eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Grado en Ingeniería Informática

Computación

Trabajo de Fin de Grado

Diagnóstico y seguimiento de enfermedades cutáneas mediante Visión por Computador y técnicas de Minería de Datos

Autor/a

Marcos Iturbe Marenco

Directore/a(s)

Basilio Sierra Araujo

Carmen Hernández Gómez

Octubre 2021

Resumen

En este proyecto se estudia la aplicación de técnicas de minería de datos para el diagnóstico de lesiones pigmentadas, en particular el melanoma. Esta es la lesión dermatológica que más muertes causa a nivel mundial. La tasa de mortalidad es significativamente menor si se hace un diagnóstico precoz, por lo que desarrollar una herramienta accesible públicamente puede evitar muchas de estas muertes, sobre todo en lugares donde tener acceso a un dermatólogo es muy costoso. Se propone la utilización de redes neuronales convolucionales para desarrollar un clasificador capaz de hacer un diagnóstico. Se ha utilizado una novedosa arquitectura conocida como `EfficientNet` que obtiene unos resultados muy prometedores. Para entrenar los modelos se han utilizado las TPUs ofrecidas por la plataforma `Kaggle`. Finalmente y para dar un uso práctico al modelo desarrollado, se ha diseñado, utilizando `Flask`, una aplicación web capaz de predecir el diagnóstico una imagen. Otra de las funciones de la aplicación es permitir a los usuarios aportar imágenes al modelo para mejorarlo.

Agradecimientos

Mis padres me inculcaron el valor del conocimiento desde una temprana edad, por lo tanto les agradezco el poder haber sido capaz de llegar hasta este punto. También quiero agradecerles enormemente a mis tutores el tiempo dedicado y sus recomendaciones.

Un agradecimiento especial para mi hermana Amaya, la persona que me habló del diagnóstico del melanoma.

Dedicado a Amaya,
que tanto me insistió para desarrollar este proyecto.

Índice general

Resumen I

Índice general IV

Lista de Figuras VII

Lista de Tablas X

1. Introducción 1

1.1. Motivación . . . 1

1.2. Planificación . . . 2

1.2.1. Objetivos . . . 2

1.2.2. Tareas . . . 3

1.3. Herramientas utilizadas . . . 4

1.3.1. Python . . . 4

1.3.2. Keras . . . 4

1.3.3. TensorFlow . . . 5

1.3.4. Flask . . . 5

1.3.5. JQuery . . . 6

1.3.6. Bulma . . . 6

1.4. Contenidos de la memoria . . . 6

2. Fundamentos Teóricos **8**

2.1. Marco contextual dermatológico 8

2.2. Minería de datos 10

2.3. Aprendizaje automático 12

2.3.1. Aprendizaje supervisado 13

2.3.2. Aprendizaje no supervisado 14

2.4. Redes Neuronales Artificiales 15

2.4.1. Redes neuronales convolucionales 18

2.4.2. Aumentación de datos 25

2.4.3. Métricas 28

3. Modelos existentes **30**

3.1. *Dataset* 30

3.1.1. *The ISIC 2020 Challenge Dataset* 30

3.2. Análisis de otras técnicas utilizadas 32

3.2.1. Primera técnica a analizar 33

3.2.2. Segunda técnica a analizar 33

3.3. Experimentación 34

3.3.1. Técnica TTA 34

3.3.2. Experimentación de entrenamiento 35

4. Modelo Propuesto **37**

4.1. Arquitectura del modelo 37

4.1.1. Mayor profundidad (d) 37

4.1.2. Mayor anchura (w) 38

4.1.3. Mayor resolución (r) 38

4.1.4. Combinación de escalados 39

4.1.5. Arquitectura base 41
4.2. Aumentaciones utilizadas 41
4.3. Planificación de la tasa de aprendizaje 44

5. Desarrollo de la aplicación **46**
5.1. Motivación 46
5.1.1. Capa del navegador 47
5.1.2. Capa del servidor 47
5.1.3. Capa de persistencia 47
5.2. Diseño de la aplicación 47
5.2.1. Casos de uso 49
5.2.2. Interfaz de usuario 49

6. Resultados obtenidos **53**
6.1. Resultados de la experimentación 53
6.1.1. TTA 53
6.1.2. Número de Épocas 58
6.1.3. Variación de la tasa de aprendizaje 58
6.1.4. Aumentaciones distintas 58
6.2. Resultados de la aplicación 58

7. Conclusiones y Líneas Futuras **60**

Glosario y Acrónimos **61**

Anexos

Bibliografía **63**

Lista de Figuras

1.1. Logo de `Python` 4

1.2. Logo de `Keras` 5

1.3. Logo de `TensorFlow` 5

1.4. Logo de `Flask` 5

1.5. Logo de `jQuery` 6

1.6. Logo de `Bulma` 6

2.1. Disciplinas que contribuyen a la minería de datos. 11

2.2. Diagrama del algoritmo *k*-NN con distintos valores de *k*. 14

2.3. Diagrama del algoritmo *k*-medias. 15

2.4. Izquierda: diagrama de una neurona real. Derecha: diagrama de una neurona artificial. 16

2.5. Ejemplo de red neuronal artificial. 17

2.6. Función sigmoide y función tangente hiperbólica. 18

2.7. Diagrama de una imagen a color pasada a tensor. 19

2.8. Arquitectura de una CNN simple de 5 capas. 21

2.9. Operación convolucional. 22

2.10. Imagen a color junto con los resultados tras haber aplicado distintos *kernel*s. 23

2.11. Imágenes procesadas con distintos *kernel*s. 23

2.12. Resultados tras haber aplicado *kernel*s complejos. 24

2.13. Imagen tras aplicar el filtro Sobel. 24

2.14. Ejemplo de *max-pooling* 26

2.15. Distintos ejemplos de combinaciones de transformaciones aplicadas a imágenes. 27

2.16. Distorsión de rejilla y de transformación elástica. 27

2.17. Representación del área bajo la curva ROC. 28

3.1. Ejemplos de melanomas. 32

3.2. Ejemplos de lesiones pigmentadas benignas. 32

4.1. Diferentes formas de escalar una red. 38

4.2. Tipos de escalado. 39

4.3. Escalado combinando las diferentes dimensiones de una red. 40

4.4. Arquitectura base de EfficientNet. 41

4.5. Imágenes obtenidas tras haber aplicado una matriz de rotación. 42

4.6. Imagen obtenida tras haber aplicado una matriz de transformación de cizalla. 43

4.7. Imagen obtenida tras haber aplicado una matriz de zoom. 43

4.8. Imagen obtenida tras haber aplicado una matriz de traslación. 44

4.9. Visualización de las distintas tasas de aprendizaje. 44

4.10. Visualización de la función de error tras reducirle la tasa de aprendizaje. . 45

4.11. Visualización de la evolución de la tasa de aprendizaje, la función de error y la precisión. 45

5.1. Estructura de la aplicación. 48

5.2. Página principal de la aplicación. 50

5.3. Predicción devuelta por la aplicación. 50

5.4. Formularios de registro e inicio de sesión. 51

5.5. Formulario para subir una imagen. 52

6.1. Captura del navegador que muestra el tiempo de espera de la petición. . . 59

Lista de Tablas

1.1. Tabla de estimación de horas. 4

3.1. Modelos utilizados en la primera técnica. 33

6.1. Resultados sin aplicar TTA. 54

6.2. Añadir la imagen original al hacer la media de las predicciones. 54

6.3. Cambiar el peso de la imagen original para hacer la media. 54

6.4. Cambiar el número de aumentaciones usadas en la TTA. 55

6.5. Cambiar el tipo de aumentaciones en la TTA. 55

6.6. Resumen de experimentación TTA. 56

6.7. Resultado de las *ensembles*. 57

6.8. Resultado con distinto número de épocas. 58

6.9. Resultado con reducción de aprendizaje ante estancamiento. 58

6.10. Resultado con distintas aumentaciones en el entrenamiento. 59

1. CAPÍTULO

Introducción

Este proyecto se propone el reto de utilizar técnicas de minería de datos y de aprendizaje automático para poder diagnosticar enfermedades cutáneas. En particular, el melanoma, una de las enfermedades de la piel más graves.

El melanoma es un tumor maligno de los **melanocitos**, que se presenta más frecuentemente en la piel, aunque también puede producirse en mucosas, retina y meninges. La función de los melanocitos es producir melanina y transferirla a los **queratinocitos** que los rodean. La proliferación excesiva de melanocitos puede dar lugar a **nevus melanocíticos** (conocidos habitualmente como "pecas") si son benignos, o a melanomas si son malignos.

1.1. Motivación

La informática ha ayudado a resolver una gran variedad de problemas en las últimas décadas, muchos de los cuales han sido en el ámbito de la medicina. Desarrollar una herramienta tecnológica en este área puede llegar a salvar millones de vidas. Este proyecto ha sido propuesto por el alumno al conocer las fases de diagnóstico de la enfermedad. Los dermatólogos analizan las lesiones buscando una serie de características. Esta tarea es trasladable a una computadora, que puede llegar a ser más efectiva que un humano al ser capaz de procesar mucha más información. También puede servir de apoyo a los dermatólogos, o incluso remplazarlos en aquellos lugares donde no los haya.

1.2. Planificación

La planificación de un proyecto es algo crucial para conseguir su éxito. A continuación se describe la planificación de este proyecto, definiendo bien los objetivos y las tareas realizadas.

1.2.1. Objetivos

En esta sección se describen brevemente tanto los objetivos generales como los objetivos específicos propuestos antes de iniciar el proyecto.

Objetivos generales

El objetivo principal de este proyecto es proponer un modelo de aprendizaje automático que, dada una imagen de una lesión pigmentada, sea capaz de discriminar si se trata de un melanoma o no.

Objetivos específicos

Para lograr el objetivo principal de este proyecto se han introducido los siguientes objetivos específicos de aprendizaje, de investigación y de desarrollo.

Objetivos de aprendizaje: Entre los objetivos de aprendizaje propuestos al comienzo de este trabajo, podemos destacar los siguientes:

- Hacer un estudio exhaustivo del **aprendizaje profundo o *Deep Learning*** (**DL**) y de las **redes neuronales convolucionales o *Convolutional Neural networks*** (**CNNs**).
- Aprender a manejar librerías de aprendizaje profundo como `Keras` y `TensorFlow`.

Objetivos de investigación: Entre los objetivos de investigación propuestos al comienzo de este trabajo, podemos destacar los siguientes:

- Recopilar los datos necesarios para poder entrenar los modelos.
- Analizar y comparar las técnicas utilizadas a día de hoy para la clasificación de lesiones pigmentadas.

Objetivos de desarrollo: Entre los objetivos de desarrollo propuestos al comienzo de este trabajo, podemos destacar los siguientes:

- Implementar mejoras a los modelos encontrados.
- Diseñar una aplicación escalable, que permita a los usuarios subir una foto y obtener una predicción.

1.2.2. Tareas

En esta sección se describen las tareas realizadas en el proyecto. Para poder visualizarlas de una manera más clara, se han dividido en los siguientes paquetes de trabajo:

- Paquete de búsqueda de datos (BD).
- Paquete de desarrollo del modelo (DM).
  - DM.T1 Aprendizaje: En esta tarea se incluye la búsqueda de información.
  - DM.T2 Análisis: En esta tarea se analizan detalladamente los algoritmos encontrados.
  - DM.T3 Experimentación: En esta tarea se incluye el proceso de experimentación con estos algoritmos.
- Paquete de aplicación (DA).
  - DA.T1 Aprendizaje: En esta tarea se incluye la elección del *framework* y su aprendizaje.
  - DA.T2 Diseño: En esta tarea se incluye el proceso de diseño de la aplicación.
  - DA.T3 Implementación: En esta tarea se incluye el proceso de implementación de la aplicación.
- Paquete de realización de la memoria (M).

En la Tabla 1.1 se presentan la estimación, en horas, de cada una de las tareas y su dedicación final.

TAREAS	HORAS PLANIFICADAS	HORAS REALES
BD	10	5
DM	135	172
DM.T1	55	68
DM.T2	35	42
DM.T3	45	62
DA	115	83
DA.T1	25	15
DA.T2	35	23
DA.T3	55	45
M	40	54
Total	300	314

Tabla 1.1: Tabla de estimación de horas.

1.3. Herramientas utilizadas

En esta sección se mencionan las herramientas más importantes utilizadas en el proyecto.

1.3.1. Python

`Python` es un lenguaje de programación de alto nivel que se utiliza para desarrollar todo tipo de aplicaciones. Es un lenguaje interpretado. Su gran potencial viene dado por la gran cantidad de librerías distintas que tiene. Este proyecto está programado en este lenguaje, tanto para la creación de los modelos, como para la creación de la aplicación web.

Figura 1.1: Logo de `Python`

1.3.2. Keras

Es una librería de código abierto escrita en `Python` cuyo objetivo es acelerar la creación de redes neuronales. Está diseñada para ser una API de alto nivel.

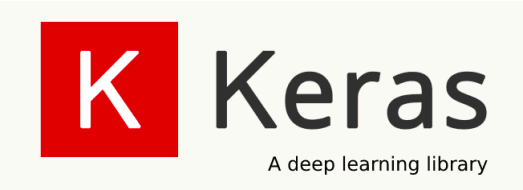

Figura 1.2: Logo de `Keras`

1.3.3. TensorFlow

`TensorFlow` es una librería de código abierto desarrollada por *Google Brain* para aplicaciones de aprendizaje automático y redes neuronales profundas. Incorpora funcionalidades de bajo y alto nivel, entre las que se incluye `Keras`.

Figura 1.3: Logo de `TensorFlow`

1.3.4. Flask

`Flask` es un micro *framework* escrito en `Python` para el desarrollo de aplicaciones web. El término "micro" no significa que solo se puedan hacer proyectos pequeños. Al instalar `Flask` se tiene lo necesario para crear una aplicación web funcional. `Flask` es fácilmente escalable, a través de extensiones, según nuestras necesidades.

Figura 1.4: Logo de `Flask`

1.3.5. JQuery

`JQuery` es una librería de código abierto que simplifica la tarea de programar en `JavaScript`.

Figura 1.5: Logo de `jQuery`

1.3.6. Bulma

`Bulma` es un *framework* de desarrollo gratuito y *open-source* que proporciona componentes web. Con esta herramienta se han modelado las plantillas de las páginas `HTML` de nuestra aplicación.

Figura 1.6: Logo de `Bulma`

1.4. Contenidos de la memoria

En esta sección se presenta la estructura de la memoria del Proyecto Fin de Grado.

En este primer capítulo se describen la planificación y las herramientas utilizadas en el desarrollo de este proyecto. En el capítulo 2, se presentan los fundamentos teóricos necesarios para el buen desarrollo del proyecto. En el capítulo 3, se describen el *dataset* utilizado y los modelos ya existentes, con los que se plantean una serie de experimentos.

En el capítulo 4 se presenta el modelo propuesto tras la experimentación y el análisis de los ya existentes. En el capítulo 5 se describe la aplicación web en la que se incluye el modelo propuesto. En el capítulo 6 se presentan los resultados de la experimentación. Finalmente, en el capítulo 7 se redactan las conclusiones y líneas futuras.

2. CAPÍTULO

Fundamentos Teóricos

2.1. Marco contextual dermatológico

Dentro del ámbito de la dermatología, el melanoma tiene una especial relevancia debido a sus altas tasas de mortalidad y a su creciente incidencia. El melanoma es la forma más grave de cáncer de piel. Según las estimaciones de la *American Cancer Society* para el año 2019 [16], el melanoma fue responsable del 62 % de las muertes por cáncer de piel en ese año. Además, en Estados Unidos, se trata del quinto cáncer más frecuente teniendo en cuenta ambos sexos.

Las tasas de supervivencia de esta enfermedad dependen del estadio del cáncer en el momento del diagnóstico y, por lo tanto, el diagnóstico precoz es crucial.

Los melanomas presentan en su mayoría dos fases de crecimiento: una primera fase de crecimiento radial/horizontal (si se detecta en esta fase, la tasa de supervivencia es del 94 % mediante la extirpación local), seguida de una fase de crecimiento vertical. Estas fases tienen una gran importancia diagnóstica ya que es durante la fase de crecimiento vertical cuando aumenta la probabilidad de producir metástasis, y es el hecho de haber metastatizado o no lo que define un buen o mal pronóstico.

Hoy en día, el diagnóstico inicial para determinar si una lesión pigmentada de la piel es sospechosa de melanoma es clínico y se fundamenta en dos clasificaciones:

1. Clasificación ABCDE.

2. Escala de los 7 puntos de Glasgow.

Este diagnóstico inicial puede complementarse con técnicas que aumenten la especificidad, como son la dermatoscopia o la dermatoscopia digital. En caso de considerarse una lesión sospechosa, está indicada su extirpación.

Los criterios ABCDE que sugieren malignidad son:

1. Asimetría.
2. Bordes irregulares.
3. Coloración heterogénea.
4. Diámetro mayor de 6 mm.
5. Evolución: cambio de tamaño, forma, color, relieve, síntomas...

La escala de Glasgow presenta los siguientes criterios:

- Criterios mayores.
  - Cambio de tamaño / lunar nuevo.
  - Cambio de forma.
  - Cambio de color.
- Criterios menores
  - Diámetro $\geq$ 7 mm.
  - Inflamación.
  - Sangrado.
  - Cambios en la sensibilidad (picor o dolor).

En muchos casos, estos criterios se pueden identificar en una imagen, y es en esta propiedad en la que se fundamenta este trabajo. El algoritmo que se entrena analiza las imágenes al igual que lo hace el ojo clínico, y da con los patrones que sugieren malignidad, permitiendo alcanzar un pronto diagnóstico y reduciendo la incidencia de casos en fase de crecimiento vertical.

2.2. Minería de datos

La "minería de datos" no nace por el desarrollo de nuevas tecnologías sino que nace al surgir nuevas necesidades. En el pasado, los datos almacenados por grandes empresas, organizaciones o instituciones, se consideraban el producto que se generaba a partir de sus sistemas de información; sin embargo, al aparecer la minería de datos, los datos se convirtieron en la materia prima, con la que se podía generar el producto final: el conocimiento. La estadística es la primera ciencia que trata los datos como su materia prima, pero este nuevo tipo de datos, especialmente por la cantidad, la diversidad y la tipología, hacen que las distintas técnicas que conforman la minería de datos sean muy variadas. El conocimiento obtenido debe ser útil para ayudar a tomar decisiones en el ámbito en el que se han extraído dichos datos.

La minería de datos se define como el proceso de extraer conocimiento a partir de un gran número de datos almacenados con distintos tipos de formatos [6]. Es decir, la tarea principal de la minería de datos es obtener modelos comprensibles a partir de los datos. Para que este proceso sea útil, la tarea debe estar automatizada y debe permitir utilizar los patrones descubiertos para tomar decisiones más fiables y por lo tanto, generar un beneficio. Esto significa que la minería de datos se enfrenta a dos grandes retos:

- Ser capaz de gestionar una gran cantidad de datos extraídos de distintos sistemas de información. Esto implica gestionar correctamente los problemas presentes en este tipo de sistemas como por ejemplo el ruido o los datos ausentes, erróneos o volátiles.
- Usar técnicas adecuadas para analizar y obtener información útil y novedosa de los datos previamente gestionados.

En las últimas décadas, el volumen de los datos que se encuentran almacenados en los sistemas de información de las organizaciones ha crecido espectacularmente. Mucha de esa información es histórica; es decir, representa las distintas transacciones o situaciones vividas por la organización. Esta información es muy valiosa ya que conocer el pasado nos ayuda a predecir el futuro. La mayor parte de las decisiones que tomamos son debidas a nuestras experiencias pasadas. Las empresas funcionan del mismo modo, incluso sin hacer uso de la minería de datos, la mayor parte de las decisiones se toman analizando las experiencias pasadas sacadas de informes o de otro tipo de fuentes. En muchos ámbitos, el método tradicional de obtener información a partir de unos datos consiste en el análisis

e interpretación de los mismos de forma manual. Un especialista en la materia analiza los datos y genera un informe o hipótesis que refleja sus tendencias o pautas. Por ejemplo, un grupo de médicos analizan el perfil de los pacientes que sufren diabetes y generan una lista de características comunes a la mayoría de ellos. Al tratar nuevos pacientes con las mismas características, tienen la opción de avisarles sobre su perfil de riesgo y darles unas pautas de prevención. Esta forma manual de actuar se está quedando desfasada por su alto coste y su lentitud. Además, debido al crecimiento exponencial de los datos, es imposible poder analizar la nueva cantidad de información sin una herramienta lo suficientemente potente. Por ello, muchas de las decisiones se toman en base a la intuición del usuario al no poder ser capaz de procesar semejante volumen de datos. Es por eso que nace la minería de datos.

Figura 2.1: Disciplinas que contribuyen a la minería de datos.

La minería de datos es un campo multidisciplinar que abarca muchas tecnologías. De ahí que la investigación y los avances individuales que se producen en estas tecnologías contribuyen a su desarrollo. En la Figura 2.1 podemos observar algunas de las disciplinas más influyentes [7] como:

- Bases de datos: el avance en las grandes bases y almacenes de datos, así como la mejora en las técnicas de indexación, tienen una gran repercusión en el diseño de algoritmos eficientes de minería de datos.
- Recuperación de la información: consiste en recuperar información a partir de datos textuales. Por ejemplo, encontrar documentos a través de palabras clave. Muchas de las técnicas utilizadas en este campo se han usado para casos más generales de minería de datos.

- La estadística: este campo ha proporcionado muchas de las técnicas y algoritmos utilizados en minería de datos. Conceptos como la media, la varianza, el análisis univariante y multivariante, la regresión lineal, la modelización paramétrica o las técnicas bayesianas entre muchos otros, son clave en este campo.
- El aprendizaje automático: es el área dentro de la inteligencia artificial que se encarga de desarrollar algoritmos capaces de aprender a través de ejemplos.
- Los sistemas para la toma de decisión: son herramientas y sistemas de información que ayudan a la toma de decisiones. Herramientas como el análisis ROC o los árboles de decisión, provienen de este campo.
- La visualización de datos: permite al usuario entender los datos de una forma más visual a través de gráficas, iconos, jerarquías...
- La computación paralela y distribuida: estos sistemas permiten distribuir el coste computacional entre varios procesadores. Una de las principales ventajas es la escalabilidad de los algoritmos.
- Otras disciplinas: dependiendo del tipo de datos a ser procesados, también se utilizan otras técnicas como el procesamiento del lenguaje natural, el análisis de imágenes, el procesamiento de señales, los gráficos por computador...

2.3. Aprendizaje automático

El **aprendizaje automático o *Machine Learning*** (**AA**) es la rama de la inteligencia artificial que tiene como objetivo, desarrollar técnicas que permitan a las computadoras aprender [2]. Es un proceso de inducción al conocimiento, ya que trata de crear algoritmos capaces de reconocer patrones y generalizar comportamientos a través de datos de ejemplo.

Hay un gran número de problemas que se pueden abordar usando el **AA**. La principal diferencia entre estos problemas es el tipo de predicción que se quiere obtener. Las más habituales son:

- Regresión: Intentar predecir un valor real. Por ejemplo, intentar predecir el valor de la bolsa mañana a partir del valor de la bolsa de la última década.

- Clasificación: Intentar hacer una clasificación de objetos entre unas clases previamente fijadas. Si solo existen dos clases, se conoce como clasificador binario. Por ejemplo, tomando en cuenta una serie de características, diagnosticar a un paciente.

Cuando se aborda un nuevo problema de **AA**, una de las primeras tareas es enmarcarlo en una de estas clases. De esta forma se puede medir el error cometido entre la realidad y la predicción del algoritmo. Dependiendo del tipo de datos que se utilicen como ejemplo y de la salida que se produzca, hay diferentes tipos de algoritmos en el **AA**.

2.3.1. Aprendizaje supervisado

Los modelos de aprendizaje supervisado son aquellos en los que se sabe de antemano los parámetros a predecir. Aprenden funciones que relacionan las variables de entrada con la salida, y ante nuevas variables donde se desconoce la salida, son capaces de predecirla basándose en estas funciones.

El aprendizaje supervisado está más extendido y tiene más aplicaciones prácticas. Esto se debe a que está muy bien definido su objetivo, lo que permite elaborar métricas sobre el desempeño de los modelos creados. Existe una gran variedad de algoritmos de aprendizaje supervisado. A continuación, se explica el **método de los k vecinos más cercanos o *k-Nearest Neighbour*** (**k-NN**), uno de los más básicos, pero que da una idea de cómo este tipo de algoritmos funcionan.

Algoritmo k-NN

El algoritmo de aprendizaje supervisado **k-NN** se basa en criterios de vecindad. La nueva entrada a predecir debe pertenecer a la misma clase que tienen sus vecinos más cercanos en el conjunto de entrenamiento. Este algoritmo tiene una gran similitud con el comportamiento humano de clasificación. Al encontrarnos con un elemento nuevo desconocido, lo comparamos con las cosas que conocemos y lo clasificamos como el elemento al que más se parezca.

Para hacer este tipo de clasificaciones es necesario tener una manera de medir la similitud entre nuestras entradas. Es por eso que los valores de entrada se consideran como una especie de coordenadas dentro de un espacio vectorial. De esta manera, los elementos más próximos son los más similares; es decir aquellos cuya distancia vectorial sea menor. La versión más simple de este algoritmo es el algoritmo del vecino más cercano. A la nueva

muestra se le asigna la clasificación de la muestra más cercana del conjunto de entrenamiento. A partir de este algoritmo, es lógico pensar que para aprovechar más a fondo el conocimiento del resto del conjunto de entrenamiento, se puede generalizar este algoritmo y obtener el resultado según los k vecinos más cercanos. En la Figura 2.2 podemos ver el valor que se le asignaría a una nueva muestra dependiendo de su posición.

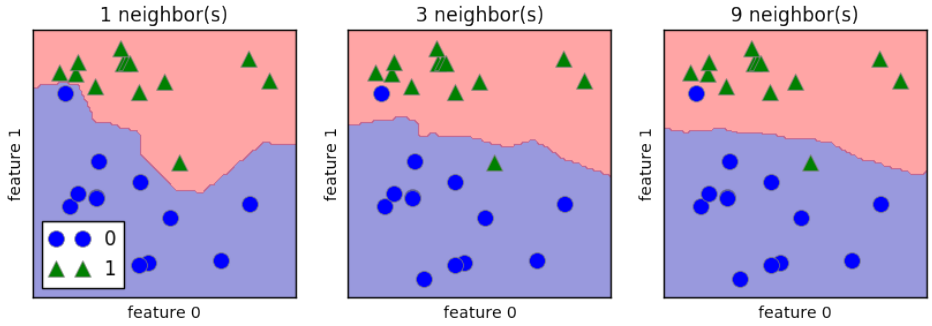

Figura 2.2: Diagrama del algoritmo k-NN con distintos valores de k.

Aplicando varias mejoras y combinándolo con otras técnicas, este algoritmo consigue muy buenas predicciones en muchos ámbitos. Por ejemplo, es capaz de alcanzar una precisión del 97,4 % en el diagnóstico de la diabetes [11].

2.3.2. Aprendizaje no supervisado

En el aprendizaje no supervisado el objetivo ya no es emparejar entradas con salidas, sino ser capaz de agrupar datos. El *clustering* es el proceso de agrupar los datos según su similitud. Este tipo de algoritmos no tienen una aplicación práctica tan directa como los de aprendizaje supervisado porque requieren de más pruebas de ensayo y error. Uno de los algoritmos más sencillos y extendidos de todos es el ***k*-medias o *k-means***.

Algoritmo k-medias

Este algoritmo de aprendizaje no supervisado intenta encontrar k subconjuntos dentro de una muestra. Lo que diferencia a este tipo de algoritmos es que no hay ningún conocimiento previo de cómo hay que agrupar los datos y que no hay ningún criterio que nos diga la bondad de nuestra solución. El algoritmo ***k*-means** utiliza un criterio artificial que permite medir el desempeño del algoritmo. Este criterio es la varianza total del sistema, y el algoritmo ***k*-means** intenta minimizarla:

$$\sum_i \sum_j d(x_j^i, c_i)^2$$

siendo c_i el centroide de la agrupación i-ésima y siendo x_j^i el conjunto de ejemplos clasificados en esa agrupación.

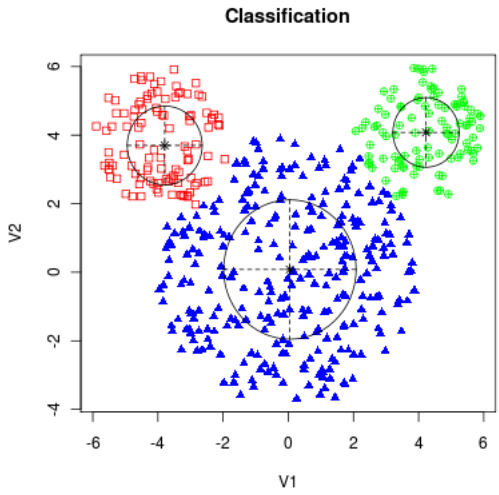

Figura 2.3: Diagrama del algoritmo k-medias.

Los pasos que sigue este algoritmo son relativamente sencillos:

1. Selecciona k puntos al azar del conjunto y les asigna una clase distinta.
2. A cada punto restante le asigna la clase del punto más cercano previamente seleccionado.
3. Se calcula el centroide de los grupos obtenidos.
4. Los centroides pasan a ser los puntos seleccionados.
5. Se vuelve al paso 2 hasta que no hayan más reasignaciones, o hasta que la varianza total no se modifique.

2.4. Redes Neuronales Artificiales

Uno de los campos que ha hecho avanzar tanto en los últimos años a la Inteligencia Artificial es el **AA**, más concretamente las **redes neuronales artificiales o *Artificial Neural***

***networks* (ANNs)**. Las **ANNs** son sistemas conexionistas que dependiendo del tipo de arquitectura que tengan pueden utilizarse en tareas de reconocimiento de patrones, clasificación y visualización. Por lo tanto las **ANNs** son una herramienta más de la minería de datos.

Las **ANNs** son un método de aprendizaje cuya finalidad inicial era emular el comportamiento del cerebro humano.

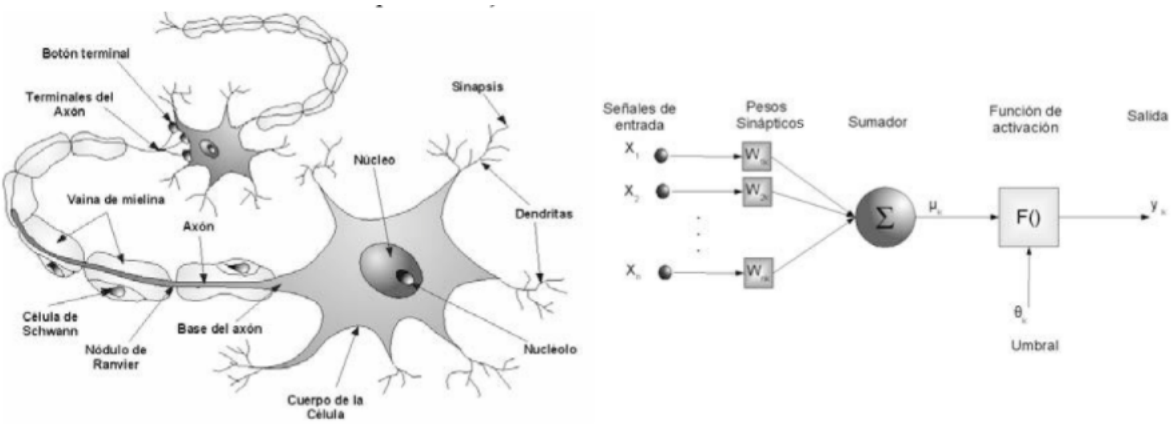

Figura 2.4: Izquierda: diagrama de una neurona real. Derecha: diagrama de una neurona artificial.

Las neuronas del cerebro humano reciben la información a través de la sinapsis de sus dendritas. Cada sinapsis representa la unión de un axón de una neurona con una dendrita de otra. Se produce una transmisión electro-química durante la sinapsis permitiendo así pasar información de una neurona a otra. La información se transmite de una dendrita a otra hasta alcanzar el cuerpo de la célula. Aquí tiene lugar un sumatorio de todos los impulsos recibidos y se aplica una función de activación a este. La neurona se activa si el resultado supera un umbral determinado, y envía una señal a lo largo de su axón para comunicarse con otras neuronas. Esta es la manera en la que la información pasa de una parte de la red a otra. Una neurona simple se muestra en la parte izquierda de la Figura 2.4.

Para modelar estas redes biológicas se utiliza la estructura mostrada en la parte derecha de la Figura 2.4. Este modelo incluye un parámetro adicional denominado *bias* y denotado por θ_k cuya finalidad es aumentar o reducir el umbral de activación de la neurona. Las entradas se representan como un vector de entrada x y el rendimiento de la sinapsis se representa como un vector de pesos w. Por lo tanto el valor de salida de esta neurona viene dado por:

$$y = f(\sum_i w_i x_i) = f(w \cdot x) = f(w^T x)$$

donde f es la función de activación. Cuando tenemos una red de neuronas, las salidas de unas se conectan con las entradas de otras, formando una red como la de la Figura 2.5.

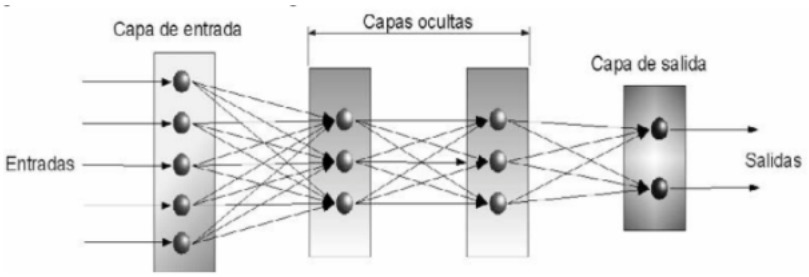

Figura 2.5: Ejemplo de red neuronal artificial.

Si el peso entre dos neuronas conectadas es positivo, el efecto producido es de excitación, en cambio, si es negativo actúa como inhibidor. Podemos observar que una neurona es una unidad de procesamiento muy simple; sin embargo, el gran potencial que tienen las **ANNs** aparece al poner a trabajar muchas neuronas, simples pero robustas, en paralelo.

La capacidad de resolver problemas de **separabilidad lineal** que tienen las **ANNs** se descubrió pronto, pero se tardó bastante tiempo en encontrar un método para construirlas a partir de ejemplos. Este método se llama el algoritmo de retropropagación.

El algoritmo es el siguiente:

1. Inicializar los pesos con valores pequeños y aleatorios.
2. Seleccionar un vector de entrada,x^P a partir de un conjunto de ejemplos de entrenamiento.
3. Propagar la activación obtenida por las funciones de activación a través de la red hasta alcanzar las neuronas de salida.
4. Calcular los valores de δ de las capas de salida.

$$\delta_j^P = (t_j^P - o_j^P) f'(Act_j^P)$$

5. Calcular los valores de δ para la capas ocultas.

$$\delta_i^P = \sum_{j=1}^{N} \delta_j^P w_{ij} f'(Act_i^P)$$

6. Actualizar los pesos: $\Delta_p w_{ji} = \gamma \delta_i^P o_j^P$

7. Repetir el paso 2-6 con todos los ejemplos de entrenamiento.

Este algoritmo es convergente.

En el algoritmo no se especifica la función de activación. Las funciones de activación más comunes son la función sigmoide:

$$\frac{1}{1+exp(-bx)}$$

y la función tangente hiperbólica, $tanh(x)$. En la Figura 2.6 podemos ver su representación gráfica. Ambas funciones son derivables y monótonas. Además tienen la propiedad de que su razón de cambio es mayor para valores intermedios y menor para valores extremos. Se sabe que la función de activación de la tangente hiperbólica alcanza antes la convergencia.

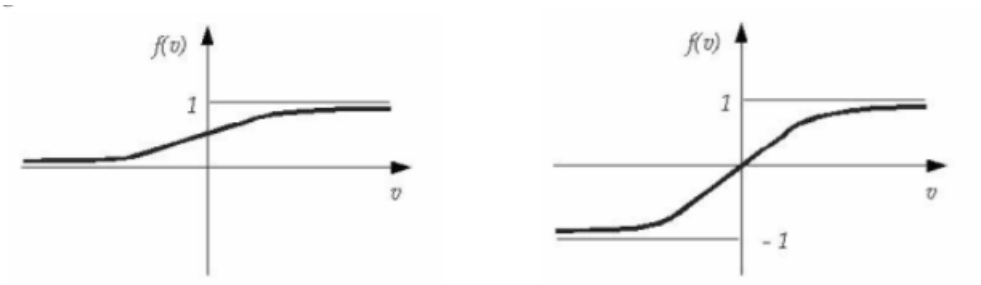

Figura 2.6: Función sigmoide y función tangente hiperbólica.

2.4.1. Redes neuronales convolucionales

Las **redes neuronales convolucionales o *Convolutional Neural networks* (CNNs)** [19] han demostrado obtener unos resultados excelentes en muchos problemas de **AA** y de visión por computador. Se han publicado numerosos artículos sobre este tema y existen gran cantidad de paquetes de software de dominio público. Las **CNNs** tienen innumerables aplicaciones, sobre todo en clasificación y análisis de imágenes.

Para entender correctamente cómo funciona una **CNN**, es necesario tener claros los siguientes conceptos que presentamos a continuación: tensor, operador de convolución, arquitectura de una red convolucional, etc.

Tensor

Un tensor es una forma de representar matrices con un grado mayor que 2. Por ejemplo:

$$x \in R^{H \times W \times D}$$

es un tensor de orden 3 que está formado por los vectores (i, j, d) siendo $0 \leq i \leq H, 0 \leq j \leq W, 0 \leq d \leq D$. Podemos representar escalares como tensores de orden 0 y vectores y matrices como tensores de primer y segundo orden respectivamente.

Una imagen a color con una resolución de $H \times W$ se puede representar como un tensor de tercer orden[1] de tamaño $H \times W \times 3$. Cada canal se compone de una matriz $H \times W$ que contiene el valor del color correspondiente a dicho canal, ya sea el canal rojo, el verde o el azul, de todos los píxeles. Esta transformación se puede ver en la Figura 2.7.

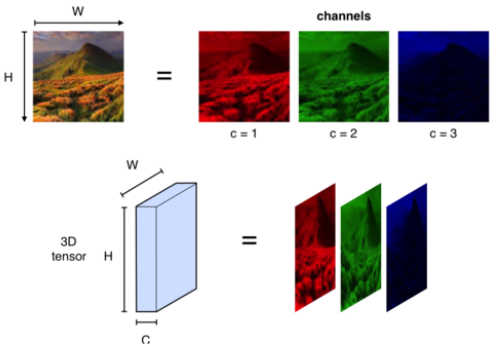

Figura 2.7: Diagrama de una imagen a color pasada a tensor.

Representar imágenes como tensores tiene ciertas ventajas. En muchas aplicaciones de visión por computador, las imágenes a color – que son tensores de tercer orden – se pasan a escala de grises, reduciendo el orden del tensor a dos, y permitiendo de esta manera poder representar las imágenes mediante matrices. Aunque es más sencillo manejar matrices, al realizar esta transformación, se pierde la información del color, la cual puede ser muy importante en el análisis y en la clasificación de las imágenes.

[1] Si una imagen está en formato RGB tiene 3 canales, uno para cada color.

Los tensores son esenciales en las CNNs. La entrada, la representación intermedia y los parámetros de las CNNs son todos tensores. También entran en juego tensores con mayor grado, como por ejemplo los *kernel*s convolucionales.

Operador convolución

El operador convolución es una operación matemática que transforma dos funciones $x()$ y $w()$ en una función $s()$. Imaginemos que tenemos un sensor láser que nos devuelve la posición de una nave $x(t)$ en un determinado instante de tiempo t. La función $x()$ es continua ya que podemos obtener un valor en cualquier instante. Ahora imaginemos que el sensor devuelve unos resultados con interferencias. Para evitar estas interferencias queremos obtener la media de las últimas mediciones donde las mediciones más recientes tienen más peso en dicha media. Se define la función de pesos $w(a)$ siendo a la edad de la medición. Si aplicamos esta función a cada instante de tiempo, se puede obtener una estimación más precisa de la posición de la nave:

$$s(t) = \int x(a)w(t-a)da$$

El operador convolución se suele representar con el símbolo $*$:

$$s(t) = (x * w)(t)$$

En **AA** los valores son discretos. La función $x()$ se conoce como entrada y suele ser un tensor multidimensional, la función $w()$, que se conoce como *kernel*, también es un tensor que la red va adaptando y la función $s()$, que es el resultado, se conoce como **mapa de características o *Feature map*** [4].

Por lo tanto el operador convolución en **AA** viene dado por:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n)$$

Arquitectura de una CNN

Las CNNs se componen de tres tipos de capas [12]: capas convolucionales, capas *pooling* y capas completamente conectadas.

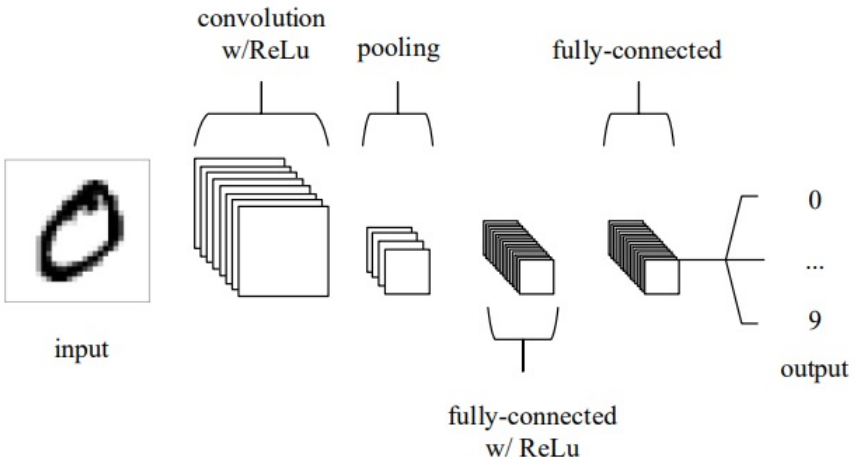

Figura 2.8: Arquitectura de una CNN simple de 5 capas.

Se pueden distinguir cuatro partes claves de la arquitectura, tal y como se muestra en la Figura 2.8.

1. En la entrada se encuentra el tensor que contiene el valor de los píxeles de la imagen.
2. La capa convolucional transforma la entrada para resaltar los aspectos más importantes a la hora de clasificar una imagen.
3. La capa *pooling* realiza un filtrado de los datos obtenidos, conservando los más importantes y reduciendo el número de parámetros significativamente.
4. Las capas totalmente conectadas (*fully-connected*) llevan a cabo las mismas tareas que se realizan en las ANNs convencionales para obtener una clasificación.

Tras este proceso de transformaciones y submuestreos, las CNNs son capaces de dar puntuaciones de clase para problemas de clasificación y regresión.

Capa convolucional

Como se puede intuir a partir del nombre, la capa convolucional es la más importante en las CNNs. Los parámetros que estas capas aprenden se denominan *kernel*s y son tensores cuyas dimensiones normalmente son pequeñas.

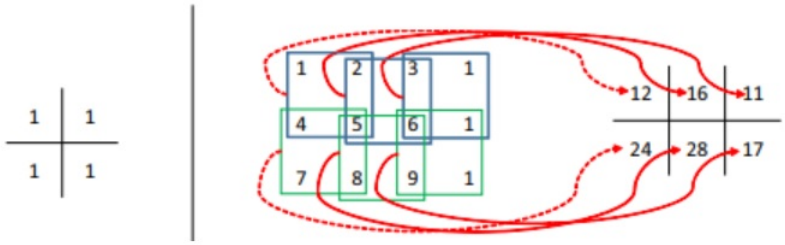

Figura 2.9: Operación convolucional.

En el ejemplo que presentamos a continuación podemos observar cómo una capa convolucional procesa una entrada.

En la Figura 2.9 aparece un tensor de segundo grado, en la parte izquierda, de dimensiones 2×2. En la parte derecha de la figura, tenemos la representación de la entrada y de la salida generada. Se divide la entrada en submatrices del tamaño del *kernel* y, por cada división, se hace el sumatorio de todos los productos de los elementos de la subdivisión de la entrada por los elementos que ocupan esa misma posición en el *kernel*. El resultado se guarda en la matriz de salida.

En la Figura 2.9, la primera subdivisión, al multiplicarse por el *kernel*, genera el siguiente resultado: $(11)+(12)+(14)+(15) = 12$. Después el *kernel* se desplaza un píxel hacia abajo para procesar la siguiente subdivisión: $(14)+(15)+(17)+(18) = 24$. Se repite esta misma operación hasta llegar al borde inferior de la matriz. El proceso vuelve a comenzar desde arriba (un píxel hacia la derecha) hasta terminar de procesar todas las subdivisiones de la imagen. En este ejemplo se han utilizado matrices en vez de tensores de mayor grado para mostrar de manera sencilla este tipo de operaciones.

Imaginemos que la entrada de la i-ésima capa es un tensor de tercer orden de tamaño $H^l \times W^l \times D^l$. El *kernel* que procesa esta entrada también es un tensor de tercer orden de tamaño $H \times W \times D^l$. Las subdivisiones, en este caso, son tensores del mismo tamaño que el *kernel*, y los sumatorios se realizan entre los productos de la primera subdivisión y dicho *kernel* de arriba hacia abajo y de izquierda a derecha.

Una vez mostrado lo que hace esta capa, pasamos a comprobar su utilidad. En la Figura 2.10 podemos ver una imagen a color como entrada, así como los resultados obtenidos tras haber aplicado dos *kernel*s distintos:

$$K = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}, K^T$$

Cuando en la imagen hay un borde horizontal en la posición (x,y), los valores del resultado de la convolución en esa posición serán grandes si aplicamos el *kernel* K. Esto es debido a que, si hay un borde horizontal, los píxeles en la posición $(x+1,y)$ y en la posición $(x-1,y)$ tendrán una gran diferencia en su magnitud. Lo mismo ocurre con el *kernel* traspuesto, que reacciona al comportamiento de los bordes verticales. En la Figura 2.10 se puede ver cómo el resultado resalta dichos bordes.

Figura 2.10: Imagen a color junto con los resultados tras haber aplicado distintos *kernel*s.

Cabe destacar que los *kernel*s son capaces de activarse; es decir, producir valores elevados cuando se encuentran con ciertas características. En la Figura 2.10 las características que podemos visualizar son los bordes. A medida que nos adentramos en capas más profundas, los *kernel*s son capaces de reaccionar a patrones más complejos. En la Figura 2.11 podemos ver estas activaciones.

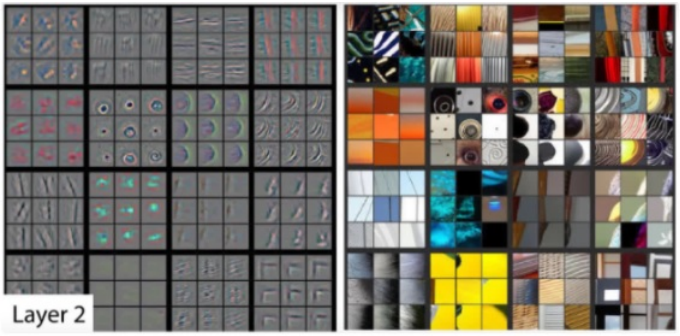

Figura 2.11: Imágenes procesadas con distintos *kernel*s.

Cuanto más nos adentramos en nuestra red, las capas son capaces de activarse al encon-

trarse con patrones más complejos. Se pueden detectar partes de objetos o incluso tipos de objetos particulares, como por ejemplo patas de pájaros, caras de perro, etc.

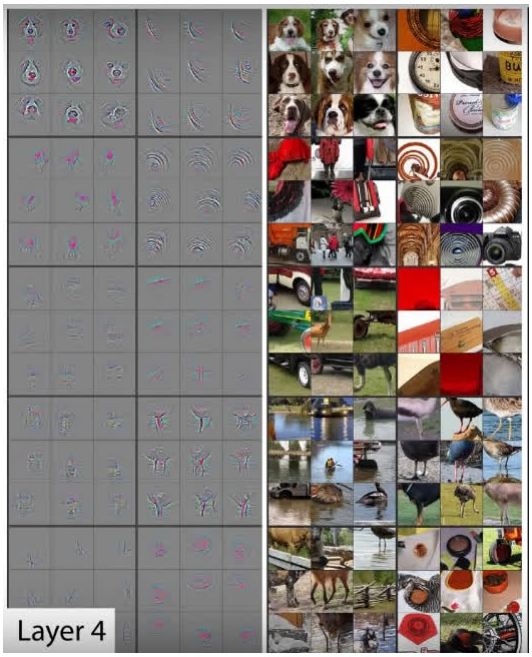

Figura 2.12: Resultados tras haber aplicado *kernel*s complejos.

La red aprende automáticamente los *kernel*s a través del entrenamiento. Los valores de dichos *kernel*s se inicializan con valores aleatorios. Las capacidades de detección de los *kernel*s se refuerzan a medida que la red aprende. En el pasado, los expertos en visión por computador eran los encargados de definir estos filtros. Un ejemplo es el filtro Sobel (ver Figura 2.13), que se utilizaba para detectar bordes. Sin embargo, con el aprendizaje profundo, estos filtros se pueden aprender de manera automática.

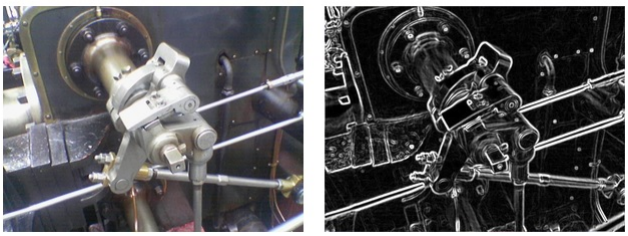

Figura 2.13: Imagen tras aplicar el filtro Sobel.

Uno de los beneficios de la capa convolucional es que todas las subdivisiones de la entrada comparten el mismo *kernel*. Esto reduce significativamente el número de parámetros

necesarios. Por ejemplo, si tenemos una imagen en la que aparecen varios perros, el *kernel* capaz de detectar este patrón se activará en las distintas posiciones.

Por otra parte, es muy habitual que haya parámetros compartidos en los diferentes *kernel*s. Imaginemos que tenemos dos *kernel*s capaces de detectar patrones de gato y patrones de perro. La **CNN** no necesita desarrollar dos subconjuntos de parámetros distintos, ya que puede aprender a detectar ciertos patrones compartidos, como ojos o texturas de pieles de animal.

Hay que dejar claro que los patrones o las características que aprende una CNN pueden no corresponder a algo tan concreto como un objeto específico. Una característica se puede activar con frecuencia ante ciertos objetos de una misma clase, y desactivarse con otros que no son de dicha clase. Sin embargo, también puede generar activaciones falsas ante objetos de otra clase. Es más, algo clave en las CNNs es su aprendizaje generalizado. Si la tarea es reconocer N tipos de objeto diferentes, la CNN extraerá M características de la imagen de entrada. Lo más probable es que cualquiera de esas M características sea útil para reconocer a los N tipos de objeto y, que para reconocer un tipo de objeto en concreto, se necesite la información de las M características.

CAPA *pooling*

La capa *pooling* se coloca normalmente justo después de una capa convolucional. Su utilidad radica en la capacidad de reducir las dimensiones de los **mapas de características o *Feature maps*** tras una capa convolucional. La operación realizada por esta capa también se llama "reducción de muestreo". A pesar de que se pierde mucha información en esta operación, resulta interesante aplicar este tipo de capas porque reduce el coste computacional del algoritmo y porque evita el sobreajuste.

Hay diferentes tipos de *pooling*. Uno de los más comunes es el *max-pooling*.

Como se puede ver en la Figura 2.14, esta operación reduce el tamaño del tensor de entrada tomando los valores máximos de una determinada zona. Existen otras estrategias de *pooling* que utilizan la media, o una media ponderada con respecto al valor central.

2.4.2. Aumentación de datos

Los modelos de aprendizaje automático de hoy en día trabajan con una infinidad de parámetros. Esto les permite generalizar lo aprendido cuando son entrenados con una gran

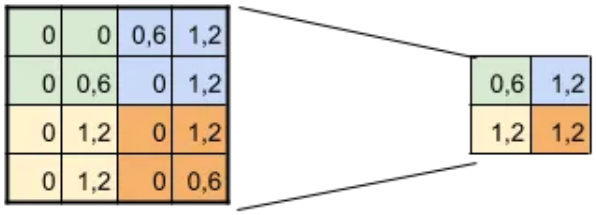

Figura 2.14: Ejemplo de *max-pooling*

cantidad de datos etiquetados. En la práctica, sin embargo, tener acceso a estos inmensos conjuntos de datos (*dataset*s) etiquetados no siempre es posible. Trabajar con datos limitados supone un elevado riesgo de sobreajuste (*overfitting*) [1]. La aumentación es una técnica frecuentemente utilizada para contrarrestar la escasez de datos.

Esta técnica consiste en hacer pequeños cambios en los datos de entrada sin cambiar su etiqueta. Así no solo se consigue aumentar la cantidad de datos sino también su diversidad.

En el ámbito de la visión por computador, la aumentación de datos se ha convertido en una herramienta indispensable para combatir el sobreajuste en las redes neuronales convolucionales. Se utiliza en todos los ámbitos para mejorar los resultados de los conjuntos de datos de referencia [17].

Se pueden aplicar transformaciones simples, como rotaciones, recortados, cambios en la saturación y el brillo, ... Pero para utilizar esta técnica, hay que tener en cuenta lo que se quiere clasificar porque, por ejemplo, si al entrenar un modelo con el MNIST *dataset* [8] se aplican rotaciones, los resultados podrían empeorar por la confusión del modelo ante las imágenes de los números 6 y 9.

En el ámbito médico, sin embargo, hacer un uso muy amplio de aumentación de imágenes es muy común. Esto se debe a que los *dataset*s para el análisis médico suelen ser de tamaño reducido y de un coste económico elevado [3]. Al combinar arquitecturas de redes de aprendizaje profundo previamente entrenadas con un amplio repertorio de técnicas de aumentación, se ha conseguido una gran precisión para la detección del cáncer de mama con un *dataset* de imágenes histológicas muy reducido: menos de 100 imágenes por clase [13]. Además de las transformaciones geométricas simples usadas habitualmente en medicina, también se pueden utilizar transformaciones elásticas o distorsiones de rejilla, puesto que, a pesar de ser más complejas, las imágenes a analizar no son imágenes de estructuras rígidas y pueden haber muchos cambios en las formas.

Figura 2.15: Distintos ejemplos de combinaciones de transformaciones aplicadas a imágenes.

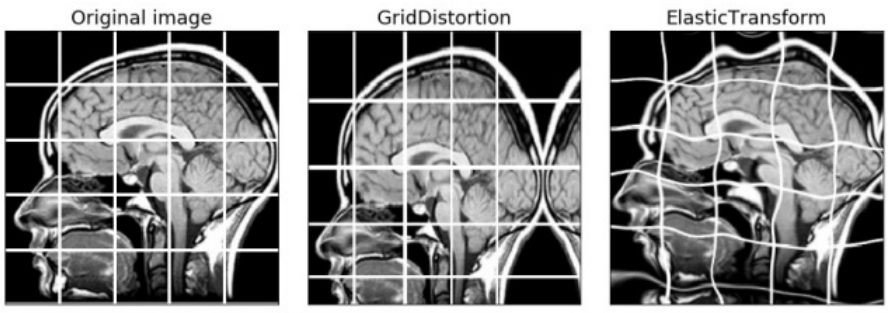

Figura 2.16: Distorsión de rejilla y de transformación elástica.

Técnica TTA

Test Time Augmentation (**TTA**) es una técnica muy similar a la de aumentación de datos, pero con la particularidad de que se utiliza con los datos de test. En vez de introducir la imagen original como entrada al modelo, se obtienen un número X de imágenes ligeramente transformadas de la misma para que el modelo las clasifique. Y se toma como respuesta el resultado de una combinación de dichas clasificaciones. Dependiendo del ámbito, esta técnica puede resultar beneficiosa [18]. También dependiendo del tipo de transformaciones y de la forma de combinar los resultados, se puede mejorar el desempeño del modelo [15].

2.4.3. Métricas

En cualquier tarea de minería de datos, para saber si nuestro modelo es eficaz, debemos tener alguna forma de medir los resultados. Hay muchas métricas para medir la eficacia de los modelos, pero en este Trabajo de Fin de Grado hemos utilizado la métrica conocida como el **área bajo la curva ROC o *Area Under ROC Curve*** (**AUC-ROC**).

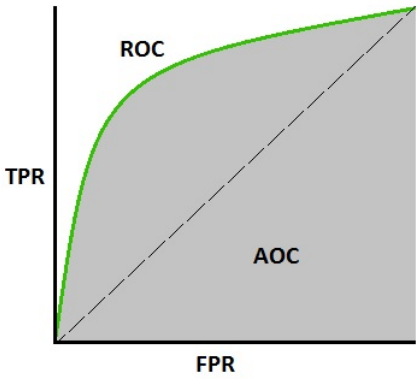

Figura 2.17: Representación del área bajo la curva ROC.

Esta métrica de evaluación es una de las más utilizadas en aprendizaje automático para verificar el rendimiento de un modelo. Nos da una idea del grado de eficacia de nuestro modelo para diferenciar entre dos clases. Cuanto más cerca esté de 1 el valor que nos dé, mejor es nuestro clasificador.

El área bajo la curva AUC-ROC es una representación gráfica de la sensibilidad en función de la especificidad según se varía el umbral de discriminación [9].

La sensibilidad (TPR) es, por definición, la tasa de "verdaderos positivos":

$$TPR = \frac{TP}{TP+FN}$$

La especificidad se define como:

$$Especificidad = \frac{TN}{TN+FP}$$

FPR es una métrica de "falsos positivos":

$$FPR = \frac{FP}{TN+FP} = 1 - Especificidad$$

A medida que cambiamos el umbral de discriminación, estos valores van cambiando y dibujando así toda la curva. Medir el área bajo esta curva permite comparar el desempeño de un clasificador frente a otro.

3. CAPÍTULO

Modelos existentes

En este capítulo se resumen los distintos modelos encontrados, y la experimentación hecha con ellos.

3.1. *Dataset*

Uno de los objetivos de este proyecto era conseguir los datos suficientes para poder crear un modelo predictivo. La idea inicial era conseguir los datos del Hospital de Basurto. Sin embargo, esta idea fue descartada porque en el hospital no existe una base de datos que recoja imágenes con su diagnóstico de los pacientes con melanoma. Para sacar la información por este medio, habría que ir sacando uno a uno de cada expediente la información, para construir así la base de datos. Esta tarea supone un gran número de horas, a las que hay que añadir las gestiones necesarias para conseguir los permisos para acceder a dichos datos.

3.1.1. *The ISIC 2020 Challenge Dataset*

Después de haber descartado la primera opción, se opta por utilizar *The ISIC 2020 Challenge Dataset* [5]. Este *dataset* está alojado en la plataforma de `Kaggle`. Esta fuente de datos es una buena opción porque los datos ya han sido tratados. Esto significa que no hay datos ni repetidos ni erróneos, por lo tanto todo el preprocesado de datos necesario ya está hecho.

Cada registro del *dataset* tiene los siguientes campos:

- *image_name*: Nombre del identificativo de la imagen.
- *patient_id*: Nombre del identificativo del paciente.
- *sex*: Sexo del paciente.
  - *male*
  - *female*
- *age_approx*: Edad aproximada del paciente.
- *anatom_site_general_challenge*: Parte anatómica de la lesión cutánea.
  - *head/neck*
  - *upper extremity*
  - *lower extremity*
  - *torso*
  - *palms/soles*
  - *oral/genital*
- *diagnosis*: Diagnostico de la lesión cutánea.
  - *unknown*
  - *nevus*
  - *melanoma*
  - *seborrheic keratosis*
  - *lentigo NOS*
  - *lichenoid keratosis*
  - *solar lentigo*
  - *cafe-au-lait macule*
  - *atypical melanocytic proliferation*
- *benign_malignant*: Malignidad de la lesión. Solo el melanoma es clasificado como maligno.

- *target*: Malignidad de la lesión en formato binario. Si la lesión cutánea es maligna este campo es 1, si es benigna es 0.

Las imágenes están disponibles en 2 formatos, en ***JPEG*** y en ***TfRecords***. La Figura 3.1 muestra ejemplos de imágenes diagnosticadas como melanomas y la Figura 3.2 muestra otras lesiones pigmentadas benignas.

Figura 3.1: Ejemplos de melanomas.

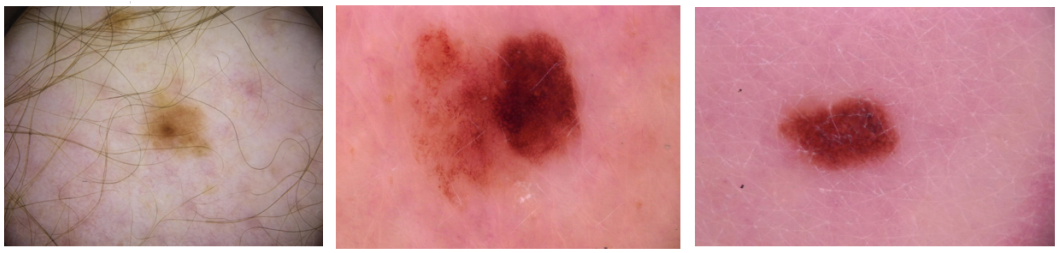

Figura 3.2: Ejemplos de lesiones pigmentadas benignas.

3.2. Análisis de otras técnicas utilizadas

En la misma plataforma donde está alojado el *dataset*, algunos de los equipos competidores comparten sus soluciones públicamente. En este apartado realizaremos un análisis exhaustivo de diferentes soluciones que se han presentado a la competición. En el siguiente enlace se puede ver la clasificación: link SIIM-ISIC Melanoma Classification.

Dentro de las soluciones con mejores puntuaciones, se han escogido las soluciones que no han necesitado una gran cantidad de recursos, o versiones adaptadas de estas.

3.2.1. Primera técnica a analizar

La primera técnica que se ha escogido consiguió la tercera posición en la competición. El código de su solución se puede encontrar en el siguiente enlace: link Kaggle SIIM-ISIC Melanoma Classification: 3rd place solution overview.

Esta técnica hace una agrupación o *ensemble* de nueve modelos distintos. Ocho de ellos son CNNs, y el último modelo hace las predicciones solo teniendo en cuenta los metadatos. La arquitectura de las ocho CNNs es igual, lo que varía es la cantidad y la resolución de las imágenes utilizadas para el entrenamiento, el tipo de aumentación y el número de épocas.

Modelo	**Datos**	**Resolución de imágenes**	**Épocas**	**Aumentación de pelo**
1	2020	256 x 256	13	
2	2020	384 x 384	15	
3	2020	512 x 512	15	
4	2020	768 x 768	15	
5	2019-2020	256 x 256	25	X
6	2019-2020	384 x 384	25	X
7	2019-2020	512 x 512	12	X
8	2019-2020	768 x 768	10	

Tabla 3.1: Modelos utilizados en la primera técnica.

3.2.2. Segunda técnica a analizar

La segunda técnica a analizar consiguió la posición más alta en la competición. Es una agrupación o *ensemble* de modelos con distintas arquitecturas y tamaños de entrada. Los modelos son todos CNNs que en su mayoría tienen como entrada imágenes, pero en algunos casos también se toma en cuenta los metadatos de los pacientes. Tener un sistema de validación estable y hacer una elección correcta del objetivo de los clasificadores son algunas de las claves de su éxito [16].

En la competición, la técnica de validación era el área bajo la curva ROC. El inconveniente de utilizar esta técnica es su inestabilidad. Las imágenes ofrecidas por la competición para hacer el entrenamiento ascienden a 33000. Sin embargo, el porcentaje de melanomas es muy bajo, aproximadamente del 1,76 %. Al haber un número tan pequeño de casos positivos, esta métrica era muy inestable, incluso con una validación cruzada de 5 iteraciones (*5-fold cross validation*). Para solventar este problema, combinan el *dataset* de la

competición de 2019, que incluye las imágenes de 2018 y 2017. Este *dataset* es más pequeño, 25000 imágenes, pero contiene un porcentaje más alto de melanomas, un 17,85 %, lo cual se traduce en una métrica mucho más estable.

Lo que diferencia esta técnica de la anterior es lo que predicen los modelos. El objetivo de la competición es clasificar las lesiones que aparecen en las imágenes en benignas y malignas. Sin embargo, como uno de los parámetros del conjunto de datos de entrenamiento es el diagnóstico, los modelos propuestos tienen como objetivo predecir el diagnóstico, convirtiéndolos de esta manera en clasificadores de clases. Dado que los melanomas son el único diagnóstico maligno, utilizan su probabilidad de clase como resultado. Esto tiene mucho sentido porque la diversidad que existe en las clases benignas es bastante grande. Usar el diagnóstico como parámetro a predecir puede aportar bastante información al modelo.

El segundo aspecto a destacar es que se utilizan metadatos dentro de las capas ocultas de la CNN.

3.3. Experimentación

En esta sección se describen una serie de experimentos realizados con los modelos propuestos. Las dos técnicas analizadas en las secciones anteriores tienen una puntuación muy elevada pero la experimentación solo se ha hecho sobre la primera. La principal razón es que estos modelos fueron implementados usando exclusivamente las **TPUs** que ofrece *Kaggle*, la otra, en cambio, utiliza 8 *NVIDIA Tesla V100* ***GPUs*** en paralelo. También se ha tenido en cuenta que el tiempo de entrenamiento no superaba en ningún caso las 3h, mientras que la otra solución tardaba entre 15 y 45 horas. La última razón que ha motivado la decisión es el uso de `TensorFlow` del que ya se tenía un conocimiento mínimo. La otra técnica utilizaba `PyTorch`, otra librería de **ANNs**.

3.3.1. Técnica TTA

Tras el análisis de las dos técnicas anteriores, se comprueba que, en ambas, se utiliza la técnica TTA. Partiendo de este hecho, se proponen los siguientes experimentos para buscar una mejora del resultado:

1. Sin hacer uso de la TTA. Tabla 6.1.

2. Añadir la imagen original al hacer la media de las predicciones. Tabla 6.2.
3. Cambiar el peso de la imagen original para hacer la media. Tabla 6.3.
4. Cambiar el número de aumentaciones usadas en la TTA. Tabla 6.4.
5. Cambiar el tipo de aumentaciones en la TTA. Tabla 6.5.

Para todos los experimentos se han utilizado los modelos propuestos por la primera técnica analizada anteriormente.

3.3.2. Experimentación de entrenamiento

La experimentación propuesta en la subsección anterior no implica volver a entrenar la **ANN**, por lo tanto, su coste computacional es mucho menor. En cambio, en los experimentos que se describen en las siguientes secciones, es necesario volver a entrenar las **ANNs**. A pesar de contar con las **TPUs** de `Kaggle`, el tiempo de cada uno de ellos es muy elevado. Se ha optado por experimentar solo con un modelo. Sus características son:

- Resolución de entrada: 256x256.
- *Dataset*: 2020.
- Épocas. 13.

Número de épocas

Este es uno de los hiperparámetros más comunes para experimentar. Los experimentos propuestos son:

- 10 épocas.
- 12 épocas.
- 15 épocas.
- 20 épocas

Los resultados obtenidos se encuentran en la Tabla 6.8.

Variación de la tasa de aprendizaje

Se propone experimentar con otra técnica para variar la tasa de aprendizaje. Esta técnica consiste en reducir la tasa de aprendizaje cuando se alcanza un estancamiento. Se ha probado la técnica variando el número de épocas de paciencia, esto es el número de épocas que espera el algoritmo antes de reducir la tasa de aprendizaje. Los parámetros especificados son los siguientes:

- Tasa de aprendizaje mínima: 0,000001.
- Factor de reducción de la tasa de aprendizaje: 0,2.

Aumentaciones

En este experimento se propone quitar algunas aumentaciones. Dos de los rasgos que tienen en cuenta los dermatólogos para diagnosticar un melanoma son la simetría y la uniformidad de los bordes de una lesión pigmentada. En este experimento se propone quitar las aumentaciones que cambien la forma geométrica de las lesiones. Los experimentos son:

- Quitar la transformación de cizalla y el zoom.
- Utilizar un zoom uniforme, es decir que el zoom vertical y horizontal sea el mismo.
- Utilizar un zoom uniforme además de la transformación de cizalla.

4. CAPÍTULO

Modelo Propuesto

Después de haber analizado y experimentado con los distintos modelos, en este capítulo se describe el modelo seleccionado y todas sus características.

4.1. Arquitectura del modelo

La arquitectura del modelo propuesto es `EfficientNet-B6` [10]. Antes de la aparición de `EfficientNet`, para mejorar los modelos se optaba por escalarlos de alguna manera. Hay tres maneras de escalar un modelo:

- Hacer la red más profunda: aumentar el número de capas de la red.
- Hacer la red más ancha: aumentar el número de *feature-map*s en cada capa.
- Aumentar la resolución de las imágenes de entrada.

4.1.1. Mayor profundidad (d)

Hacer la red más profunda es la manera más habitual de mejorar un modelo. Se pueden añadir o quitar capas de una red neuronal. Por ejemplo existe `ResNet-18` y `ResNet-200`. La intuición nos dice que a mayor número de capas las características encontradas serán mejores y generalizarán mejor. Sin embargo esto no es así. El principal problema es el

desvanecimiento del gradiente, esto significa que en algunos casos el gradiente irá reduciéndose a valores muy pequeños impidiendo cambiar el peso de su valor eficazmente. Incluso utilizando técnicas para evitar este problema, llega un punto en que añadir más capas es inútil. Por ejemplo, `ResNet-101` tiene una puntuación muy similar a `ResNet-1000`.

4.1.2. Mayor anchura (w)

Esta técnica de escalar una red se suele utilizar cuando se quiere mantener la red pequeña. También se ha demostrado que al hacer más ancha una CNN se obtienen características más detalladas [20]. Además, las redes pequeñas son más fáciles de entrenar. El problema es que llega un punto en que la precisión del modelo se satura si solo se añade anchura a la red.

4.1.3. Mayor resolución (r)

Podemos pensar que al haber mayor resolución, la red es capaz de diferenciar un mayor número de detalles, haciendo que la clasificación sea mejor. Esa es la razón por la que a la hora de hacer tareas de reconocimiento de objetos, se utilizan resoluciones de 300×300, o de 512×512, o de 600×600. Sin embargo, este tipo de escalada también se satura rápido. Por ejemplo, no hay mucha diferencia entre una red entrenada con imágenes de 500×500 que con imágenes de 560×560.

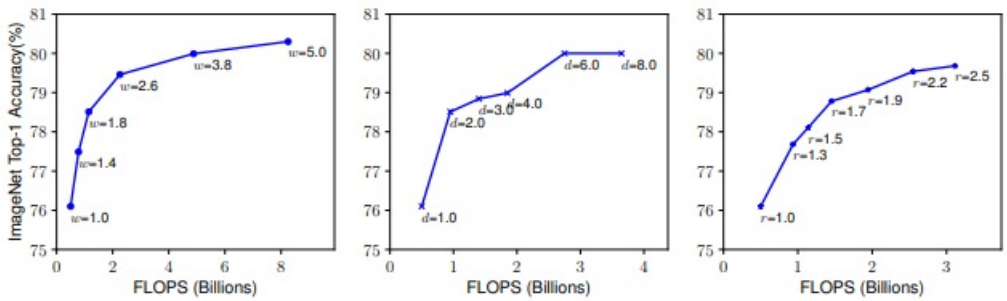

Figura 4.1: Diferentes formas de escalar una red.

En la Figura 4.1 podemos ver que al escalar solo uno de los parámetros de una red (*d*=profundidad, *w*=anchura, *r*=resolución), la mejora se satura rápidamente.

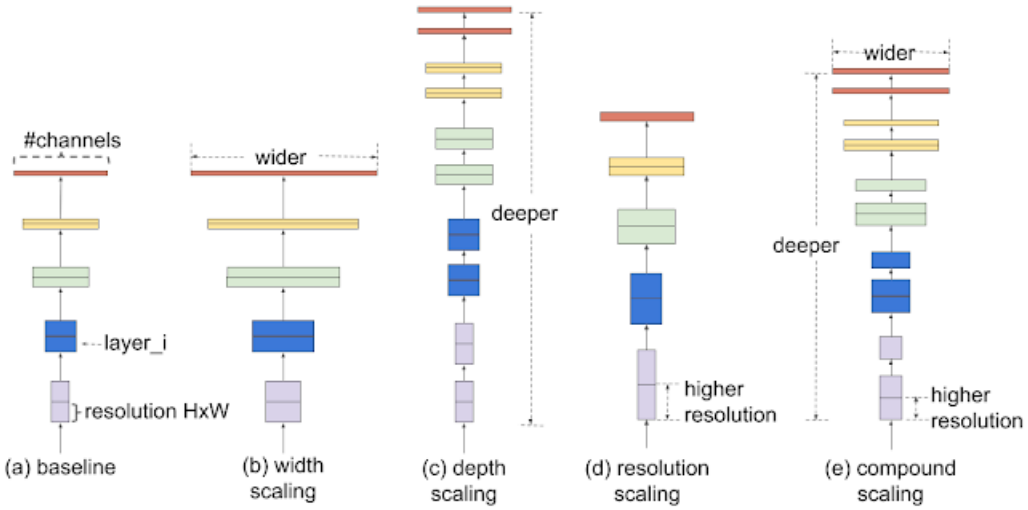

Figura 4.2: Tipos de escalado.

4.1.4. Combinación de escalados

La combinación de los distintos tipos de escalado se puede hacer de forma arbitraria, sin embargo el resultado obtenido puede no ser el más óptimo en cuanto a precisión y eficiencia. La hipótesis inicial consiste en que, al aumentar el número de capas, la anchura de esas capas y la resolución de las imágenes de entrada, se obtendrá un mejor resultado. Se hacen múltiples experimentos para demostrar esta hipótesis. En la Figura 4.2 se muestran los distintos tipos de escalado.

En la Figura 4.3 se parte de una resolución de 224×224 y una profundidad de 18 capas ($d = 1$, $r = 1$). Cada punto de la línea representa un valor distinto de la anchura de la red (w). Se puede ver que se consiguen mejores puntuaciones con igual número de **FLOPS (operaciones de coma flotante por segundo o *Floating point operations per second*)** si se combina de forma adecuada el escalado de la red.

Los creadores de `EfficientNet` proponen una forma de escalar las dimensiones de una red que es sencilla pero eficiente. Su propuesta utiliza el coeficiente compuesto Φ para escalar de una forma uniforme la anchura, la profundidad y la resolución de la red.

$$Profundidad : d = \alpha^{\Phi}$$

$$Anchura : w = \beta^{\Phi}$$

$$Resolución : r = \gamma^{\Phi}$$

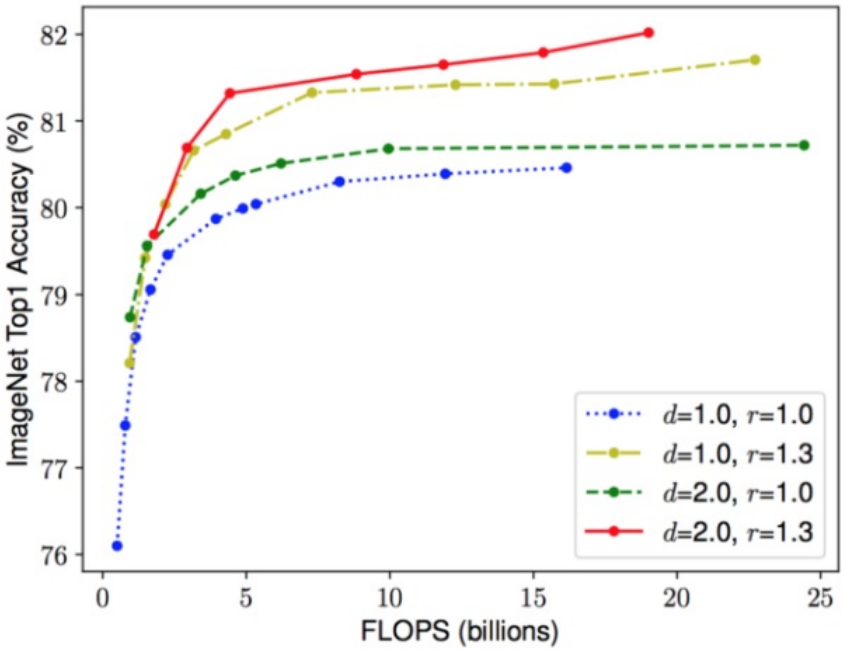

Figura 4.3: Escalado combinando las diferentes dimensiones de una red.

$$verificando:$$

$$\alpha \cdot \beta^{\Phi} \cdot \gamma^{\Phi} \approx 2$$

$$\alpha \geq 1, \beta \geq 1, \gamma \geq 1$$

Φ es un parámetro especificado por el usuario que controla la cantidad de recursos disponibles. α, β y γ especifican cómo se distribuyen estos recursos entre la profundidad, la anchura y la resolución de la red respectivamente.

En una CNN las capas convolucionales son las más costosas computacionalmente. Los **FLOPS** de una operación convolucional son directamente proporcionales a d, w^2, r^2. Esto significa que si duplicamos el número de capas, se duplican los FLOPS. Sin embargo, si duplicamos la anchura de las capas, o la resolución de las imágenes, el número de FLOPS se multiplica por cuatro. Para que los FLOPS no crezcan de manera desmesurada, se aplica la restricción $\alpha \cdot \beta^{\Phi} \cdot \gamma^{\Phi} \approx 2$.

4.1.5. Arquitectura base

Al hacer el escalado no se alteran el tipo de operaciones dentro de la red, por lo tanto es importante partir de una buena arquitectura. Los autores de *EfficientNet* partieron de una arquitectura similar a la de *MobileNetV2* y *MnasNet* 4.4.

Figura 4.4: Arquitectura base de EfficientNet.

En la Figura 4.4 se puede ver que la mayoría de capas de la red son **MVConv (bloques residuales invertidos o *Inverted Residual Blocks*)** [14].

Si tomamos la arquitectura reflejada en la Figura 4.4 como base, tenemos que encontrar los parámetros de escalada óptimos. Si nos guiamos por la ecuación propuesta y tomamos $\Phi = 1$, lo que significa que contamos con el doble de recursos, los valores óptimos son $\alpha = 1{,}2, \beta = 1{,}1, \gamma = 1{,}15$. Esta es la arquitectura de `EfficientNet B0`. Si se fijan estos valores de α, β, γ y se experimenta con distintos valores de Φ, se obtienen las arquitecturas de `EfficientNet B1-B7`. Con estas arquitecturas se consiguen puntuaciones que superan el estado del arte en muchas tareas de clasificación, reduciendo significativamente el número de parámetros de la red.

4.2. Aumentaciones utilizadas

La estrategia utilizada para crear las aumentaciones de los datos es la siguiente. Se utiliza una distribución normal centrada en 1 para sacar números aleatorios. Esos números aleatorios generados se utilizan para calcular las matrices de transformación de las imágenes. Los tipos de aumentaciones utilizados son:

- Rotaciones.
- Transformación de cizalla.

- Zoom uniforme.
- Traslaciones.

Para calcular la matriz de rotación, Figura 4.5, se utiliza el número obtenido aleatoriamente como *rot*.

$$r = \frac{\pi \cdot rot}{180}$$

$$\text{Matriz de rotación} = \begin{pmatrix} \cos r & \sin r & 0 \\ -\sin r & \cos r & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Figura 4.5: Imágenes obtenidas tras haber aplicado una matriz de rotación.

Para calcular la matriz de transformación de cizalla, Figura 4.6, se utiliza el número obtenido aleatoriamente como *shear*.

$$\text{Matriz de transformación} = \begin{pmatrix} 1 & \sin shear & 0 \\ 0 & \cos shear & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Para calcular la matriz de zoom, Figura 4.7, se utiliza el número obtenido aleatoriamente como *zoom*.

$$\text{Matriz de transformación} = \begin{pmatrix} \frac{1}{zoom} & 0 & 0 \\ 0 & \frac{1}{zoom} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Figura 4.6: Imagen obtenida tras haber aplicado una matriz de transformación de cizalla.

Figura 4.7: Imagen obtenida tras haber aplicado una matriz de zoom.

Para calcular la matriz de traslación, Figura 4.8, se utilizan los números obtenidos aleatoriamente como $hshift$ y $vshift$.

$$\text{Matriz de transformación} = \begin{pmatrix} 1 & 0 & hshift \\ 0 & 1 & wshift \\ 0 & 0 & 1 \end{pmatrix}$$

También se utiliza las aumentaciones disponibles en la librería de *Tensorflow*:

- Voltear la imagen (`tf.image.random_flip_left_right`).
- Cambiar el tono (`tf.image.random_hue`).
- Cambiar la saturación (`tf.image.random_saturation`).
- Cambiar el contraste (`tf.image.random_contrast`).
- Cambiar el brillo (`tf.image.random_brightness`).

Figura 4.8: Imagen obtenida tras haber aplicado una matriz de traslación.

4.3. Planificación de la tasa de aprendizaje

Planificar la tasa de aprendizaje es algo que puede marcar la diferencia a la hora de entrenar un modelo. La tasa de aprendizaje controla el tamaño del cambio de los pesos en cada iteración. Si este tamaño es muy pequeño, la velocidad de convergencia del modelo será muy lenta, y es probable que acabe en un mínimo local. En cambio, si es muy grande, la función del error será muy inestable, y dará saltos alrededor del mínimo sin llegar nunca a alcanzarlo. En la Figura 4.9 se puede percibir esto de forma más visual.

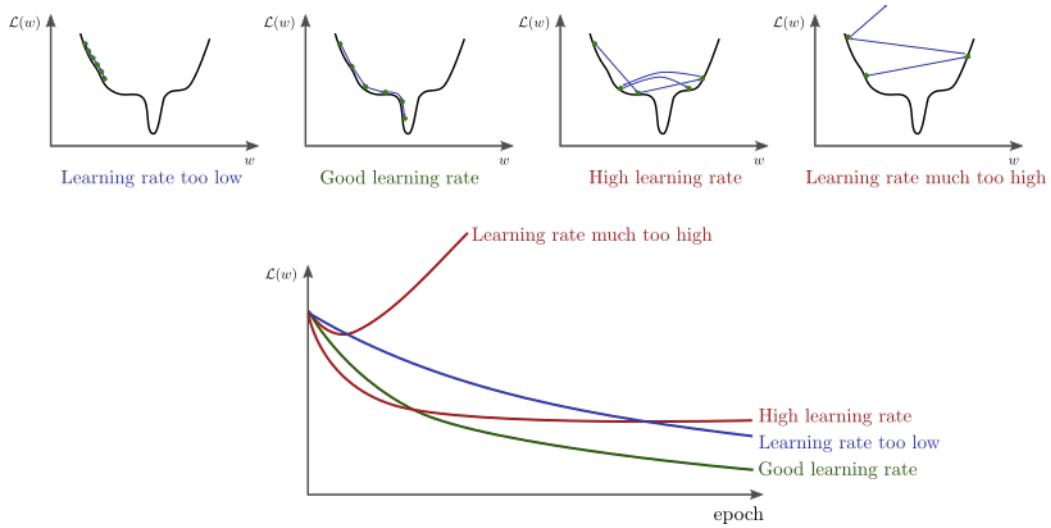

Figura 4.9: Visualización de las distintas tasas de aprendizaje.

Una de las maneras más efectivas de planificar la tasa de aprendizaje, es reducirla cuando

se encuentra con un estancamiento. La Figura 4.10 muestra cómo afecta la reducción de la tasa de aprendizaje a la función del error.

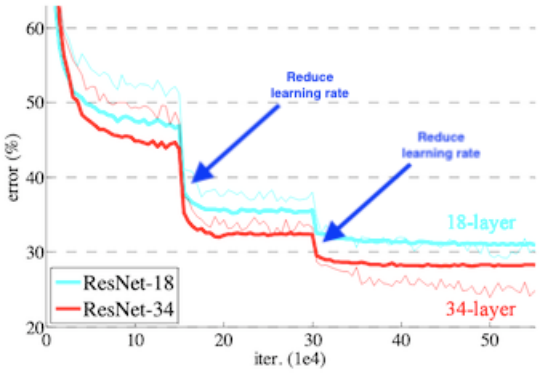

Figura 4.10: Visualización de la función de error tras reducirle la tasa de aprendizaje.

En el modelo propuesto se utiliza la siguiente técnica. En las primeras épocas, las de calentamiento, se aumenta la tasa de aprendizaje hasta alcanzar el máximo en la última época de calentamiento. Luego se decrementa hasta llegar al mínimo en la última época. En la Figura 4.11 se puede ver cómo evolucionan la tasa de aprendizaje, la función de error y la precisión.

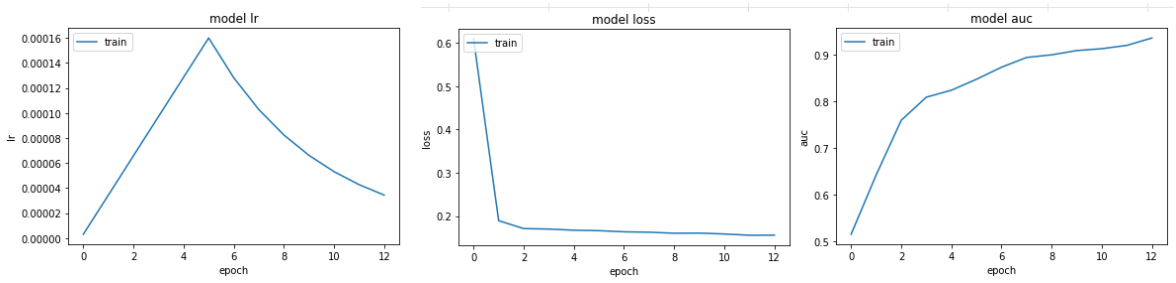

Figura 4.11: Visualización de la evolución de la tasa de aprendizaje, la función de error y la precisión.

5. CAPÍTULO

Desarrollo de la aplicación

En este capítulo se describe la forma de darle una utilidad al modelo creado. Esto es algo muy importante, ya que de nada sirve una herramienta si no hay una forma útil de hacer uso de ella. En este proyecto se ha decidido que la forma más práctica es creando una aplicación web. A lo largo de este capítulo se explica el porqué de esta decisión, las herramientas utilizadas y el diseño de la aplicación.

5.1. Motivación

Después de crear el modelo, hay que buscar una forma para que los usuarios puedan utilizarlo. En este proyecto se ha optado por crear una aplicación web. Se ha elegido esta opción por las siguientes razones:

1. Es multi-plataforma. Cualquier equipo con un navegador instalado puede hacer uso de la aplicación independientemente del tipo de dispositivo que sea y de su sistema operativo.

2. Permite a los usuarios colaborar con el proyecto. Una de las opciones para mejorar el modelo es utilizar más imágenes a la hora de entrenarlo. Si en la aplicación se permite a los usuarios subir sus propias imágenes con sus respectivos diagnósticos, se puede ir mejorando el modelo con las nuevas imágenes.

Las aplicaciones web normalmente están formadas por tres capas. Existen muchas variaciones posibles, pero en la mayoría se pueden distinguir claramente la capa del navegador, la capa del servidor y la capa de persistencia.

5.1.1. Capa del navegador

Esta capa, también conocida como interfaz de usuario, es la capa desde donde el usuario interactúa con la aplicación. Hay que destacar que esta capa se ejecuta en el propio equipo del usuario, a través del navegador. En esta capa se le pide al usuario que introduzca los datos a procesar por el modelo. La aplicación utiliza `JavaScript` para transformar los datos y poder enviarlos a través del protocolo `HTTP`. Mediante el método "`POST`" envía los datos al servidor. Cuando recibe la respuesta, utiliza `JavaScript` para insertar en `DOM` la predicción del modelo, permitiendo que el usuario pueda verlo.

5.1.2. Capa del servidor

Esta capa es donde se encuentra toda la lógica de negocio de la aplicación. El servidor aloja distintos servicios web escritos en `Python`, en uno de ellos está nuestro modelo. Este servicio, al recibir los datos de la capa del cliente, los introduce al modelo y devuelve una predicción. La capa del servidor se ha implementado utilizando `Flask`, un *framework* de desarrollo de aplicaciones web y APIs escrito en `Python`.

5.1.3. Capa de persistencia

En esta capa es donde se guardan los datos. En nuestra aplicación se utiliza para tener un registro de los usuarios y para almacenar las imágenes que aporten mejoras en el modelo. Para esta aplicación se ha utilizado `SQLAlchemy`, un ORM (*Object Relational Mapper*) escrito en `Python` y se ha utilizado la base de datos relacional: `SQLite`.

5.2. Diseño de la aplicación

Esta aplicación es una prueba de concepto, ya que el objetivo era probar la viabilidad de una aplicación web que albergase un modelo predictivo. Se ha utilizado uno de los modelos generados por los *notebooks* de `Kaggle`.

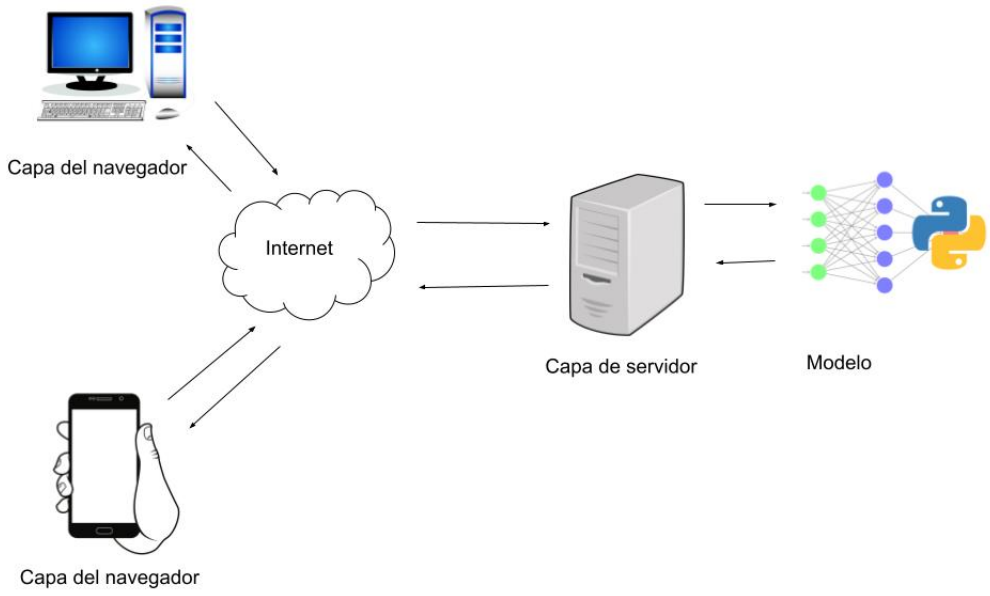

Figura 5.1: Estructura de la aplicación.

La primera tarea ha sido definir los casos de uso. Después, se han implementado las interfaces de usuario necesarias.

5.2.1. Casos de uso

A la hora de diseñar una aplicación, lo primero que hay que tener en cuenta para poder hacer un buen diseño son los casos de uso. La idea de hacer una aplicación web surgió para dar una utilidad al modelo creado, permitiendo a usuarios de todo el mundo utilizarlo para obtener predicciones de sus propias imágenes. Sin embargo, hemos caído en cuenta de que también es una forma de poder mejorar el modelo. Uno de los modos de mejorar el modelo es utilizando más imágenes a la hora de entrenarlo, por lo que es interesante permitir a los usuarios de todo el mundo subir sus propias imágenes.

Teniendo todo lo anterior en cuenta, los casos de uso de nuestra aplicación serán los siguientes:

1. Permitir a un usuario subir una foto, añadirle meta-datos y obtener una predicción del modelo.
2. Permitir a un usuario registrarse e iniciar sesión. Hay dos opciones: fuente médica o usuario común.
3. Permitir a un usuario registrado subir una foto.

Cabe destacar que, en el futuro, cuando la aplicación esté en marcha, se podrán devolver dos predicciones: la obtenida con el modelo entrenado con imágenes de usuarios comunes y la obtenida con fuentes médicas.

5.2.2. Interfaz de usuario

A continuación, se describen las distintas interfaces de usuario para cada caso de uso. Cada uno tiene su propia página `HTML`.

Caso de uso 1: Obtener una predicción

Este es el principal caso de uso de nuestra aplicación. Por ello, tal y como podemos ver en la Figura 5.2, en nuestra página de inicio tenemos la opción de subir una imagen. Una vez subida, como podemos ver en la Figura 5.3, se obtiene la predicción de nuestro modelo.

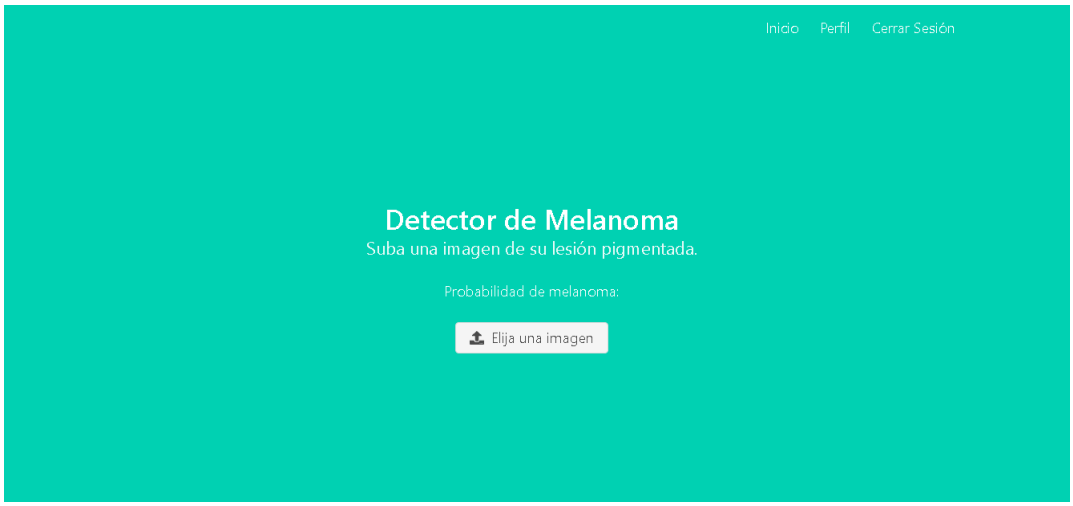

Figura 5.2: Página principal de la aplicación.

Figura 5.3: Predicción devuelta por la aplicación.

En la parte superior de la Figura 5.2 podemos ver la barra de navegación. Desde aquí se puede navegar por la aplicación a los distintos casos de uso.

Caso de uso 2: Registro e Inicio de Sesión

Figura 5.4: Formularios de registro e inicio de sesión.

Desde la barra de navegación tenemos acceso a las opciones: Registro e Inicio de Sesión. En la Figura 5.4 podemos ver los formularios necesarios para estas opciones. El formulario de registro propuesto en esta prueba de concepto es muy simple, ya que cualquier persona se puede registrar como usuario médico. Sin embargo, en la práctica este formulario debe ser más estricto para que las imágenes que proceden de una fuente médica se puedan agregar al modelo como fuentes verificadas.

Caso de uso 3: Subir nuevos datos

Al iniciar una sesión el usuario tiene tres opciones en la barra de navegación: Inicio (desde donde se accede al primer caso de uso), Perfil (desde donde puede aportar imágenes al modelo) y Cerrar Sesión. En la página de perfil, tal y como muestra la Figura 5.5, para subir una imagen hay que completar el formulario. Los campos obligatorios son:

- la imagen.
- el diagnóstico.

Los campos opcionales son:

- el sexo.
- el lugar anatómico.
- la edad aproximada.
- el diagnóstico.

Este caso de uso, aunque parezca el más trivial, es muy importante ya que permite generar una base de datos de imágenes médicas. Este tipo de bases de datos son escasas y tienen un alto coste. Mediante esta aplicación se puede crear una base de datos centralizada y accesible para todo el público y de un tamaño considerable.

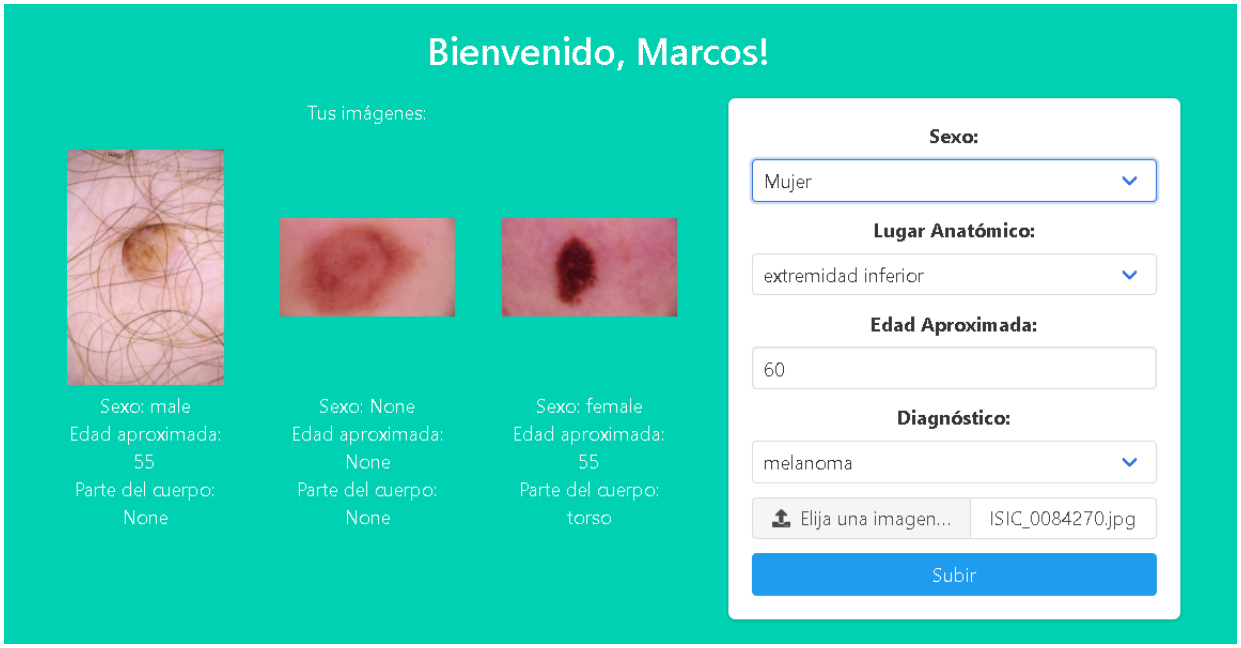

Figura 5.5: Formulario para subir una imagen.

Cada usuario puede ver en su perfil las imágenes que ha subido.

6. CAPÍTULO

Resultados obtenidos

En este capítulo se describen los resultados de este Trabajo de Fin de Grado. Este trabajo ha tenido dos tareas bastante diferenciadas: la elección del modelo, donde se ha experimentado con variaciones de modelos existentes, y el desarrollo de la aplicación.

6.1. Resultados de la experimentación

En la tercera sección del tercer capítulo, ver Sección 3.3, se proponen una serie de experimentos para los cuales se muestran los correspondientes resultados a continuación.

6.1.1. TTA

Los resultados obtenidos se resumen en las siguientes tablas. El **AUC** inicial es el resultado de los modelos sin ninguna alteración, es decir, con una **TTA** de 25 pasos.

Los resultados de la Tabla 6.1 demuestran que la aplicación de la técnica **TTA** mejora el resultado en todos y cada uno de los modelos.

Experimento	Resolución	*Dataset*	AUC Inicial	AUC Final	Diferencia
1	768	2019-2020	0,9280	0,9161	-1,28 %
2	768	2020	0,8946	0,8537	-4,57 %
3	512	2019-2020	0,9239	0,9117	-1,32 %
4	512	2020	0,8983	0,8751	-2,58 %
5	384	2019-2020	0,9243	0,8944	-3,23 %
6	384	2020	0,9111	0,8738	-4,09 %
7	256	2019-2020	0,9230	0,9099	-1,42 %
8	256	2020	0,9106	0,9037	-0,76 %

Tabla 6.1: Resultados sin aplicar TTA.

Experimento	Resolución	*Dataset*	AUC Inicial	AUC Final	Diferencia
1	768	2019-2020	0,9280	0,9314	0,37 %
2	768	2020	0,8946	0,8962	0,18 %
3	512	2019-2020	0,9239	0,9270	0,34 %
4	512	2020	0,8983	0,8987	0,04 %
5	384	2019-2020	0,9243	0,9217	-0,28 %
6	384	2020	0,9111	0,9134	0,25 %
7	256	2019-2020	0,9230	0,9243	0,14 %
8	256	2020	0,9106	0,9102	-0,04 %

Tabla 6.2: Añadir la imagen original al hacer la media de las predicciones.

Experimento	Resolución	*Dataset*	AUC Inicial	AUC TTA 1	AUC TTA 2
1	768	2019- 2020	0,9280	0,9314	0,9281
2	768	2020	0,8946	0,8962	0,8849
3	512	2019- 2020	0,9239	0,9270	0,9243
4	512	2020	0,8983	0,8987	0,8858
5	384	2019- 2020	0,9243	0,9217	0,9241
6	384	2020	0,9111	0,9134	0,9021
7	256	2019- 2020	0,9230	0,9243	0,9225
8	256	2020	0,9106	0,9102	0,9092

Tabla 6.3: Cambiar el peso de la imagen original para hacer la media.

En la Tabla 6.2 se ve que, al añadir la imagen original, se mejora el resultado de la mayoría de los modelos.

Para cambiar el peso de la imagen original, al hacer la media se ha optado por añadir la imagen original dos veces (AUC TTA 2). En la Tabla 6.3 se puede ver que esta opción

mejora la **TTA** Inicial, pero parece que añadir solamente una imagen (AUC TTA 1) es una mejor opción.

Experimento	Resolución	*Dataset*	AUC Inicial	AUC 30	AUC 40
1	768	2019- 2020	0,9280	0,9302	0,9296
2	768	2020	0,8946	0,8930	0,8947
3	512	2019- 2020	0,9239	0,9259	0,9253
4	512	2020	0,8983	0,8982	0,8999
5	384	2019- 2020	0,9243	0,9247	0,9263
6	384	2020	0,9111	0,9143	0,9144
7	256	2019- 2020	0,9230	0,9276	0,9247
8	256	2020	0,9106	0,9101	0,9124

Tabla 6.4: Cambiar el número de aumentaciones usadas en la TTA.

En la Tabla 6.4 se puede ver que añadir más aumentaciones a la hora de hacer la **TTA** mejora la mayoría de los resultados individuales de los modelos.

Experimento	Resolución	*Dataset*	AUC Inicial	AUC Final	Mejora
1	768	2019- 2020	0,9280	0,9321	0,44 %
2	768	2020	0,8946	0,8958	0,13 %
3	512	2019- 2020	0,9239	0,9286	0,51 %
4	512	2020	0,8983	0,8956	-0,30 %
5	384	2019- 2020	0,9243	0,9226	-0,18 %
6	384	2020	0,9111	0,9118	0,08 %
7	256	2019- 2020	0,9230	0,9248	0,20 %
8	256	2020	0,9106	0,9103	-0,03 %

Tabla 6.5: Cambiar el tipo de aumentaciones en la TTA.

En la Tabla 6.5 se ha reducido el tipo de aumentaciones. Se han quitado las aumentaciones que alteraban la forma geométrica del melanoma. Es interesante ver que este cambio es el que se comporta de una manera más irregular ya que mejora para algunos modelos y empeora para otros. Hay que destacar que, con este tipo de **TTA**, se consigue el mejor resultado individual de un solo modelo $AUC = 0{,}9321$.

Experimento	Resolución	*Dataset*	Actual	TTA30	Mejora	TTA40	Mejora	TTA + original	Mejora	Cambiar Aumentaciones	Mejora
1	768	2019-2020	0,9280	0,9302	0,24 %	0,9296	0,17 %	0,9314	0,37 %	0,9321	0,44 %
2	768	2020	0,8946	0,8930	-0,18 %	0,8947	0,01 %	0,8962	0,18 %	0,8958	0,13 %
3	512	2019-2020	0,9239	0,9259	0,22 %	0,9253	0,15 %	0,927	0,34 %	0,9286	0,51 %
4	512	2020	0,8983	0,8982	-0,01 %	0,8999	0,18 %	0,8987	0,04 %	0,8956	-0,30 %
5	384	2019-2020	0,9243	0,9247	0,04 %	0,9263	0,22 %	0,9217	-0,28 %	0,9226	-0,18 %
6	384	2020	0,9111	0,9143	0,35 %	0,9144	0,36 %	0,9134	0,25 %	0,9118	0,08 %
7	256	2019-2020	0,9230	0,9276	0,50 %	0,9247	0,18 %	0,9243	0,14 %	0,9248	0,20 %
8	256	2020	0,9106	0,9101	-0,05 %	0,9124	0,20 %	0,9102	-0,04 %	0,9103	-0,03 %
Media					0,14 %		0,18 %		0,12 %		0,10 %

Tabla 6.6: Resumen de experimentación TTA.

Ensemble	Puntuación Total	Puntuación Parcial
TTA 40	0,9491	0,9592
Cambiar Aumentaciones	0,9489	0,9584
TTA + original	0,9491	0,9588
Puntuación del modelo propuesto versión original	0,9484	0,9654
Mejor puntuación de la competición	0,9490	0,9586

Tabla 6.7: Resultado de las *ensemble*s.

En la Tabla 6.6 se resume la experimentación con la técnica **TTA**. Como podemos ver, el cambio que mejora la media de todos los modelos es el de realizar 40 pasos en la **TTA**. Es decir, se realizan 40 aumentaciones aleatorias para cada imagen y se devuelve la media de predicciones como resultado. Este cambio aumenta la puntuación media un 0,18 %.

Sin embargo, que todos los modelos sean mejores individualmente no asegura que su *ensemble* tenga un mejor resultado. Para comprobar su comportamiento, hemos implementado la *ensemble* de los modelos mejorados.

Ensemble

Una vez obtenidos los modelos mejorados, se han hecho las predicciones de los datos de validación. Estos resultados se han guardado en ficheros `.csv` para luego poder procesarlos. Para todas las *ensemble*s se han utilizado los metadatos a la hora de hacer la media. Es decir, una vez hecho la *ensemble* de los modelos de las CNNs mejorados, se hace una media ponderada con la predicción de los metadatos con un peso del 20 %. Los resultados se muestran en la Tabla 6.7.

En la Tabla 6.7 se comparan los resultados obtenidos con la puntuación del modelo ganador y con la puntuación del modelo original sin ninguna mejora. Con los modelos obtenidos se habría ganado la competición. Cabe destacar que, a pesar de que los modelos individuales obtenidos al añadir una imagen original a la hora de hacer la **TTA** tengan un peor resultado que los modelos obtenidos de hacer la **TTA** con 40 pasos (véase Tabla 6.6), en conjunto se obtiene la misma puntuación haciendo 14 predicciones menos. Por lo tanto, es una mejor opción. En la Tabla 6.7 se muestra la puntuación parcial con el 30 % de los datos de validación y la puntuación total con el resto de los datos. En la competición, es esta puntuación la que cuenta.

Épocas	AUC	Mejora
10	0,9157	0,56 %
12	0,9165	0,65 %
15	0,9096	-0,11 %
20	0,8772	-3,67 %

Tabla 6.8: Resultado con distinto número de épocas.

Épocas	Paciencia	AUC
13	3	0,9044 %
13	2	0,9007 %
13	1	0,9052 %

Tabla 6.9: Resultado con reducción de aprendizaje ante estancamiento.

6.1.2. Número de Épocas

Los resultados obtenidos en este experimento son sorprendentes. En la Tabla 6.8 se puede ver que el modelo mejora con menos épocas de entrenamiento. Esto se debe al sobreajuste ocasionado por el exceso de entrenamiento.

6.1.3. Variación de la tasa de aprendizaje

En esta experimentación no se han obtenido muy buenos resultados, ya que no han mejorado la estrategia propuesta. El modelo propuesto obtenía un **AUC** de 0,9106.

En la Tabla 6.9 podemos observar los resultados obtenidos.

6.1.4. Aumentaciones distintas

En este caso, los resultados obtenidos se muestran en la Tabla 6.10. Hay que destacar que, cambiando el tipo de zoom, se consigue la mayor mejora de un modelo: un $1,81\,\%$. Por lo cual parece que realizar un zoom uniforme mejora considerablemente las predicciones. Tiene sentido que esto sea así porque el zoom varía mucho de una imagen a otra.

6.2. Resultados de la aplicación

La aplicación se ha desarrollado como una prueba de concepto, por lo que le faltan muchos aspectos relevantes para poder ponerse en un entorno de producción. Sin embargo,

Aumentaciones	AUC	Mejora
Sin Zoom Sin Shear	0,8980	-1,38 %
Zoom Uniforme Sin Shear	0,9171	0,71 %
Zoom Uniforme	0,9213	1,18 %

Tabla 6.10: Resultado con distintas aumentaciones en el entrenamiento.

sirve para demostrar la viabilidad de alojar un modelo predictivo en un servidor.

El servidor utilizado usa una **CPU** `intel CORE i5` y es capaz de cargar el modelo en menos de 5 minutos. Una vez cargado, el modelo tiene una velocidad media de respuesta inferior a los 3 segundos, como se ve en la Figura 6.1. No obstante, debemos destacar que con una **GPU** se pueden reducir enormemente los tiempos de espera.

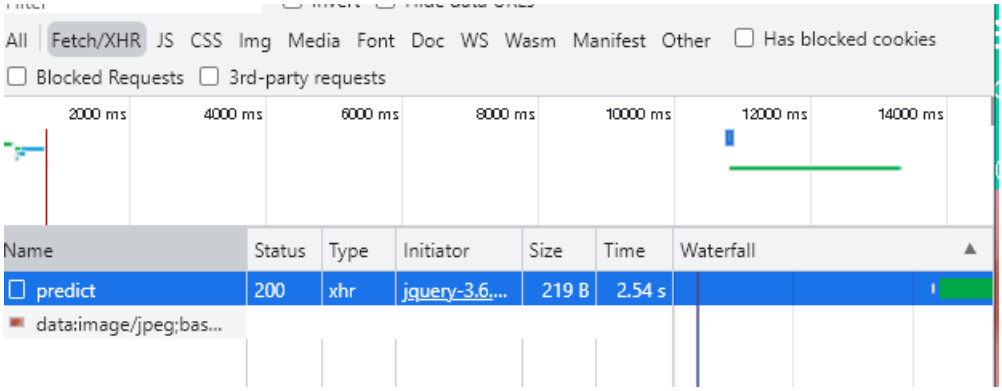

Figura 6.1: Captura del navegador que muestra el tiempo de espera de la petición.

7. CAPÍTULO

Conclusiones y Líneas Futuras

En este último capítulo se describen las conclusiones después de haber terminado el proyecto y las futuras líneas del mismo. Los objetivos se han cumplido en su totalidad, sobre todo los objetivos de aprendizaje. Gracias a la realización de este trabajo se ha profundizado en algunos campos del área del **AA** y se ha descubierto un apasionante mundo sin explorar. El resultado del modelo propuesto alcanza unos resultados muy prometedores que sugieren que, con más tiempo de dedicación al proyecto, pueden llegar a ser de gran utilidad.

Los principales caminos a seguir para continuar con este proyecto son:

- Mejorar la aplicación web: La aplicación existente está desarrollada en un entorno de pruebas no apto para la producción. No se han tenido en cuenta aspectos de ciberseguridad ni de protección de datos. Se pueden crear procesos automáticos que generen nuevos modelos con las imágenes aportadas tanto por los usuarios médicos como por los usuarios comunes y mostrar ambos resultados. Esto permitiría generar nuevos modelos y clasificarlos según su puntuación.
- Mejorar la fuente de datos: Sería interesante ponerse en contacto con hospitales para que aportaran imágenes.
- Mejorar el modelo: En este proyecto se han propuesto una serie de mejoras. Sería interesante combinarlas para mejorar el modelo así como añadir otras técnicas nuevas.

Glosario y Acrónimos

***k*-NN** método de los *k* vecinos más cercanos o *k-Nearest Neighbour*. 13

***k*-means** *k*-medias o *k-means*. 14

JPEG Formato de archivo de imágenes.. 32

TfRecord Formato de archivo para almacenar datos en `Tensorflow`. 32

AA aprendizaje automático o *Machine Learning*. 12, 13, 15, 18, 20, 60

ANN red neuronal artificial o *Artificial Neural network*. 15–17, 34, 35

AUC área bajo la curva o *Area Under Curve*. 53, 58

AUC-ROC área bajo la curva ROC o *Area Under ROC Curve*. 28

CNN red neuronal convolucional. 2, 18, 19, 25

CPU Unidad central de procesamiento o *Central Processing Unit)*. 59

DL aprendizaje profundo o *Deep Learning*. 2

FLOPS operaciones de coma flotante por segundo o *Floating point operations per second*. 39, 40

GPU Unidad de procesamiento gráfico o *Graphics Processing Unit)*. 34, 59

mapa de características o *Feature map* . 20, 25

melanocito El melanocito es una célula dendrítica que deriva de la cresta neural y que migra hacia la epidermis y el folículo piloso durante la embriogénesis. Su principal función es la producción de melanina que tiene importancia en la protección contra los rayos solares. 1

MVConv bloques residuales invertidos o *Inverted Residual Blocks*. 41

nevus melanocíticos Los nevus melanocíticos (NM), vulgarmente conocidos como lunares, pecas o nevus, son lesiones circunscritas del sistema melanocitario cutáneo y se definen como la proliferación benigna de melanocitos.. 1

queratinocito Los queratinocitos son las células predominantes ($\%80 - \%90$) de la epidermis, la capa más superficial de la piel. 1

separabilidad lineal Un problema es linealmente separable si en el espacio de patrones de entrada se puede encontrar un hiperplano que separe los patrones de una clase de los de la otra clase.. 17

TPU Unidad de procesamiento tensorial o *Tensor Processing Unit)*. 34, 35

TTA *Test Time Augmentation*. 28, 53, 55, 57

Bibliografía

[1] Alexander Buslaev, Vladimir I. Iglovikov, Eugene Khvedchenya, Alex Parinov, Mikhail Druzhinin, and Alexandr A. Kalinin. Albumentations: Fast and flexible image augmentations. *Information*, 11(2):125, Feb 2020.

[2] Fernando Sancho Caparrini. Introducción al aprendizaje automático, 2017.

[3] Travers Ching, Daniel S. Himmelstein, Brett K. Beaulieu-Jones, Alexandr A. Kalinin, Brian T. Do, Gregory P. Way, Enrico Ferrero, Paul-Michael Agapow, Michael Zietz, Michael M. Hoffman, Wei Xie, Gail L. Rosen, Benjamin J. Lengerich, Johnny Israeli, Jack Lanchantin, Stephen Woloszynek, Anne E. Carpenter, Avanti Shrikumar, Jinbo Xu, Evan M. Cofer, Christopher A. Lavender, Srinivas C. Turaga, Amr M. Alexandari, Zhiyong Lu, David J. Harris, Dave DeCaprio, Yanjun Qi, Anshul Kundaje, Yifan Peng, Laura K. Wiley, Marwin H. S. Segler, Simina M. Boca, S. Joshua Swamidass, Austin Huang, Anthony Gitter, and Casey S. Greene. Opportunities and obstacles for deep learning in biology and medicine. *Journal of The Royal Society Interface*, 15(141):20170387, 2018.

[4] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[5] Qishen Ha, Bo Liu, and Fuxu Liu. Identifying melanoma images using efficientnet ensemble: Winning solution to the siim-isic melanoma classification challenge. *ArXiv*, abs/2010.05351, 2020.

[6] Eibe Frank Ian H. Witten and Mark A. Hall. *Data Mining Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Publishers, Elsevier 30 Corporate Drive, Suite 400, Burlington, MA 01803, USA, 2011.

[7] Mª José Ramírez Quintana Cèsar Ferri Ramírez José Hernández Orallo. *Introducción a la minería de datos*. Pearson, 2004.

[8] Y LeCun, C Cortes, and CJC Burgess. The mnist database of handwritten images, 2012.

[9] Jayawant N. Mandrekar. Receiver operating characteristic curve in diagnostic test assessment. *Journal of Thoracic Oncology*, 5(9):1315–1316, September 2010.

[10] Quoc V. Le Mingxing Tan. Efficientnet: Rethinking model scaling for convolutional neural networks. *ArXiv*, 2019.

[11] M NirmalaDevi, S Appavu alias Balamurugan, and U. V Swathi. An amalgam knn to predict diabetes mellitus. In *2013 IEEE International Conference ON Emerging Trends in Computing, Communication and Nanotechnology (ICECCN)*, pages 691–695, 2013.

[12] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *CoRR*, abs/1511.08458, 2015.

[13] Alexander Rakhlin, Alexey Shvets, Vladimir Iglovikov, and Alexandr A. Kalinin. Deep convolutional neural networks for breast cancer histology image analysis. *Image Analysis and Recognition*, page 737–744, 2018.

[14] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. *ArXiv*, abs/1801.04381, 2019.

[15] Divya Shanmugam, Davis Blalock, Guha Balakrishnan, and John Guttag. When and Why Test-Time Augmentation Works. *arXiv e-prints*, page arXiv:2011.11156, November 2020.

[16] Rebecca L. Siegel, Kimberly D. Miller, and Ahmedin Jemal. Cancer statistics, 2019. *CA: A Cancer Journal for Clinicians*, 69(1):7–34, 2019.

[17] P.Y. Simard, D. Steinkraus, and J.C. Platt. Best practices for convolutional neural networks applied to visual document analysis. In *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.*, pages 958–963, 2003.

[18] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[19] Jianxin Wu. Introduction to convolutional neural networks. In *National Key Lab for Novel Software Technology. Nanjing University. China*, volume 5, 2017.

[20] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In Edwin R. Hancock Richard C. Wilson and William A. P. Smith, editors, *Proceedings of the British Machine Vision Conference (BMVC)*, pages 87.1–87.12. BMVA Press, September 2016.