenados clasificados en categorías como las ***DenseNet-121***, que cuenta con modelos de 121 capas densas que van de los 5 a los 7 millones de parámetros. También podemos encontrar conjuntos más complejos como ***DenseNet-264*** que llegan hasta los 70 millones de parámetros.

- ***NASNet:*** Esta familia de arquitecturas ha sido desarrollada con técnicas de búsqueda de arquitectura neural. Es decir, buscar un bloque sobre un conjunto de datos pequeño y luego transferirlo a un conjunto mayor. Esta arquitectura ha sido una de las que mejores resultados ha dado en tareas de reconocimiento de imágenes en relación a su tamaño. Concretamente nos centraremos en la versión pequeña de esta familia llamada ***NASNet-Mobile***.

9.3. Conjuntos de datos

Durante los diferentes años el ISIC ha provisto distintos ***Dataset*** de imágenes obtenidas mediante dermatoscopio. Los metadatos ofrecidos con las imágenes cambian dependiendo del año y el objetivo del concurso por lo que vamos a profundizar en los diferentes dataset, los metadatos y las clasificaciones de los mejores modelos desarrollados, para tener una visión objetiva sobre la meta a la que podemos llegar en nuestro desarrollo.

9.3.1. Métricas de evaluación

Es importante destacar que al igual que estos concursos, vamos a utilizar ***balanced multiclass accuracy*** como métrica de referencia para nuestros resultados, teniendo también en cuenta, pero de manera más subjetiva el recall de las clases cancerígenas y potencialmente cancerígenas, pues nos interesa una prueba sensible.

$$\textbf{Balanced Multiclass Accuracy} = \frac{1}{N}\sum_{i=1}^{N}\frac{TP_i}{TP_i + FN_i}$$

Esta métrica es bastante buena para medir el desempeño de los modelos de clasificación multiclase, sobre todo usada cuando se trabaja con muestras desbalanceadas. Como se puede ver la métrica se calcula como la media de los recall de las distintas clases, donde **N** es el número de clases de la base de datos.

9.3.2. ISIC Challenge 2018

El ***challenge*** de 2018[52], [53] consistió en 3 subtareas: segmentación de lesiones, segmentación de atributos en las lesiones y categorizar en 7 clases las lesiones. Nos centraremos en el diagnóstico multiclase, donde se puede observar que el mejor modelo obtiene una puntuación de 0.88, siendo este un ensamble de 10 modelos distintos. Por lo general, los mejores resultados en este campo se consiguen aplicando la técnica de ***averaging ensemble***, por lo cual más adelante enfocaremos el desarrollo en esta dirección. Para hacer una valoración en igualdad de condiciones hay que tener en cuenta el uso de datos externos para el desarrollo de los modelos en este campeonato, si solo nos fijamos en los modelos que no usan ningún tipo de base de datos externa a la del concurso, el mejor modelo tiene una puntuación de 0.84, y el TOP 10 estaría en la puntuación 0.74, lo cual tendremos en cuenta para valorar nuestros resultados.

Las bases de datos que utiliza el conjunto ofertado son HAM10000[26] Dataset[53] y MSK[27] Dataset[52], [54], el primero de los dos es bastante conocido y formado por 10.000 imágenes su nombre hace referencia a la comparación entre el diagnóstico realizado por los humanos y las máquinas. En conjunto, forman un set de ***training*** de 10.015 imágenes, que corresponde al dataset HAM10000, 195 imágenes para el set de ***validation*** y de 1.514 imágenes para el set de ***test***. Nos basaremos principalmente en este ***dataset*** para el desarrollo de la práctica así como fusionaremos los set ***training*** y ***validation*** ofertados para hacer la división a nuestro gusto.

[26] Human Against Machine
[27] Memorial Sloan Kettering Cancer Center

9.3.3. ISIC Challenge 2019

El challenge de 2019 también cuenta con una tarea de diagnóstico de lesiones, aunque se introduce alguna clase más al modelo, siendo esta la razón por la que aunque se ofrece una base de datos con más del doble de muestras los resultados del concurso son peores. Por esto, he decidido prescindir de este ***dataset***.

Las bases de datos que utiliza el ***challenge*** de 2019 son HAM10000 Dataset[53], MSK Dataset[52], [54] y además BCN_20000 Dataset, este último es lo que marca la diferencia con el conjunto de 2018. En total contamos con 25.331 imágenes para el set de ***training***, que es el único que nos ofrecen.

9.3.4. ISIC Challenge 2020

Este concurso[55] cambió el enfoque respecto a los anteriores, buscaron conseguir un modelo de clasificación binaria, para detectar las lesiones malignas. Esta técnica, aunque es interesante, es menos útil a mi parecer de la clasificación por tipos. Además, si nos enfocamos en clasificar las lesiones según sus atributos, podemos componer diferentes conjuntos, y después clasificar los conjuntos como malignos, benignos o potencialmente malignos. Este conjunto pese a clasificar las muestras de forma binaria, cuenta con un pequeño porcentaje de muestras clasificadas según el tipo de lesión, lo que se podría aprovechar para aumentar los otros ***dataset***. Sin embargo se ha preferido no utilizar estas muestras para evitar añadir posibles sesgos, pues es seguro que la fuente de las imágenes que contienen esta información es la misma.

Este ***dataset***[56] es el más grande que ofrece el ISIC, contando con un total de 33.126 imágenes para el set de entrenamiento y 10.982 para el set de ***test***, por esto y por tratarse de una clasificación binaria las puntuaciones de los diferentes modelos son bastante altas. Esta base de datos está formada por bases de datos de múltiples entidades médicas y académicas, siendo esto bueno para obtener una buena variabilidad en los datos.

9.3.5. Conclusión

En conclusión, se ha decidido optar por utilizar el dataset del 2018 para el desarrollo de la práctica, y de esta forma evitar introducir algún sesgo, aunque de cara a futuros trabajos se podría investigar la viabilidad de añadir estos datos.

Del dataset de 2018 se usarán los sets de ***training*** y ***validation*** conjuntamente para generar los nuestros, y el set de test se usará exclusivamente para probar los resultados de los entrenamientos.

Todo el código del desarrollo que se va a exponer a continuación se encuentra en un archivo de ***Jupyter Notebooks*** en el repositorio https://github.com/Arturo7201/ML_Lesion_Diagnosis

Además, el código se encuentra documentado en el Anexo I, donde se explican los diferentes bloques con más profundidad.

9.4. Análisis exploratorio

Empezaremos esta sección comentando las diferentes clases que contiene la base de datos:

- **MEL**[28]**:** Clase que representa las lesiones de tipo melanoma, un tipo de cáncer de piel ya comentado, es el cáncer de piel más peligroso y agresivo.

- **NV**[29]: Clase que representa a los lunares comunes, con gran variabilidad dentro de la misma, algunos nevus o lunares tienen capacidad de evolucionar a melanoma.

- **BCC**[30]**:** Clase que representa al tipo más común de cáncer de piel pero menos invasivo que el MEL.

- **AKIEC**[31]: Clase que representa a un conjunto de lesiones cancerosas o precancerosas, entre ellas la queratosis solar, la enfermedad de Bowen o el carcinoma de células escamosas.

- **BKL**[32]: Clase que en contraposición con la anterior representa el conjunto de queratosis benignas, como la seborreica o el lentigo solar.

- **DF**[33]: Una lesión benigna por la proliferación del tejido fibroso, sin peligro que se puede confundir con MEL por su apariencia.

- **VASC**[34]: Clase que representa problemas benignos en los vasos sanguíneos de la piel.

Teniendo definidas las clases de la base de datos, también las agrupamos en 3 conjuntos, teniendo en cuenta su peligrosidad:

- **Lesiones Cancerígenas**: Incluye la clase MEL y la clase BCC, aunque la primera sea más peligrosa, ambas clases representan lesiones cancerosas.

- **Lesiones Potencialmente Cancerígenas**: Claramente la clase AKIEC forma parte de este conjunto, además para asegurarnos la sensibilidad de la prueba debemos incluir la clase NV, pues dependiendo de la lesión puede necesitar seguimiento profesional.

- **Lesiones Benignas**: Esta clase representará el conjunto de lesiones que no suponen peligro alguno para la salud, BKL, DF y VASC.

Estas 3 superclases las usaremos como métrica de apoyo calculando los ***recall*** de cada una, interesandonos que el ***recall*** de las superclases lesiones cancerígenas y lesiones potencialmente cancerígenas sea más sensible que el de las lesiones benignas, en caso de tener la métrica ***balanced multiclass accuracy*** parecida en algunas pruebas.

[28] Melanoma
[29] Nevus Melanocítico
[30] Carcinoma Basocelular
[31] Queratosis Actínica
[32] Queratosis Benigna
[33] Dermatofibroma
[34]Lesiones Vasculares

9.4.1. Visualización y análisis descriptivo de los datos

Procedemos a analizar los metadatos disponibles del set correspondiente a HAM10000 de la base de datos del 2018. Primero analizaremos la distribución de las edades de los pacientes con un histograma. Como podemos ver en la figura 10, la media de edad está en torno a los 50 años, destaca la zona central comparándolo con lo estudiado en la epidemiología donde observamos un crecimiento más grande en edades avanzadas y no tanto en la zona central. Se puede suponer que hay algún tipo de sesgo en la selección de las muestras, pues así lo refleja esta figura.

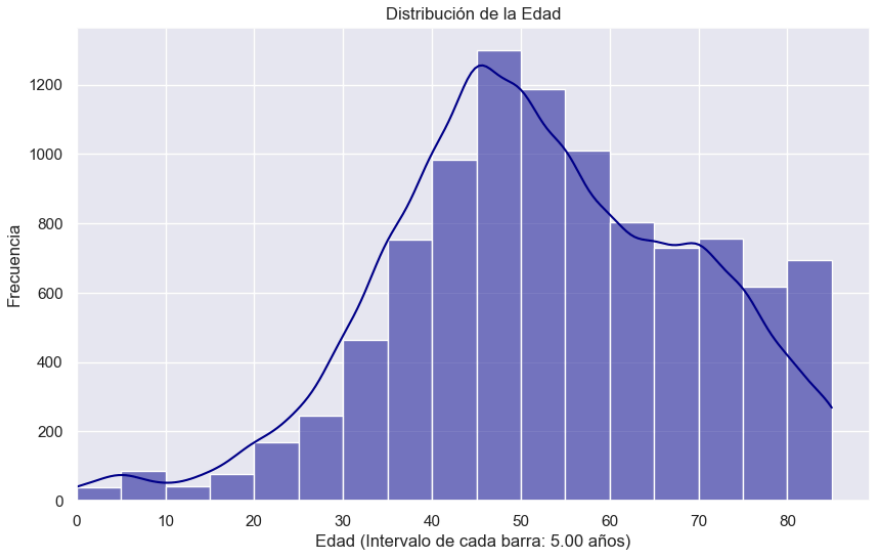

Figura 11: Histograma de la edad de los pacientes.

En la figura 11 observamos la distribucion por sexo, donde podemos ver que se refleja bastante lo visto en la epidemiologia, con una leve superioridad en los casos del sexo masculino, por lo cual, respecto a esta información el set de datos está bastante equilibrado.

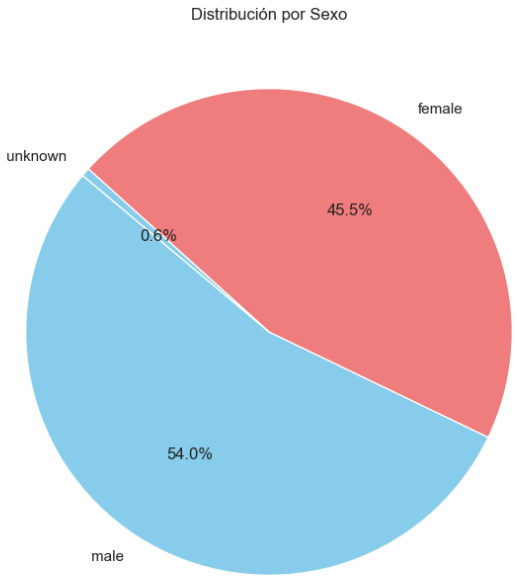

Figura 12: *Pie chart* de la distribución por sexo de los pacientes.

Por último analizaremos la meta información respectiva al lugar donde aparece la lesión. Como podemos ver en la figura 12, la mayor parte de las lesiones están en el tronco, incluyendo la espalda, y las extremidades, suelen ser zonas más susceptibles de quemarse por tener una mayor exposición al sol.

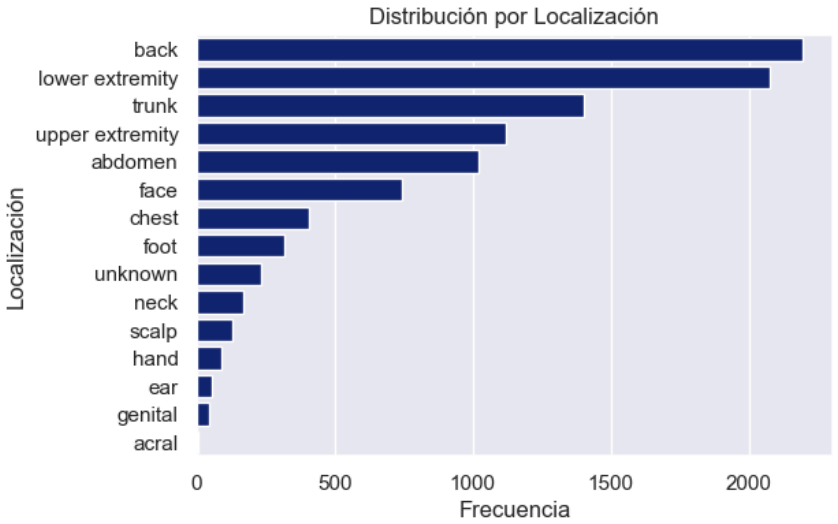

Figura 13: Distribución de la localización de las lesiones en los pacientes.

9.5. Preprocesamiento de los datos

Para empezar el desarrollo, debemos preparar los datos en el comienzo del pipeline, y adaptarlos a la estructura que vamos a necesitar, dependiendo del ***framework*** con el que estemos trabajando. En nuestro caso lo primero que vamos a hacer es importar las diferentes librerías que son utilizadas durante el desarrollo la práctica, además tenemos que declarar las rutas a la carpeta de las imágenes y al csv[35] que contiene la información relevante al tipo de lesión de cada imagen. Posteriormente hemos de importar el csv como ***dataframe***, que es una estructura de datos existente en ***python*** semejante a un csv, esta estructura es de fácil manipulación y las librerías de ***TensorFlow*** están diseñadas para trabajar con ella.

El csv está estructurado con 8 columnas, la primera corresponde con el nombre de la imagen y las siguientes representan cada una de las clases que hay que predecir, de forma que en cada fila podemos encontrar el nombre de una imagen jpg y la relación binaria con cada clase, dicho de otro modo, cada imagen se asocia a una clase por el 1 que contiene su fila, lo demás son 0.

Debemos aquí modificar el ***dataframe*** y adecuarlo a la estructura que necesitamos, tenemos que añadir 2 columnas para que posteriormente, el objeto ***datagen***, que será el generador de imágenes que alimentara el modelo pueda funcionar. La primera columna que debemos añadir tiene que contener la ruta a cada una de las imágenes. La segunda fila que debemos añadir se llamará ***class***, y tendrá el papel de indicarle a ***TensorFlow*** la clase de cada imagen, como un dato de tipo ***string***, pues para realizar un modelo de clasificación multiclase el ***framework*** necesita obtener las clases de esta manera, y no con la especificación binaria que contiene el csv. También se tiene que realizar el mismo proceso con los datos que se usarán para el set de ***test***, pues estos están separados de los demás.

[35] Comma-Separated Values

En la figura 13 podemos observar una lesión perteneciente a cada clase de la base de datos, esta figura se obtiene directamente del ***dataframe*** y nos vale de comprobación para saber que está bien implementado. Como se puede observar comparando una pareja de cada lesión, a simple vista se denotan características similares, por ejemplo en los dermatofibromas es muy notable el tejido fibroso, con apariencia de cicatriz. Se puede destacar también la similitud que tienen los carcinomas basocelulares con la clase AKIEC, esto nos puede dar una idea de la complejidad intrínseca que tiene este problema.

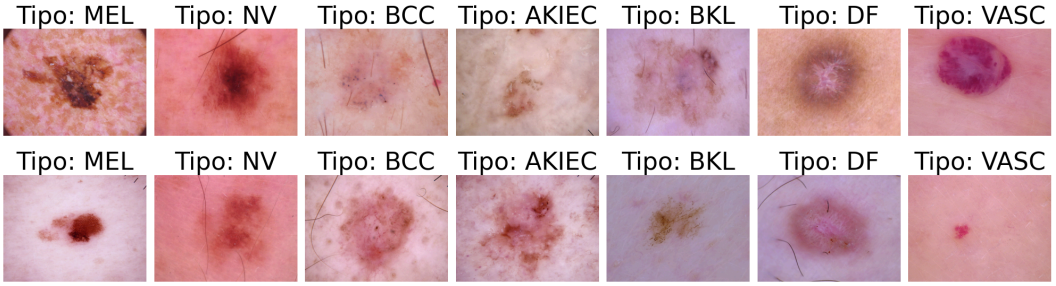

Figura 14: Imágenes de lesiones pertenecientes a cada clase de la base de datos.

Una vez hemos construido estos primeros dataframes, tenemos que dividir el primero en el ***dataframe*** que usará el generador de ***training*** y el ***dataframe*** que usara el generador de ***validation***. Con la función ***train_test_split*** de la biblioteca ***scikit-learn***. dividimos el primer dataframe con el porcentaje que le queramos asignar al set de ***validation***, esta función nos permite dividir los datos de manera aleatoria en dos subconjuntos. El tamaño de los sets de ***training*** y ***validation*** los decidiremos después de realizar una prueba de optimización, en la que se probará con las siguientes proporciones: 80% 20%, 70% 30% y 55% 45%.

Para realizar la prueba de optimización en la división de los ***dataframe*** de datos, se ha utilizado la red ***EfficientNet-b0***. Se han usado los mismos hiperparámetros en los 3 modelos y se han obtenido los resultados que podemos ver en la tabla 2. Como podemos observar claramente la clásica división de 80/20, es la que mejor funciona. Podríamos pensar a la luz de los resultados, en realizar una división 90/10, pero esta queda descartada pues con un set ***validation*** tan pequeño el comportamiento de la función de error se vuelve muy errático.

EXP.	DIVISION DATOS		ACCURACY	BMA[36] (normal)	BMA (superclases)
	TRAINING	VALIDATION			
1	80%	20%	0.7791	0.7502	0.7817
2	70%	30%	0.7711	0.7351	0.7765
3	55%	45%	0.7632	0.7181	0.7719

Tabla 2: Pruebas con distintas proporciones en los sets de *training* y *validation*.

[36] Balanced Multiclass Accuracy

Cuando ya tenemos nuestros set de datos en forma de ***dataframes*** listos y separados, tenemos que tomar una serie de decisiones para construir el generador de datos. Primero vamos a analizar la cantidad de muestras que forman cada clase, con el fin de saber si es un set de datos equilibrado. Como vemos en la figura 14, el set está extremadamente desbalanceado, de la clase NV tenemos 6.705 muestras, y de la segunda clase con más muestras, la clase MEL, solo tenemos 1113. En contraste, la clase DF solo tiene 115 muestras, esto es problemático y tenemos que modificar el set de datos para contrarrestar el desbalance. Podemos realizar dos tratamientos diferentes, el ***subsampling***, que queda descartado por el pequeño tamaño de las clases con menos muestras, o el ***oversampling***, que será la técnica elegida. Como vamos a realizar ***oversampling*** a un conjunto tan desbalanceado, podríamos acabar teniendo problemas de ***overfitting***, pues la red se entrenaria demasiadas veces con las mismas imágenes de las clases pequeñas, pero este problema lo resolveremos más adelante aplicando técnicas de aumento de datos complejas.

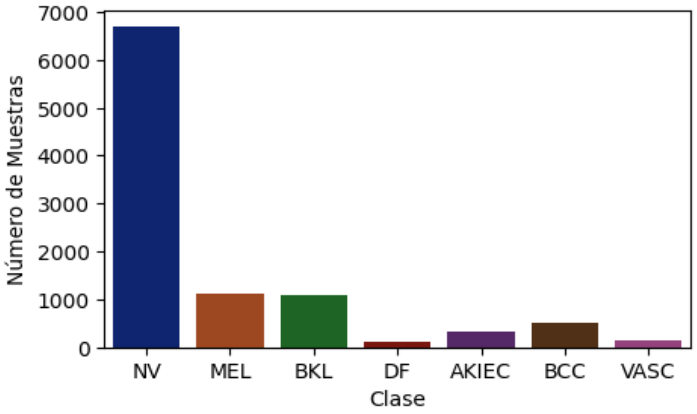

Figura 15:Muestras pertenecientes a cada clase de la base de datos.

9.6. Oversampling y factor de reducción

Para realizar el ***oversampling*** de los datos usaremos la función ***resample*** de la biblioteca ***scikit-learn.*** Implementamos una función que se encarga de contar el número de muestras que tiene una clase y que a través de la función ***resample*** las clone si fuera necesario para alcanzar el target de muestras que necesitamos por clase, en nuestro caso el número de muestras de la clase con más muestras de todo el set de datos. Podría establecerse un target más pequeño si no quisiera hacerse un oversampling total, o si quisiera hacerse un undersampling.

Podemos ver en la figura 15 el ***dataframe*** de ***training*** después de aplicarle el clonado de muestras aleatorio, ahora el set de datos está equilibrado, hemos de realizar este proceso tanto al set de ***training*** como al set de ***validation***. En cambio el set de ***test*** se dejará desbalanceado para ofrecer unas métricas más realistas, ya que la frecuencia de las distintas afecciones en un escenario realista no es uniforme.

Llegados a este punto tenemos que tener en cuenta la cantidad de veces que se han clonado las muestras de las clases más pequeñas, por ejemplo, de la clase DF se han multiplicado las muestras en un factor de casi 56. Esto es mucho y si le enseñaramos todo el set de ***training*** a la red neuronal cuando la entrenemos le estaríamos pasando 56 veces cada imagen de la clase. Nos interesa que todo el conjunto de NV, que no se ha clonado, se le pase a la red pero pasarle una cantidad de veces tan elevada los conjuntos pequeños nos llevará a ***overfitting***. En total, el set de ***training*** ha aumentado de 8012 imágenes a 37485, ha aumentado en un factor de 4,7.

Sabiendo que tendremos problemas con el ***overfitting*** es de suma importancia desarrollar un buen generador de imágenes, que introduzca la variabilidad suficiente como para contrarrestar el problema, de hecho es común entre los modelos generados por participantes de los concursos del ISIC, desarrollar generadores con aumentos de datos medianamente complejos.

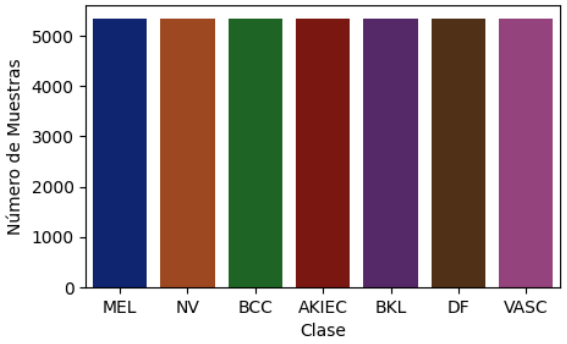

Figura 16: Muestras de cada clase del set de training después de hacer *oversampling*.

Podemos experimentar también acerca del ***target*** óptimo para realizar el ***overfitting***. Observando la tabla 3, podemos ver diferentes pruebas, donde se han aplicado los mismos hiperparámetros a excepción del factor de reducción. Este parámetro es el factor en el que se reducen los ***steps per epoch*** durante el entrenamiento. Aunque normalmente es un hiperparámetro que no se modifica, y se establece de forma que se le enseña todo el set de ***training*** en cada época, en nuestro caso lo vamos a reducir por 2 razones. La primera es la capacidad computacional, el entrenamiento de redes neuronales relativamente grandes se hace muy pesado con los componentes del dispositivo de trabajo. Por lo tanto se ha optado por reducir la carga de trabajo en cada época para la red. Esto podría afectar al rendimiento del modelo en circunstancias normales, pero al estar tan desbalanceado y realizar un ***oversampling*** tan grande no afecta tanto. La segunda razón es precisamente controlar el ***overfitting*** durante el entrenamiento, pues a pesar de usar generadores con complejos aumentos de datos, la cantidad de imágenes repetidas acaba afectando al entrenamiento.

El experimento se ha realizado con la red ***EfficientNet-b0***, sin congelar ninguna capa de la misma, también se ha aplicado el aumento de datos para realizar este experimento. Para medir la eficiencia de los modelos se usa la métrica ya mencionada BMA**,** aplicándola tanto a las clases del set de datos como a las superclases, conjuntos de las primeras, que hemos creado.

Se ha probado con dos ***target*** distintos, primero realizando un ***oversampling*** a 1.200 muestras de cada clase, y luego realizando un ***oversampling*** total, es decir a 6.705 muestras de cada clase. Se puede apreciar que los modelos entrenados con todo el dataset son más estables y que el modelo entrenado con un ***oversampling*** de 1200 muestras requiere un factor de reducción concreto para funcionar.

Además se puede apreciar cómo al realizar un ***oversampling*** con un ***target*** bajo, tenemos mucha sensibilidad a la hora de ajustar los ***steps per epoch*** según el factor de reducción, esto se debe a que hay muchas menos imágenes repetidas en cada generador. Si reducimos mucho los ***steps per epoch***, en el caso de realizar un ***oversampling*** así de bajo, estamos capando la capacidad de generalización de la red neuronal y podemos observar claramente una reducción de la eficiencia del modelo con un factor de reducción de más de 4.

La reducción de la eficiencia del modelo no se da así en caso de realizar un ***oversampling*** total, pues al tener tantas imágenes de las clases pequeñas repetidas, el hecho de aplicar un factor de reducción ayuda a reducir el ***overfitting***. Esto pasa porque pasamos al modelo más imágenes de las clases pequeñas, pero dándole todavía una cantidad muy considerable, a la vez que aprovechamos todo el conjunto de imágenes de las clases grandes.

En conclusión, decidimos optar por un ***oversampling*** total, por estabilidad a la hora de obtener buenas métricas y tener más control del ***overfitting***. Aunque viendo los resultados se observa que el experimento 3 ha obtenido mejor puntuación, si tenemos en cuenta la métrica BMA para las superclases, el modelo del experimento 7 supera al modelo del experimento 3. Además de optar por realizar un ***oversampling*** total se ha decidido establecer como factor de reducción para el entrenamiento de todos los modelos en 4, por ser el que mejor rendimiento arroja.

EXP.	FACTOR REDUCCIÓN	Oversampling Target	BMA (normal)	BMA (superclases)	Overfitting
1	5	1200	0.6338	0.6338	0.10
2	4	1200	0.7062	0.7597	0.10
3	2	1200	0.7539	0.7520	0.00
4	1	1200	0.7175	07587	0.06
5	7	TOTAL	0.7347	0.7461	0.01
6	5	TOTAL	0.7347	0.7462	0.05
7	**4**	**TOTAL**	**0.7502**	**0.7817**	**0.10**
8	2	TOTAL	0.7355	0.7930	0.13

Tabla 3: Pruebas con distintos targets de oversampling.

9.7. Data augmentation

Durante el desarrollo de los modelos se han probado diferentes formas de aumentar los datos, cambiando el número y tipo de transformaciones. Se ha podido concluir que la complejidad intrínseca del problema, es relativamente elevada, por lo que la complejidad de la red usada también ha de serlo.

Para la elección de las redes que se han usado en la práctica, se han hecho una serie de pruebas, en las cuales se fue aumentando la complejidad de los modelos pre entrenados con imagenet. Según los resultados que se fueron obteniendo alcanzamos la complejidad necesaria para generalizar con aproximadamente 3 millones de parámetros. Aunque con esta cantidad de parámetros los modelos ya generalizaron, no arrojaban las mejores métricas de precisión, por lo que se ha necesitado subir la complejidad hasta los 4 millones de parámetros, como por ejemplo con la ***EfficientNet-b0***.

Al estar usando modelos pre entrenados con tal grado de complejidad, el set de datos se quedaba corto, y obtenemos un fuerte overfitting con pocas épocas, lo que nos ha llevado a la necesidad de aplicar técnicas de data augmentation más complejas. En la figura 15 podemos ver la comparativa de dos entrenamientos de la ***EfficientNet-b0***, en el de la izquierda se ha usado un aumento de datos relativamente simple, usando las propias opciones de los generadores. En esta versión simple tenemos, de forma resumida las siguientes técnicas de aumento:

- Rotación de la imagen.
- Giro horizontal.
- Giro vertical.
- Ajuste del brillo en un rango parametrizado.
- Zoom aleatorio en cierto rango.
- Cizallamiento de la imagen en cierto rango.

Como podemos observar el entrenamiento con este aumento de datos produce bastante ***overfitting***. Habiendo una diferencia de 0.2 entre ***training accuracy*** y ***validation accuracy***. En cambio, los resultados aplicando el aumento de datos complejo son mucho mejores y se ve claramente cómo se reduce el ***overfitting***. Viendo esta comparación podemos concluir que usar un aumento complejo arroja resultados mucho mejores en nuestro caso, por lo tanto será el que se usará en el desarrollo de las evaluaciones.

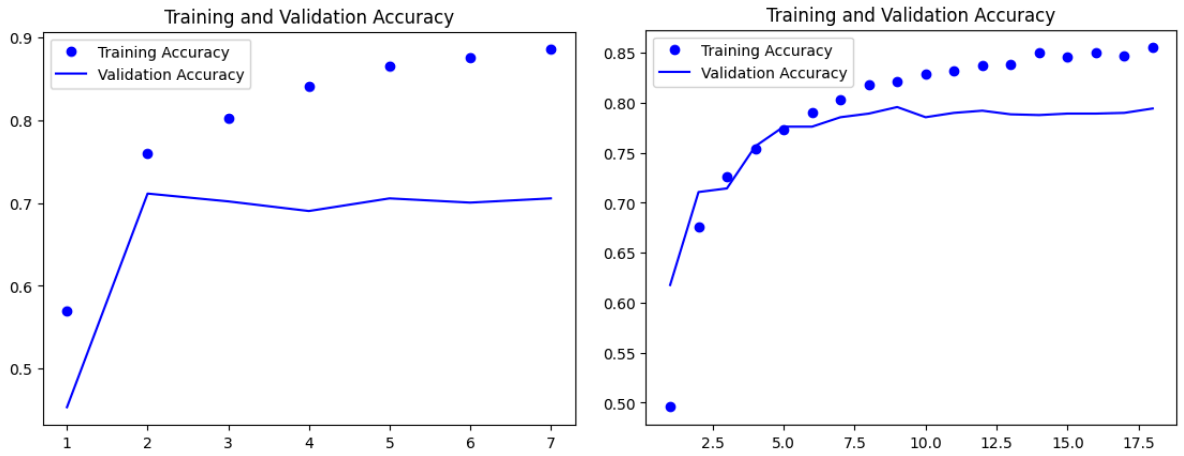

Figura 17: Comparativa con aumento de datos simple (izquierda) y complejo (derecha)

Para el aumento de datos complejo se ha usado ***albumentations*** de ***PyTorch***, por la versatilidad de transformaciones y configuraciones de las mismas que nos ofrece. El primer aumento de datos que se construyó se hizo directamente con el generador de imágenes de ***TensorFlow,*** pero este ofrece menos opciones, por eso se decidió cambiar de ***framework***.

Hay que destacar que a cada transformación se le configuran unos parámetros, entre ellos la probabilidad de que la transformación en concreto se aplique a cada imagen que procesa el generador.

A continuación redacto todas las transformaciones que aplica el aumento de datos complejo, en caso de no especificar la probabilidad de aplicar la transformación, esta será del 50%:

- ***Transpose***: Realiza una transposición de la imagen.
- ***VerticalFlip***: Aplica un volteo vertical a la imagen.
- ***HorizontalFlip***: Aplica un volteo horizontal a la imagen.
- ***RandomBrightnessContrast***: Modifica aleatoriamente el brillo y el contraste entre los valores parametrizados con una probabilidad del 75%.
- ***OneOf***: Aplica una de las siguientes transformaciones con una probabilidad del 70%.
    - ***MotionBlur***: Aplica desenfoque de movimiento a la imagen.
    - ***MedianBlur***: Desenfoca la imagen usando el filtro mediano.
    - ***GaussianBlur***: Aplica el desenfoque gaussiano.
    - ***GaussNoise***: Añade ruido gaussiano.
- ***OneOf:*** Aplica una de las siguientes transformaciones con una probabilidad del 70%.
    - ***OpticalDistortion***: Aplica distorsión óptica.
    - ***GridDistortion***: Aplica distorsión en cuadrícula.
    - ***ElasticTransform***: Aplica transformación elástica.
- ***CLAHE***: Aplica ecualización adaptativa del histograma con probabilidad del 70%.
- ***HueSaturationValue***: Ajusta el tono, la saturación entre los valores parametrizados.
- ***ShiftScaleRotate***: Aplica desplazamiento, escala y rotación entre los valores parametrizados con una probabilidad del 85%
- ***Resize(224, 224)***: Redimensiona la imagen a 224x224 píxeles, que es el tamaño que usaremos como ***input*** en nuestros modelos.
- ***ToTensorV2():*** Transforma la imagen en un tensor, que es el formato necesario para ser utilizada como ***input*** de los modelos de ***deep learning***.

Para la elección de las transformaciones que aplicó en el generador, me he inspirado en los mejores ***dataminers*** que participaron en los diferentes challenges del ISIC, usando las transformaciones que los mejores solían implementar.[57]

9.8. Selección de hiperparámetros

Una vez se ha completado el proceso anteriormente descrito a la base de datos y se han creado los generadores de datos, es el momento de establecer los hiperparámetros que vamos a usar para entrenar nuestro modelo.

Nos centraremos en el tamaño de los lotes, la tasa de aprendizaje y el número de épocas. Estos parámetros son los que normalmente se van probando hasta dar con una combinación óptima para la eficiencia del modelo. En nuestro caso, en vez de ir probando combinaciones vamos a hacer uso de diferentes bibliotecas de ***TensorFlow*** para evitar tener que realizar un montón de pruebas combinando los diferentes parámetros.

9.8.1. Tamaño de batch y learning rate

El ***batch*** o tamaño del lote, es el número de muestras que utiliza la red para actualizar los pesos en cada iteración. Se trata de un parámetro crítico y depende de varios factores. Para empezar, es un parámetro que se suele poner como potencia de 2 por convención. Tenemos que tener en cuenta el límite de la memoria, normalmente de la GPU y en nuestro caso de la memoria RAM. En nuestro caso, con un ***batch size*** de 128 o más el sistema tenía que hacer uso de la memoria de los SSD[37] para apoyarse y esto ralentiza el entrenamientos de forma desmedida. Además de la capacidad computacional tenemos que hacer hincapié en las diferencias de tener un ***batch*** más grande o pequeño y su relación con el ***learning rate***.

Con un tamaño de ***batch*** mayor, la red generaliza más, pues tiene más ejemplos para cada actualización que con un ***batch*** pequeño. De hecho si usamos un ***batch*** muy pequeño, la función de pérdida actuará de manera errática, pues actualiza los pesos muchas veces, basándose en pocos ejemplos, lo que cambia demasiado la arquitectura de la red y hace que no generalize. Esto sería así si no fuera por el ***learning rate***, el parámetro que le indica a la red en qué medida puede modificar los pesos en cada iteración. Por lo tanto, podremos usar un ***batch*** de relativa comodidad para nuestra capacidad computacional si optimizamos bien el parámetro ***learning rate***, esto a costa de que aumente el número de épocas, pues tardará más en entrenar.

Al igual que con el número de épocas, podemos establecer este parámetro de forma dinámica, y reducir el número de pruebas necesarias. Con la función ***ReduceLROnPlateau***, podemos ajustar la tasa de aprendizaje dinámicamente. Esta función monitoriza una métrica, en nuestro caso la pérdida del set ***validation*** y si esta no mejora después de un número determinado de épocas reduce el ***learning rate*** en un factor parametrizado. Concretamente, el ***callback*** se implementa con la siguiente línea de código:

```
ReduceLROnPlateau(monitor='val_loss', factor=0.2,  patience=2, min_lr=0.000001, min_delta=0.0001)
```

Podemos ver como se ha declarado la métrica ***val_loss*** como la monitoreada, un factor de reducción de 0.2, lo que quiere decir que la tasa de aprendizaje se reduce un 80% cuando se actualiza. También declaramos la paciencia en 2 épocas, que será lo esperaremos sin mejoras antes de reducir la tasa de aprendizaje. Por último, tenemos la tasa mínima de aprendizaje como 0.000001, y el delta mínimo como 0.0001. Este último parámetro hace referencia al cambio mínimo que debe producirse en la variable monitoreada para que se considere una mejora.

[37] Solid State Disk

En la figura 16 podemos ver la comparación de 2 entrenamientos, el de la izquierda con tamaño de lote 32 y el de la derecha con un tamaño de lote 16. Como se observa la diferencia en la eficiencia de los modelos es inexistente, esto gracias al sistema que hemos implementado con los ***callbacks***. Se puede apreciar un poco más de variabilidad en las métricas del modelo entrenado con un lote de tamaño 16. Como también busco realizar entrenamientos eficientes se usará un tamaño de ***batch*** de 32, pues aprovecha mejor el procesamiento en paralelo.

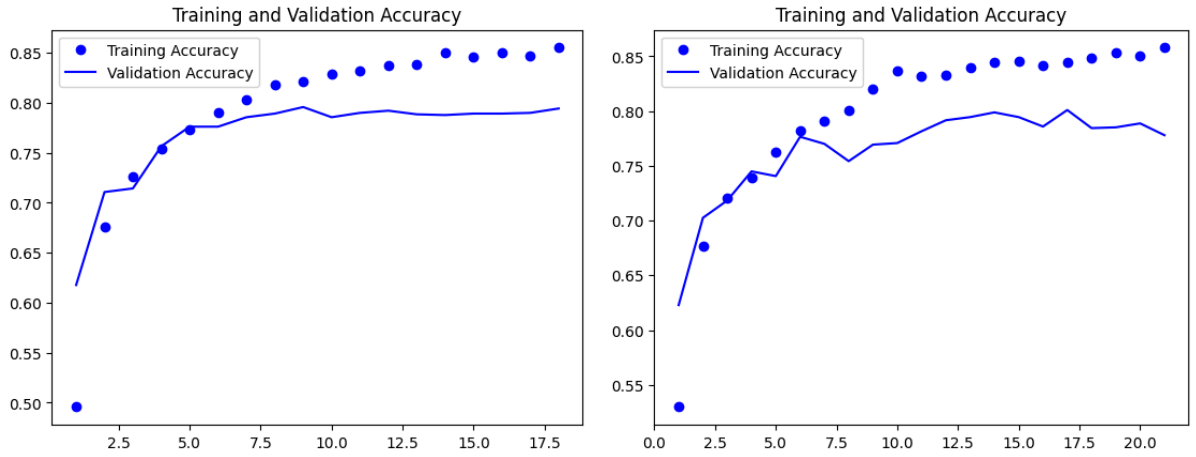

Figura 18: Comparativa de entrenamientos con batch 32 (izquierda) y batch 16 (derecha).

9.8.2. Epoch

Es común establecer el parámetro ***epoch***, es decir las épocas de entrenamiento, después de haber realizado varias pruebas, y establecer este número optimizando la pérdida del set ***validation***. En nuestro caso en vez de basarnos en otras pruebas vamos a utilizar los ***callbacks*** que nos ofrece ***TensorFlow*** durante el entrenamiento para establecer un número dinámico de épocas, este límite se basará en la mejora de las predicciones del set ***validation***.

```
EarlyStopping(patience=4, restore_best_weights=True)
```

Con la función ***EarlyStopping*** de los ***callbacks***, establecemos la paciencia de mejora en 4 épocas, por lo que si tras 4 épocas las métricas del set de ***validation*** no mejoran el entrenamiento acabará. Además esta función nos permite restaurar en el modelo los pesos de la época con mejores resultados con el parámetro ***restore_best_weights***.

Esto nos permite sacar de la ecuación el parámetro epoch, ya que no tenemos que optimizarlo para el entrenamiento y esto nos facilita en gran medida la realización de múltiples pruebas. Además usaremos otra función de los ***callbacks*** llamada ***ModelCheckpoint***, esta función guardará en el dispositivo los pesos del modelo entrenado de manera automática al terminar el entrenamiento.

10. RESULTADOS

10.1. EfficientNet

En primer lugar analizamos los resultados con la red ***EfficientNet***, concretamente se ha probado con los modelos b0 y b1, que por potencia son los que mejor se adaptan a la complejidad del problema.

En los experimentos se ha probado lo siguiente: el modelo b0 entrenado por completo, el modelo b0 añadiendo una serie de capas densas al final del mismo, el modelo b1 entrenado por completo y el modelo b1 donde se han congelado los pesos del 60% de la red.

EXP.	MODELO	ACCURACY	BMA (normal)	BMA (superclases)
1	***EfficientNet-b0***	0.7791	0.7502	0.7817
2	***EfficientNet-b0 + Dense Layers***	0.7672	0.7062	0.7717
3	***EfficientNet-b1***	0.7731	0.72445	0.7815
4	***EfficientNet-b1 (40%)***	0.6799	0.6278	0.6729

Tabla 4: Resultados EfficientNet.

Se puede observar como el mejor resultado se ha obtenido con el modelo del primer experimento, entrenando por completo la red ***EfficientNet-b0***. En la figura 16 podemos ver las matrices de confusión del modelo con el mejor resultado.

Como se puede observar, y teniendo en cuenta los resultados de los demás experimentos, la clase que más cuesta predecir a los modelos es AKIEC, en este caso con un ***recall*** del 67%. La sensibilidad de las diferentes superclases está bastante equilibrada, y si tenemos en cuenta los resultados del concurso ISIC 2018 este modelo entraría en el TOP 15. Evaluando lo conseguido mediante esta arquitectura se puede concluir que el objetivo ha quedado satisfecho.

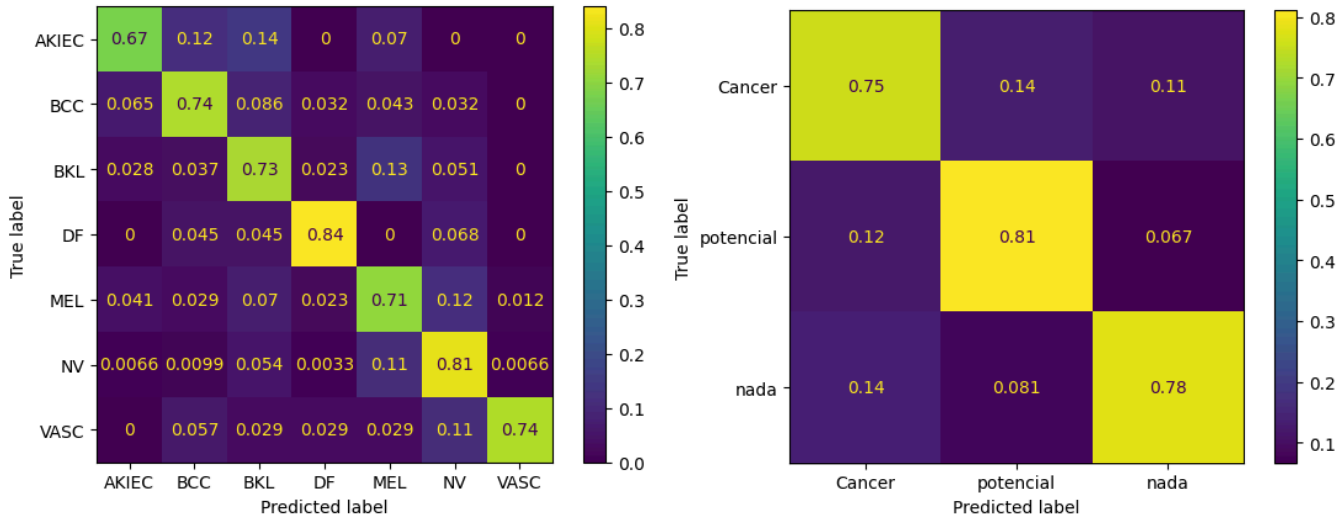

Figura 19: Matrices de confusión de la red EfficientNet

10.2. DenseNet

Para evaluar la arquitectura ***DenseNet***, se ha usado el modelo ***DenseNet-121***, el cual cuenta con 121 capas densas, donde cada capa se conecta a todas las posteriores, es por esto que el número de parámetros del modelo es bastante grande, de en torno a 7 millones.

En los experimentos se ha probado lo siguiente: el modelo entrenado por completo, el modelo donde se ha congelado los pesos del 30% de la red, que corresponde a las 200 primeras capas, y por último el modelo entrenado completo y aplicando regularización L1.

Con este modelo se ha aplicado la técnica de regularización porque al elevar el número de parámetros aumenta considerablemente el overfitting en comparación a la arquitectura anterior, que solo consta de 4 millones. Esto añade una penalización al modelo basado en la magnitud absoluta de los coeficientes de los parámetros.

EXP.	MODELO	ACCURACY	BMA (normal)	BMA (superclases)
1	***DenseNet-121***	0.7745	0.7148	0.7789
3	***DenseNet-121 (70%)***	0.7434	0.6838	0.7583
4	***DenseNet-121 + Regularización L1***	0.7751	0.7136	0.7813

Tabla 5: Resultados DenseNet.

Se puede observar como el mejor resultado se ha obtenido entrenando el modelo completo y aplicando la regularización L1. En la figura 17 podemos ver las matrices de confusión del modelo con el mejor resultado.

Al igual que con la arquitectura ***EfficientNet***, la clase que peor predice el modelo sigue siendo AKIEC, en este caso con un ***recall*** del 58%, peor que con la primera arquitectura. Sin embargo, han mejorado los resultados de las superclases, de hecho tenemos más ***recall*** en las clases que necesitamos sensibles que con la arquitectura anterior. En conjunto los resultados son parecidos a los de ***EfficientNet*** e igualmente cumplen las expectativas.

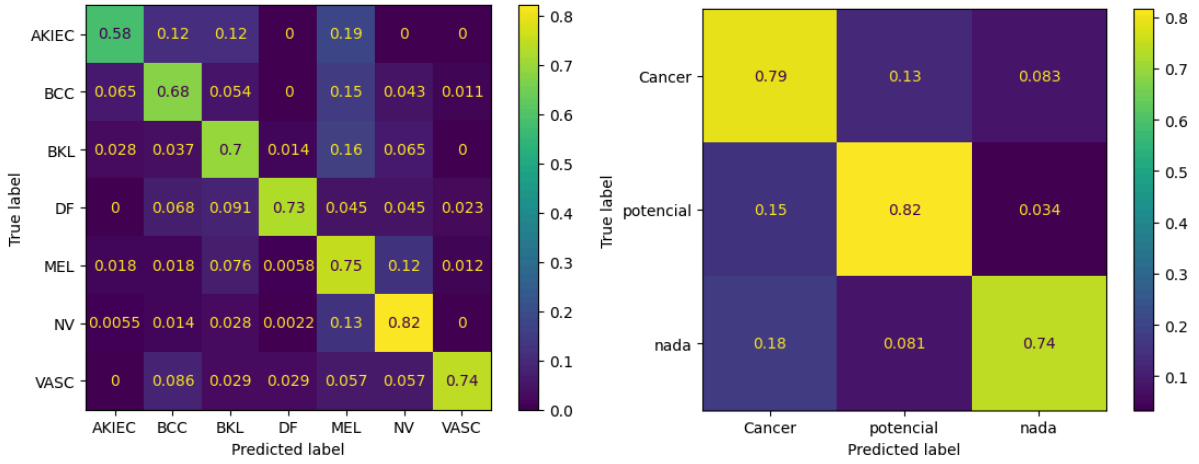

Figura 20: Matrices de confusión de la red DenseNet

10.3. NASNet

Para evaluar la arquitectura ***NASNet***, se ha usado el modelo ***NASNet Mobile***, que se trata de la versión optimizada de esta arquitectura con 4 millones de parámetros, diseñada para trabajar con poca capacidad computacional, en comparación con sus versiones grandes ***NASNet Medium*** o ***NASNet Large***.

En los experimentos se ha probado lo siguiente: el modelo entrenado por completo, el modelo donde se ha congelado los pesos del 15% de la red, que corresponde a las 400 primeras capas, y por último el modelo entrenado completo y aplicando regularización L1.

En este caso también se ha probado a aplicar la técnica de regularización L1, porque a pesar de tener la misma cantidad de parámetros que ***EfficientNet*** se denota una diferencia sustancial entre la ***accuracy*** y la ***val_accuracy*** desde las primeras épocas.

EXP.	MODELO	ACCURACY	BMA (normal)	BMA (superclases)
1	***NASNet Mobile***	0.7573	0.6991	0.7359
3	***NASNet Mobile (85%)***	0.7123	0.6672	0.7140
4	***NASNet Mobile + Regularización L1***	0.7731	0.6586	0.7390

Tabla 6: Resultados NASNet.

Se puede observar como el mejor resultado se ha obtenido entrenando el modelo completo, sin ninguna de las modificaciones, es curioso que a pesar de mostrar un ***overfitting*** desde el principio del entrenamiento la regularización L1 no consigue mejorar los resultados.

A diferencia de las anteriores arquitecturas, en este caso las métricas de las clases han cambiado bastante, por ejemplo la clase AKIEC aquí es la tercera con mejor recall. En el resto de clases salvo en NV se han obtenido peores métricas, lo que ha llevado a empeorar la métrica BMA. Las métricas de las superclases también arrojan peores resultados que en las arquitecturas anteriores, en conclusión esta arquitectura funciona un poco peor.

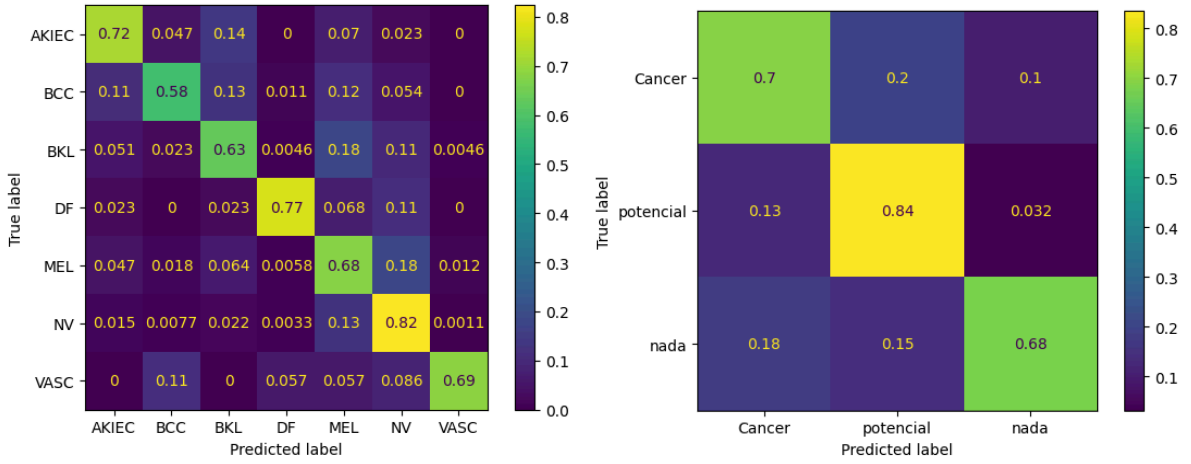

Figura 21: Matrices de confusión de la red NASNet

10.4. ENSAMBLADO

Para acabar las evaluaciones se ha realizado una prueba con la técnica de ensamblado, la cual es muy usada en los challenges de ISIC para mejorar las métricas. Para esto se ha optado por usar el mejor modelo de cada arquitectura y promediar sus predicciones. De esta forma, podemos ver una mejora en las métricas que se observan en la tabla .

EXP.	MODELO	ACCURACY	BMA (normal)	BMA (superclases)
1	***Ensemble learning***	0.8069	0.7645	0.8053

Tabla 7: Resultados *Ensemble*.

La mejora respecto a las arquitecturas por separado es sustancial, teniendo en cuenta el objetivo al que queremos llegar, si observamos la tabla de clasificación del ***challenge*** de 2018, podemos ver como este ensamblado entraría en el TOP 10 si no tenemos en cuenta los modelos que usan datos externos, lo cual es un muy buen resultado usando modelos que usan poca capacidad computacional.

Se puede observar en las matrices de confusión como se han equilibrado bastante las diferentes clases, aunque sigue siendo la clase AKIEC la que peor funciona en el modelo.

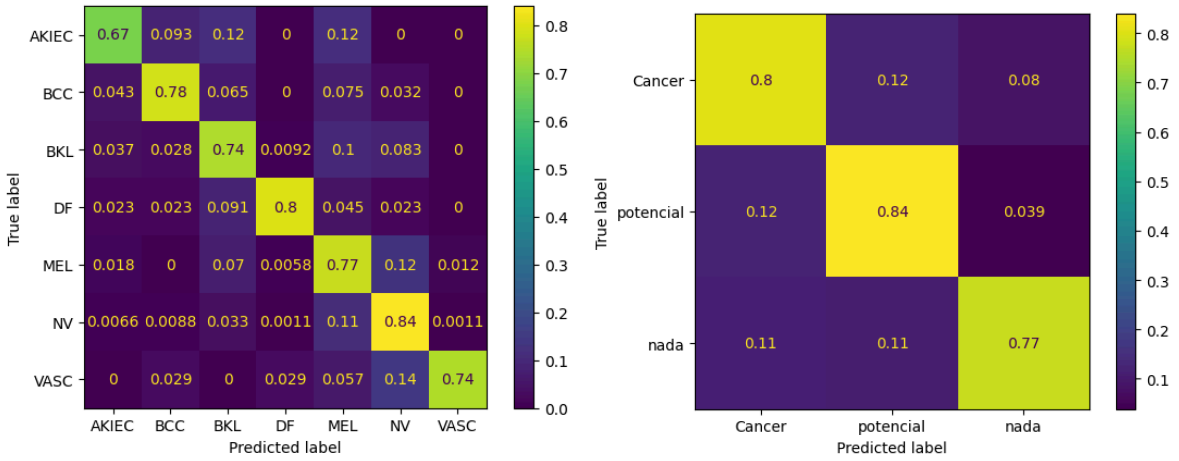

Figura 22: Matrices de confusión del ensamblado

11. CONCLUSIONES Y LÍNEAS DE MEJORA

Para concluir el proyecto de desarrollo, se puede decir que se han alcanzado los objetivos propuestos al principio del mismo. Si nos basamos en los resultados obtenidos, se han satisfecho las previsiones, pues se han conseguido métricas muy competentes comparándolas con el challenge del 2018 en el ISIC.

Si nos basamos en los resultados de las arquitecturas de forma individual:

- ***EfficientNet:*** Es la arquitectura con la que se han conseguido los mejores resultados, y los más estables, esto puede ser debido a que es el modelo con el que se han ido ajustando todos los hiperparámetros del pipeline.

- ***DenseNet:*** Con esta arquitectura también se han conseguido resultados muy convincentes aunque su gran tamaño ha penalizado tanto computacionalmente como con el overfitting. Esto se ha arreglado congelando capas y con la regularización L1, aun así no supera las métricas de ***EfficientNet.***

- ***NASNet:*** Esta arquitectura ha sido la que peores resultados ha arrojado en comparación. Sin embargo las clases con buenas y malas métricas no son similares a las anteriores, por lo cual es de gran utilidad a la hora de realizar el ensamblado, ya que predice mejor alguna clase que funcionaba mal en ***EfficientNet*** y ***DenseNet***.

En líneas generales, en cuanto al desarrollo teórico, ha sido de gran utilidad para contextualizar y entender mejor las bases del Machine Learning. El aprendizaje teórico que me ha aportado el desarrollo, desde la idea más primitiva de la neurona artificial hasta las arquitecturas más complejas y profundas, ha sido de gran ayuda en la comprensión de la práctica.

Los modelos conseguidos son perfectos para aplicarlos a un sistema web o dispositivo, pero si quisiéramos mejorar los resultados, se podría estudiar como línea de mejora ampliar la base de datos con otros conjuntos. Esta mejora permitiría aumentar la complejidad de los modelos usados, aunque también se necesitaría aumentar la capacidad computacional para optimizar los procesos de entrenamiento.

Como segunda línea de mejora, y quizás la más práctica si no queremos crear modelos muy pesados, sería realizar todo el proceso de optimización de hiperparametros para cada una de las arquitecturas probadas. Esto podría mejorar un poco las métricas de dichos modelos, aunque sí una meta optimista puede estar en torno a una puntuación de 0.8 BMA, ya nos encontramos bastante cerca.

Finalmente, quiero destacar la importancia de aplicar las técnicas y tecnologías desarrolladas durante este proyecto en un campo como es el de la medicina, donde actualmente, con modelos tan desarrollados como los estudiados, se podría aplicar el Machine Learning en diferentes campos sin mucha capacidad computacional. Tan solo con desarrollar herramientas de cribado o toma de decisiones en el triaje, donde se busca tener una buena sensibilidad, se podría liberar gran carga de trabajo de los profesionales sanitarios.

12. BIBLIOGRAFÍA

[1] «¿Qué es el cáncer? American Cancer Society». Accedido: 15 de febrero de 2024. [En línea]. Disponible en: https://www.cancer.org/es/cancer/entendimiento-del-cancer/que-es-el-cancer.html

[2] «Dimensiones del cáncer | AECC Observatorio». Accedido: 16 de febrero de 2024. [En línea]. Disponible en: https://observatorio.contraelcancer.es/explora/dimensiones-del-cancer

[3] R. L. Siegel, K. D. Miller, N. S. Wagle, y A. Jemal, «Cancer statistics, 2023», *CA. Cancer J. Clin.*, vol. 73, n.º 1, pp. 17-48, 2023, doi: 10.3322/caac.21763.

[4] I. W. G. on the E. of C. R. to Humans, *Solar and Ultraviolet Radiation*. International Agency for Research on Cancer, 1992.

[5] M. F. Cruz Mahecha y M. F. Vargas Martínez, «Detección de melanomas a partir de imágenes dermatoscópicas», dic. 2018, Accedido: 10 de abril de 2024. [En línea]. Disponible en: http://repository.udistrital.edu.co/handle/11349/14814

[6] P. Castañeda Gameros y J. Eljure Téllez, «El cáncer de piel, un problema actual», *Rev. Fac. Med. UNAM*, vol. 59, n.º 2, Art. n.º 2, 2016.

[7] «Cáncer de piel». Accedido: 10 de marzo de 2024. [En línea]. Disponible en: https://www.contraelcancer.es/es/todo-sobre-cancer/tipos-cancer/cancer-piel

[8] «Factores de riesgo para los cánceres de piel de células basales y de células escamosas». Accedido: 10 de abril de 2024. [En línea]. Disponible en: https://www.cancer.org/es/cancer/tipos/cancer-de-piel-de-celulas-basales-y-escamosas/causas-riesgos-prevencion/factores-de-riesgo.html

[9] T. I. A. for R. on Cancer (IARC), «Global Cancer Observatory». Accedido: 10 de abril de 2024. [En línea]. Disponible en: https://gco.iarc.fr/

[10] «Cancer Today». Accedido: 10 de abril de 2024. [En línea]. Disponible en: https://gco.iarc.who.int/today/

[11] A. Tejera-Vaquerizo *et al.*, «Incidencia y mortalidad del cáncer cutáneo en España: revisión sistemática y metaanálisis», *Actas Dermo-Sifiliográficas*, vol. 107, n.º 4, pp. 318-328, may 2016, doi: 10.1016/j.ad.2015.12.008.

[12] M. F. Cruz Mahecha y M. F. Vargas Martínez, «Detección de melanomas a partir de imágenes dermatoscópicas», dic. 2018, Accedido: 10 de abril de 2024. [En línea]. Disponible en: http://repository.udistrital.edu.co/handle/11349/14814

[13] Luis Alonso Romero, «Fundamentos de Sistemas Inteligentes», [En línea]. Disponible en: https://studium23.usal.es/pluginfile.php/300835/mod_resource/content/3/FSI.RNA2022.pdf

[14] Y. Zeng *et al.*, «Brain-inspired and Self-based Artificial Intelligence». arXiv, 28 de febrero de 2024. doi: 10.48550/arXiv.2402.18784.

[15] «FSI.Introduccion2021.pdf». Accedido: 10 de abril de 2024. [En línea]. Disponible en: https://studium23.usal.es/pluginfile.php/300804/mod_resource/content/3/FSI.Introduccion2021.pdf

[16] Z.-H. Zhou, «Open-environment Machine Learning». arXiv, 9 de agosto de 2022. doi: 10.48550/arXiv.2206.00423.

[17] «Archivo:Perceptrón 5 unidades.svg - Wikipedia, la enciclopedia libre». Accedido: 13 de junio de 2024. [En línea]. Disponible en: https://commons.wikimedia.org/wiki/File:Perceptr%C3%B3n_5_unidades.svg

[18] «Función de activación», *Wikipedia, la enciclopedia libre*. 7 de octubre de 2023. Accedido: 10 de abril de 2024. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=Funci%C3%B3n_de_activaci%C3%B3n&oldid=154390554

[19] «Paso a paso: entrenamiento de la red Neuronal YOLOv8 con anylabeling - OpenSistemas». Accedido: 10 de abril de 2024. [En línea]. Disponible en: https://opensistemas.com/paso-a-paso-entrenamiento-de-la-red-neuronal/
[20] J. Leonel, «Backpropagation», Medium. Accedido: 10 de abril de 2024. [En línea]. Disponible en: https://medium.com/@jorgesleonel/backpropagation-cc81e9c772fd
[21] «Algoritmo backpropagation: funciones y aplicaciones», UNIR. Accedido: 10 de abril de 2024. [En línea]. Disponible en: https://www.unir.net/ingenieria/revista/backpropagation/
[22] D. Calvo, «Función de coste - Redes neuronales», Diego Calvo. Accedido: 4 de abril de 2024. [En línea]. Disponible en: https://www.diegocalvo.es/funcion-de-coste-redes-neuronales/
[23] I. P. Borrero y M. E. G. Arias, *DEEP LEARNING*. Servicio de Publicaciones de la Universidad de Huelva, 2021.
[24] «¿Qué es Deep Learning? | IBM». Accedido: 4 de abril de 2024. [En línea]. Disponible en: https://www.ibm.com/es-es/topics/deep-learning
[25] S. Jinnai, N. Yamazaki, Y. Hirano, Y. Sugawara, Y. Ohe, y R. Hamamoto, «The Development of a Skin Cancer Classification System for Pigmented Skin Lesions Using Deep Learning», *Biomolecules*, vol. 10, n.º 8, p. 1123, jul. 2020, doi: 10.3390/biom10081123.
[26] «Perceptrón multicapa», *Wikipedia, la enciclopedia libre*. 29 de septiembre de 2023. Accedido: 10 de abril de 2024. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=Perceptr%C3%B3n_multicapa&oldid=154167162
[27] T. original uploader was G. at S. Wikipedia, *Red Neuronal Artificial de tipo es:Perceptrón simple con n neuronas de entrada, m neuronas en su capa oculta y una neurona de salida.* 2004. Accedido: 13 de junio de 2024. [En línea]. Disponible en: https://commons.wikimedia.org/w/index.php?curid=2175731
[28] S. Santillana Quesada, «Introducción a las redes neuronales para el tratamiento de imágenes», 2022, Accedido: 10 de abril de 2024. [En línea]. Disponible en: https://digibug.ugr.es/handle/10481/76426
[29] C. Bonilla Carrión, «Redes Convolucionales», jun. 2020, Accedido: 10 de abril de 2024. [En línea]. Disponible en: https://idus.us.es/handle/11441/115221
[30] «Red neuronal convolucional», *Wikipedia, la enciclopedia libre*. 11 de marzo de 2024. Accedido: 10 de abril de 2024. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=Red_neuronal_convolucional&oldid=158743241
[31] V. Mirjalili y S. Raschka, *Python Machine Learning*. Marcombo, 2020.
[32] W. Whitney, *English: A diagram of a convolutional layer and a pooling layer applied to an image.* 2014. Accedido: 28 de junio de 2024. [En línea]. Disponible en: https://commons.wikimedia.org/wiki/File:Convolutional_Network_(vector).svg
[33] K. He, X. Zhang, S. Ren, y J. Sun, «Deep Residual Learning for Image Recognition». arXiv, 10 de diciembre de 2015. doi: 10.48550/arXiv.1512.03385.
[34] LunarLullaby, *English: This is a building block (Residual Block) in a deep Residual Network. This block has a residual connection that skips two sequential layers.* 2015. Accedido: 13 de junio de 2024. [En línea]. Disponible en: https://commons.wikimedia.org/wiki/File:ResBlock.png
[35] G. Huang, Z. Liu, L. van der Maaten, y K. Q. Weinberger, «Densely Connected Convolutional Networks». arXiv, 28 de enero de 2018. doi: 10.48550/arXiv.1608.06993.
[36] «Red neuronal recurrente», *Wikipedia, la enciclopedia libre*. 8 de octubre de 2023. Accedido: 10 de abril de 2024. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=Red_neuronal_recurrente&oldid=154429535

[37] L. R. Calcagni, «Redes Generativas Antagónicas y sus aplicaciones», Tesis, Universidad Nacional de La Plata, 2020. Accedido: 10 de abril de 2024. [En línea]. Disponible en: http://sedici.unlp.edu.ar/handle/10915/101507
[38] «Comprensión de la Matriz de Confusión y Cómo Implementarla en Python», DataSource.ai. Accedido: 11 de abril de 2024. [En línea]. Disponible en: https://www.datasource.ai/es/data-science-articles/view-source:https://www.datasource.ai/es/data-science-articles/comprension-de-la-matriz-de-confusion-y-como-implementarla-en-python
[39] «Métricas De Evaluación De Modelos En El Aprendizaje Automático», DataSource.ai. Accedido: 11 de abril de 2024. [En línea]. Disponible en: https://www.datasource.ai/es/data-science-articles/view-source:https://www.datasource.ai/es/data-science-articles/metricas-de-evaluacion-de-modelos-en-el-aprendizaje-automatico
[40] P. I. Dorado Díaz, «Contribuciones de las técnicas machine learning a la cardiología. Predicción de reestenosis tras implante de stent coronario», http://purl.org/dc/dcmitype/Text, Universidad de Salamanca, 2019. Accedido: 13 de junio de 2024. [En línea]. Disponible en: https://produccioncientifica.usal.es/documentos/5e7d3f502999524bdbea60d1
[41] «F-score», *Wikipedia*. 20 de marzo de 2024. Accedido: 11 de abril de 2024. [En línea]. Disponible en: https://en.wikipedia.org/w/index.php?title=F-score&oldid=1214710515
[42] P. Gaspar Peral, «Elaboración y evaluación de modelos predictivos de negocio». Accedido: 11 de abril de 2024. [En línea]. Disponible en: https://oa.upm.es/42909/
[43] MartinThoma, *Deutsch: ROC-Kurve - die Abszisse ist die Falsch-Positiv-Rate und die Ordinate die Richtig-*. 2018. Accedido: 13 de junio de 2024. [En línea]. Disponible en: https://commons.wikimedia.org/wiki/File:Roc-draft-xkcd-style.svg
[44] B. Ozenne, F. Subtil, y D. Maucort-Boulch, «The precision–recall curve overcame the optimism of the receiver operating characteristic curve in rare diseases», *J. Clin. Epidemiol.*, vol. 68, n.º 8, pp. 855-859, ago. 2015, doi: 10.1016/j.jclinepi.2015.02.010.
[45] «Differences between Receiver Operating Characteristic AUC (ROC AUC) and Precision Recall AUC (PR AUC)». Accedido: 11 de abril de 2024. [En línea]. Disponible en: https://www.chioka.in/differences-between-roc-auc-and-pr-auc/?source=post_page-----1489fbd9a527----------
[46] R. KeepCoding, «La división de datos en Deep Learning». Accedido: 11 de abril de 2024. [En línea]. Disponible en: https://keepcoding.io/blog/division-datos-deep-learning/
[47] M. Tan y Q. V. Le, «EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks». arXiv, 11 de septiembre de 2020. Accedido: 10 de abril de 2024. [En línea]. Disponible en: http://arxiv.org/abs/1905.11946
[48] J. M. Rodríguez Corral, J. Civit-Masot, F. Luna-Perejón, I. Díaz-Cano, A. Morgado-Estévez, y M. Domínguez-Morales, «Energy efficiency in edge TPU vs. embedded GPU for computer-aided medical imaging segmentation and classification», *Eng. Appl. Artif. Intell.*, vol. 127, p. 107298, ene. 2024, doi: 10.1016/j.engappai.2023.107298.
[49] «SIIM-ISIC Melanoma Classification | Kaggle». Accedido: 30 de mayo de 2024. [En línea]. Disponible en: https://www.kaggle.com/competitions/siim-isic-melanoma-classification
[50] «ImageNet», *Wikipedia*. 25 de abril de 2024. Accedido: 31 de mayo de 2024. [En línea]. Disponible en: https://en.wikipedia.org/w/index.php?title=ImageNet&oldid=1220649686
[51] Daniel, «¿Qué es el Transfer Learning?», Formación en ciencia de datos | DataScientest.com. Accedido: 30 de mayo de 2024. [En línea]. Disponible en:

https://datascientest.com/es/que-es-el-transfer-learning
[52] N. Codella *et al.*, «Skin Lesion Analysis Toward Melanoma Detection 2018: A Challenge Hosted by the International Skin Imaging Collaboration (ISIC)». arXiv, 29 de marzo de 2019. doi: 10.48550/arXiv.1902.03368.
[53] «The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions | Scientific Data». Accedido: 31 de mayo de 2024. [En línea]. Disponible en: https://www.nature.com/articles/sdata2018161
[54] N. C. F. Codella *et al.*, «Skin Lesion Analysis Toward Melanoma Detection: A Challenge at the 2017 International Symposium on Biomedical Imaging (ISBI), Hosted by the International Skin Imaging Collaboration (ISIC)». arXiv, 8 de enero de 2018. doi: 10.48550/arXiv.1710.05006.
[55] «The ISIC 2020 Challenge Dataset», ISIC 2020 Challenge Dataset. Accedido: 31 de mayo de 2024. [En línea]. Disponible en: https://challenge2020.isic-archive.com/
[56] V. Rotemberg *et al.*, «A patient-centric dataset of images and metadata for identifying melanomas using clinical context», *Sci. Data*, vol. 8, n.º 1, p. 34, ene. 2021, doi: 10.1038/s41597-021-00815-z.
[57] «SIIM-ISIC Melanoma Classification». Accedido: 4 de junio de 2024. [En línea]. Disponible en: https://kaggle.com/competitions/siim-isic-melanoma-classification