ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

IMPLEMENTACIÓN DE UN SISTEMA DE DETECCIÓN TEMPRANA DE MELANOMA UTILIZANDO REDES NEURONALES CONVOLUCIONALES

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

RIVERA CHICAIZA DIEGO ALEXIS

diego.rivera01@epn.edu.ec

DIRECTOR: Ph.D. FELIPE LEONEL GRIJALVA ARÉVALO

felipe.grijalva@epn.edu.ec

CODIRECTOR: Ph.D. ROBIN GERARDO ÁLVAREZ RUEDA

robin.alvarez@epn.edu.ec

Quito, marzo 2020

AVAL

Certificamos que el presente trabajo fue desarrollado por Diego Alexis Rivera Chicaiza, bajo nuestra supervisión.

Ph.D. FELIPE LEONEL GRIJALVA ARÉVALO
DIRECTOR DEL TRABAJO DE TITULACIÓN

Ph.D. ROBIN GERARDO ÁLVAREZ RUEDA
CODIRECTOR DEL TRABAJO DE TITULACIÓN

DECLARACIÓN DE AUTORÍA

Yo, DIEGO ALEXIS RIVERA CHICAIZA, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración dejo constancia de que la Escuela Politécnica Nacional podrá hacer uso del presente trabajo según los términos estipulados en la Ley, Reglamentos y Normas vigentes.

DIEGO ALEXIS RIVERA CHICAIZA

DEDICATORIA

Este trabajo está dedicado a:

A mis padres, Julio César y Bélgica Azucena
por siempre haber creído en mí,
y haberme brindado todo el cariño y apoyo
que me permitieron lograr esta gran meta.

A mi segunda madre, mi abuelita María
que aunque no pudo estar a mi lado
espero enorgullecerla
desde donde sea que se encuentre.

Finalmente dedico este trabajo de titulación
a las personas que nunca me abandonaron,
y a las que ya no están junto a mí
pero una parte mía está con ellas.

Esto es por, y para ustedes.

AGRADECIMIENTO

Agradezco a mis padres, que gracias a su trabajo y preocupación, jamás permitieron que me faltaran los recursos necesarios para poder cumplir con todos mis objetivos durante todo este tiempo.

Al Ph.D. Felipe Grijalva por proporcionarme la guía correcta que me permitió desarrollar este proyecto de la mejor forma, sin mencionar los reconocimientos que me permitió alcanzar fuera de la universidad.

A la Escuela Politécnica Nacional por brindarme los conocimientos que dotan de prestigio a sus estudiantes, por enseñarme el valor de la paciencia y por forjar en mí el carácter necesario para desenvolverme en el mundo laboral.

Finalmente, agradezco a mis amigos, aquellos que siempre estuvieron conmigo y a los que hice mientras recorría el camino. Sin ellos, mi paso por las aulas de la Escuela Politécnica Nacional no hubiera sido igual.

Gracias por todo.

ÍNDICE DE CONTENIDO

AVAL I
DECLARACIÓN DE AUTORÍA II
DEDICATORIA III
AGRADECIMIENTO IV
ÍNDICE DE CONTENIDO V
RESUMEN VI
ABSTRACT VIII
1. INTRODUCCIÓN 1
1.1. OBJETIVOS 2
1.2. ALCANCE 3
1.3. MARCO TEÓRICO 3
1.3.1. EL MELANOMA 3
1.3.1.1 . Método ABCD de Stolz 3
1.3.1.2 . Método de los 7 punto de Argenziano 4
1.3.1.3 . Método de Menzies 4
1.3.2. EL PERCEPTRÓN 5
1.3.2.1 . Pesos sinápticos 6
1.3.2.2 . Función de activación 7
1.3.3. PERCEPTRONES MULTICAPA (MLP) 7
1.3.3.1 . Capa de entrada 8
1.3.3.2 . Capas ocultas 8
1.3.3.3 . Capa de salida 9
1.3.4. REDES NEURONALES CONVOLUCIONALES (CNN) 11
1.3.4.1 . Capa convolucional 11
1.3.4.2 . Convolución separable en profundidad (Depthwise separable convolution) 13
1.3.4.3 . Unidad Lineal Rectificada (ReLU) 16
1.3.4.4 . Capa de agrupamiento (Pooling) 18
1.4. TRANSFERENCIA DE APRENDIZAJE (TRANSFER LEARNING) 20

1.5. SOBREAJUSTE (OVER-FITTING) Y DROPOUT 22
1.5.1. OVER-FITTING 22
1.5.2. DROPOUT 23
1.6. HERRAMIENTAS DE IMPLEMENTACIÓN 24
1.6.1. LIBRERÍA DE APRENDIZAJE PROFUNDO TENSORFLOW 24
1.6.2. LIBRERÍA DE REDES NEURONALES KERAS 25
2. METODOLOGÍA 27
2.1. BASE DE DATOS 27
2.2. AUMENTO DE DATOS 30
2.3. TRANSFERENCIA DE APRENDIZAJE CON LA CNN MOBILENET 35
2.3.1. ALGORITMO DE OPTIMIZACIÓN 36
2.3.2. TASA DE APRENDIZAJE 37
2.3.3. FUNCIÓN DE PÉRDIDA 37
2.4. Experimentos 38
3. RESULTADOS Y DISCUSIÓN (APLICACIÓN METODOLÓGICA) 40
3.1. NOTEBOOK INTERACTIVO 44
4. CONCLUSIONES Y RECOMENDACIONES 47
4.1. CONCLUSIONES 47
4.2. RECOMENDACIONES 47
5. REFERENCIAS BIBLIOGRÁFICAS 49
ANEXOS 52

RESUMEN

La detección temprana del melanoma es crucial para evitar muertes por cáncer de piel, pero solo con los avances recientes de arquitecturas de redes neuronales convolucionales profundas, como MobileNet, es posible crear un sistema lo suficientemente confiable para detectar al melanoma, que se puede implementar en entornos con recursos limitados como teléfonos móviles o sistemas integrados.

Con este objetivo, este trabajo evalúa el desempeño de la implementación de un sistema de reconocimiento temprano de melanoma utilizando MobileNet entrenado con la base de datos HAM10000. Además, se explica en detalle dos estrategias para mejorar la tarea de clasificación del melanoma, es decir, el aumento de datos en un conjunto de datos desbalanceado y un enfoque multiclase para abordar un problema de clasificación binaria.

Los resultados numéricos en términos de métricas AUC y curvas ROC corroboran la validez de nuestro modelo. El rendimiento del modelo propuesto también se compara con el rendimiento del promedio dermatológico. Adicionalmente, se implementa una interfaz interactiva con el usuario capaz realizar predicciones en tiempo real.

Finalmente, en base a los resultados obtenidos de las métricas de rendimiento, se concluye que el modelo posee un buen rendimiento, y es capaz de predecir correctamente la afección en la gran mayoría de los casos de análisis.

PALABRAS CLAVE: red neuronal, aumento de datos, trasferencia de aprendizaje, validación cruzada, ROC, AUC.

ABSTRACT

Early detection of melanoma is crucial to avoid skin cancer deaths, but only with the recent advances of deep convolutional neural networks architectures, such as MobileNet, it is possible to create a reliable enough system to detect melanoma, that can be implemented on resource constrained environments such as mobile phones or embedded systems.

With this aim, this work assesses the performance of the implementation of an early melanoma recognition system using MobileNet trained from the HAM10000 database. Besides, we explain in detail two strategies to improve melanoma classification task, i.e., data augmentation on an unbalanced dataset and a multiclass approach to address a binary classification problem.

Numerical results in terms of AUC metric and ROC curves corroborate the validity of our model. The performance of the proposed model is also compared to the average dermatologist performance. Additionally, an interactive interface with the user capable of real-time predictions is implemented.

Finally, based on the results obtained from the performance metrics, we conclude that the model has a good performance, and it is able to correctly predict the condition in the vast majority of analysis cases.

KEYWORDS: neural network, data augmentation, transfer learning, cross validation, ROC, AUC.

1. INTRODUCCIÓN

El melanoma de la piel es una de las afecciones que está teniendo mayor incidencia a nivel mundial y en nuestro país está siendo diagnosticado cada vez con mayor frecuencia debido al alto índice de radiación ultravioleta que se ha presentado en los últimos años. Específicamente en la ciudad de Quito se ha registrado la tasa más alta de incidencia de cáncer de piel y se estima que esta tendencia vaya en aumento, pues en comparación con el último registro de tumores ocasionados debido a la presencia de afecciones de la piel, ha habido un incremento del 57 % en registros más actuales [1].

Tradicionalmente el proceso de diagnóstico del cáncer de piel consta de varios pasos a seguir. Este inicia con una evaluación clínica visual de cualquier tipo de irregularidades que se presentan de forma repentina en la piel que toma en cuenta factores como forma, color y tamaño, entre otros. Seguidamente se realiza un examen dermatoscópico para el cual es necesaria la utilización de instrumentos más sofisticados que permiten obtener imágenes con una resolución más detallada en comparación con una cámara fotográfica ordinaria (e.g. dermatoscopio) para la obtención y procesamiento de imágenes de la lesión y finalmente para obtener un resultado certero se recurre a una biopsia con su posterior análisis histopatológico que determinará la presencia o no de la enfermedad [2]. Los médicos recurren a varios métodos o algoritmos que introducen el análisis de los factores anteriormente mencionados tales como el método ABCD de Stolz, el método de los 7 puntos de Argenziano o el método de Menzies [3]. Estos métodos en cierta forma permiten cuantificar los factores y asignar una puntuación que indicará la posibilidad de existencia de melanoma, mas no ofrecerá un resultado totalmente certero. Además, estos métodos requieren en mayor o menor nivel un grado de experiencia por parte del especialista encargado del estudio sin mencionar que los análisis toman un tiempo considerable y recursos económicos que no siempre están al alcance del paciente.

A pesar de la experiencia que posee el médico, una clasificación visual siempre tiene un rango de error considerable, ya que diferentes condiciones de la piel pueden tener características similares, como se puede ver en la Figura 1.1.

Las redes neuronales convolucionales (CNN) en el análisis de imágenes médicas han demostrado un enorme potencial debido a la capacidad de encontrar patrones en una gran cantidad de datos.

La mayoría de los enfoques para el análisis de imágenes de dermatoscopía utilizan CNN para la clasificación del melanoma. Sin embargo, hay varios factores [4] que impactan en el desempeño de la tarea de clasificación del melanoma (por ejemplo, tamaño del conjunto de datos, arquitectura, etc.). Además, el mayor recurso público de imágenes, para probar y validar sistemas de diagnóstico automatizados para el melanoma, es el conjunto de datos de la *International Skin Imaging Collaboration (ISIC)*, que se creó para estandarizar las métricas de evaluación, de los diferentes enfoques que buscan mejorar la realización de la tarea de clasificación de melanoma, a través de su desafío anual. Pero entrenar todas las capas de una CNN profunda desde cero es inviable, debido al pequeño tamaño del conjunto de

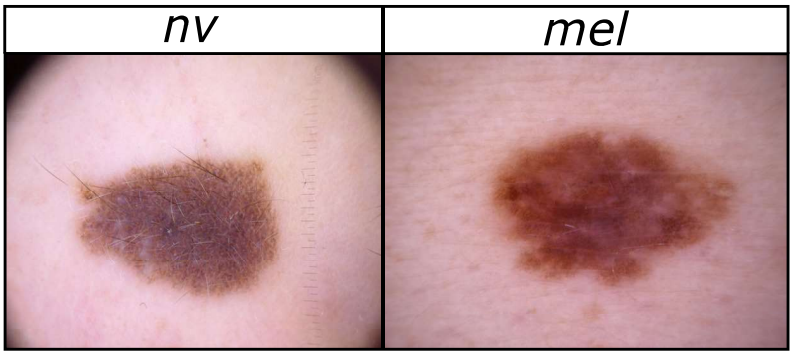


Figura 1.1: Comparación entre Nevos melanocíticos (nv) y melanoma (mel)

datos disponible públicamente (por ejemplo, conjunto de datos HAM10000 [5] con 10000 imágenes).
Por lo tanto, para hacer frente a esta restricción, los investigadores utilizan CNN pre-entrenadas [6] y estrategias de aumento de datos [7]. De hecho, cada año, el desafío de la ISIC muestra una diversificación progresiva de las CNN pre-entrenadas en ImageNet (un gran conjunto de datos con más de 14 millones de imágenes y más de 20,000 categorías). Por lo tanto, el rendimiento de las arquitecturas CNN en la tarea de clasificar el melanoma está fuertemente correlacionado con el rendimiento de las arquitecturas en ImageNet (revisar [8] y [9] para más discusión sobre esto).

Este trabajo fue aceptado para ser publicado y presentado en el *6th LatinAmerican Conference on Computational Intelligence LA-CCI* desarrollado en la ciudad de Guayaquil del 13 al 15 de noviembre del presente año. Adicionalmente, con este trabajo se obtuvo el 1er lugar en el concurso *2nd Contest of Student Papers in the EPN* organizado por el capítulo ecuatoriano de la *IEEE Computational Intelligence Society*.

1.1. OBJETIVOS

El objetivo general de este Proyecto Técnico es: Implementar un sistema de detección temprana de melanoma utilizando redes neuronales convolucionales.

Los objetivos específicos del Proyecto Técnico son:

- Estudiar los aspectos más importantes del melanoma y sus métodos de detección.
- Estudiar las redes neuronales convolucionales como extractores de características.
- Implementar un clasificador binario para detectar la presencia de melanoma.
- Determinar la métrica de rendimiento AUC (*Area Under the Curve*) con la ayuda de la curva ROC (*Receiver Operating Characteristic*) para el clasificador implementado.
- Implementar un *notebook* interactivo utilizando la herramienta *Jupyter Notebook* que permita visualizar el desempeño del clasificador.

1.2. ALCANCE

Para la realización de este proyecto se pretende desarrollar una interfaz interactiva con la ayuda de la herramienta *Jupyter Notebook*, capaz de evaluar imágenes de lesiones cutáneas. Un modelo implementado en la interfaz interactiva se encargará del procesamiento de la imagen y le mostrará al usuario una respuesta que le permita tomar las acciones que considere necesarias. El software no solo estará orientado hacia los pacientes, sino que podrá ser utilizado por dermatólogos que optimizarán su trabajo y obtendrán resultados de una forma más rápida y eficaz.

1.3. MARCO TEÓRICO

El campo de las redes neuronales es muy extenso, pues desde sus inicios han evolucionado de una forma impresionante. Para iniciar con esta sección se hará un breve análisis acerca del melanoma y los métodos que utilizan los dermatólogos para poder detectarlo de una forma visual. También se hará un repaso de todos los conceptos necesarios para comprender el funcionamiento de una red neuronal básica, hasta las principales características que diferencian a una red neuronal convolucional y finalmente se hablará acerca de las herramientas computacionales necesarias para llevar a cabo la implementación de dichas redes, con sus respectivas librerías.

1.3.1. EL MELANOMA

El melanoma es el tipo de cáncer de la piel más perjudicial. Este se genera en las células llamadas melanocitos, encargadas de producir la melanina que dota de pigmento a la piel.
El melanoma también puede presentarse en lugares no tan comunes como en los ojos, y hasta puede aparecer en los órganos internos como los intestinos. Estos son casos aislados, ya que se dan con muy poca frecuencia.
No se conoce a ciencia cierta la causa exacta de la aparición del melanoma, pero si se conoce que la exposición a los rayos ultravioleta que puede provenir de fuentes como el sol o camas bronceadoras puede aumentar la probabilidad de que se produzca el melanoma [10].

Existen métodos utilizados por dermatólogos que permiten obtener un diagnóstico relativamente rápido en comparación con métodos de verificación patológica. Se realizará una breve descripción de estos métodos. Cabe aclarar que la mayoría de estos métodos visuales se basa en la valoración de criterios, por lo que se cuantificarán los distintos criterios para poder y así se podrá decidir si la lesión es o no melanoma. Cada método tiene su propio umbral establecido.

1.3.1.1. Método ABCD de Stolz

Este es un método desarrollado por Stolz en el año 1994, y tiene como objetivo facilitar el análisis para los observadores con menos experiencia. Este método es poco útil si se quiere analizar mucosidades o piel con poca presencia de vello [3].

Tabla 1.1: Regla ABCD de Stolz [3]

Criterio	Puntuación	Factor de corrección
Asimetría	0 a 2	$\times 1.3$
Bordes	0 a 8	$\times 0.1$
Color	1 a 6	$\times 0.5$
Estructuras drmatoscópicas	1 a 5	$\times 0.5$
Índice dermatoscópico total		
<4.75 Benigna	4.8 - 5.45 Sospechosa	>5.45 >Maligna

El método de Stolz valora 4 criterios:

Asimetría: Se divide a la lesión en dos ejes y se puntúan asimetrías de distintos tipos con una puntuación que va de 0 a 2.

Bordes: La lesión se divide en 8 segmentos y se le da un valor de 1 a cada segmento que presente variación abrupta de color en su finalización, dando valores que van de 0 a 8.

Color: Se pueden presentar 6 diferentes colores y se valora con 1 a cada color que se presente. Se tienen valores en un rango de 1 a 6.

Estructuras dermatoscópicas: Las estructuras dermatoscópicas son diferentes particularidades en forma y color presentes en la lesión. Se pueden tener 5 diferentes y la valoración tiene un rango de 1 a 5.
Para obtener el valor del índice dermatoscópico total se debe multiplicar la valoración obtenida para cada criterio por un factor de corrección específico y se suman los valores. El proceso se encuentra resumido en la Tabla 1.1 [3].

1.3.1.2. Método de los 7 punto de Argenziano

Este método fue creado por Argenziano en el año 1998, y toma en cuenta 7 diferentes criterios, estos se pueden clasificar en criterios mayores y menores.

Criterios mayores: Se trata de tres criterios que reciben una mayor puntuación debido a las irregularidades que presentan tanto en forma y color. Reciben una valoración máxima de 2.

Criterios menores: Son cuatro criterios que presentan una menor relevancia, presentando irregularidades mayormente en cuanto a color. Reciben una valoración máxima de 1.
En la Tabla 1.2 se especifican los criterios mayores y menores, junto con el umbral que debe alcanzar la lesión para ser diagnosticada como maligna o benigna [3].

1.3.1.3. Método de Menzies

Este método fue desarrollado por Menzies en el año 1996 y toma en cuenta 11 criterios que se dividen en criterios positivos y negativos.

Criterios negativos: Son dos criterios que de no estar presentes, puede diagnosticarse a la lesión como melanoma.

Tabla 1.2: Método de los 7 puntos de Argenziano [3]

Criterio	Puntuación
CRITERIOS MAYORES	
Retículo pigmentado atípico	2
Velo azul-blanquecino	2
Patrón vascular atípico	2
CRITERIOS MENORES	
Proyecciones irregulares	1
Puntos/glóbulos irregulares	1
Manchas de pigmento irregulares	1
Estructuras asociadas a regresión	1
PUNTUACIÓN TOTAL	
<3 Benigna	$\geq$3 Maligna

Tabla 1.3: Método de Menzies [3]

Criterios Negativos (No deben estar presentes)
Simetría
Monocromía
Criterios Postivos (Al menos uno debe estar presente)
Velo Azul-blanquecino
Despigmentación pseudocicatricial
Múltiples colores (5 a 6)
Retículo pigmentado prominente
Pseudópodos
Proyecciones radiales
Múltiples puntos marrones
Puntos/glóbulos negros periféricos
Múltiples puntos azul-gris

Criterios positivos: Son los 9 criterios restantes, que de estar presentes en la lesión, se la puede diagnosticar como una lesión maligna. Es necesaria la presencia de un único criterio positivo para el diagnóstico.

En la Tabla 1.3 se mencionan los 11 criterios y se puede notar que este método no es cuantitativo como los otros dos mencionados, simplemente se basa en la presencia o ausencia de los criterios.

A pesar de que estos métodos ofrecen un diagnóstico rápido, no ofrecen una confiabilidad óptima. La implementación de las redes neuronales ha permitido disminuir tanto el tiempo de diagnóstico como la precisión con que se la realiza [3].

1.3.2. EL PERCEPTRÓN

El perceptrón, también conocido como neurona, es en realidad el precursor de las redes neuronales que se utilizan hoy en día y se basa en el funcionamiento del cerebro biológico, es decir, es capaz de resolver problemas de distinta índole por medio del aprendizaje. El desarrollo del perceptrón fue de gran utilidad para la resolución de problemas complejos que

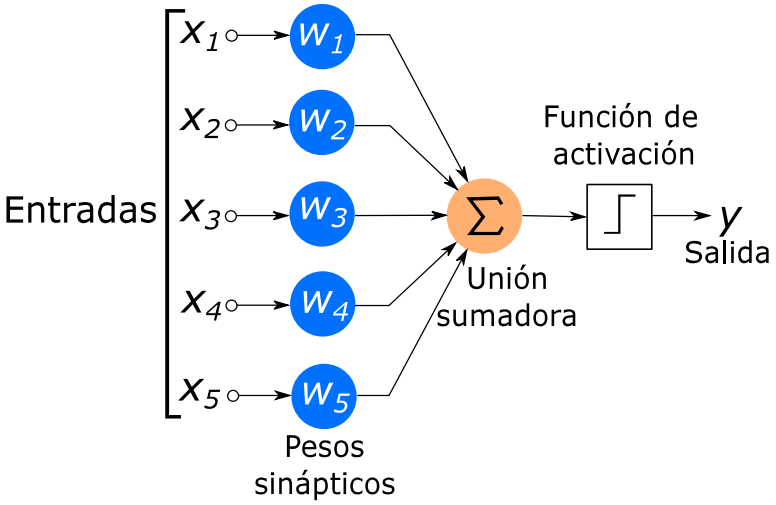


Figura 1.2: Arquitectura básica del perceptrón [13]

requerían de modelos capaces de llevar a cabo la clasificación entre distintas categorías y modelos predictivos.

Un perceptrón por sí mismo no es capaz de realizar tareas sumamente complejas. Si se habla de tareas de clasificación, un sólo perceptrón únicamente será capaz de discriminar dos clases, lo que se conoce como patrones *linealmente separables*.

Los patrones linealmente separables están está ligados al concepto de **hiperplano**. Un hiperplano no es más que un concepto generalizado de la función que cumple un plano en un ambiente tridimensional, es decir, separa el espacio en dos partes. Este concepto puede ser aplicado a rectas, planos, espacios, etc [11].

El objetivo de la creación del perceptrón no fue desarrollar un sistema que simule al cerebro humano, sino implementar estructuras y algoritmos (*teorema de convergencia del perceptrón*) que permitan desempeñar funciones de aprendizaje similares a las del cerebro [12].

El trabajo que una neurona realiza en términos generales no es complejo. Se tiene un conjunto de datos de entrada que en la Figura 1.2 se representan desde X_1 hasta X_5 a las cuales se asocian pesos sinápticos representados desde W_1 hasta W_5 y con la ayuda de una función de activación se obtiene la salida deseada. Como los pesos sinápticos influyen en gran medida en la salida obtenida, estos varían constantemente en los procesos de entrenamiento de modelos más complejos para aumentar la confiabilidad en el resultado final [13].

Ahora es necesario describir dos elementos fundamentales que presenta una neurona simple, y estos son: Pesos sinápticos y función de activación.

1.3.2.1. Pesos sinápticos

Para poder explicar de una manera sencilla la función de estos pesos se puede llevar a cabo una analogía con una neurona biológica. Como se puede apreciar en la Figura 1.3, las neuronas comparten información por medio del proceso denominado *sinapsis* durante el cual las neuronas de hecho tienen una separación mínima en el orden de las micras.

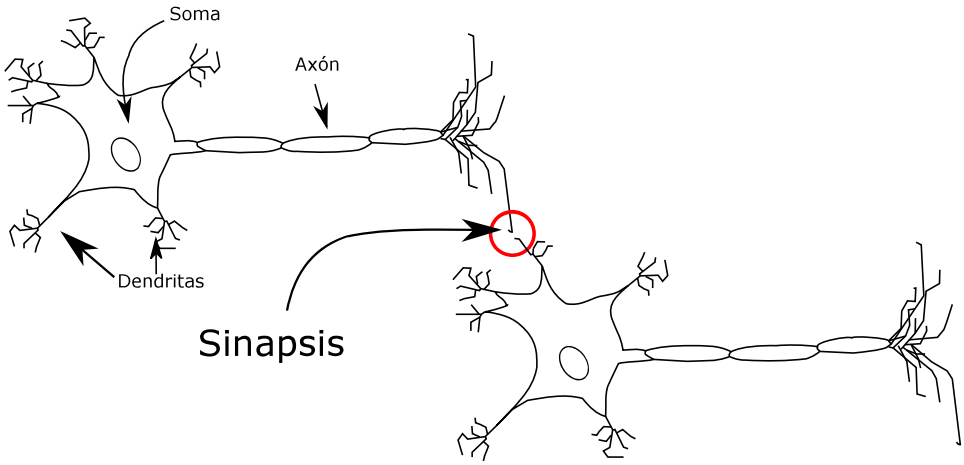


Figura 1.3: Representación de la sinapsis entre neuronas [14]

A la neurona de la izquierda se la denomina *neurona presináptica* y a la de la derecha se la denomina *neurona postsináptica*. El peso sináptico es la representación del nivel de intensidad con que estas neuronas realizan diferentes interacciones entre ellas en el caso de referirnos a dos perceptrones o nodos de la red neuronal [14].

1.3.2.2. Función de activación

La función de activación en cambio es la encargada de la activación de la neurona, es decir, maneja un umbral que determina el valor en la salida de la esta. En el caso de que se esté llevando a cabo un proceso de clasificación binaria, suponiendo un umbral de 0.5, si la sumatoria de la información de entrada de la neurona supera este valor umbral, entonces la salida será un 1, de lo contrario será 0. Existen diferentes tipos de funciones de activación, entre las más conocidas se encuentran la rectificadora, la sigmoide (clasificación binaria) y la softmax (clasificación multiclase) [15].
Más adelante se describirán las funciones de activación que fueron relevantes para este proyecto.

1.3.3. PERCEPTRONES MULTICAPA (MLP)

Como se mencionó en la sección anterior, un sólo perceptrón no es capaz de realizar tareas con un alto grado de complejidad e incluso existen métodos estadísticos que podrían realizar las tareas que realiza un perceptrón de una manera más eficiente. Es a partir de la necesidad de implementar modelos capaces de realizar labores de clasificación multiclase y modelos predictivos mucho más precisos que aparecen los perceptrones multicapa o más conocidos como redes neuronales en la actualidad.
La Figura 1.4 representa de una manera muy simplificada la arquitectura de una red neuronal genérica. Cada uno de los círculos de color representa una neurona, y a la disposición de neuronas en forma de columnas se la llamará capa de ahora en adelante. Se puede distinguir muy claramente que consta de tres elementos básicos: Una capa de entrada, las capas ocultas y la capa de salida [13].

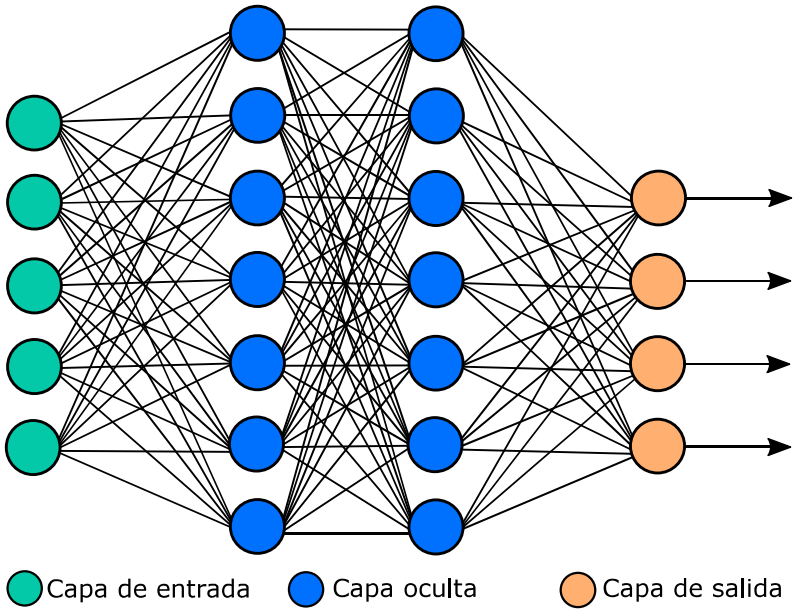


Figura 1.4: Arquitectura básica de un perceptrón multicapa [13]

1.3.3.1. Capa de entrada

En la capa de entrada se encuentran presentes los datos que se utilizarán para llevar a cabo todo el proceso de entrenamiento de la red. Podría decirse que esta es una capa visible de la red neuronal, pero es importante aclarar que esta no es una capa en sí, debido a que en esta capa no se realiza procesamiento de los datos de entrada por parte de las neuronas, sino que sólo se cumple con la función de propagar los valores de las entradas a las siguientes capas de la red [15].

1.3.3.2. Capas ocultas

Es bastante simple comprender el por qué del nombre de estas capas considerando el esquema de la Figura 1.4, pues ni el usuario ni los datos de entrada están en contacto directo con estas. El número de capas ocultas dependerá del nivel de complejidad que se quiera implementar en nuestro modelo. Cabe recalcar que mientras más capas ocultas se implementen en la red, mejor será el resultado obtenido en la capa de salida, ya sea en tareas de clasificación, así como en tareas predictivas. En sus primeras etapas, las redes neuronales constaban de muy pocas capas ocultas que se encargaban de realizar el cómputo de la información, incluso se podría tener un capa oculta que conste de una sola neurona. Estas limitaciones para poder implementar redes neuronales más complejas iban de la mano con las limitaciones de hardware de aquellas épocas, sin embargo, el poder de procesamiento en la actualidad, debido al enorme avance que ha habido en cuanto al desarrollo de hardware dio paso a la creación de redes con un gran número de capas ocultas. Una sola máquina hoy en día es capaz de llevar a cabo millones de procesos en cuestión de minutos e incluso es posible agrupar un conjunto de máquinas y utilizar sus recursos como una unidad para aumentar el poder de procesamiento (clúster). De aquí nace el concepto de ***deep learning (aprendizaje profundo)***, pues mientras más capas ocultas tenga nuestra red, se dice que

esta es más profunda y el proceso de aprendizaje será mucho más confiable y eficiente [15].

1.3.3.3. Capa de salida

En realidad esta capa es la última de las capas ocultas pero cumple con una función más específica, que es la de emitir un resultado final que vaya acorde al propósito par el cual fue creado el modelo, es decir, puede existir un único valor de salida si se trata de un propósito predictivo o varios valores en l salida en el caso de un problema de clasificación multiclase. Los escenarios más comunes en la capa de salida de una red neuronal pueden ser los siguientes:

- Un caso predictivo, también conocido como uno de regresión, por lo general suele contar con una neurona en su capa de salida y puede o no tener un función de activación.
- Cunado se trata de un caso de clasificación binaria, por lo general se tiene una neurona en la capa de salida utilizando una función de activación sigmoide, que ha demostrado ser óptima para casos de clasificación binaria.
- Si se trata de un caso de clasificación multiclase, el número de neuronas en la capa de salida es variable, pues se tienen tantas neuronas como posibles categorías de clasificación. Generalmente, la función de activación para estos casos es la softmax [15].

La *función de activación softmax* es clave en la salida de una red neuronal encargada de llevar a cabo una clasificación multiclase como la que se implementó.

Función de activción softmax: La última capa de una red neuronal es la encargada de entregar la salida esperada, es decir, después del procesamiento de la información de entrada se obtienen los parámetros necesarios para que la función de activación pueda realizar su trabajo. La función *softmax* al igual que cualquier función, cuenta con una ecuación característica:

$$\frac{e^{Y_j}}{\sum_{k=1}^{K} e^{Y_k}} \tag{1.1}$$

En la Ecuación 1.1 la variable Y representa las salidas que se obtienen a partir del procesamiento en las capas ocultas, mientras que la variable K representa el número de clases presentes en nuestro modelo.
Una vez que la información de entrada de la red neuronal ha sido procesada por las capas ocultas se obtiene una salida, que será la información de entrada de la capa de *softmax*, cuyo número de neuronas dependerá del número de clases presentes. Como se puede ver en la Figura 1.5, la salidas de las capas ocultas ingresan a la capa de *softmax*, en donde esta finalmente nos entregará las respectivas probabilidades de que un dato de entrada pertenezca a cada una de las clases. Evidentemente, la categoría con el porcentaje más alto, será la categoría a la que el objeto de análisis tendrá una mayor probabilidad de pertenecer.

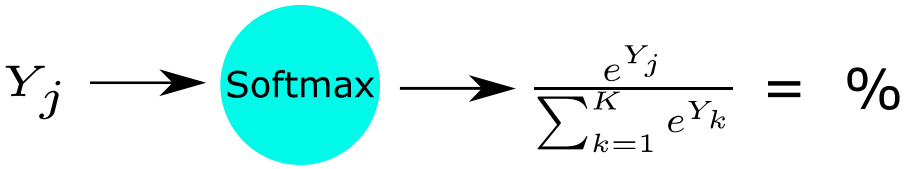


Figura 1.5: Principio de funcionamiento de softmax

Bajo estas premisas se puede implementar una red neuronal óptima capaz de cumplir nuestros requerimientos, sin embargo, la implementación de una red neuronal es sólo el primer paso en la serie de procesos que implican desarrollar todo un modelo capaz de realizar predicciones confiables
Se tomará en consideración aspectos como *datos de entrada* y el *algoritmo de descenso de gradiente estocástico*, el cual guarda una estrecha relación con la *tasa de aprendizaje*.

Datos de entrada: Existe una gran variedad de datos que pueden ser procesados por una red neuronal, sin embargo, es importante saber que hay distintos tipos de redes y unas pueden ser más apropiadas que otras dependiendo del tipo de datos que se utilizarán. En una red neuronal MLP o *fully connected* por lo general se utilizan datos numéricos, que bien pueden ser importados desde distintas fuentes o incluso pueden ser generados si se lo requiere. Estos datos requieren ser normalizados a valores que van de 0 a 1, y si se trata de cadenas de caracteres pueden ser codificados a valores numéricos. Más adelante se analizarán las redes neuronales convolucionales, que son ideales para trabajar con imágenes como datos de entrada [15].

Algoritmo de descenso de gradiente estocástico (SGD): Este tal vez es el algoritmo de optimización para redes neuronales más conocido y comúnmente utilizado hasta hace pocos años atrás, pues se considera como la base para algoritmos de optimización más complejos que existen en la actualidad. El objetivo en la implementación de este algoritmo de optimización en una red neuronal es simple pero imprescindible, mejorar la confiabilidad en las predicciones en base al entrenamiento. Para lograr este objetivo al algoritmo se basa en el cálculo del error. Una vez que los datos de entrada pasan por toda la red hasta la última capa y se ha definido cuál es la salida, esta se compara con la salida esperada, obteniendo así una medida del error que la red tiene hasta el momento. Mediante el llamado *algoritmo de retropropagación (backpropagation)*, el error obtenido se propaga a través de todas las capas de la red para que los pesos puedan ser actualizados en función del impacto que tuvieron sobre dicho error y disminuirlo en la siguiente iteración. Como este algoritmo es aplicado a cada uno de los datos de entrenamiento, una vez que todos los datos han concluido el proceso se habrá completado un ciclo del entrenamiento y este recibe el nombre de época (*epoch*). El número de épocas en una red neuronal es configurable al momento de implementarla y dependerá de la aplicación, la confiabilidad deseada y la tasa de aprendizaje [15].

Tasa de aprendizaje: La tasa de aprendizaje es de los parámetros más importantes que se deben configurar y está fuertemente ligado al algoritmo de optimización. Esta tasa toma valores positivos muy pequeños que casi siempre se encuentran en el rango de 0.0 hasta 1.0. Se la puede definir como la intensidad con la que varían los pesos al momento de ser actualizados mediante el algoritmo de retropropagación. Si el valor que toma la tasa de aprendizaje es relativamente alto, la variación que sufrirá el peso será más abrupta, mientras que cuando este valor es pequeño la variación, por el contrario será leve.
Una buena práctica de entrenamiento de una red neuronal es el continuo monitoreo del progreso en la confiabilidad de la red con respecto al avance a las épocas de entrenamiento. Teniendo en cuenta el parámetro de la confiabilidad se debe implementar una disminución dinámica en la tasa de aprendizaje. El motivo de la implementación de esta técnica es un mejor ajuste de los pesos, se fija un valor adecuado en la tasa de aprendizaje y este irá disminuyendo siempre y cuando la confiabilidad de la red no sufra mayores cambios, de esta manera se consigue obtener actualizaciones significativas en los pesos al inicio del entrenamiento y modulaciones más leves y precisas posteriormente. Así se logra una eficiencia óptima en el proceso de entrenamiento y se evita que la red tarde un tiempo excesivo en ser entrenada o que tenga una baja eficiencia debido tiempos de entrenamiento demasiado cortos [15].

Sin duda las redes neuronales incorporan un sinnúmero de beneficios al agrupar una gran cantidad de perceptrones capaces de realizar tareas mucho más complejas. Sin embargo, existen redes neuronales que fueron desarrolladas para lograr una mayor eficiencia al resolver problemas más específicos. Uno de estos casos es el de las ***redes neuronales convolucionales***, que son un tipo de redes neuronales a las cuales se les añade capas específicas ideales para el procesamiento de imágenes.

1.3.4. REDES NEURONALES CONVOLUCIONALES (CNN)

Las redes neuronales convolucionales parten del concepto de las redes MLP o perceptrones multicapa, pero nacen de la necesidad de automatizar el procesamiento de imágenes a partir de la implementación de capas dedicadas. En la Figura 1.6 se puede encontrar una arquitectura genérica de las redes neuronales convolucionales, y se nota la presencia de distintas capas que no se encontraban en las MLP.
A simple vista se puede notar al menos dos capas de las cuales no se ha hablado, estas son las de *convolución* y la de *pooling*. Adicionalmente se puede ver una capa de *ReLU*, que usualmente trabaja en conjunto con la capa de convolución. Al final de la arquitectura se nota la estructura de una red neuronal MLP para realizar una clasificación multiclase (si se lo requiere) con la ayuda de una capa de *softmax* como se analizó en la Sección 1.3.3.

1.3.4.1. Capa convolucional

Esta es la capa que caracteriza este tipo de redes, pues es la encargada de cuantificar a las imágenes para que puedan ser representadas como arreglos de datos de distintas

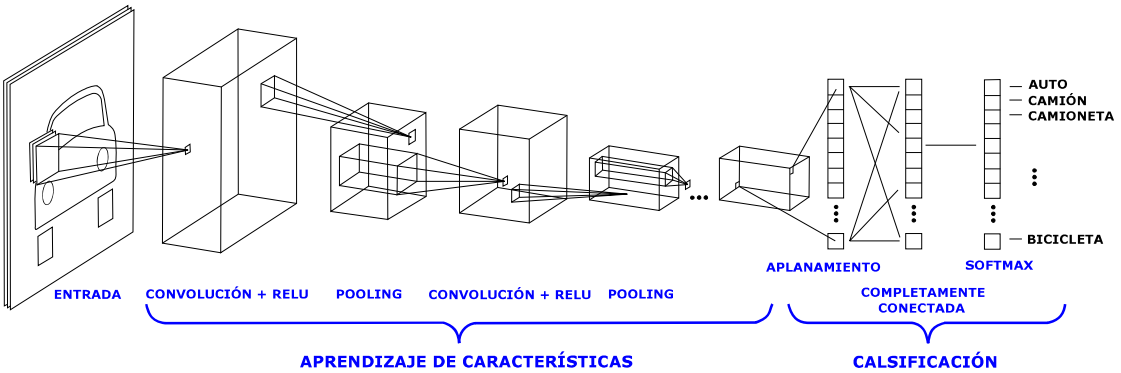


Figura 1.6: Arquitectura básica de una CNN [13]

dimensiones. Suponiendo que se tiene como entrada una imagen en blanco y negro y los valores de sus respectivos píxeles se encuentran representados en la Figura 1.7. Se tiene un filtro de dimensiones específicas, que para facilidad de comprensión también tiene valores entre 0 y 1 como el de la imagen [16]. El proceso de convolución consta de un procedimiento en el cual se realiza el producto punto entre la sección de la imagen que cubre el filtro y dicho filtro, obteniendo de este proceso un solo valor escalar. Este filtro repite el mismo proceso a lo largo de toda la imagen, moviéndose un determinado número de pasos a la vez. El número de pasos que se mueve el filtro es conocido como *stride* y se encuentra representado en la Figura 1.8 con un valor de 1 [13].

Se vio como el proceso de convolución es capaz de conformar el mapa de características de la imagen asumiendo que se utilizó simplemente un imagen en blanco y negro, y que la red neuronal que se utilizó consta de un solo filtro, pero en la práctica este sería un caso muy aislado. La realidad es que casi la totalidad de imágenes que requieren ser analizadas en una red neuronal cuentan con más de un único canal de entrada (RGB), es decir, son imágenes a color.

En la Figura 1.9 se puede observar el proceso de extracción del mapa de características de una caso más práctico. Es una variación del proceso que se describió para realizar la extracción del mapa para una imagen a blanco y negro, pero en este caso se tiene una imagen multicanal (RGB), lo que supone que es a color. Además de esta variación, se añade un filtro adicional, con lo que se tiene una representación más práctica de lo que se puede encontrar en la realidad. Generalmente, las implementaciones de redes neuronales constan de al menos 64 a 128 filtros, y cuando se trata de transferencia de aprendizaje, las redes neuronales pre-entrenadas tienen muchos más filtros [16].

También se puede notar que se han introducido filtros con mayor dimensionalidad, en este caso en particular se trata de una convolución estándar. Como la profundidad del filtro es la misma que el número de canales de la imagen de entrada, después de un proceso de combinación, se obtiene finalmente una reducción en a profundidad para cada filtro aplicado, y la profundidad del mapa de características dependerá del número de filtros que se decida implementar o el de la red neuronal que se utilice.

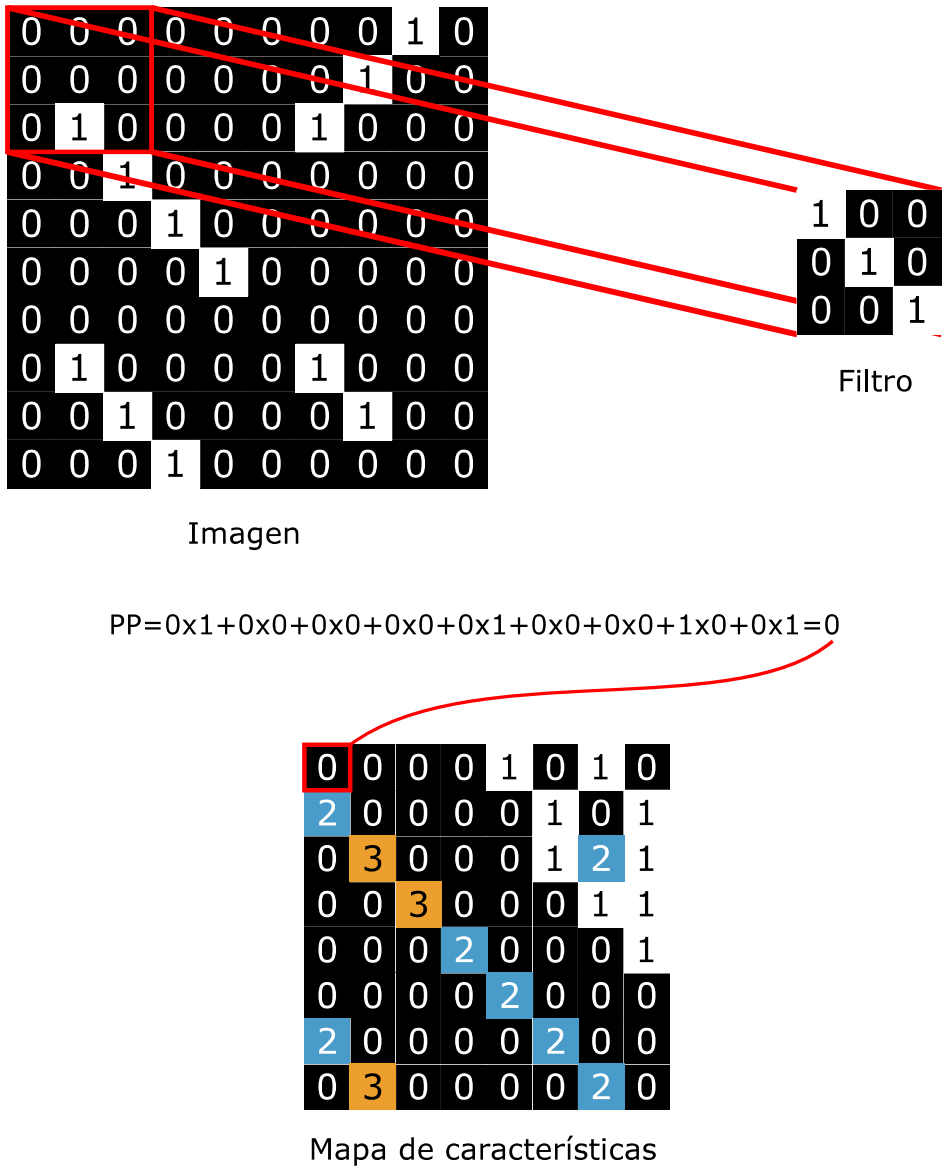


Figura 1.7: Extracción del mapa de características [16]

1.3.4.2. Convolución separable en profundidad (Depthwise separable convolution)

La convolución separable en profundidad o *depthwise separable convolution* es una variación de la convolución estándar. La *depthwise separable convolution* ofrece una ventaja clara frente a la convolución estándar, y es en cuanto a eficiencia computacional. Para esto se puede analizar el ejemplo que se plantea en la Figura 1.10.

En el ejemplo se tiene una entrada de dimensiones $7 \times 7 \times 3$, a la cual se le aplica un filtro de dimensiones $3 \times 3 \times 3$. Este filtro es más conocido en el proceso de convolución como *kernel*. Mediante el proceso analizado en la Sección 1.3.4.1 se obtiene una salida de dimensiones $5 \times 5 \times 1$, suponiendo que la red neuronal utilizada tenga un total de 128 filtros la dimensión final de la salida obtenida será de $5 \times 5 \times 128$.

Con estos datos es posible calcular el costo computacional del proceso de una convolución

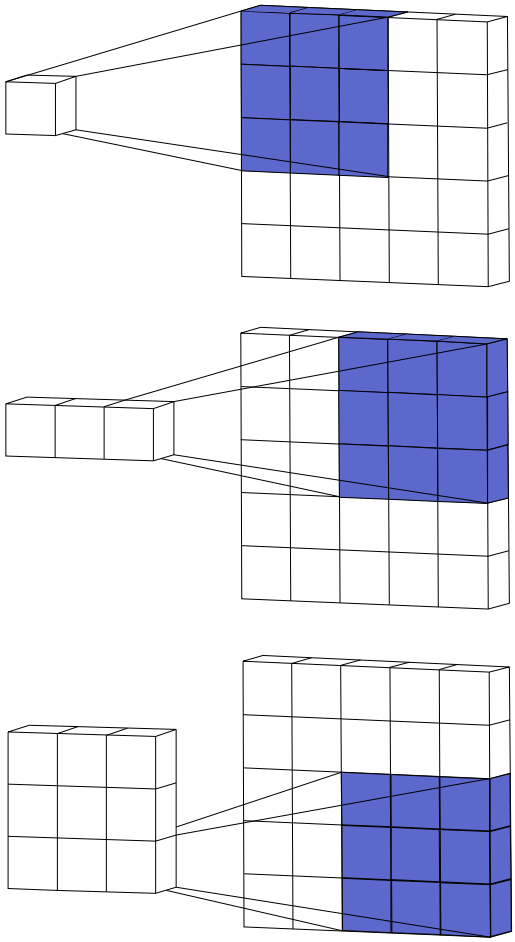


Figura 1.8: Representación del stride [13]

estándar. Se sabe que se utilizaron 128 *kernels* de dimensiones $3 \times 3 \times 3$ y que cada uno de estos, debido a que el tamaño del *stride* normalmente es de 1, cada kernel se desplazó a lo largo de la entrada 5×5 veces. El costo computacional es el resultado de multiplicar $128 \times 3 \times 3 \times 3 \times 5 \times 5$, dando un total de 86400 operaciones en todo el proceso.
Ahora que se conoce el proceso que se realiza al aplicar una convolución estándar y además el costo computacional que este tipo de convolución genera, es momento de analizar el proceso que se lleva a cabo al aplicar una *depthwise separable convolution*. Para esto se puede prestar atención a la Figura 1.11.

La información de entrada es la misma que se utiliza para analizar el caso de la convolución estándar. Para este caso, el proceso se divide en dos etapas conocidas como *depthwise convolution* y *pointwise convolution*. En la primera etapa (*depthwise convolution*) se aplica un kernel de dimensiones $3 \times 3 \times 1$ a cada uno de los canales de la entrada, que en este caso son tres. De aquí se obtiene una primera salida de dimensiones $5 \times 5 \times 3$.
En la segunda etapa (*pointwise convolution*) se aplica un solo kernel, que en este caso será de dimensiones $1 \times 1 \times 3$, para obtener la misma salida de dimensiones $5 \times 5 \times 1$ que se obtuvo con la convolución estándar, y de igual forma se asume que la CNN cuenta con 128 filtros para finalmente tener una salida de dimensiones $5 \times 5 \times 128$.

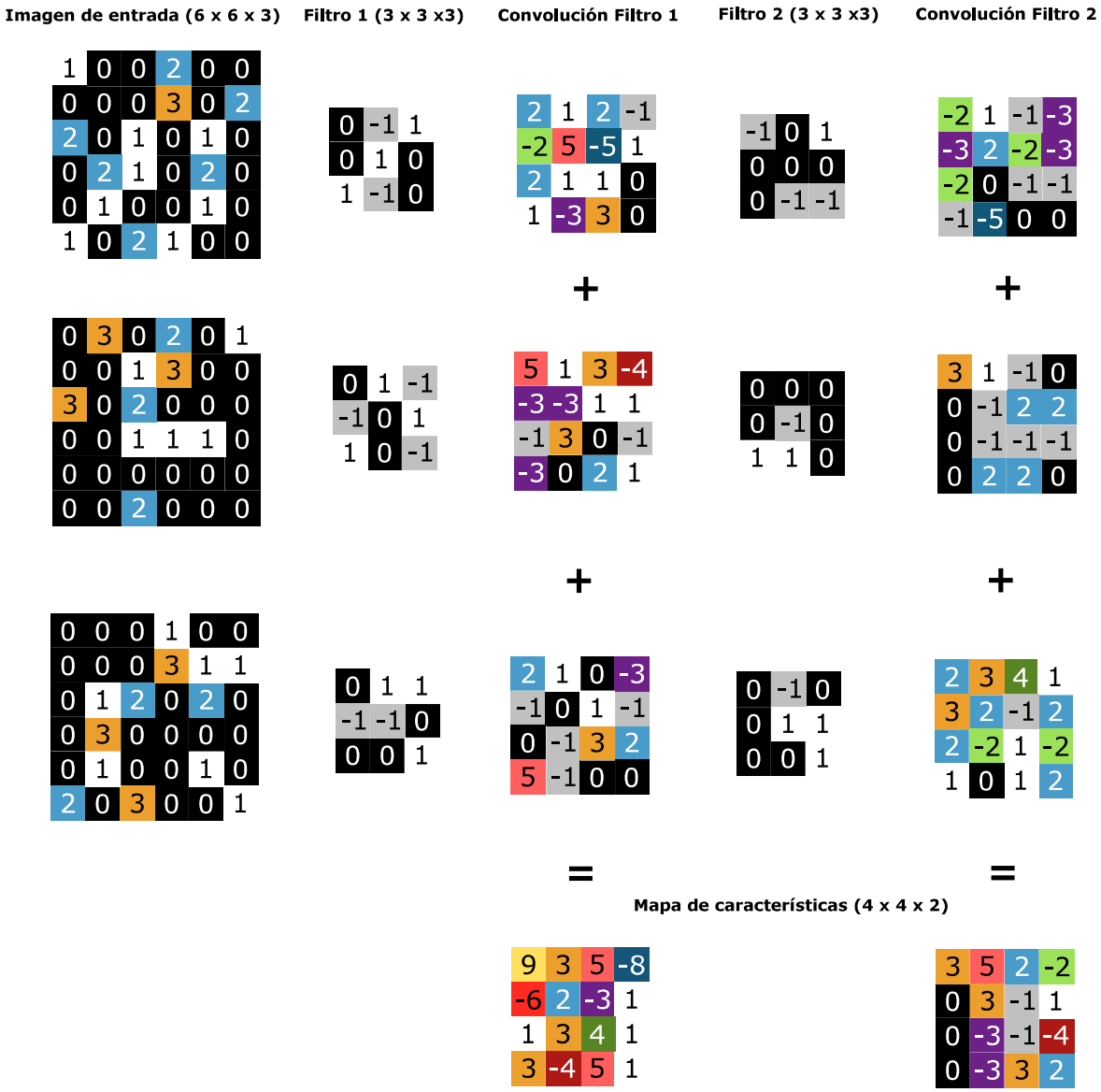


Figura 1.9: Convolución de imagen multicanal [16]

El resultado final parece ser el mismo, pero si se analiza ahora el costo computacional se nota una gran diferencia. El costo computacional de la etapa de *depthwise convolution* será de $3 \times 3 \times 3 \times 1 \times 5 \times 5$, dando un total de 675 operaciones. En la segunda etapa el costo computacional será de $128 \times 1 \times 1 \times 3 \times 5 \times 5$, con un total de 9600 operaciones. Si se calcula el número total de operaciones para todo el proceso de *depthwise separable convolution*, esto suma un total de 10275 operaciones, que es un número evidentemente más bajo que el obtenido en el proceso de la convolución estándar [16].
Se puede generalizar el costo computacional de ambos procesos de la siguiente forma:

El costo computacional para la convolución estándar (CCCE) se representa como:

$$\text{CCCE} = Nc \times h \times h \times D \times (H - h + 1) \times (W - h + 1) \tag{1.2}$$

El costo computacional para la *depthwise separable convolution* se representa como:

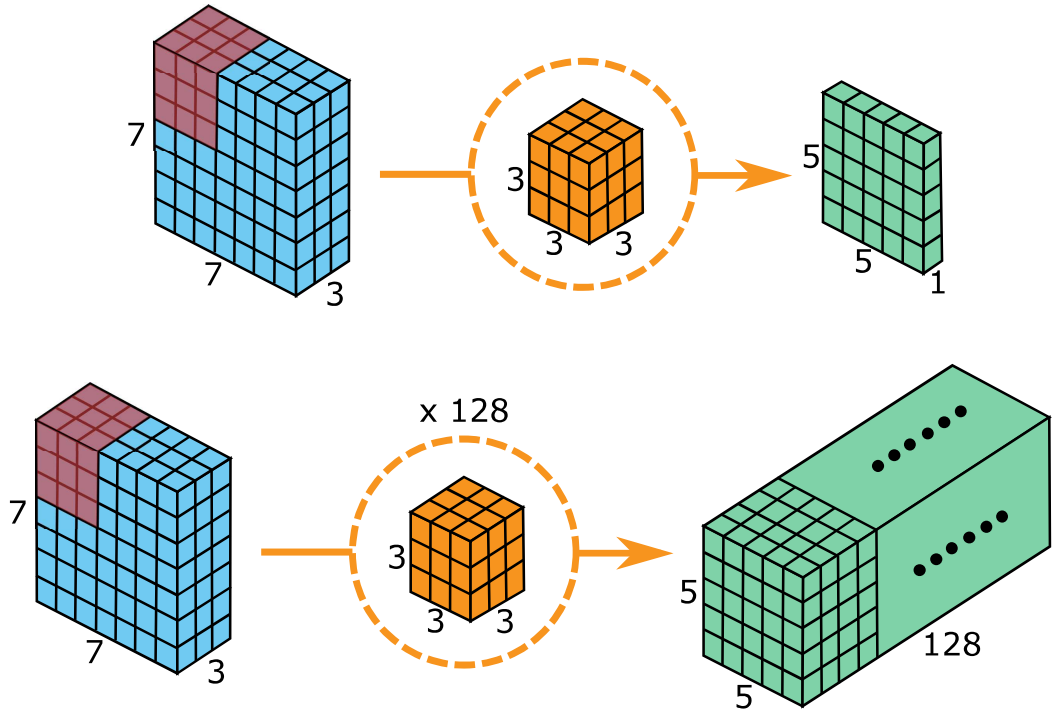


Figura 1.10: Convolución estándar [16]

$$\text{CCDSC} = (h \times h + Nc) \times D \times (H - h + 1) \times (W - h + 1) \tag{1.3}$$

En donde H y W representan las dimensiones de la entrada, mientras que h representa las dimensiones del kernel, suponiendo que este es cuadrado. El factor D representa la profundidad tanto de la entrada, como del kernel y Nc representa el número de *kernels*.

El resultado de la relación entre el costo computacional de la *depthwise separable convolution* y la convolución estándar lo se lo puede ver en la Ecuación 1.4.

$$\frac{\text{CCDSC}}{\text{CCCE}} = \frac{1}{Nc} + \frac{1}{h^2} \tag{1.4}$$

Si se considera que generalmente Nc es mucho mayor que h, finalmente el primer factor de la relación es despreciable y, tomando en cuenta el ejemplo analizada, con un valor de h igual a 3, entonces se puede deducir que el costo computacional de una convolución estándar es aproximadamente 9 veces mayor que el de una *depthwise separable convolution*.

Esta premisa es particularmente importante debido a que MobileNet incorpora la *depthwise separable convolution*, red utilizada para llevar a cabo este proyecto, y que demandó un menor consumo de recursos computacionales [16].

1.3.4.3. Unidad Lineal Rectificada (ReLU)

La capa de *ReLU* cumple con varias funciones que ayudan a mejorar en gran medida la eficiencia con la que la red se desenvuelve. En la Sección 1.3.4.1 se analizó el proceso

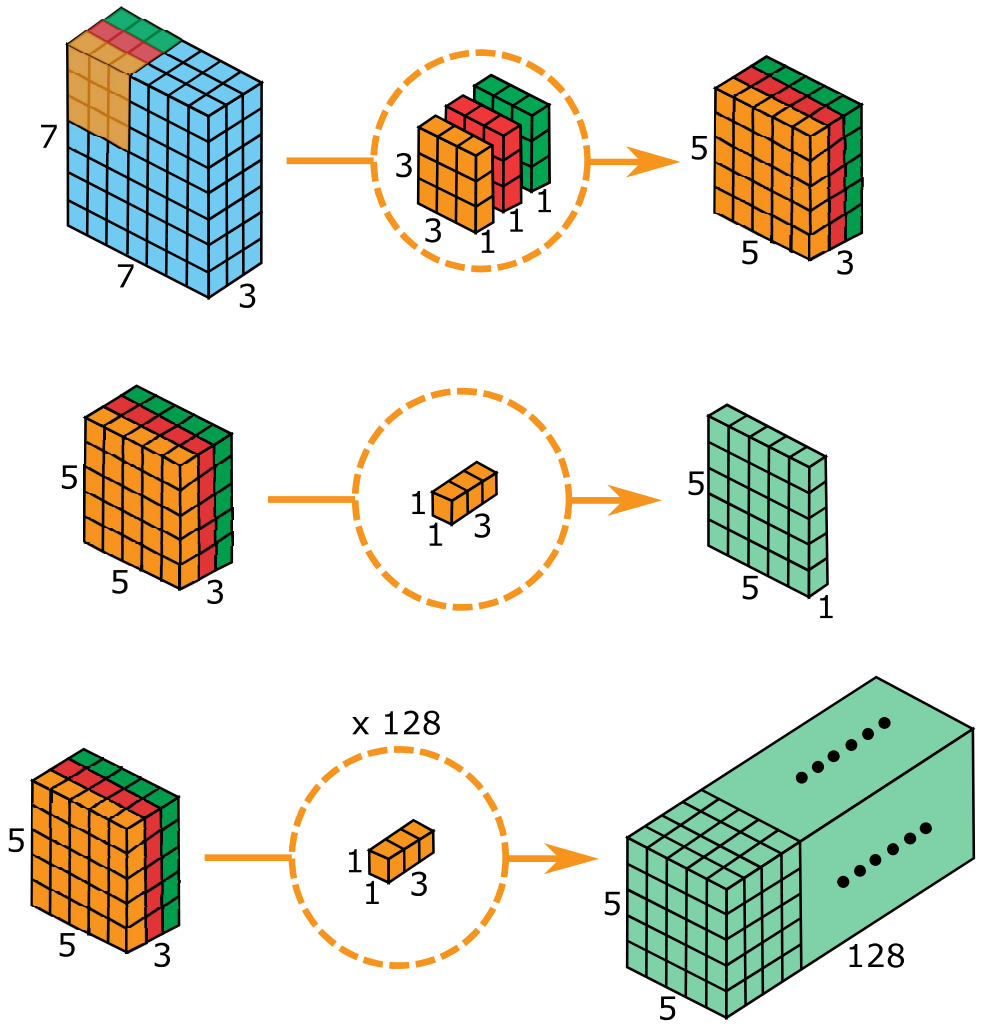


Figura 1.11: Depthwise separable convolution [16]

que una imagen sufre dentro de la capa convolucional para poder extraer su mapa de características. Todo el proceso consta de multiplicaciones y sumas (operaciones lineales), lo que hace que el proceso de convolución sea lineal en sí. El problema de la linealidad en un proceso de clasificación como el del presente proyecto es que no permite obtener una adecuada eficiencia al momento de entrenar la red, pues en la mayoría de los casos la linealidad en una imagen impide resaltar los aspectos característicos que la distinguen de otra. Como se ve en la Figura 1.12, la función que caracteriza a la ReLU se comporta de dos maneras distintas dependiendo del valor de entrada. Los valores negativos que ingresan a la función se suprimen la salida, mientras que los valores positivos no sufren ningún cambio. Este proceso relativamente sencillo introduce no linealidad en la red y, por consiguiente, resalta las características relevantes de una imagen para facilitar el trabajo de discriminación. En una imagen esta no linealidad conlleva cambios abruptos en patrones, colores y otras características [17].

Si se aplicas la capa de ReLU al ejemplo planteado en la sección anterior, se obtendrá un resultado como el de la Figura 1.13, en donde los valores negativos son reemplazados por cero en el mapa de características, y los valores positivos no han sufrido cambio alguno.

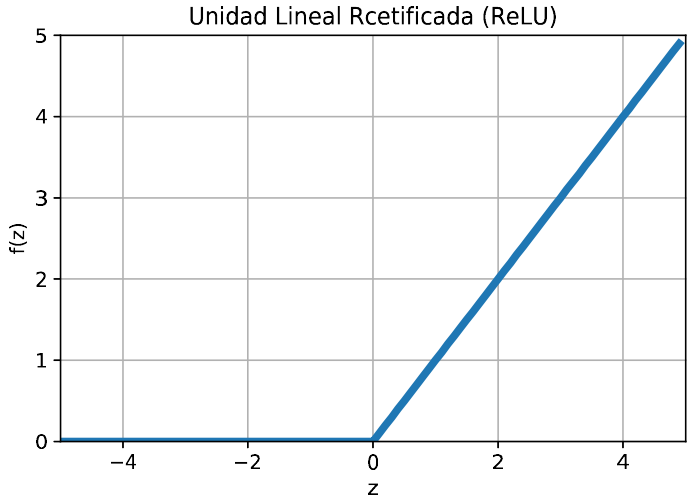


Figura 1.12: Función de acivación ReLU

Mapa de características del Filtro 1

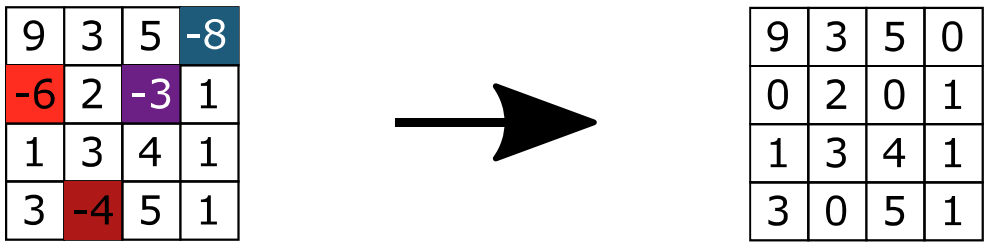


Figura 1.13: ReLU sobre mapa de características [16]

1.3.4.4. Capa de agrupamiento (Pooling)

La capa de agrupamiento, o capa de *pooling* como se la conoce más comúnmente, cumple varias funciones de vital importancia para el desenvolvimiento de la red. Esta capa suele ser aplicada después de añadir no linealidad al proceso, como por ejemplo la capa de *ReLU* analizada previamente.

A pesar del potente hardware del que se dispone actualmente, todo el proceso que conlleva el entrenamiento de una CNN puede llegar a consumir una gran cantidad de recursos computacionales, pues la cantidad de procesos que se llevan a cabo simultáneamente puede llegar a ser extremadamente grande, dependiendo del tipo de red neuronal utilizada y la cantidad de información de entrada disponible.

Es aquí donde la capa de *pooling* se vuelve fundamental para reducir la carga de procesamiento. El objetivo de la capa de *pooling* es reducir las dimensiones de los mapas de características que se obtuvieron de la capa convolucional y que ya pasaron por la capa de *ReLU*. Para lograr esta reducción en la dimensionalidad de estos mapas de características se aplica un proceso muy similar al filtrado que se vio en la capa convolucional, en el cual el filtro seleccionado es más pequeño que el mapa de características y al desplazarse a lo largo del mapa realiza una agrupación en sus valores que da como resultado final un ma-

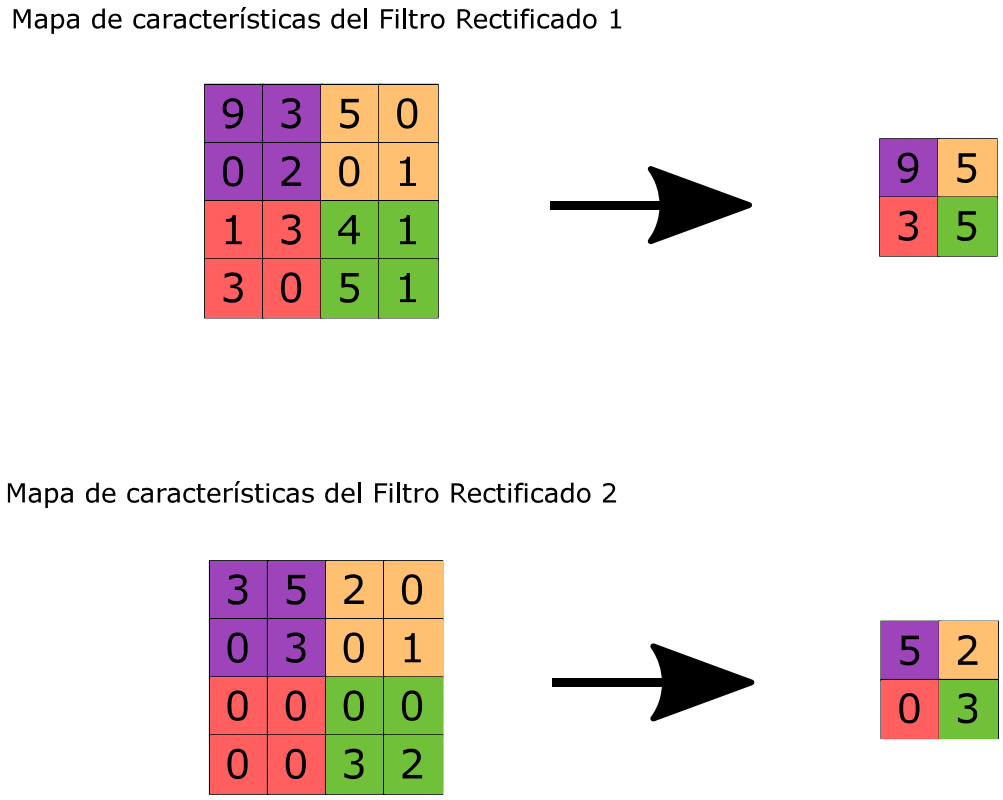


Figura 1.14: Capa de max pooling

pa más compacto. En la mayoría de los casos, es común utilizar un filtro de dimensiones 2×2 con un valor de *stride* de 2. Existen varias técnicas que permiten realizar el proceso de *pooling*, las más utilizadas son las de *average pooling* y *max pooling*. Estas son las más utilizadas debido a que han demostrado tener los mejores resultados en la práctica [16].

Average pooling: Esta técnica obtiene un valor promedio en los valores que agrupa el filtro sobre el mapa de características.

Max pooling: Su funcionamiento es un poco más simple que el caso anterior, pues en el *max pooling* sólo se escoge el valor más alto en el conjunto de valores a los que el filtro es aplicado en el mapa de características [13].

Una vez más, tomando el ejemplo planteado en la Sección 1.3.4.1, se aplica un proceso de *max pooling* al mapa de características obtenido de la Sección 1.3.4.3, en donde se introduce no linealidad en los mapas correspondientes a cada filtro y se obtiene un resultado como el que se visualiza en la Figura 1.14, en donde se ha reducido la dimensión del mapa de características de 4×4 a 2×2, es decir, se ha reducido la cantidad de información que será procesada en la capa de MLP en un 50 %.

La segunda funcionalidad de la capa de *pooling* es dotar a la CNN la capacidad de poder identificar las características relevantes independientemente de dónde estas se localicen en la imagen. Esta propiedad recibe el nombre de invarianza traslacional, y proporciona a la red el beneficio de la reducción de sensibilidad ante pequeñas variaciones en la localización

de las características de una imagen [16].

Adicionalmente, en la Figura 1.6 se puede ver una capa de aplanamiento, más conocida como *flatten*, la cual realiza un procesamiento sencillo a la información obtenida a la salida de todas las capas convolucionales de las que conste la red implementada. La capa de *flatten* estructura la información en un formato aceptable para poder ser ingresada a la MLP y así poder continuar el procesamiento de esta [13].

1.4. TRANSFERENCIA DE APRENDIZAJE (TRANSFER LEARNING)

En la Sección 1.3.4.1 se mencionó el término *"transferencia de aprendizaje"*, más conocido como *transfer learning*, y en esta sección se describirá con más detalle su concepto y la utilidad de éste método implementado adecuadamente en una red neuronal.

Ya se conocen los conceptos básicos acerca del perceptrón, las redes neuronales MLP y las redes neuronales convolucionales (CNN), pero aún no se conocen las distintas estrategias con que se pueden implementar estos conceptos en la práctica para un modelo funcional. Antes de implementar una red neuronal, primero se debe tomar en cuenta aspectos como el objetivo de nuestro modelo, los datos de entrenamiento de los que se dispone, bases de datos de acceso público, el tiempo necesario para su entrenamiento, el rendimiento que se quiere alcanzar, entre otros.
La primera opción a considerar es la implementación de un modelo totalmente personalizado, al cual se puede añadir las capas y parámetros que nos parezcan adecuadas para su correcto desenvolvimiento. Este es un proceso complejo y requiere de períodos de prueba extensos para poder obtener resultados óptimos. En muchos de los casos es más eficiente utilizar la técnica de *transfer learning*, cuyo principio de funcionamiento se muestra en la Figura 1.15.

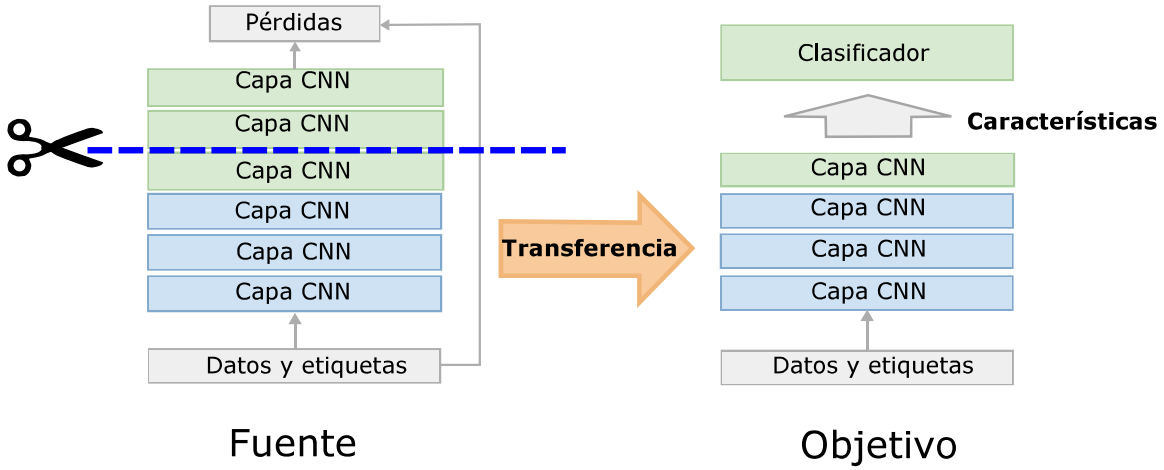


Figura 1.15: Principio básico de transfer learning [18]

La técnica de *transfer learning* se basa en la reutilización de un modelo previamente entrenado para una tarea específica, pero enfocándolo hacia nuestros requerimientos. Esto

	Base de datos similar	Base de datos diferente
Base de datos pequeña	Transfer learning: características de nivel más alto + clasificador	Transfer learning: características de nivel más bajo + clasificador
Base de datos grande	Fine-tune	Fine-tune

Figura 1.16: Casos de uso en transfer learning [19]

supone una gran mejora en cuanto a tiempos de entrenamiento y también en el rendimiento de nuestro modelo, pues el modelo pre-entrenado puede cumplir con una función similar a nuestro objetivo. El método tradicional de *transfer learning* es el que se describe en la Figura 1.15, es decir, se toma la red previamente entrenada y se descartan las capas que sean necesarias. Los pesos que la red obtuvo después de su proceso de entrenamiento quedan estáticos y estos se transfieren a nuestro modelo. Se dice que estos parámetros transferidos a nuestra red es el conocimiento previo de la red que ya fue entrenada con un distinto conjunto de datos. A nuestra red se le añade las capas específicas para la tarea que fue planteada, que generalmente incluye a la capa de clasificación. De esta manera, el conocimiento de la red que se usó como fuente fue aprovechado para ser aplicado a una tarea distinta y se han tomado los beneficios de cortos períodos de prueba y una mejor eficiencia para nuestro modelo [19].
El método descrito se conoce como *extractor de características*, y no es el único que existe al momento de aplicar *transfer learning*, existen distintos escenarios para los cuales se han desarrollado variaciones de este método, los más frecuentes se muestran en la Figura 1.16, en donde se ven 4 posibles variaciones de aplicación de *tranasfer learning* y la aparición de un nuevo término: *Fine-tune*.
Fine-tune (ajuste sutil) es el término que describe a un caso específico de *transfer learning*, en el cual los pesos no permanecen estáticos o congelados (*freeze*) al momento de utilizarse en el nuevo modelo después de ser transferidos, sino que la red es nuevamente entrenada con una base de datos propia y los pesos sufren pequeñas variaciones gracias a la utilización de una tasa de aprendizaje y una de decaimiento [19].

Ahora que se tiene una familiarización con el concepto de *fine tune* se puede pasar a describir cada uno de los casos que se presentan en la Figura 1.16, tomando en cuenta que la tarea original es la tarea a la que se enfocó el modelo previo, y la nueva tarea es a la que el nuevo modelo se enfoca.

La nueva tarea tiene una base de datos más pequeña pero similar a la tarea original: En este caso es oportuno utilizar *transfer learning* tradicional, debido a que con pocos datos de entrenamiento e información poco diversa la red puede sufrir un *sobre-entrenamiento (overfitting)*. Se analizará el problema del *overfitting* más adelante.
Como los nuevos datos son similares a los originales, se tiene la libertad de utilizar las capas de nivel más alto de la red, pues aquí se encuentran las características más específicas que serán similares a las de los nuevos datos de entrada.

La nueva tarea tiene una base de datos más pequeña y diferente a la tarea original: Como los datos de los que se dispone son demasiado pequeños para volver a entrenar la red, no es factible utilizar *fine-tune*, además estos son diferentes a los originales y tampoco se puede utilizar información de las últimas capas del modelo previo. La solución para este problema es simplemente utilizar el mismo esquema del caso previo, pero utilizando la información de capas más bajas de la red, en donde las características recolectadas son más genéricas y pueden ser utilizadas para tareas no tan similares.

La nueva tarea tiene una base de datos grande y similar a la tarea original: Con una cantidad grande de datos sería factible entrenar una red neuronal desde cero, pero es más eficiente optar por el fine-tune y simplemente realizar ajustes pequeños para obtener un mejor rendimiento.

La nueva tarea tiene una base de datos grande y diferente a la tarea original: Este caso es el que más se adapta a las condiciones de este proyecto en particular. Se tiene una base de datos considerablemente grande, pero esta es diferente a la del modelo original, entonces la opción que optimizaría el rendimiento del nuevo modelo es la del fine-tune [19].

1.5. SOBREAJUSTE (OVER-FITTING) Y DROPOUT

El *over-fitting* es uno de los problemas más frecuentes en el campo del *deep learning*, sin embargo, existen métodos como el *dropout* para mitigar en gran medida los efectos del *over-fitting* sobre el rendimiento de una red neuronal.

1.5.1. OVER-FITTING

Se tiende a pensar que un rendimiento sobresaliente como resultado final del entrenamiento de la red es siempre beneficioso, pero esto no es del todo correcto. El problema del *over-fitting* suele ser engañoso y hay que saber lidiar con él.
En la Figura 1.17 se puede ver una comparación entre los distintos casos de ajuste que se pueden obtener del entrenamiento de una red neuronal. El primer caso es evidente, el *under-fitting* es el resultado de un mal entrenamiento y se puede dar debido a pobres bases de datos o modelos muy simples.
Los casos que nos interesan son el *apropiado* y el *over-fitting*. Para obtener un ajuste apropiado es necesario considerar el aspecto de generalización de un modelo.

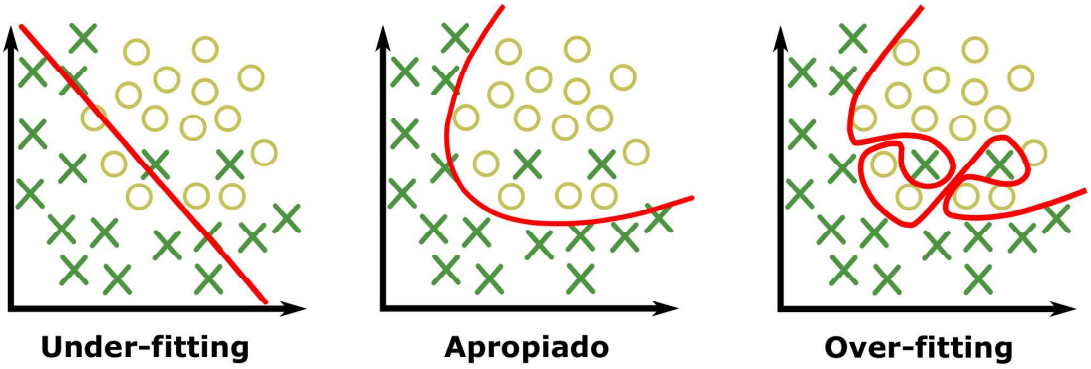


Figura 1.17: Comparación entre distintos casos de ajuste [20]

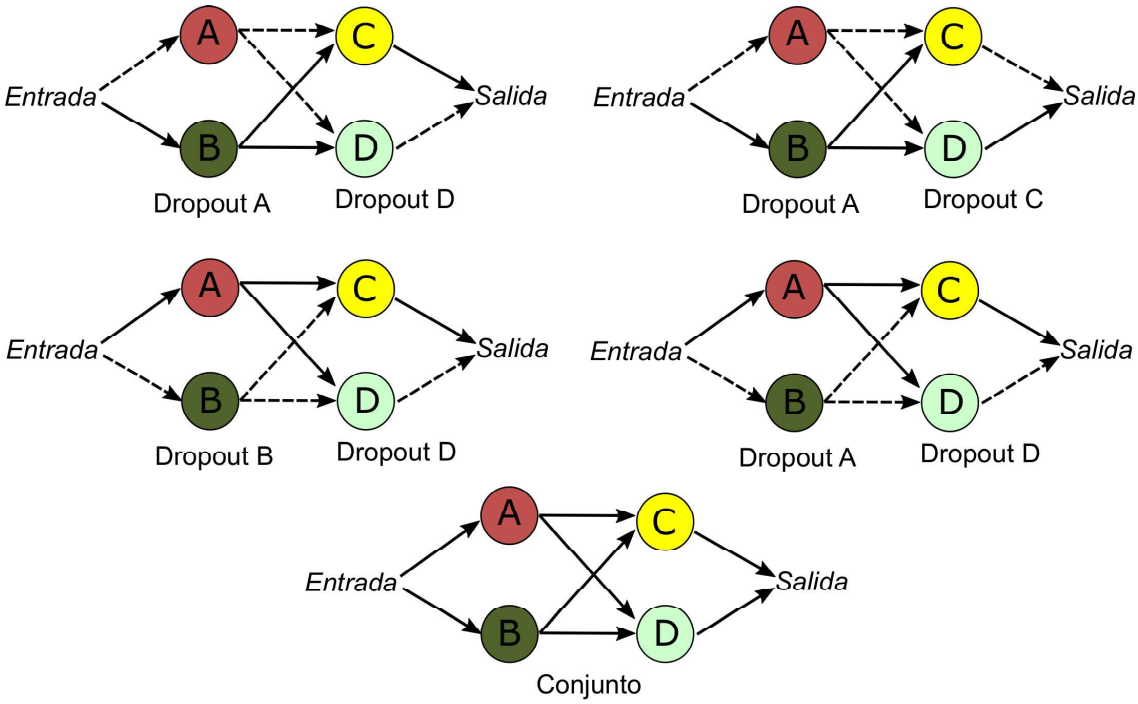


Figura 1.18: Concepto de dropout [21]

Generalización: En aprendizaje de máquina, el término generalización hace referencia a la capacidad que tiene el modelo para generalizar los conceptos aprendidos para información que no ha sido vista o introducida a dicho modelo.

De aquí se puede notar que el principal problema del *over-fitting* es su pobre capacidad de generalización, lo que conlleva a un rendimiento muy bajo, y será especialmente notorio al realizar predicciones sobre información que no ha pasado por la red [20].

1.5.2. DROPOUT

El *dropout* es un método muy sencillo de aplicar en CNNs y a la vez es muy efectivo cuando se trata de casos en que la variedad de datos es muy limitada, es decir, casos en que es muy probable que se produzca *over-fitting*, como se vio en la Sección 1.4.
En la Figura 1.18 se representa el concepto básico del funcionamiento del *dropout*. Para

que una red neuronal tenga un alto rendimiento existen varios factores de gran importancia, como la cantidad de datos con la que es entrenada o la variedad de los mismos. Si los datos de los que se dispone no cumplen con estos factores existe la opción de generar varios modelos que aprendan distintas características de los datos para luego formar un modelo más robusto y de alto rendimiento.

La idea del *dropout* es descartar aleatoriamente cierto porcentaje de neuronas en cada época de entrenamiento. A pesar de utilizar la misma red neuronal para cada época, el proceso aleatorio de descarte de neuronas hace que el modelo varíe y se obtenga un aprendizaje más completo acerca de las características de los datos de entrada.

Adicionalmente, la aplicación del *dropout* en el proceso de entrenamiento introduce un efecto de dispersión en las curvas de aprendizaje, lo que se traduce como una mejora en la capacidad de generalización del modelo, evitando así que se presente un posible *over-fitting* [22].

1.6. HERRAMIENTAS DE IMPLEMENTACIÓN

Una vez que se han analizado todos los conceptos básicos referentes al ámbito de las redes neuronales y el aprendizaje profundo o *deep learning*, se puede familiarizarse con el ambiente en el que se desarrollarán los modelos de aprendizaje que serán capaces de realizar las tareas que se requieren para cumplir con los objetivos de este proyecto.
Como se sabe, el software a utilizar para toda la programación es Python a través de su distribución Anaconda, que facilita la creación de distintos ambientes que serán utilizados según los requerimientos específicos de cada tarea. Además, Anaconda ofrece una interfaz sencilla para la instalación de las diversas librerías necesarias para cada aplicación. Para el desarrollo de modelos de aprendizaje automático, una de las librerías más populares debido a su eficiencia de compilación es Tensorflow, la cual fue utilizada para nuestro propósito.
En la Figura 1.19 se aprecia la interfaz principal de Anaconda en donde se ha creado el ambiente llamado *tensorflow* remarcado en el recuadro rojo, en el cual se instalaron todos los elementos necesarios para llevar a cabo la implementación del modelo de clasificación. La Figura 1.20 en cambio muestra el apartado *Enviroments* de Anaconda, en el cual se puede hallar con facilidad los diversos paquetes que se encuentran disponibles para ser instalados, como Tensorflow y Keras entre otros. Las librerías necesarias pueden ser instaladas utilizando este método o mediante comandos simples dentro de la interfaz de programación de Python.

1.6.1. LIBRERÍA DE APRENDIZAJE PROFUNDO TENSORFLOW

TensorFlow es una librería de código abierto que fue creada por el equipo de *Google Brain* en el año 2015. Está orientada al aprendizaje automático y su lenguaje base puede ser C++ o popularmente Python.

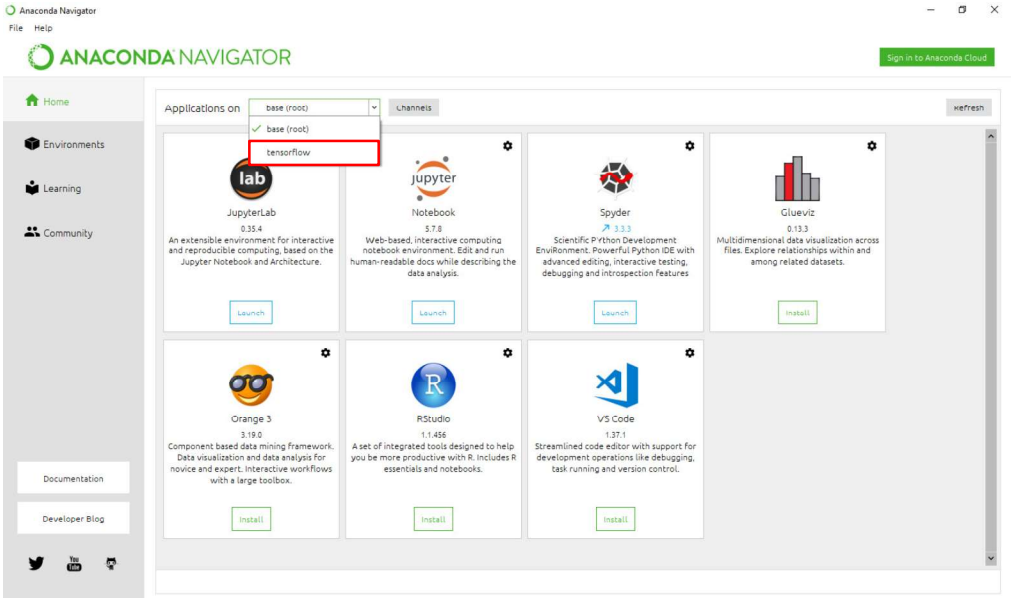


Figura 1.19: Interfaz principal de Anaconda Navigator con sus respectivos ambientes

Esta librería se ha convertido en una de las más populares debido a las ventajas que ofrece en cuanto eficiencia con respecto a su competencia. Es posible implementarla sobre varios CPUs o GPUs, y su tiempo de compilación es menor, lo que la hace ideal para el procesamiento de imágenes que en otras plataformas podría resultar en un proceso largo y tedioso. Además, Tensorflow ofrece herramientas visuales como TensorBorad, en la cual se puede obtener estadísticas, gráficas analíticas de nuestro modelo e incluso se puede visualizar gráficas estructurales de la red neuronal con las respectivas modificaciones que se le hayan aplicado [23].

1.6.2. LIBRERÍA DE REDES NEURONALES KERAS

Keras es en sí una API (*Application Programming Interface*) de alto nivel orientada a las redes neuronales desarrollada en Python, y es capaz de ejecutarse sobre la librería TensorFlow, descrita previamente. El principal enfoque de Keras es la experimentación rápida bajo el concepto de que un rápido salto de la idea a la obtención de resultados es el mejor método de investigación.
Existen cuatro principios que rigen el funcionamiento de Keras, y debido a estos se debe la gran acogida de Keras entre quienes buscan desarrollar prototipos de una manera rápida y sencilla mediante la utilización de redes neuronales.

Facilidad de uso: Bajo la premisa de que Keras fue diseñada para humanos y no para máquinas, esta se enfoca primordialmente en la experiencia del usuario, es decir, el usuario no debe tener necesariamente un alto grado de conocimiento para poder trabajar con Keras. Las APIs que implementa son muy simples y concretas, además las acciones que el usuario debe realizar se minimizan y la retroalimentación que se recibe al momento de cometer un error es muy clara y fácil de procesar para su corrección.

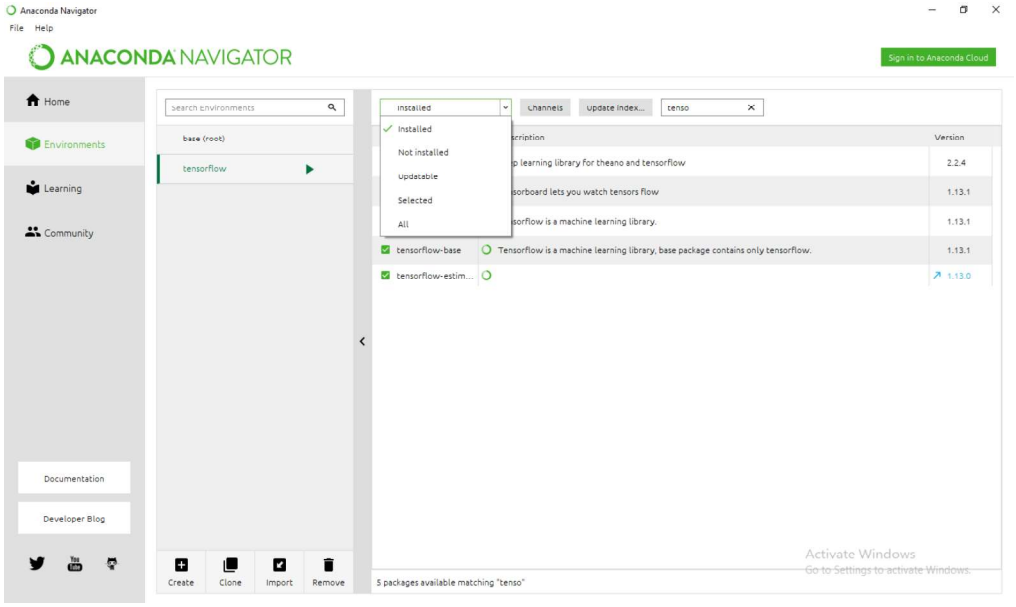


Figura 1.20: Interfaz de instalación de paquetes de Anaconda

Modularidad: La creación de un modelo depende de la implementación de múltiples módulos, entre los cuales se puede mencionar a los algoritmos de optimización, las distintas capas de la red, las funciones de activación, etc. Cada uno de estos elementos puede ser combinado en Keras con la menor cantidad de restricciones posibles y así obtener un modelo que se adapte a nuestros requerimientos específicos con la mejor eficiencia.

Fácil extensibilidad: Está relacionado con el principio de modularidad, pues el hecho de que los módulos que componen el modelo sean independientes permite que agregar nuevos módulos para implementar funcionalidades o desarrollar nuevas aplicaciones se muy sencillo. Otra ventaja de la extensibilidad es que permite una mayor personalización de nuestro modelo y así incursionar en investigaciones más avanzadas.

Trabaja con Python: Los modelos creados con Keras son escritos en su totalidad en Python, lo que hace que sean más compactos y fáciles de depurar. También facilitan el proceso de extensibilidad [24].

Ahora que se tiene un concepto más claro acerca del funcionamiento de las redes neuronales y el ambiente en el cual se van a implementar, podemos proceder al análisis de las distintas técnicas para obtener un desempeño óptimo.

2. METODOLOGÍA

El proceso de implementación del sistema consta de etapas muy marcadas, la Figura 2.1 muestra el trabajo a realizar de una manera simplificada y sencilla de entender a través de un diagrama de bloques.

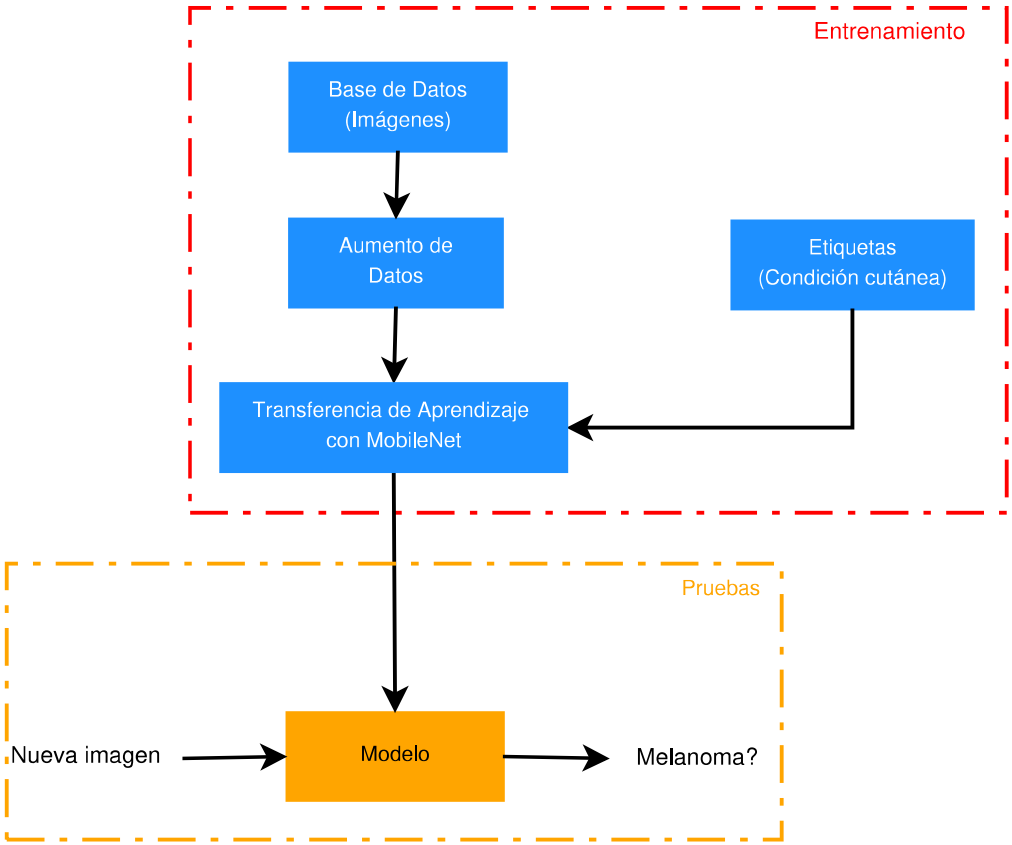


Figura 2.1: Etapas de la implementación del sistema

Para iniciar el proceso de entrenamiento es necesario tener una base de datos, que en este caso consistirá en imágenes de diferentes condiciones de la piel. Estas imágenes deben pasar por un proceso de aumento de datos, debido a que la cantidad de imágenes debe ser equilibrada para obtener un mejor rendimiento en la etapa de entrenamiento. La etapa de transferencia de aprendizaje con MobileNet (arquitectura orientada a dispositivos con recursos limitados [25]) es la más importante en este proceso, ya que en esta etapa se define el modelo a usar y se lleva a cabo el entrenamiento de la red. Finalmente, el modelo resultante de la fase de entrenamiento se utilizará para determinar el rendimiento del clasificador en un conjunto de pruebas.

2.1. BASE DE DATOS

Como se mencionó previamente, la base de datos utilizada para la etapa de entrenamiento de la red neuronal es la HAM10000, que contiene imágenes de diferentes tipos de enfermedades de la piel y, entre ellas, se pueden encontrar enfermedades benignas y cancerígenas.

Tabla 2.1: Comparación entre distintas bases de datos [5]

Base de datos	Total de imágenes	Verificación patológica (%)
PH2	200	20.5 %
Atlas	1024	desconocido
ISIC 2017[b]	13786	26.3 %
Rosendahl	2259	100 %
ViDIR Legacy	439	100 %
ViDIR Current	3363	77.1 %
ViDIR MoleMax	3954	1.2 %
HAM10000	**10015**	53.3 %

En la Tabla 2.1 podemos ver una comparación entre distintas bases de datos que se consideraron para llevar a cabo el proyecto. Notamos que HAM10000 a pesar de no contar con una verificación patológica del 100 % como las de Rosendahl o ViDIR Legacy, si cuenta con un porcentaje mayor al 50 %.
La *verificación patológica* hace referencia al conjunto de información relevante que se puede obtener a partir de una biopsia (extracción de muestras de tejido). Esta información puede ser macro y microscópica, y es el método más confiable para determinar la presencia o ausencia de la afección [26].)
El aspecto fuerte que HAM10000 nos ofrece es la cantidad de imágenes que posee y la variedad de las mismas. Se puede decir que es la base de datos más equilibrada de las mencionadas en la comparación en cuanto a los aspectos de verificación y de cantidad de imágenes.

Tabla 2.2: Base de datos HAM10000 [5]

Lesión cutánea	Imágenes totales
Queratosis Actínica (akiec)	327
Carcinoma de Células Basales (bcc)	514
Queratosis Benigna (bkl)	1099
Dermatofibroma (df)	115
Nevos Melanocíticos (nv)	6705
Melanoma (mel)	1113
Lesiones Cutáneas Vasculares (vasc)	142
Total	10015

En la Tabla 2.2 podemos ver que la base de datos contiene 10015 imágenes, que se han clasificado en 7 categorías diferentes. El surgimiento de la base de datos HAM10000 (*"Human Against Machine with 10000 training images"*) se da debido a la necesidad de solucionar el problema de las deficientes y poco variadas bases de datos públicas existentes en el ámbito de las lesiones cutáneas. Las imágenes fueron recolectadas de diferentes poblaciones a lo largo de 20 años utilizando únicamente dos fuentes, el Departamento de Dermatología de la Universidad de Medicina de Viena en Austria y la práctica del cáncer de piel de Cliff Rosendahl en Queensland, Australia. Las imágenes pasaron por un proceso de normalización y limpieza, pues una parte de las imágenes recolectadas ni siquiera contaban con un formato

digital debido a su antigüedad. El resultado final es un dataset enfocado a aplicaciones de aprendizaje automático para fines académicos [5]. A continuación se presenta una breve descripción de las afecciones presentes en el dataset:

Queratosis Actínica (akiec): Es una variante no invasiva del carcinoma de células escamosas, y puede ser tratada localmente por medio de cirugía. Sin embargo, si la enfermedad progresa se puede convertir en un carcinoma invasivo.

Carcinoma de Células Basales (bcc): Es una variación del cáncer de piel epitelial, pero esta no suele resultar en metástasis. Además se puede presentar en variedad de formas (plana, nodular, pigmentada o quística).

Queratosis Benigna (bkl): Es en realidad una condición genérica, pues está compuesta por tres subgrupos que dermatoscópicacmente lucen diferentes pero biológicamente son similares, por lo que generalmente se los agrupa bajo una misma denominación histopatológica.

Dermatofibroma (df): Es una condición cutánea que se considera benigna y se presenta a partir de la proliferación o inflamación de un trauma mínimo. La forma típica en que se aprecia dermatoscópicamente es con líneas reticulares en su periferia con un parche central blanco que indica fibrosis.

Nevos Melanocíticos (nv): Se trata de neoplasias de melanocitos de carácter benigno y que se pueden presentar en una infinidad de variantes. Difieren del melanoma en la simetría que presentan, tanto en color como estructura.

Melanoma (mel): Al igual que los nevos melanocíticos, el melanoma también es una neoplasia derivada de los melanocitos, pero en este caso de carácter maligno. El melanoma puede ser curado si se los detecta en etapas tempranas y se lo extirpa mediante escisión quirúrgica simple. Su tratamiento depende del sitio anatómico en donde se ubique.

Lesiones Cutáneas Vasculares (vasc): Es un conjunto de condiciones cutáneas entre las que se incluyen angiomas cereza, angioqueratomas, granulomas piógenos e incluso hemorragias. Estos angiomas se caracterizan por sus colores rojos o púrpuras y se los conoce también como terrones rojos o lagunas [5].

La base de datos HAM10000 puede ser obtenida fácilmente de varias fuentes de acceso libre. Para este proyecto se la obtuvo libremente de la siguiente dirección web: https://www.kaggle.com/kmader/skin-cancer-mnist-ham10000, en la cual puede encontrarse el archivo con la metadata necesaria para llevar a cabo el preprocesamiento inicial de las imágenes.

En la Figura 2.2 encontramos una recopilación de ejemplos visuales de cada una de las enfermedades descritas anteriormente.

Un aspecto importante que debemos considerar es que el número de imágenes en la categoría *nv* es notablemente mayor que el resto de las categorías (es decir, conjunto de datos desbalanceado).

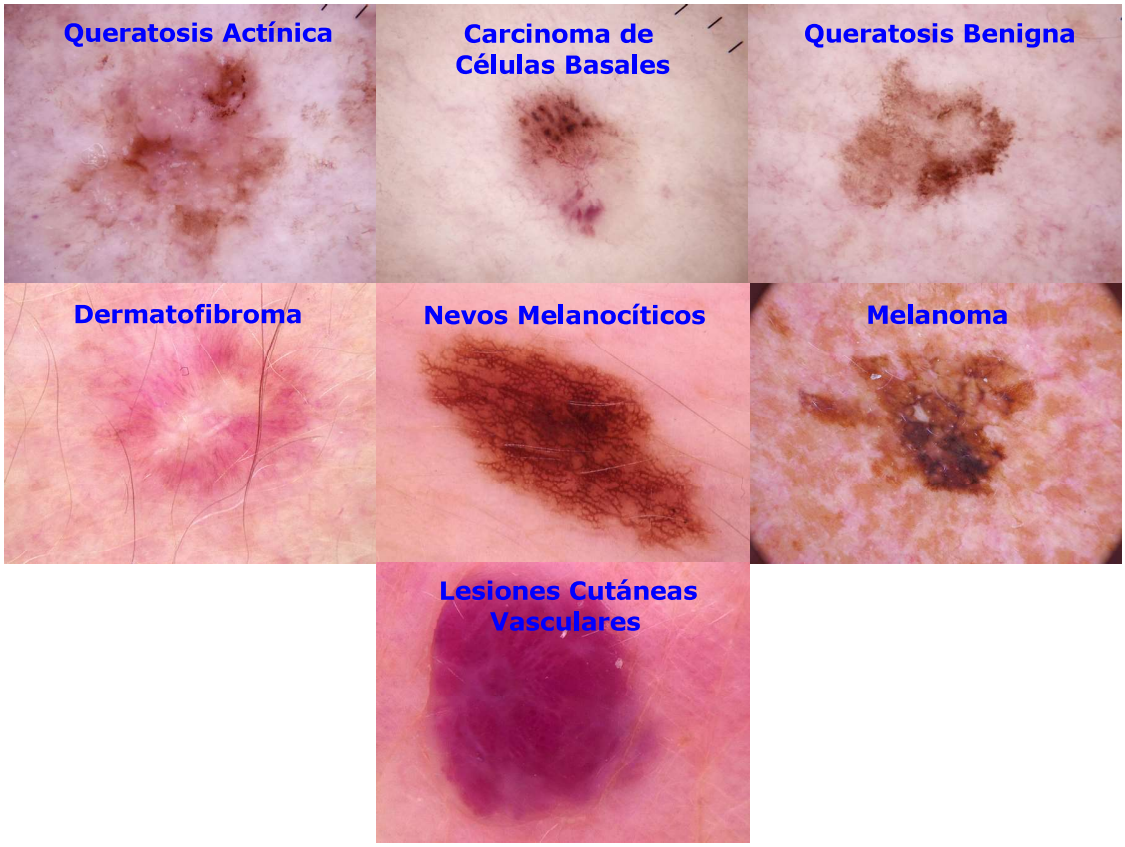


Figura 2.2: Lesiones cutáneas de la base de datos HAM10000

Un conjunto de datos desbalanceado puede ocasionar que la predicción favorezca a la clase con un mayor número de datos, pues el modelo tiene una menor probabilidad de fallar en su predicción, pero no será efectivo al momento de realizar predicciones con información que no haya pasado a través del modelo (información nueva) [27]. Esto puede causar problemas en la fase de entrenamiento, teniendo un impacto negativo en el rendimiento del modelo, por lo que en la siguiente etapa se aborda este problema.

Antes de empezar a describir todos los parámetros que definen al modelo, es importante aclarar que estos valores han sido fijados en base a resultados obtenidos en trabajos previos [13], con los cuales se han obtenido resultados satisfactorios.

2.2. AUMENTO DE DATOS

La técnica de aumento de datos o *data augmentation* es utilizada principalmente para ampliar una base de datos cuando esta es relativamente pequeña, o bien para mejorar el rendimiento de un modelo que no presente inconvenientes con el tamaño de su base de datos. El propósito de la etapa de aumento de datos es generar nuevas imágenes a partir de las originales. Para lograr este objetivo, se parte de las imágenes originales como la que se muestra en la Figura 2.3, a las cuales se aplica una variedad de transformaciones. La herramienta que se utiliza para poder generar nuevas imágenes aplicando distintos parámetros es `ImageDataGenerator`, que se utiliza generalmente en Keras para el preprocesamiento de

bases de datos, específicamente, para trabajar con imágenes como en nuestro caso.

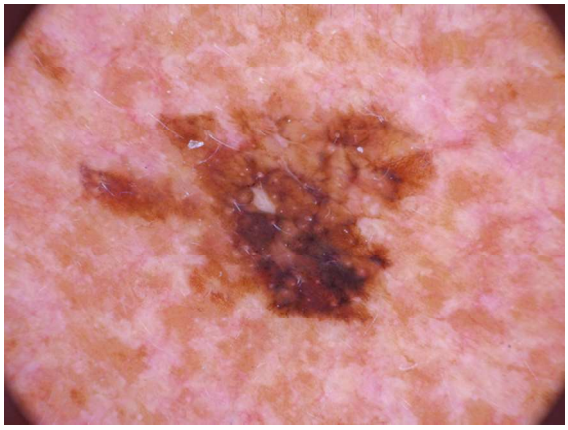


Figura 2.3: Imagen original de clase *mel*

Segmento de código 2.1: Parámetros para aumento de datos

```
1    datagen = ImageDataGenerator(
2        rotation_range=180,
3        width_shift_range=0.1,
4        height_shift_range=0.1,
5        zoom_range=0.1,
6        horizontal_flip=True,
7        vertical_flip=True,
8        brightness_range=(0.9,1.1),
9        fill_mode='nearest')
```

A continuación se describen cada uno de los argumentos que se incluyen dentro de la función `ImageDataGenerator`.

Rango de rotación: Es el rango de grados para rotaciones aleatorias. Al fijar este parámetro en 180, se define un rango entre -180 y +180 grados. Se puede implementar este argumento mediante el comando `rotation_range=180`. Se muestra la comparativa en la Figura 2.4.

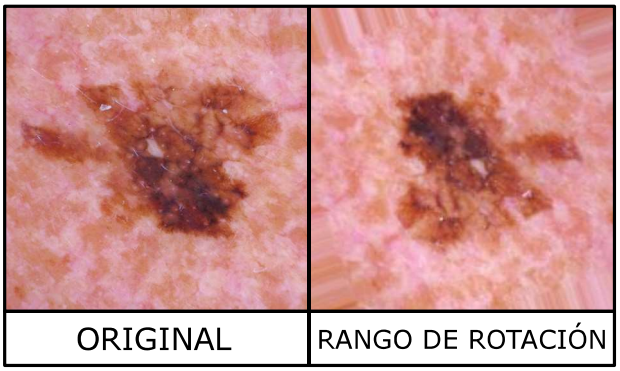


Figura 2.4: Visualización del rango de rotación

Rango de desplazamiento en ancho: La imagen original se desplaza aleatoriamente en un porcentaje proporcional al ancho de la imagen. Se estableció este parámetro en 0.1, es

decir, 10 %. Se puede implementar este argumento mediante el comando `width_shift_range =0.1`. Se muestra la comparativa en la Figura 2.5.

Figura 2.5: Visualización del rango de desplazamiento en ancho

Rango de desplazamiento en altura: Es similar a la transformación anterior, pero utiliza la altura de la imagen para realizar el cambio. Se estableció este parámetro en 0.1, es decir, 10 %. Se puede implementar este argumento mediante el comando `height_shift_range=0.1`. Se muestra la comparativa en la Figura 2.6.

Figura 2.6: Visualización del rango de desplazamiento en altura

Rango de enfoque: Esta transformación genera un zoom aleatorio. Se estableció un rango entre 90 % a 110 % de la imagen original. Se puede implementar este argumento mediante el comando `zoom_range=0.1`. Se muestra la comparativa en la Figura 2.7.

Giro horizontal: Da la vuelta al azar a la imagen horizontalmente. Se puede implementar este argumento mediante el comando `horizontal_flip=True`. Se muestra la comparativa en la Figura 2.8.

Giro vertical: Da la vuelta al azar a la imagen verticalmente. Se puede implementar este argumento mediante el comando `vertical_flip=True`. Se muestra la comparativa en la Figura 2.9.

Rango de brillo: Aumenta o disminuye el brillo de la imagen. Se estableció un rango entre 0.9 y 1.1. Se puede implementar este argumento mediante el comando `brightness_range`

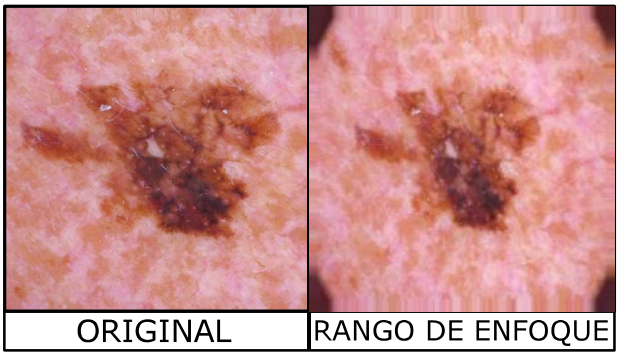


Figura 2.7: Visualización del rango de enfoque

Figura 2.8: Visualización del giro horizontal

`=(0.9,1.1)`. Se muestra la comparativa en la Figura 2.10.

Modo de relleno: Se utilizó el parámetro *nearest* para llevar a cabo esta transformación, el cual rellena puntos fuera de los límites de la imagen con información similar a la que se encuentra dentro de sus límites. Se puede implementar este argumento mediante el comando `fill_mode='nearest'`.

Estas transformaciones se aplican a cada imagen de todas las categorías excepto *nv*, ya que el objetivo es que todas las categorías tengan un número similar de imágenes entre ellas.

La categoría *nv* tiene aproximadamente 6000 imágenes, por lo que el algoritmo aplicado a las categorías restantes para generar el aumento de datos genera aproximadamente el mismo número de imágenes que la categoría *nv*.

Segmento de código 2.2: Proceso de aumento de datos

```
    batch_size = 50

    aug_datagen = datagen.flow_from_directory(path,
                                              save_to_dir=save_path,
                                              save_format='jpg',
                                              target_size=(224, 224),
                                              batch_size=batch_size)
```

La generación de las nuevas imágenes es un proceso que no posee gran complejidad, y se muestra su implementación en el Segmento de código 2.2.

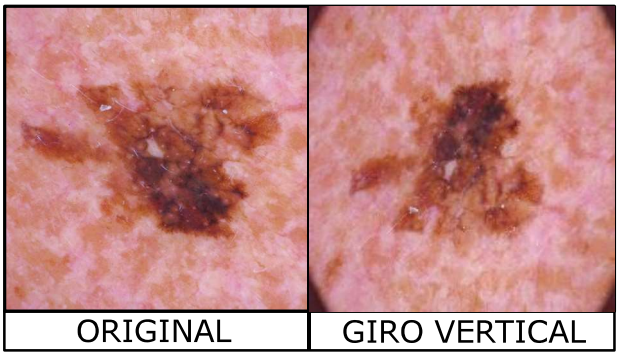


Figura 2.9: Visualización del giro vertical

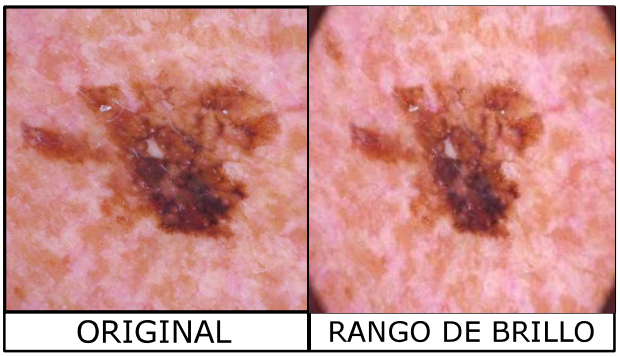


Figura 2.10: Visualización del rango de brillo

Para explicar este código se empieza por el parámetro `batch_size`, que define el número de imágenes que serán procesadas simultáneamente. Su valor dependerá del tiempo disponible para el proceso y el rendimiento deseado, pues mientras más alto sea este valor, más imágenes serán procesadas y el tiempo de procesamiento será menor. Sin embargo, puede que el rendimiento del modelo se vea afectado. El parámetro `aug_datagen` es el encargado de generar las imágenes nuevas, y lo hace de la siguiente manera: al aplicar las modificaciones vistas en el Segmento de código 2.1 a las imágenes originales, estas son primero almacenadas en una ruta temporal, la cual se define en el parámetro `path`. Una vez terminado el proceso de generación de las imágenes, estas deben migrar hacia su ruta final para ser distribuidas en sus respectivas categorías. Esta ruta final se define en el parámetro `save_path`. El resto de parámetros implementados en el código únicamente definen el formato y tamaño de la imagen generada.
En la Tabla 2.3 se puede ver los resultados obtenidos del proceso de aumento de datos.
A diferencia de la Tabla 2.2, la Tabla 2.3 muestra claramente una distribución mucho más equilibrada, pero a su vez, esto aumentará el tiempo de procesamiento de las imágenes en la red.

Tabla 2.3: Base de datos HAM10000 con aumento de datos

Lesión cutánea	Imágenes totales
Queratosis Actínica (akiec)	5644
Carcinoma de Células Basales (bcc)	5604
Queratosis Benigna (bkl)	6163
Dermatofibroma (df)	4944
Nevos Melanocíticos (nv)	6705
Melanoma (mel)	6237
Lesiones Cutáneas Vasculares (vasc)	6033
Total	41330

2.3. TRANSFERENCIA DE APRENDIZAJE CON LA CNN MOBILENET

La idea de la transferencia de aprendizaje es que se toma una red previamente entrenada con una gran cantidad de datos y se la usa para una nueva tarea.
Se utilizó la red MobileNet cuya principal característica es su velocidad en comparación con otras arquitecturas de redes neuronales. Originalmente, esta red se entrenó con el conjunto de datos de ImageNet que contiene alrededor de 14 millones de imágenes y tiene un mejor rendimiento que el modelo Resnet50 en términos de velocidad de procesamiento, tanto en el proceso de entrenamiento como al realizar predicciones en tiempo real. Para el modelo propuesto, el proceso de entrenamiento toma tiempos en el orden de las horas, mientras que el proceso de predicciones no toma más que unos pocos segundos.
MobileNet se basa principalmente en dos bloques, un bloque de convolución estándar (*standard convolution*) y un bloque de convolución separable en profundidad (*depthwise separable convolution*). El primer bloque consiste en una capa de convolución estándar, una capa de normalización por lotes (*batch normalization*) y una unidad lineal rectificadora (ReLU). El segundo bloque consiste en una capa de convolución en profundidad (*depthwise convolution*), seguida de una capa de normalización por lotes, una ReLU, una convolución puntual (*pointwise convolution*) y, de nuevo, una capa de normalización por lotes y una ReLU.
La combinación de estos bloques forma la arquitectura original de MobileNet representada en la Figura 2.11, sobre la cual se tiene una capa totalmente conectada con activación softmax que la convierte en un clasificador multiclase de 1000 clases.
La ventaja que MobileNet nos brinda es que utiliza convoluciones separables en profundidad para reducir el tiempo de procesamiento de las imágenes, en lugar de usar el proceso de convolución estándar, que en un solo paso realiza un proceso de filtrado y combina sus entradas para obtener un nuevo conjunto de salidas. La Figura 2.12 muestra cómo funciona una convolución estándar, donde D_k es el tamaño del núcleo, suponiendo que este es cuadrado, y M y N son el número de canales de entrada y salida, respectivamente.
Por otro lado, la convolución en profundidad aplica un filtro a cada canal de la imagen, es decir, si es una imagen en color, tendrá tres canales (rojo, verde y azul), como se puede ver en la Figura 2.13. Después de aplicar los filtros, es necesario llevar a cabo una combinación

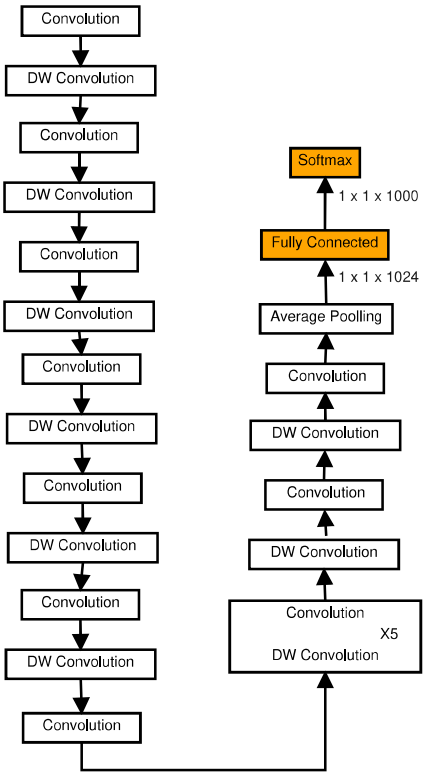


Figura 2.11: Resumen de la arquitectura de MobileNet

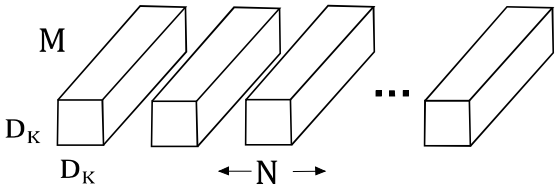


Figura 2.12: Proceso de Convolución Estándar [25]

lineal de la salida de la capa de convolución en profundidad en la capa de convolución puntual, como se muestra en la Figura 2.14. Estos dos pasos reducen el tiempo de cálculo de aproximadamente 8 a 9 veces el tiempo de una convolución estándar [25].
Existen parámetros importantes en la implementación del modelo cuya operación es necesario explicar.

2.3.1. ALGORITMO DE OPTIMIZACIÓN

Se utilizó el algoritmo de optimización llamado Adam, basado en el algoritmo de Descenso de Gradiente Estocástico (SGD). A diferencia del algoritmo SGD, Adam puede variar la tasa de aprendizaje a lo largo del proceso de capacitación para obtener un modelo de mejor

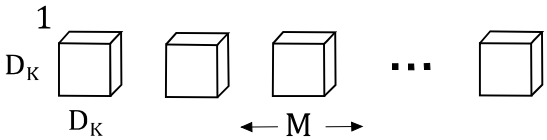


Figura 2.13: Capa de Convolución en Profundidad [25]

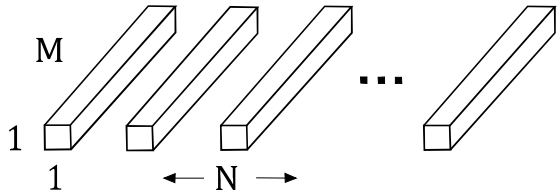


Figura 2.14: Capa de Convolución Puntual [25]

rendimiento.

2.3.2. TASA DE APRENDIZAJE

La tasa de aprendizaje es el parámetro que controla la variación de los pesos de la red para cada época de entrenamiento [28]. Se usó un valor de 0.001. Cuando la red no mejora su rendimiento a lo largo de las épocas, es necesario reducir la velocidad de aprendizaje, por lo que la red mejorará hasta las etapas finales de su entrenamiento. Se estableció una reducción en la tasa de aprendizaje en un factor de 0.5 si la red no obtiene una mejora en 4 épocas con respecto a su precisión categórica de validación. Finalmente, entrenó la red por un máximo de 50 épocas. El parámetro `mode` del Segmento de código 2.3 nos indica que la tasa de aprendizaje será reducida siempre y cuando el parámetro monitoreado haya dejado de crecer. En este caso se utilizó la precisión del modelo como parámetro de monitorización.

Segmento de código 2.3: Reducción de la tasa de aprendizaje

```
reduce_lr = ReduceLROnPlateau(monitor='val_categorical_accuracy',
                              factor=0.5, patience=4,
                              verbose=1,
                              mode='max',
                              min_lr=0.0000001)
```

2.3.3. FUNCIÓN DE PÉRDIDA

Es un parámetro muy importante para estimar el error de nuestro modelo y así poder mejorarlo. Dado que nuestro problema es un problema de clasificación, la función de pérdida que se utilizó es la entropía cruzada categórica (*categorical crossentropy*), que es la función óptima para esquemas de clasificación multiclase.

Segmento de código 2.4: Definición de algoritmo de optimización y función de pérdida

```
model.compile(Adam(lr=0.001),
              loss='categorical_crossentropy',
              metrics=[categorical_accuracy,
                       top_2_accuracy,
                       top_3_accuracy])
```

En el Segmento de código 2.4 se puede observar la implementación de los parámetros descritos, además de la definición de las métricas de precisión categórica para el caso del modelo multiclase. Se puede notar también las métricas `top_2_accuracy` y `top_3_accuracy`, que nos brindan información acerca de los valores más altos de precisión que el modelo alcanzó entre todas las predicciones realizadas.

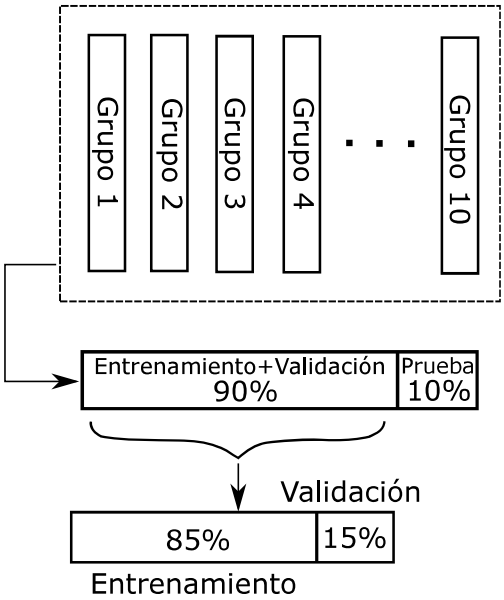


Figura 2.15: Esquema Double k-Fold Cross Validation

2.4. Experimentos

Se obtuvo el resultado óptimo para este trabajo a través de múltiples experimentos, variando diversos factores que nos permitieron obtener una alta precisión.
Primero, se utilizó el enfoque *Double Stratified K-Fold Cross Validation*, que permite evaluar el rendimiento del modelo con nuevos datos. Bajo este enfoque, todo el conjunto de datos se divide en 10 grupos, uno para prueba y los grupos restantes para entrenamiento. Luego, se separó un porcentaje del conjunto de entrenamiento para usarlo como un conjunto de validación. Se utilizó el conjunto de validación para medir el rendimiento del modelo y para evitar el sobreajuste. La Figura 2.15 muestra el esquema de validación cruzada doble que se implementó para este trabajo, especificando los porcentajes utilizados para cada parte del proceso. Este procedimiento nos permite obtener una idea más realista del rendimiento del modelo [28].
Para llevar a cabo el proceso de validación cruzada, se utilizó la función `StratifiedKFold` del Segmento de código 2.5, en la cual se puede definir el número grupos a utilizar en la validación, y además se puede modificarla para poder dividir a su vez el conjunto de datos de prueba y así obtener una validación cruzada doble. Por medio de la obtención de índices para cada imagen del conjunto, las imágenes son distribuidas en sus correspondientes grupos.

Segmento de código 2.5: Distribución de imágenes para validación cruzada

```
kf = StratifiedKFold(n_splits=10, random_state=101, shuffle=True)
kf.get_n_splits(df, y)
j=1

for train_index, test_index in kf.split(df,y):

        print("TRAIN:", train_index, "TEST:", test_index)
```

```
        X_train, X_test = df.iloc[train_index], df.iloc[test_index]
        y_train, y_test = y[train_index], y[test_index]

        df_train, df_val = train_test_split(X_train,
                                            test_size=0.15,
                                            random_state=101,
                                            stratify=y_train)

        df_test = X_test
```

Las curvas ROC fueron muy útiles para determinar el rendimiento de nuestro modelo de una manera más confiable. Nuestra red devuelve probabilidades de pertenecer a una categoría. Cuando la red predice un caso positivo que de hecho es positivo, esto se conoce como un verdadero positivo (TP), pero si la predicción dice que es positivo pero en realidad es negativo, esto se llama falso positivo (FP). La curva ROC se obtiene relacionando en un gráfico la tasa de falsos positivos (FPR) en el eje X y la tasa de verdaderos positivos (TPR) en el eje Y. Los diferentes valores de FPR y TPR se obtienen variando un umbral de clasificación que tiene valores entre 0 y 1.
El parámetro TN de la Ecuación 2.2 se conoce como verdadero negativo, lo que indica que la red predijo un caso negativo que de hecho es negativo.
El área bajo la curva ROC (AUC) es nuestro indicador del rendimiento de la red, ya que cuanto más cerca está de 1, mejor es la capacidad de la red para distinguir entre las diferentes categorías [29].

$$\text{TPR} = \frac{TP}{TP + FN} \tag{2.1}$$

$$\text{FPR} = \frac{FP}{TN + FP} \tag{2.2}$$

Para comparar nuestro clasificador multiclase, también se implementó un clasificador binario, tomando las categorías bkl, df y nv como benignas (de aquí en adelante etiquetadas como ben) y la categoría mel como maligna. Para el clasificador binario seguimos el mismo esquema de validación cruzada doble.

3. RESULTADOS Y DISCUSIÓN (APLICACIÓN METODOLÓGICA)

En esta sección, se presentan los resultados de una evaluación experimental para ayudar a estimar los factores clave que influyen en el desempeño de MobilNet en la tarea de clasificación de melanoma.

Como primer resultado del modelo se obtiene la matriz de confusión de la Figura 3.1, en donde es apreciable la efectividad del entrenamiento. Sin embargo, también es notorio el alto grado de confusión entre las categorías mel y nv, lo que nos indica que la red ha identificado un gran número de características similares entre las lesiones pertenecientes a estas dos categorías.

La Figura 3.2 muestra las características internas aprendidas por MobilNet, cada punto es una representación bidimensional (t-SNE [30]) de imágenes de lesiones de la piel de la Tabla 2.2, coloreado por categorías, donde se puede ver una clara separación entre los grupos *bcc*, *df* y *vasc* y un alto nivel de confusión entre *nv* y el resto de categorías, especialmente *mel*. La Figura 3.2 demuestra la efectividad de cómo Mobilenet organiza la información de la imagen de la lesión de la piel cuando se usa un enfoque multiclase, visualizando las decisiones de clasificación antes de la salida final softmax.

La Figura 3.4 muestra las curvas de aprendizaje del conjunto de entrenamiento y validación para el modelo multiclase. Se puede ver, tanto en las curvas de entrenamiento como en las de validación, que las pérdidas se redujeron cuando MobileNet comenzó a entrenar, esto se debe a un entrenamiento previo con Imagenet, lo que significa que las capas superficiales de MobileNet tuvieron un alto nivel de extracción de características cuando se inició el entrenamiento. Si bien se puede ver ocasionalmente vibraciones en la pérdida de entrenamiento, se puede apreciar que la pérdida de validación sigue disminuyendo.

Durante la etapa de capacitación, se pudo obtener un valor promedio de precisión categórica de validación de 88.02 %, mientras que el valor de precisión categórica de prueba fue de 87.48 %. En cuanto al clasificador binario, se obtuvo un valor promedio de precisión binaria

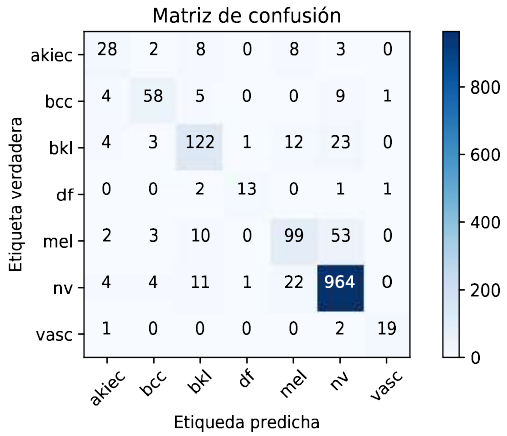


Figura 3.1: Matriz de confusión para el modelo multiclase

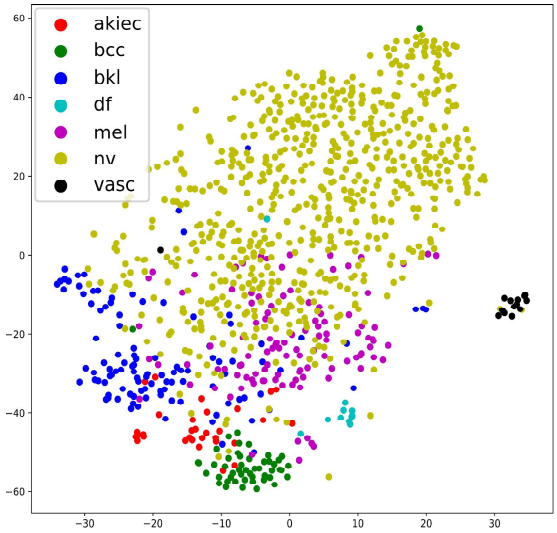


Figura 3.2: Representación t-SNE del modelo multiclase

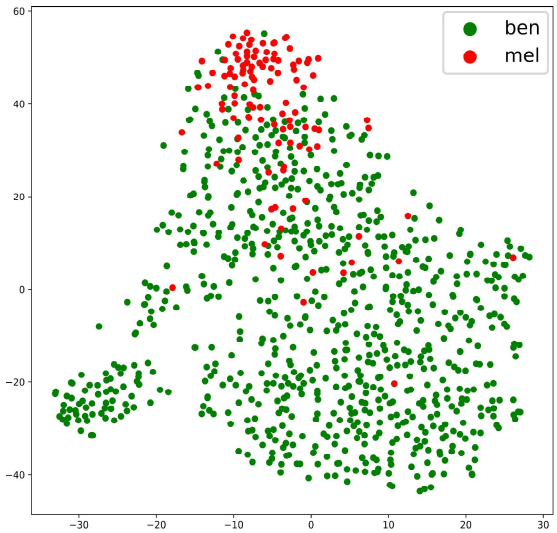


Figura 3.3: Representación t-SNE del modelo binario

de validación de 92.72 % y un valor de precisión binaria de prueba de 92.87 %.
A partir de estos valores de precisión, se puede suponer que el clasificador binario supera al clasificador multiclase, pero esto se debe a que el clasificador binario se entrenó con un número menor de clases. En términos generales, el indicador más confiable del rendimiento de la red son las métricas AUC y ROC que se describieron en la sección 2.4.
Para evaluar la calidad de salida de MobileNet, se muestra en la Fig. 3.5 la curva Característica Operativa del Receptor (ROC) para el enfoque multiclase, donde el eje Y es la tasa de verdaderos positivos y el eje X es la tasa de falsos positivos de nuestro modelo. Debido a nuestro enfoque multiclase, la Figura 3.5 tiene siete curvas ROC, una por clase. Observe que para la clase de melanoma, se logró un AUC de 0,95.

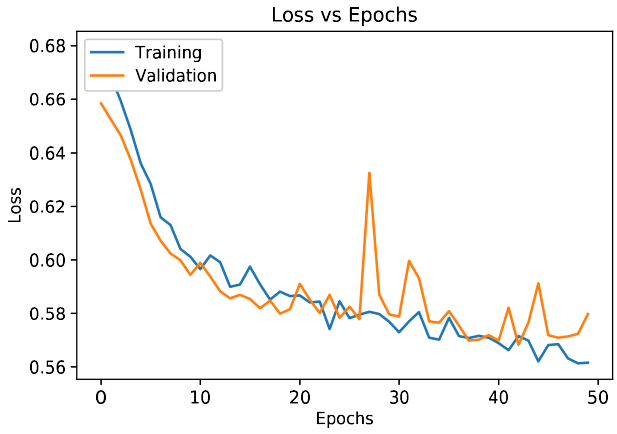


Figura 3.4: Valor de la función de pérdida para el modelo multiclase

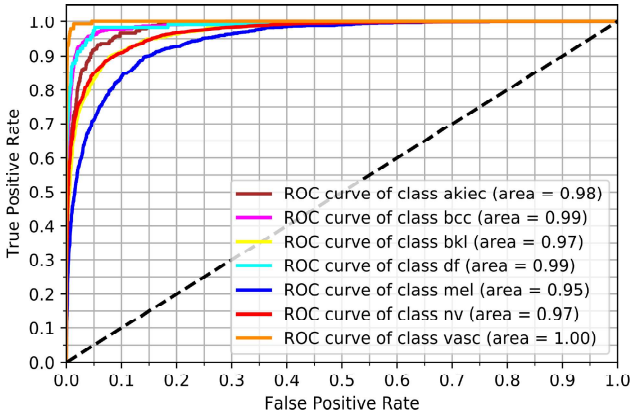


Figura 3.5: Curva ROC para el modelo multiclase

Segmento de código 3.1: Curvas ROC multiclase

```
# 7 clases por clasificar
n_classes = 7

y_test = label_binarize(test_labels, classes=[0, 1, 2, 3, 4, 5, 6])
# Grosor de la línea
lw = 2

# Cómputo de la curva ROC con su respectiva área
fpr = dict()
tpr = dict()
thr = dict()
roc_auc = dict()
for i in range(n_classes):
    fpr[i], tpr[i], thr[i] = roc_curve(y_test[:, i], predictions[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])
```

En el Segmento de código 3.1 se encuentra la implementación de las curvas ROC para cada una de las clases, en donde se definen los diferentes umbrales, además de los valores de TPR y FPR con la ayuda de los valores almacenados en la variable `predictions` obtenidos después del proceso de entrenamiento.

También se trazó la curva ROC para el enfoque de clasificación binaria en comparación con

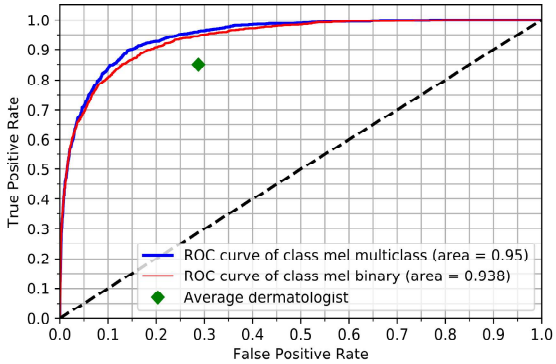


Figura 3.6: El punto verde indica el rendimiento dermatológico promedio según [2]

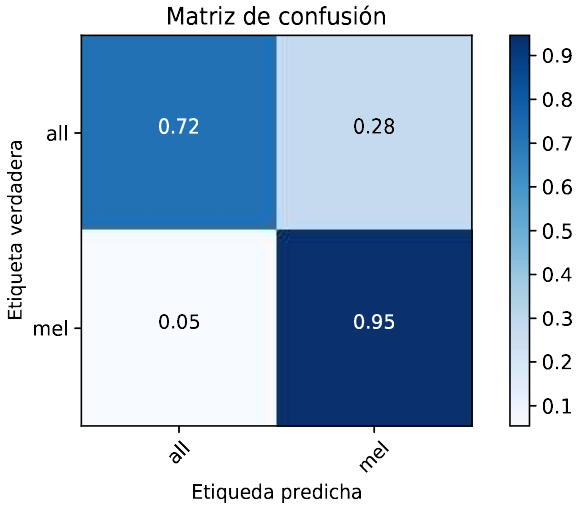


Figura 3.7: Matriz de confusión binaria

la curva ROC del melanoma del modelo multiclase de la Figura 3.6 para ilustrar la mejora obtenida cuando se usa el enfoque multiclase. Observe en la Figura 3.6 que ambos enfoques superan el rendimiento promedio dermatológico de acuerdo con [2]. Sin embargo, es importante tener en cuenta que a pesar de que nuestro AUC es comparable al obtenido en [2], ellos abordaron un problema de clasificación mucho más difícil ya que se entrenaron en un gran conjunto de datos de cientos de enfermedades de la piel. Aquí solo se considera siete enfermedades y un conjunto de datos mucho más pequeño. Además, también realizaron una validación de conjuntos de datos cruzados en lugar del doble enfoque de validación cruzada que se utilizó.

Si se analiza la Figura 3.6 se puede notar que la tasa de falsos positivos se aproxima a un valor de 0.28, valor que se ha utilizado como referencia para obtener la matriz de confusión binaria de la Figura 3.7 a partir de la matriz multicalse.

Cada punto en la curva ROC representa una diferente matriz de confusión, por lo que utilizando al rendimiento dermatológico de la Figura 3.6 como referencia se nota una ventaja en cuanto a la tasa de verdaderos positivos del modelo, es decir, aproximadamente 0.85 en el rendimiento dermatológico contra un 0.95 obtenido del modelo multiclase desarrollado en

este proyecto.
Adicionalmente, debido al enfoque binario de la matriz de confusión de la Figura 3.7, es posible obtener valores adicionales para medir el desempeño de la red como la *sensibilidad* y la *especificidad*.

Sensibilidad: La sensibilidad es la habilidad que tiene nuestro modelo de predecir correctamente los casos que poseen la condición cutánea [31]. La tasa de verdaderos positivos de la matriz de confusión representa a la sensibilidad y tiene un valor de 0.95 para este caso.

Sensibilidad: La especificidad, al contrario de la sensibilidad, es la habilidad del modelo de identificar correctamente los casos que no poseen la condición cutánea [31]. En la matriz de confusión, la tasa de verdaderos negativos representa a la especificidad con un valor de 0.72.

3.1. NOTEBOOK INTERACTIVO

Como resultado final del proyecto, se ha implementado una interfaz interactiva (*notebook*) desarrollada en la aplicación web *Jupyter Notebook*, la cual consta de varios módulos que se aprecian en la Figura 3.8.
En el primer módulo se tiene una introducción acerca del propósito de la herramienta y estadísticas acerca del melanoma. Además se tiene información importante acerca de librerías necesarias para el correcto funcionamiento de la herramienta.
El segundo módulo proporciona información acerca de la CNN utilizada, junto con las modificaciones que sufrió esta red.
El tercer módulo brinda información de la base de datos utilizada para entrenar el modelo (HAM10000), y también se despliega una breve descripción de cada afección presente en la base.
Finalmente se aprecia el último módulo, en donde se encuentran dos botones. El primer botón nos permite cargar una imagen almacenada en el disco de nuestro ordenador, y el segundo botón permite hacer la predicción que clasificará a la imagen en una de las 7 categorías.
Al dar click en el botón *Upload* se desplegará una ventana como la de la Figura 3.9, en la cual se podrá escoger una imagen para analizar, en la ruta en la que esta se encuentre almacenada.
Seguidamente, al dar click en *Predecir*, se podrá tener una afección maligna que se mostrará en rojo como en la Figura 3.10 o una afección benigna que se mostrará en verde como en la Figura 3.11.

Sistema de Detección de Melanoma

Módulo 1

El melanoma de la piel es una de las afecciones que está teniendo mayor incidencia a nivel mundial y en nuestro país está siendo diagnosticado cada vez con mayor frecuencia debido al alto índice de radiación ultravioleta que se ha presentado en los últimos años. Específicamente, en la ciudad de Quito se ha registrado la tasa más alta de incidencia de cáncer de piel y se estima que esta tendencia vaya en aumento, pues en comparación con el último registro de tumores ocasionados debido a la presencia de afecciones de la piel, ha habido un incremento del 57% en registros más actuales.

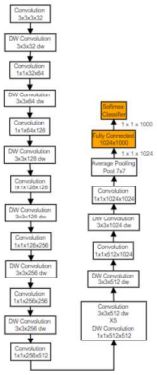
Esta herramienta ha sido desarrollada con el propósito de ayudar a los profesionales de la salud y a todo tipo de usuarios en general a detectar posibles afecciones cutáneas, de una manera rápida e intuitiva. La herramienta es capaz de realizar predicciones entre distintas afecciones, entre las cuales se encuentra el **Melanoma**.

- ***Nota:*** *Esta herramienta requiere la previa instalación de los siguientes módulos para su correcto funcionamiento:*
  - *ipywidgets*
  - *fileupload*
  - *pillow*

Out[2]: Modelo

Módulo 2

La CNN impementada fue ***MobileNet***. El modelo ha sido modificado en sus **dos últimas capas** para realizar predicciones entre **7 diferentes categorías**.



Módulo 3

El modelo fue entrenado con la base de datos **HAM10000** que cuenta con distintos tipos de enfermedades de la piel:

Afección	Imágenes totales
Queratosis Actínica	237
Carcinoma de Células Basales	514
Queratosis Benigna	1099
Dermatofibroma	115
Nevos Melanocíticos	6705
Melanoma	1113
Lesiones Cutáneas Vasculares	142
Total	10015

1. **Queratosis Actínica (akiec):** Es una variante no invasiva del carcinoma de células escamosas, y puede ser tratada localmente por medio de cirugía. Sin embargo, si la enfermedad progresa se puede convertir en un carcinoma invasivo.
2. **Carcinoma de Células Basales (bcc):** Es una variación del cáncer de piel epitelial, pero esta no suele resultar en metástasis. Además se puede presentar en variedad de formas (plana, nodular, pigmentada o quística).
3. **Queratosis Benigna (bkl):** Es en realidad una condición genérica, pues está compuesta por tres subgrupos que dermatoscópicamente lucen diferentes pero biológicamente son similares, por lo que generalmente se los agrupa bajo una misma denominación histopatológica.
4. **Dermatofibroma (df):** Es una condición cutánea que se considera benigna y se presenta a partir de la proliferación o inflamación de un trauma mínimo. La forma típica en que se aprecia dermatoscópicamente es con líneas reticulares en su periferia con un parche central blanco que indica fibrosis.
5. **Nevos Melanocíticos (nv):** Se trata de neoplasias de melanocitos de carácter benigno y que se pueden presentar en una infinidad de variantes. Difieren del melanoma en la simetría que presentan, tanto en color como estructura.
6. **Melanoma (mel):** Al igual que los nevos melanocíticos, el melanoma también es una neoplasia derivada de los melanocitos, pero en este caso de carácter maligno. El melanoma puede ser curado si se los detecta en etapas tempranas y se lo extirpa mediante escisión quirúrgica simple. Su tratamiento depende del sitio anatómico en donde se ubique.
7. **Lesiones Cutáneas Vasculares (vasc):** Es un conjunto de condiciones cutáneas entre las que se incluyen angiomas cereza, angioqueratomas, granulomas piógenos e incluso hemorragias. Estos angiomas se caracterizan por sus colores rojos o púrpuras y se los conoce también como terrones rojos o lagunas.

Diagnóstico

Es importante destacar que el único método certero para detectar una afección cutánea es mediante verificación patológica. Por lo tanto esta herramienta de diagnóstico temprano únicamente proporciona una pauta para llevar a cabo las acciones que que el usuario crea pertinentes.

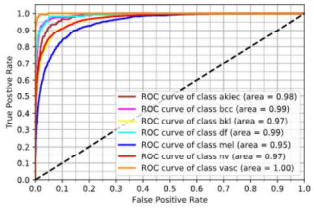
Out[3]: Algoritmo de importación

Upload (0)

Out[5]: Algoritmo de predicción

Predecir

Módulo 4



El **área bajo la curva (AUC)** indica el rendimiento del modelo para cada categoría. Mientras más cercano es este valor a 1, mejor es el rendimiento del modelo.

Figura 3.8: Módulos de la interfaz interactiva

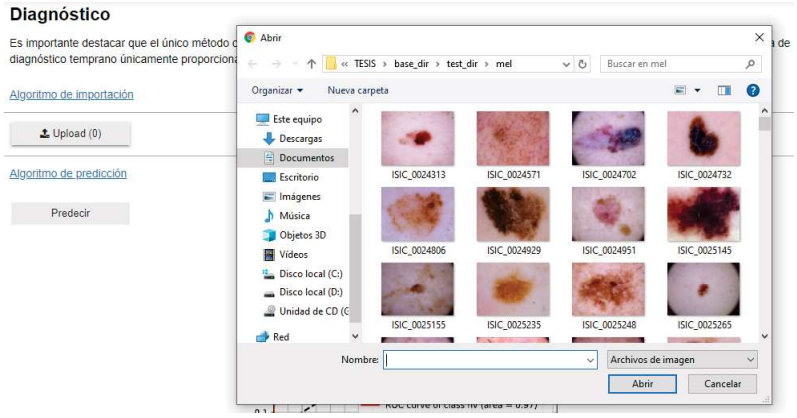


Figura 3.9: Ventana de carga de imágenes

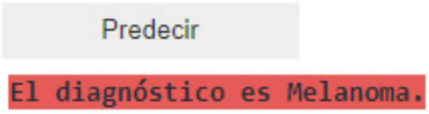


Figura 3.10: Predicción de afección maligna

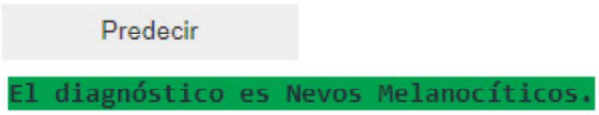


Figura 3.11: Predicción de afección benigna

4. CONCLUSIONES Y RECOMENDACIONES

4.1. CONCLUSIONES

Fue importante realizar un breve estudio acerca de los aspectos más importantes del melanoma y los métodos visuales de detección que utilizan los dermatólogos para detectarlo, pero también es importante entender que el estudio a profundidad de la enfermedad no fue el objetivo principal de este proyecto. Este estudio fue enfocado a la implementación de un método alternativo a los métodos visuales, capaz de proporcionar un mayor nivel de confiabilidad de una forma intuitiva para el usuario.

En la actualidad, cada vez es más frecuente encontrar dispositivos móviles con gran capacidad de procesamiento capaces de llevar a cabo tareas bastante complejas. Sin embargo, estos dispositivos como teléfonos inteligentes (*smartphones*) o tabletas aún no son capaces de realizar las tareas que un ordenador actual puede realizar con total facilidad. Tomando en cuenta que la gente busca agilizar su tiempo añadiendo movilidad a sus actividades cotidianas, nuestro sistema de detección de melanoma fue diseñado para plataformas con recursos limitados como los mencionados. Esta ventaja es proporcionada gracias al uso de la CNN MobileNet, que ofrece un costo computacional más bajo que otras arquitecturas de redes neuronales. Esto, a su vez, podría permitir que los dispositivos con recursos limitados obtengan un buen rendimiento al hacer predicciones en tiempo real.

Los resultados presentados en este trabajo resaltan dos estrategias para mejorar el rendimiento del modelo de clasificación de melanoma, el aumento de datos en un conjunto de datos no equilibrado y un enfoque multiclase para abordar un problema de clasificación binaria. Si se comparan los dos casos que se estudiaron en este documento (binario y multiclase), el clasificador multiclase tiene un mejor rendimiento que confirma los hallazgos en trabajos anteriores.

Para obtener un rendimiento óptimo del modelo, la red debe estar entrenada con un gran número de imágenes. Por ejemplo, se entrenó a MobileNet usando la base de datos ImageNet, que tiene millones de imágenes. Nuestra red no fue entrenada de manera convencional, es decir, se utilizó la base de datos HAM10000, que por sí sola no sería lo suficientemente grande como para entrenar la red correctamente. Teniendo en cuenta estos detalles, la implementación de un proceso de aumento de datos bajo un esquema de aprendizaje por transferencia fue crucial no solo para aumentar el tamaño de la base de datos sino también para balancear el número de imágenes por categoría.

4.2. RECOMENDACIONES

Los resultados de esta aplicación ya han sido implementados en una interfaz que no requiere ningún tipo de conexión hacia redes específicas o al internet para poder trabajar correctamente (*app standalone*), pero además, en futuros trabajos, es factible desarrollar una

aplicación para dispositivos móviles que implemente los beneficios de este estudio, probando las ventajas que se podrían descubrir a lo largo de su desarrollo. A su vez, los usuarios tendrán a mano una herramienta que les permitirá obtener diagnósticos inmediatos y tomar las decisiones adecuadas con respecto al cuidado de su piel.

Sin embargo, se debe elaborar una investigación más profunda para analizar el desempeño de la implementación en el mundo real.

Como parte de estas investigaciones se recomienda la recolección de bases de datos más grandes y completas que permitirán implementar modelos mucho más robustos y confiables. Por ejemplo, se nota que las bases de datos carecen de muestras correspondientes a pacientes con piel obscura, debido a que este tipo de lesiones son más difíciles de diferenciar en este tipo de piel. Contar con imágenes de muestra de este tipo podría lograr una gran contribución al desempeño en futuros trabajos que se desarrollen tomando como punto de partida a este proyecto.

5. REFERENCIAS BIBLIOGRÁFICAS

[1] "Quito con la tasa de incidencia más alta de cáncer de piel en Ecuador." [Online]. Disponible en: https://www.redaccionmedica.ec/secciones/salud-publica/quito-con-la-tasa-de-incidencia-m-s-alta-de-c-ncer-de-piel-en-ecuador-90411

[2] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun, "Dermatologist-level classification of skin cancer with deep neural networks," *Nature*, vol. 542, no. 7639, p. 115, 2017.

[3] P. Zaballosa, C. Carrerab, S. Puigb, and J. Malvehyb, "Criterios dermatoscópicos para el diagnóstico del melanoma," p. 16.

[4] E. Valle, M. Fornaciali, A. Menegola, J. Tavares, F. V. Bittencourt, L. T. Li, and S. Avila, "Data, depth, and design: learning reliable models for melanoma screening," *arXiv preprint arXiv:1711.00441*, 2017.

[5] P. Tschandl, C. Rosendahl, and H. Kittler, "The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions," *Scientific Data*, vol. 5, p. 180161, Aug. 2018.

[6] C. Barata, M. E. Celebi, and J. S. Marques, "A survey of feature extraction in dermoscopy image analysis of skin cancer," *IEEE J. Biomed. Health Inform.*, vol. 23, no. 3, pp. 1096–1109, May 2019.

[7] F. Perez, C. Vasconcelos, S. Avila, and E. Valle, "Data augmentation for skin lesion analysis," in *OR 2.0 Context-Aware Operating Theaters, Computer Assisted Robotic Endoscopy, Clinical Image-Based Procedures, and Skin Image Analysis*. Springer, 2018, pp. 303–311.

[8] S. Kornblith, J. Shlens, and Q. V. Le, "Do better imagenet models transfer better?" in *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2019, pp. 2661–2671.

[9] F. Perez, S. Avila, and E. Valle, "Solo or ensemble? choosing a cnn architecture for melanoma classification," in *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 0–0.

[10] "Melanoma - Síntomas y causas - Mayo Clinic." [Online]. Disponible en: https://www.mayoclinic.org/es-es/diseases-conditions/melanoma/symptoms-causes/syc-20374884

[11] Marmolejo and Villegas, "Álgebra lineal," pp. 157–163.

[12] A. M. Munt, "Introducción a los modelos de redes neuronales artificiales El Perceptrón simple y multicapa," p. 47.

[13] A. Misra, “Classifying Skin Lesions with Convolutional Neural Networks,” Dec. 2018. [Online]. Disponible en: https://towardsdatascience.com/classifying-skin-lesions-with-convolutional-neural-networks-fc1302c60d54

[14] “CONCEPTOS BÁSICOS SOBRE REDES NEURONALES.” [Online]. Disponible en: http://grupo.us.es/gtocoma/pid/pid10/RedesNeuronales.htm

[15] “Curso intensivo en Deep Learning,” May 2018. [Online]. Disponible en: https://unipython.com/curso-en-multicapa-perceptrones/

[16] I. A. o. August 22 and 2017, “A Friendly Introduction to Convolutional Neural Networks.” [Online]. Disponible en: https://hashrocket.com/blog/posts/a-friendly-introduction-to-convolutional-neural-networks

[17] “Convolutional Neural Networks (CNN): Step 1(b) - ReLU Layer - Blogs SuperDataScience - Big Data | Analytics Careers | Mentors | Success.” [Online]. Disponible en: https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-step-1b-relu-layer

[18] D. D. Sarkar, “A Comprehensive Hands-on Guide to Transfer Learning with Real-World Applications in Deep Learning,” Nov. 2018. [Online]. Disponible en: https://towardsdatascience.com/

[19] “Transfer Learning : Approaches and Empirical Observations.” [Online]. Disponible en: https://hackernoon.com/transfer-learning-approaches-and-empirical-observations-efeff9dfeca6

[20] S. N. Kasturi, “Underfitting and Overfitting in machine learning and how to deal with it !!!” Jun. 2019. [Online]. Disponible en: https://towardsdatascience.com/underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6fe4a8a49dbf

[21] V. Yadav, “Why dropouts prevent overfitting in Deep Neural Networks,” Nov. 2016. [Online]. Disponible en: https://medium.com/@vivek.yadav/why-dropouts-prevent-overfitting-in-deep-neural-networks-937e2543a701

[22] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting,” p. 30.

[23] S. Deoras, “Tensorflow Vs Theano : What Do Researchers Prefer As An AI Framework,” Dec. 2017. [Online]. Disponible en: https://www.analyticsindiamag.com/tensorflow-vs-theano-researchers-prefer-artificial-intelligence-framework/

[24] “Home - Keras Documentation.” [Online]. Disponible en: https://keras.io/

[25] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications,” *arXiv:1704.04861 [cs]*, Apr. 2017, arXiv: 1704.04861.

[26] “Diagnóstico Histopatológico.” [Online]. Disponible en: https://www.roche.com.mx/es_mx/Diagnstica/diagnostico-de-tejidos.html

[27] B. Rocca, “Handling imbalanced datasets in machine learning,” Mar. 2019. [Online]. Disponible en: https://towardsdatascience.com/handling-imbalanced-datasets-in-machine-learning-7a0e84220f28

[28] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[29] T. Fawcett, “An introduction to roc analysis,” *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861 – 874, 2006, rOC Analysis in Pattern Recognition.

[30] L. v. d. Maaten and G. Hinton, “Visualizing data using t-sne,” *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[31] A. G. Lalkhen and A. McCluskey, “Clinical tests: sensitivity and specificity,” *Continuing Education in Anaesthesia Critical Care & Pain*, vol. 8, no. 6, pp. 221–223, Dec. 2008. [Online]. Disponible en: https://linkinghub.elsevier.com/retrieve/pii/S1743181617304572

ANEXOS

ANEXO A. tSNE.py

ANEXO B. learning_curves.py

ANEXO C. multiclass_ROC.py

ANEXO D. model_test.py
(El código se encuentra en el CD adjunto a este documento)

ANEXO E. preprocessing_test.py
(El código se encuentra en el CD adjunto a este documento)

ANEXO F. cross_validation.py
(El código se encuentra en el CD adjunto a este documento)

ANEXO A: tSNE.py

```
from sklearn.manifold import TSNE

X = features
y = test_labels
target_names=['akiec', 'bcc', 'bkl', 'df', 'mel', 'nv', 'vasc']

tsne = TSNE(n_components=2, perplexity=30.0, early_exaggeration=12.0,
    random_state=0)

X_2d = tsne.fit_transform(X)

target_ids = range(len(target_names))

from matplotlib import pyplot as plt
plt.figure(figsize=(10, 10))
colors = 'r', 'g', 'b', 'c', 'm', 'y', 'k'
for i, c, label in zip(target_ids, colors, target_names):
    plt.scatter(X_2d[y == i, 0], X_2d[y == i, 1], c=c, label=label)
plt.legend()
plt.show()
```

ANEXO B: learning_curves.py

```
from keras.models import Sequential
from keras.layers import Dense
import matplotlib.pyplot as plt
import numpy
import json

acc=[]
val_acc=[]
loss=[]
val_loss=[]

for i in range(50):

    acc.append(history_dict['acc'][i])
    val_acc.append(history_dict['val_acc'][i])
    loss.append(history_dict['loss'][i])
    val_loss.append(history_dict['val_loss'][i])

# hacemos las grafica 1 precision
plt.plot(acc)
plt.plot(val_acc)
plt.title('Accuracy vs Epochs')
plt.ylabel('Accuracy')
plt.xlabel('Epochs')
plt.legend(['Training', 'Validation'], loc='upper left')
plt.savefig('ACCvsEP.png')
plt.show()
# hacemos las grafica 2 perdidas
plt.plot(loss)
plt.plot(val_loss)
plt.title('Loss vs Epochs')
plt.ylabel('Loss')
plt.xlabel('Epochs')
plt.legend(['Training', 'Validation'], loc='upper left')
plt.savefig('LOSSvsEP.png')
plt.show()
```

ANEXO c: multiclass_ROC.py

```
1  from sklearn.datasets import make_classification
2  from sklearn.preprocessing import label_binarize
3  import numpy as np
4  from scipy import interp
5  import matplotlib.pyplot as plt
6  from itertools import cycle
7  from sklearn.metrics import roc_curve, auc
8
9
10 # 7 clases a clasificar
11 n_classes = 7
12
13 y_test = label_binarize(test_labels_MUL, classes=[0, 1, 2, 3, 4, 5, 6])
14
15 lw = 2
16
17 # Cómputo de la curva ROC y el AUC para cada clase
18 fpr = dict()
19 tpr = dict()
20 thr = dict()
21 roc_auc = dict()
22 for i in range(n_classes):
23     fpr[i], tpr[i], thr[i] = roc_curve(y_test[:, i], predictions_MUL[:,
           i])
24     roc_auc[i] = auc(fpr[i], tpr[i])
25
26 # Compute micro-average ROC curve and ROC area
27 fpr["micro"], tpr["micro"], _ = roc_curve(y_test.ravel(),
       predictions_MUL.ravel())
28 roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])
29
30 all_fpr = np.unique(np.concatenate([fpr[i] for i in range(n_classes)]))
31
32 mean_tpr = np.zeros_like(all_fpr)
33 for i in range(n_classes):
34     mean_tpr += interp(all_fpr, fpr[i], tpr[i])
35
36 mean_tpr /= n_classes
37
38 fpr["macro"] = all_fpr
39 tpr["macro"] = mean_tpr
40 roc_auc["macro"] = auc(fpr["macro"], tpr["macro"])
41
42 # Graficar todas las curvas
43 plt.figure(1)
44
45 plt.plot(fpr[0], tpr[0], color='brown', lw=lw,
46          label='ROC curve of class akiec (area = {1:0.2f})'
47                ''.format(0, roc_auc[0]))
48
49 plt.plot(fpr[1], tpr[1], color='fuchsia', lw=lw,
50          label='ROC curve of class bcc (area = {1:0.2f})'
51                ''.format(1, roc_auc[1]))
52
53 plt.plot(fpr[2], tpr[2], color='yellow', lw=lw,
```

```
         label='ROC curve of class bkl (area = {1:0.2f})'
               ''.format(2, roc_auc[2]))

plt.plot(fpr[3], tpr[3], color='aqua', lw=lw,
         label='ROC curve of class df (area = {1:0.2f})'
               ''.format(3, roc_auc[3]))

plt.plot(fpr[4], tpr[4], color='blue', lw=lw,
         label='ROC curve of class mel (area = {1:0.2f})'
               ''.format(4, roc_auc[4]))

plt.plot(fpr[5], tpr[5], color='red', lw=lw,
         label='ROC curve of class nv (area = {1:0.2f})'
               ''.format(5, roc_auc[5]))

plt.plot(fpr[6], tpr[6], color='darkorange', lw=lw,
         label='ROC curve of class vasc (area = {1:0.2f})'
               ''.format(6, roc_auc[6]))

plt.plot([0, 1], [0, 1], 'k--', lw=lw)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc="lower right")
plt.grid(b=True, which='both')
plt.minorticks_on()
major_ticks = np.arange(0, 1.01, 0.1)
minor_ticks = np.arange(0, 1, 0.05)
ax = plt.gca()
ax.set_xticks(major_ticks)
ax.set_xticks(minor_ticks, minor=True)
ax.set_yticks(major_ticks)
ax.set_yticks(minor_ticks, minor=True)
plt.savefig('ROCmult.svg')
plt.show()


# Acercamiento a la esquina superior izquierda
plt.figure(2)
plt.xlim(0, 0.3)
plt.ylim(0.6, 1)
plt.plot(fpr["micro"], tpr["micro"],
         label='micro-avg ROC (area = {0:0.2f})'
               ''.format(roc_auc["micro"]),
         color='deeppink', linestyle=':', linewidth=4)

plt.plot(fpr["macro"], tpr["macro"],
         label='macro-avg ROC (area = {0:0.2f})'
               ''.format(roc_auc["macro"]),
         color='navy', linestyle=':', linewidth=4)

colors = cycle(['aqua', 'darkorange', 'cornflowerblue'])
for i, color in zip(range(n_classes), colors):
    plt.plot(fpr[i], tpr[i], color=color, lw=lw,
             label='ROC class {0} (area = {1:0.2f})'
             ''.format(i, roc_auc[i]))
```

```python

plt.plot([0, 1], [0, 1], 'k--', lw=lw)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Some extension of Receiver operating characteristic to multi-class')
plt.legend(loc="lower right")
plt.grid(True)
plt.show()
```

ORDEN DE EMPASTADO