UNIVERSIDAD AUTÓNOMA DEL ESTADO DE HIDALGO
INSTITUTO DE CIENCIAS BÁSICAS E INGENIERÍA

MAESTRÍA EN CIENCIAS EN COMPUTACIÓN AVANZADA Y ELECTRÓNICA

TESIS

DISEÑO DE UNA ARQUITECTURA DE RED NEURONAL CONVOLUCIONAL PARA LA DETECCIÓN DE CÁNCER DE PIEL EN IMÁGENES DERMOSCÓPICAS

Para obtener el grado de

Maestro en Ciencias en Computación Avanzada y Electrónica

PRESENTA

Ing. Luis Rey Jaime Calva

Director

Dr. Omar López Ortega

Comité tutorial

Dra. Anilu Franco Arcega
Dr. Pedro Amado Miranda Romagnoli
Dr. Omar López Ortega
Dr. Obed Pérez Cortes

Mineral de la Reforma, Hgo, México, Febrero 2023

UNIVERSIDAD AUTÓNOMA DEL ESTADO DE HIDALGO
Instituto de Ciencias Básicas e Ingeniería
School of Engineering and Basic Sciences
Área Académica de Computación y Electrónica
Department of Electronics and Computer Science

Mineral de la Reforma Hidalgo, a 13 de febrero de 2023

Número de control: ICBI-AACyE/151/2023
Asunto: Autorización de impresión de tema de tesis.

MTRA. OJUKY DEL ROCIÓ ISLAS MALDONADO
DIRECTORA DE ADMINISTRACIÓN ESCOLAR DE LA UAEH

El Comité Tutorial de la **TESIS** del Programa Educativo de Posgrado titulado "**Diseño de una arquitectura de red neuronal convolucional para la detección de cáncer de piel en imágenes dermoscópicas**", realizado por el sustentante Luis Rey Jaime Calva con número de cuenta 454709 perteneciente al programa de Maestría en Ciencias en Computación Avanzada y Electrónica, una vez que ha revisado, analizado y evaluado el documento recepcional de acuerdo a lo estipulado en el Artículo 110 del Reglamento de Estudios de Posgrado, tiene a bien extender la presente:

AUTORIZACIÓN DE IMPRESIÓN

Por lo que el sustentante deberá cumplir con los requisitos del Reglamento de Estudios de Posgrado y con lo establecido en el proceso de grado vigente.

UNIVERSIDAD AUTÓNOMA DEL ESTADO DE HIDALGO

Atentamente
"Amor. Orden y Progreso"

Dr. Omar López Ortega
Director de Tesis

UNIVERSIDAD AUTONOMA DEL ESTADO DE HIDALGO
DIRECCIÓN DE ADMINISTRACIÓN ESCOLAR
TITULACIÓN
DÍA 20 MES 02 AÑO 23
RECIBIÓ
NOMBRE:
FIRMA:

Instituto de Ciencias Básicas e Ingeniería
Área Académica de Computación y Electrónica

Comité Tutorial

Dra. Anilu Franco Arcega	Presidente	UAEH
Dr. Pedro Amado Miranda Romagnoli	Secretario	UAEH
Dr. Omar López Ortega	Vocal	UAEH
Dr. Obed Pérez Cortes	Suplente	UAEH

OPC/APL

Ciudad del Conocimiento
Carretera Pachuca-Tulancingo km 4.5 Colonia Carboneras, Mineral de la Reforma, Hidalgo, México. C.P. 42184
Teléfono: +52 (771) 71 720 00 ext. 2250, 2251
Fax 2109
aacye_icbi@uaeh.edu.mx

Dedicatoria

*A **mi familia**... por el apoyo, por estar siempre al pendiente de mí y porque han colaborado en mi formación personal y académica.*

*A **mis padres**... por su amor, por su apoyo incondicional, por su paciencia, por su tiempo, por sus consejos, por los sacrificios y esfuerzos que han hecho a lo largo de mi vida, por estar conmigo siempre en los buenos y malos momentos, por creer y confiar en mí.*

*A **mi hermano** y a **mi sobrina**... por su cariño, por su apoyo y por sus consejos.*

*A **mis amigos**... por su compañía, por las palabras de aliento y por los buenos momentos de convivencia que compartimos.*

Agradecimientos

A Dios por cuidarme, por darme salud y fortaleza, por estar conmigo y guiarme a lo largo de mi vida.

A la Universidad Autónoma del Estado de Hidalgo UAEH en particular al Instituto de Ciencias Básicas e Ingeniería ICBI por abrirme sus puertas y darme la oportunidad de seguir aprendiendo y de crecer en el ámbito profesional, al personal académico de la Maestría en Ciencia en Computación Avanzada y Electrónica MCCAE que en algún momento han sido parte de mi formación profesional con sus enseñanzas, al programa de becas CONACYT por el apoyo económico.

Al Dr. Omar López por la oportuna dirección de este trabajo, por su apoyo, por su profesionalismo, por compartir sus conocimientos y su experiencia, por el tiempo y esfuerzo que ha invertido en mí durante la realización de este trabajo.

A la Dra. Heydy Castillejos por su apoyo, por compartir sus conocimientos y por el tiempo dedicado en el desarrollo de este trabajo.

A los miembros del jurado por sus recomendaciones, por sus observaciones y por el tiempo que dedicaron en la revisión de este trabajo.

A quienes de alguna manera directa o indirecta me brindaron su apoyo para concluir con éxito este trabajo.

Muchas gracias a cada uno de ustedes por ayudarme y guiarme para lograr una meta más en mi vida

Índice general

Índice de figuras

Resumen

En este proyecto de investigación se construyeron tres arquitecturas de red neuronal convolucional, con distinta secuencia de capas de convolución y agrupación, las cuales, fueron utilizadas para detectar si una lesión de piel pigmentada en imágenes dermoscópicas es de clase Benigna o Maligna. Se realizaron varios experimentos con cada arquitectura utilizando 8, 16 y 32 filtros de convolución de tamaño 3×3, 4×4 y 5×5. Se evaluaron y compararon las tres arquitecturas mediante distintas métricas, en cuyo caso, fue utilizando la métrica de exactitud que se encontró que la arquitectura diseñada con una secuencia de dos capas de convolución más una capa de agrupación, seguidas por dos capas de convolución; utilizando 8 filtros de tamaño 3×3, obtuvo el mejor desempeño con una probabilidad de 96.37 % de que el diagnóstico sea correcto. Para realizar los experimentos se utilizó el conjunto de imágenes conocido como Archivo ISIC Tschandl et al. [2018], Codella et al. [2017], Combalia et al. [2019] y Codella et al. [2019] (International Skin Imaging Collaboration, por sus siglas en inglés) del año 2019, constituido por 25,331 imágenes de entrenamiento y 8,238 imágenes de prueba. De este conjunto de imágenes se utilizaron tres muestras, constituidas por 50 % de imágenes de lesiones de piel Malignas y 50 % de imágenes de lesiones de piel Benignas. Por otro lado, también se utilizaron algunos algoritmos de clasificación clásicos tales como Árboles de Decisión, Bosques Aleatorios, entre otros. Finalmente, tomando en cuenta los requisitos de procesamiento de la arquitectura, se propone que sea desplegada en la plataforma Amazon SageMaker para poder desarrollar e integrar un sistema computacional que sirva como apoyo a los médicos especialistas en el diagnóstico de cáncer de piel.

Capítulo 1

Introducción

El aprendizaje automático implica el desarrollo de algoritmos de autoaprendizaje para obtener conocimiento de la gran cantidad de datos estructurados y no estructurados que existen actualmente, con el fin de hacer predicciones y tomar decisiones basadas en datos Raschka et al. [2016]. Hoy en día, la mayoría de las empresas y las tecnologías de consumo de alto nivel utilizan el aprendizaje profundo, el cual, se ha mostrado especialmente prometedor en el tipo de problemas sin semántica fácilmente extraíble, como imágenes, audio y datos de texto Salvaris et al. [2018]. El diagnóstico de enfermedades es otro de los usos que tiene el aprendizaje profundo. Y es que, diagnosticar una enfermedad desde etapas muy tempranas puede permitir que sea tratada de forma eficaz y oportuna originando que el paciente pueda salvar la vida. En este proyecto de investigación, se va a utilizar el enfoque del aprendizaje profundo para diseñar y construir tres arquitecturas de red neuronal convolucional, las cuales, serán evaluadas y comparadas para elegir aquella que tenga el mejor desempeño en la detección de cáncer de piel en imágenes dermoscópicas de lesiones de piel pigmentada. También se utilizaran algunos algoritmos de aprendizaje automático clásicos, tales como, Árboles de Decisión, Bosques Aleatorios, Regresión Logística, K Vecinos más Cercanos, Máquinas de Vectores de Soporte y Perceptrón Multicapa, los cuales, serán evaluados para comparar su desempeño con el obtenido por las arquitecturas propuestas.

1.1. Planteamiento del problema

El diagnóstico y tratamiento del cáncer de piel es de suma importancia y cabe mencionar algunas estadísticas relacionadas con esta enfermedad. De acuerdo a la Sociedad Americana Contra El Cáncer Society [2022] el cáncer de piel es el más común entre todos los tipos de cáncer y el melanoma conforma solo el 1 % de los casos de cáncer de piel, pero causa la gran mayoría de las muertes por este tipo de cáncer.

Por su parte, la Skin Cancer Foundation Foundation [2022] menciona que en todo el mundo:

- Cada año se diagnostican más de 13 millones de casos de cáncer de piel.
- Uno de cada tres cánceres que se diagnostican es un cáncer de piel.
- Más de 65,000 personas mueren, cada año, por culpa de este cáncer de piel.
- El 90 % de los cánceres de piel no melanoma se asocian a la exposición a la radiación ultravioleta procedente del sol.

En México, de acuerdo a Roldan [2019], el melanoma es responsable del 80 % de las muertes por cáncer de piel y va en aumento en el mundo, más que cualquier neoplasia maligna, convirtiéndose en un problema de salud.

Por otro lado, la detección de cáncer de piel utilizando imágenes digitales mediante algoritmos de clasificación clásicos puede estar influenciada de manera negativa por la forma en que se adquirieron tales imágenes, ya que, pueden existir alteraciones en la luz, imágenes tomadas desde diferentes enfoques, etc. originando que deba existir una etapa de preprocesamiento en la que se utilicen técnicas como la eliminación de ruido y la segmentación de la lesión de piel, implicando que exista cierta pérdida de información en las imágenes, impactando el desempeño de los algoritmos clásicos para llevar a cabo la clasificación. En contraste, se puede utilizar un modelo de aprendizaje profundo como las redes neuronales

convolucionales, las cuales, traen consigo una etapa de extracción de características que les permite mejorar su desempeño para clasificar las imágenes. Pero, utilizar redes neuronales convolucionales implica una flexibilidad muy amplia para construir este tipo de arquitecturas, por lo tanto, encontrar aquella que tenga el mejor desempeño puede resultar una tarea complicada.

1.2. Justificación de la propuesta

Es por ello que se propone diseñar tres arquitecturas de red neuronal convolucional, de las cuales se elegirá aquella que tenga el mejor desempeño para clasificar las imágenes de lesiones de piel pigmentada. Esto debido a que, encontrar cuál es la arquitectura de red neuronal convolucional con el mejor desempeño en la tarea de detección de cáncer de piel, resulta ser complejo, dada la gran variedad de secuencias de capas de convolución y agrupación con las que se pueden diseñar, además del número de hiperparámetros que utilizan, por ejemplo, la cantidad de filtros, el tamaño del filtro, entre otros.

1.3. Hipótesis

Es posible encontrar la arquitectura de red neuronal convolucional, entre varios diseños propuestos, que tenga el mejor desempeño en la tarea de detección de cáncer de piel en imágenes dermoscópicas y que el diagnóstico sea altamente confiable, inclusive, que el realizado por algunos algoritmos clásicos.

1.4. Objetivos

1.4.1. Objetivo general

Construir varias arquitecturas de red neuronal convolucional variando la secuencia de capas de convolución y agrupación para encontrar aquella que tenga el mejor desempeño en la tarea de detección de cáncer de piel en imágenes dermoscópicas.

1.4.2. Objetivos específicos

- Modificar la estructura de la red neuronal convolucional, utilizando distintas secuencias de capas de convolución y agrupación, para generar diversas arquitecturas.
- Experimentar cada arquitectura de red neuronal convolucional, modificando el número y el tamaño de los filtros de convolución, para validar cuáles son los valores que mejor se ajustan a cada arquitectura.
- Comparar el desempeño de cada arquitectura de red neuronal convolucional, mediante diferentes métricas de evaluación, para elegir la mejor arquitectura en la detección de cáncer de piel.
- Analizar el desempeño obtenido por las arquitecturas de red neuronal convolucional propuestas con el desempeño de algunos clasificadores clásicos, utilizando diferentes métricas de evaluación, para elegir el tipo de clasificador a emplear.
- Proponer el desarrollo de un sistema computacional, analizando diferentes plataformas, para desplegar la arquitectura de red neuronal convolucional y que sea utilizada en la práctica médica.

1.5. Desarrollo de la solución

Como meta se propone la construcción y evaluación de tres arquitecturas de red neuronal convolucional, variando en cada una de ellas la secuencia de capas de convolución y agrupación, así como, la cantidad de filtros de convolución y el tamaño de los mismos. Se evaluará el desempeño de cada arquitectura mediante las métricas de exactitud, precisión, sensibilidad, puntuación F1 y área bajo la curva. Utilizando la métrica de exactitud se va a comparar el desempeño entre las tres arquitecturas, para elegir aquella que tenga el mejor desempeño. De igual manera, se utilizarán los algoritmos de Árboles de Decisión, Bosques Aleatorios, Regresión Logística, K Vecinos más Cercanos, Máquinas de Vectores de Soporte y Perceptrón Multicapa para evaluar y comparar su desempeño con la mejor arquitectura.

1.6. Aportaciones de la tesis

Con el desarrollo de este proyecto de investigación, se descubre cuál es la arquitectura de red neuronal convolucional con el mejor desempeño en la tarea de detección de cáncer de piel en imágenes dermoscópicas entre las tres arquitecturas propuestas. Para ello, además de variar la secuencia de capas de convolución y agrupación en cada arquitectura, también se variaron dos parámetros importantes como lo son la cantidad de filtros de convolución y el tamaño de los mismos. Se graficaron y analizaron los resultados obtenidos ocupando la métrica de exactitud para elegir la arquitectura con el mejor desempeño. Por otro lado, se realizó una comparación con el desempeño que tienen algunos algoritmos clásicos, tales como árboles de decisión, bosques aleatorios, regresión logística, k vecinos más cercanos, máquinas de vectores de soporte y perceptrón multicapa.

1.7. Estructura general de la tesis

La estructura general de este proyecto de investigación consta de siete capítulos, cuyo contenido se describe a continuación.

El Capítulo 1 (Introducción) presenta el planteamiento del problema y la justificación para realizar este trabajo. Se muestra el objetivo general y objetivos específicos, además, se describe el desarrollo de la solución y las aportaciones que se realizan con este trabajo.

El Capítulo 2 (Marco teórico) describe el fundamento teórico que enmarca la investigación.

El Capítulo 3 (Herramientas utilizadas) describe las principales bibliotecas utilizadas para el desarrollo de la solución.

El Capítulo 4 (Trabajos relacionados) presenta una revisión de los trabajos que han realizado otros autores. Da una descripción de las metodologías de diseño seguidas, técnicas utilizadas y resultados que han obtenido en sus investigaciones.

El Capítulo 5 (Metodología de investigación) describe la secuencia de actividades que componen cada Fase para la clasificación de las imágenes dermoscópicas.

El Capítulo 6 (Resultados) se detallan los resultados obtenidos en cada Fase de experimentación.

El Capítulo 7 (Conclusiones generales) analiza las conclusiones y describe las líneas futuras de este trabajo.

Capítulo 2

Marco teórico

En este capítulo se revisa la teoría vista para llevar a cabo este proyecto de investigación. Se comienza por ver algunos temas relacionados al cáncer de piel, tales como, los tipos existentes, el proceso para diagnosticar está enfermedad y los factores de riesgo que originan que ocurra el cáncer de piel. Posteriormente, se ven algunos temas importantes acerca de los fundamentos de imágenes digitales que permiten entender cómo se define una imagen desde un punto de vista matemático y además, cómo se componen las imágenes a color. De igual manera, se continúa con el estudio de la mecánica del filtrado espacial como punto importante para entender el funcionamiento de la redes neuronales convolucionales. Por un lado, se revisa la teoría sobre el funcionamiento de los algoritmos de Árboles de Decisión, Bosques Aleatorios, Regresión Logística, K Vecinos más Cercanos, Máquinas de Vectores de Soporte y Perceptrón Multicapa; y por otro lado, se busca entender el funcionamiento de las capas de convolución y agrupación que componen a las redes neuronales convolucionales. Se analizan las ventajas que las redes convolucionales tienen sobre los algoritmos clásicos y el porque mejoran en cuanto a desempeño. Finalmente, se ven algunas de las métricas utilizadas para evaluar el desempeño de los algoritmos de clasificación, tales como, exactitud, precisión, sensibilidad, puntuación F1 y área bajo la curva.

2.1. Cáncer de piel

La piel es una importante barrera frente a los innumerables estímulos del medio exterior, para lo cual se encuentra preparada mediante una gran variedad de funciones específicas tales como protección, sensorial, termorreguladora, metabólica e inmunológica Manzur et al. [2002]. La piel constituye un órgano cutáneo y consta de tres capas: la epidermis, la dermis y la hipodermis (ver Figura 2.1) Gatti and Cardama [1975].

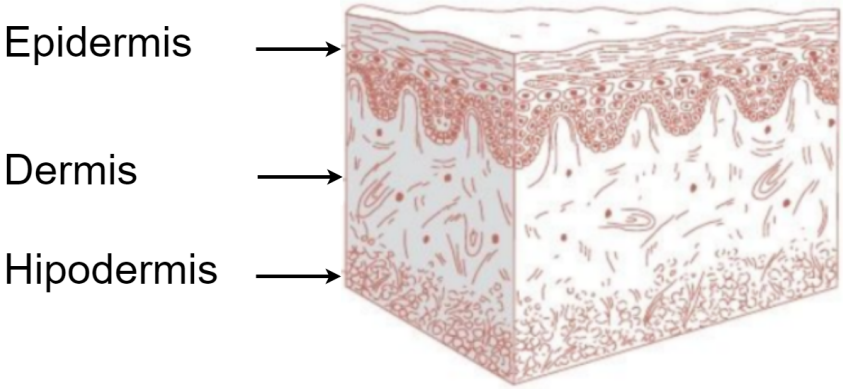


Figura 2.1: Estructura general de la piel Falabella et al. [2017].

Los tres tipos más comunes de cáncer de piel son el carcinoma de células basales (BCC, por sus siglas en inglés), el carcinoma de células escamosas (SCC, por sus siglas en inglés) y el melanoma maligno cutáneo (CMM, por sus siglas en inglés). El carcinoma de células basales y el carcinoma de células escamosas a menudo se denominan juntos como cáncer de piel "no melanoma" (NMSC, por sus siglas en inglés) Nouri [2008].

Diagnóstico de cáncer de piel

El diagnóstico de las enfermedades de la piel comienza con la determinación del lugar y tipo de las erupciones primarias (mácula o mancha, roncha, pápula, nódulo, placas, vesículas, ampollas, pústulas) y secundarias (escamas o caspa, hiperqueratosis, costras, escaras (piel necrosada), necrobrosis (avance gradual hacia la necrosis), erosiones, escoriaciones, úlceras,

atrofia, cicatriz), su forma, su color, las características de su superficie, su consistencia, su número, su distribución general y sitios de localización preferencial Steigleder [1985].

La biopsia es un comienzo crítico para el diagnóstico de cáncer de piel. Es el proceso de extracción de tejido de los pacientes para un examen de diagnóstico. Las numerosas técnicas de biopsia disponibles para un médico incluyen: escisión, incisión, afeitado, saucerización y punción. Factores como la profundidad de la invasión, la ulceración, la microsatellitosis, la invasión angiolinfática y el índice mitótico pueden afectar el pronóstico y el manejo adecuado. Hay varias herramientas disponibles para ayudar a decidir si se debe o no realizar una biopsia de una lesión pigmentada en particular. Estos incluyen la dermatoscopia, es un método que utiliza un microscopio de mano para examinar las lesiones de la piel. Es una técnica no invasiva in vivo que permite a los médicos visualizar mejor las estructuras celulares en la epidermis, la unión dermoepidérmica y la dermis; en contraste con la inspección a simple vista, donde los médicos solo pueden detectar las características morfológicas macroscópicas de las lesiones, como el tamaño, la forma, los colores, las elevaciones o las ulceraciones Rigel et al. [2011].

Existe una serie de enfoques y algoritmos de diagnóstico con el objetivo de diagnosticar correctamente el melanoma. En general, los algoritmos se pueden clasificar como basados en puntajes (la regla de dermatoscopia ABCD, CASH o lista de verificación de siete puntos) o basados en la forma (análisis de patrones).

Factores de riesgo

Una combinación de factores hereditarios y constitucionales, con la exposición a factores ambientales, determina la probabilidad de que ocurra NMSC en cualquier individuo. El color de la piel y la respuesta de la piel a la luz solar son factores constitucionales. Este hecho es evidente en los caucásicos que tienen una combinación de piel clara, ojos azules y cabello pelirrojo o rubio; muchos de ellos se queman con el sol en lugar de broncearse cuando

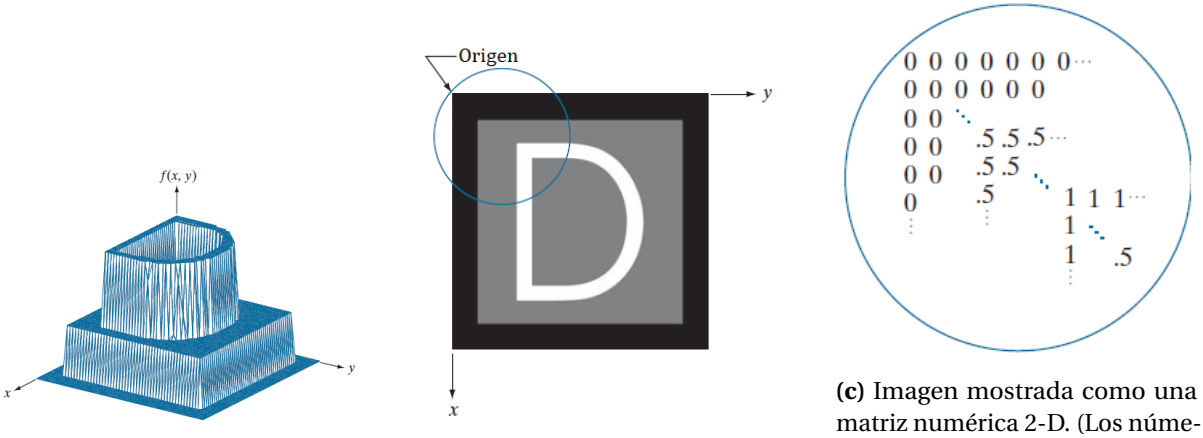
se exponen a la luz solar directa. NMSC es poco común en poblaciones negras, asiáticas e hispanas. Los principales factores son: radiación ultravioleta (rayos UV), envejecimiento (personas mayores de 75 años), condiciones médicas (pacientes receptores de un trasplante o afecciones médicas como úlceras crónicas, cicatrices de quemaduras o papiloma humano), radiación ionizante, ocupación (agricultores, soldadores, marineros, policías o pilotos), lesiones precursoras (queratosis actiníca o la enfermedad de Bowen), carcinógenos químicos (exposición al arsénico), Reichrath [2006].

2.2. Fundamentos de imágenes digitales

Una imagen en el mundo de las computadoras puede ser vista como una colección de puntos conocidos como píxeles. Esta imagen se conoce generalmente como imagen digital y se define formalmente como una matriz de píxeles cuyos valores indican la intensidad de la luz del flujo en el elemento de imagen representado por ese píxel. Cada píxel tiene su posición horizontal y vertical y un valor. Las posiciones y el valor son escalares no negativos cuyo rango depende de las características de la unidad de digitalización. Se pueden distinguir tres tipos diferentes de imágenes digitales: imagen en blanco y negro, imagen a escala de grises, e imagen en color Stipaničev [1994].

Por otro lado, desde un punto de vista matemático, una imagen puede definirse como una función bidimensional, $f(x,y)$, donde x y y son coordenadas espaciales (planas), y la amplitud de f en cualquier par de coordenadas (x,y) se denomina intensidad o nivel de gris de la imagen en ese punto. Cuando x, y, y los valores de intensidad de f son cantidades discretas finitas, se conoce a la imagen como imagen digital. Se debe tener en cuenta que una imagen digital se compone de un número finito de elementos, cada uno de los cuales tiene una ubicación y un valor determinados. En general, el valor de una imagen digital en cualquier coordenada (x,y) se denota como $f(x,y)$, donde x y y son números enteros. Cuando se hace referencia a coordenadas específicas (i,j), se utiliza la notación $f(i,j)$, donde

los argumentos son números enteros. La sección del plano real que abarcan las coordenadas de una imagen se denomina dominio espacial, y x y y se denominan variables espaciales o coordenadas espaciales Rafael C. Gonzalez [2018].



(a) Imagen mostrada como una superficie.

(b) Imagen mostrada como una matriz de intensidad visual.

(c) Imagen mostrada como una matriz numérica 2-D. (Los números 0, 0,5 y 1 representan negro, gris y blanco, respectivamente).

Figura 2.2: Formas de representar una imagen Rafael C. Gonzalez [2018].

La función $f(x, y)$ se puede representar de tres formas, como se muestra en la Figura 2.2. La Figura 2.2a es un gráfico de la función, con dos ejes que determinan la ubicación espacial y el tercer eje son los valores de f como una función de x y y. Esta representación se utiliza cuando se trabaja con conjuntos en escala de grises cuyos elementos se expresan como tripletes de la forma (x, y, z), donde x y y son coordenadas espaciales y z es el valor de f en las coordenadas (x, y).

En la Figura 2.2b se muestra como aparecería $f(x, y)$ en una pantalla de computadora o en una fotografía. La intensidad de cada punto en la pantalla es proporcional al valor de f en ese punto. En la Figura 2.2b, solo hay tres valores de intensidad igualmente espaciados. Si la intensidad se normaliza al intervalo $[0, 1]$, cada punto de la imagen tiene el valor 0, 0.5 o 1.

La Figura 2.2c se muestra una matriz compuesta por los valores numéricos de $f(x, y)$. Esta es la representación utilizada para el procesamiento informático. En forma de ecuación, esta representación de una matriz numérica $M \times N$ se define como la Ecuación 2.1

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0,N-1) \\ f(1,0) & f(1,1) & \dots & f(1,N-1) \\ \dots & \dots & & \dots \\ f(M-1,0) & f(M-1,1) & \dots & f(M-1,N-1) \end{bmatrix}. \tag{2.1}$$

En la Ecuación 2.1 el lado derecho es una imagen digital representada como una matriz de números reales. La Figura 2.3 muestra una representación gráfica de una matriz de imágenes, donde los ejes x y y se utilizan para indicar las filas y columnas de la matriz. Los píxeles específicos son valores de la matriz en un par fijo de coordenadas.

Otra forma de representar una imagen digital se muestra en la Ecuación 2.2, en donde, $a_{ij} = f(i,j)$, por lo tanto las Ecuaciones 2.1 y 2.2 denotan matrices idénticas:

$$A = \begin{bmatrix} a_{0,0} & a_{0,1} & \dots & a_{0,N-1} \\ a_{1,0} & a_{1,1} & \dots & a_{1,N-1} \\ \dots & \dots & & \dots \\ a_{M-1,0} & a_{M-1,1} & \dots & a_{M-1,N-1} \end{bmatrix}. \tag{2.2}$$

Las imágenes en color se forman en el espacio de color RGB (de las siglas en inglés, Red, Green y Blue), descrito más adelante en la sección 2.2.1, mediante el uso de imágenes en sus componentes rojo, verde y azul, como se muestra en la Figura 2.4. Cada píxel de una imagen RGB tiene tres componentes, que se pueden organizar en forma de vector columna representado por la Ecuación 2.3

$$z = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix}, \tag{2.3}$$

donde z_1 es la intensidad del píxel en la imagen roja, z_2 en la imagen verde y z_3 en la imagen azul. Por lo tanto, una imagen en color RGB de tamaño $M \times N$ se puede representar mediante tres imágenes componentes de este tamaño, o mediante un total de MN vectores

de tamaño 3×1. Un caso multiespectral general que involucre imágenes de componentes n dará como resultado vectores n-dimensionales cuya reperentación se da con la Ecuación 2.4:

$$z = \begin{bmatrix} z_1 \\ z_2 \\ \dots \\ z_n \end{bmatrix}. \tag{2.4}$$

El origen de una imagen esta en la esquina superior izquierda, como muestra la Figura 2.3. Elegir el origen de $f(x, y)$ en ese punto tiene sentido matemáticamente porque las imágenes digitales en realidad son matrices.

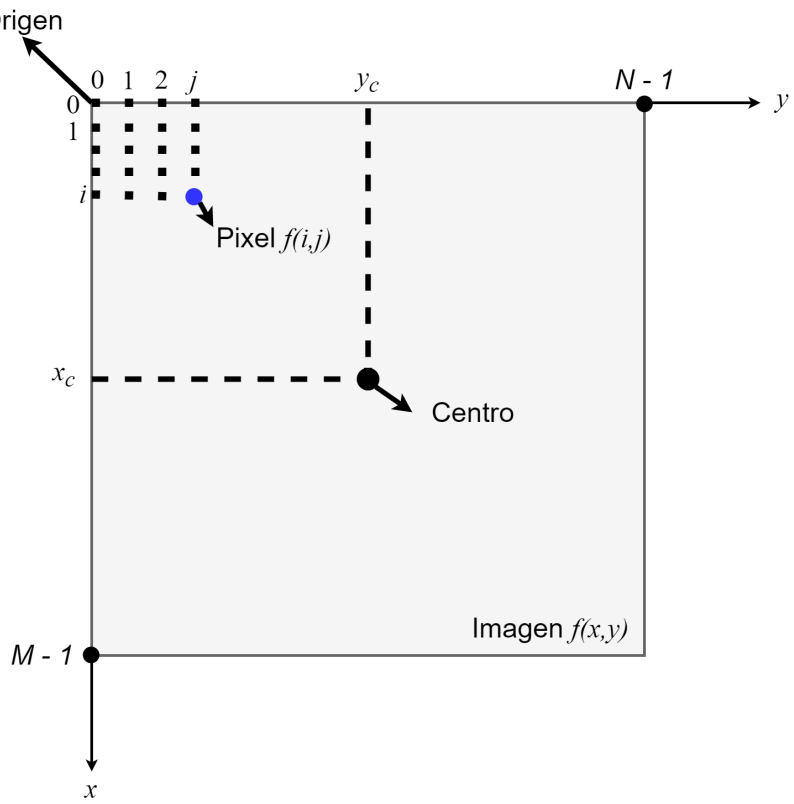


Figura 2.3: Convención de coordenadas utilizada para representar una imagen digital Rafael C. Gonzalez [2018].

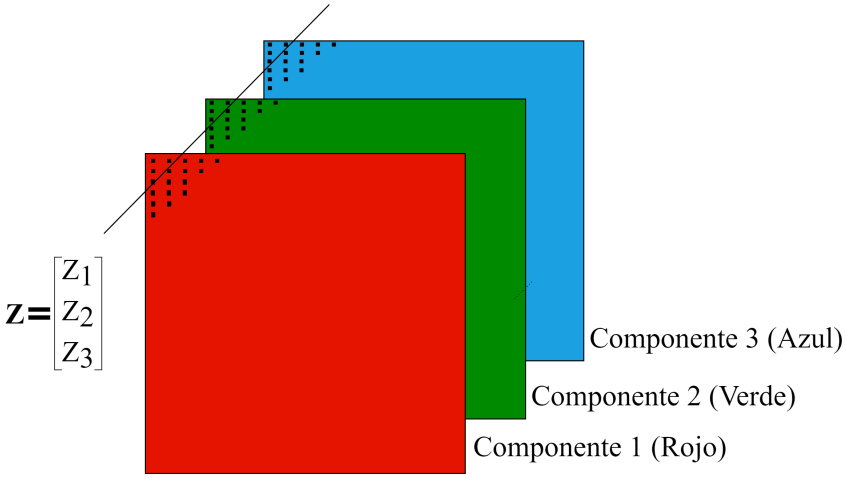


Figura 2.4: Representación de un vector con los valores de píxeles correspondientes en tres imágenes de componentes RGB Rafael C. Gonzalez [2018].

Vecinos de un píxel

Un píxel p en las coordenadas (x, y) tiene dos vecinos horizontales y dos verticales con coordenadas $(x+1, y)$, $(x-1, y)$, $(x, y+1)$ y $(x, y-1)$.

Este conjunto de píxeles, se conoce como los 4 vecinos de p, y son denotados como $N_4(p)$. Los cuatro vecinos diagonales de p tienen coordenadas $(x+1, y+1)$, $(x-1, y-1)$, $(x-1, y+1)$ y $(x-1, y-1)$ y se denotan como $N_D(p)$. Estos vecinos, junto con los 4 vecinos, se denominan los 8 vecinos de p, denotados por $N_8(p)$. El conjunto de ubicaciones de imágenes de los vecinos de un punto p se denomina vecindad de p.

2.2.1. Modelo RGB

El propósito de un modelo de color (también llamado espacio de color o sistema de color) es facilitar la especificación de colores de alguna manera estándar. Un ejemplo es el modelo RGB,en donde, cada color aparece en sus componentes espectrales primarios de rojo, verde y azul. Este modelo se basa en un sistema de coordenadas cartesianas. El subespacio de color de interés es el cubo que se muestra en la Figura 2.5, en el que los valores primarios RGB

están en tres esquinas; los colores secundarios cian, magenta y amarillo están en otras tres esquinas; el negro está en el origen; y el blanco está en la esquina más alejada del origen. En este modelo, la escala de grises (puntos de valores RGB iguales) se extiende de negro a blanco a lo largo de la línea que une estos dos puntos. Los diferentes colores en este modelo son puntos sobre o dentro del cubo y están definidos por vectores que se extienden desde el origen.

Las imágenes representadas en el modelo de color RGB constan de tres imágenes componentes, una para cada color primario. Los planos de color rojo, verde o azul tienen la misma estructura que el plano de imagen de valor gris, normalmente 256 intensidades para cada color. Combinando varias intensidades de píxeles RGB, cada píxel de imagen podría dar uno de $256 \times 256 \times 256 = 16,777,216$ valores de color diferentes. Al definir el valor del píxel de la imagen a color en la ubicación (x, y), se da un triplete en la forma (rojo, verde, azul), por ejemplo $(0,255,255)$, lo que significa la intensidad mínima para rojo e intensidad máxima para verde y azul Stipaničev [1994].

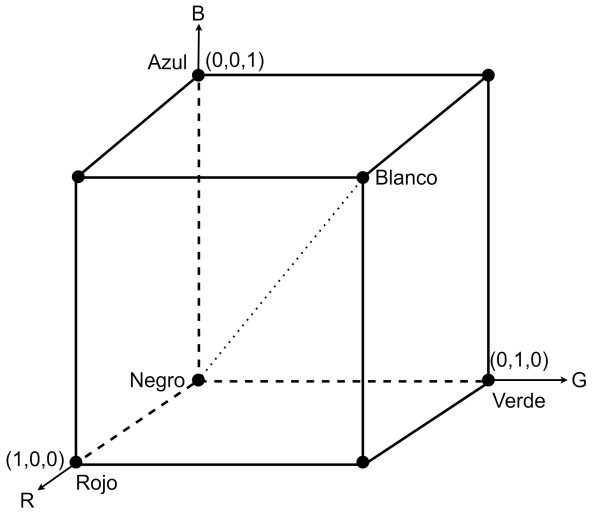


Figura 2.5: Esquema del cubo de color RGB Rafael C. Gonzalez [2018].

2.2.2. Filtrado espacial

Los procesos del dominio espacial están basados en la Ecuación 2.5

$$g(x,y) = T[f(x,y)], \tag{2.5}$$

donde f(x,y) es una imagen de entrada, $g(x,y)$ es la imagen de salida y T es un operador en f definido sobre una vecindad del punto (x,y). El operador se puede aplicar a los píxeles de una sola imagen o a los píxeles de un conjunto de imágenes. La Figura 2.6 muestra la implementación básica de la Ecuación 2.5 en una sola imagen. El punto (x_0, y_0) que se muestra es una ubicación arbitraria en la imagen, y la pequeña región que se muestra es una vecindad de (x_0, y_0). Normalmente, la vecindad es rectangular, centrada en (x_0, y_0) y mucho más pequeña que la imagen.

El proceso que muestra la Figura 2.6 consiste en mover el centro de la vecindad de píxel a píxel y aplicar el operador T a los píxeles de la vecindad para producir un valor de salida en esa ubicación. Por lo tanto, para cualquier ubicación específica (x_0, y_0), el valor de la imagen de salida g en esas coordenadas es igual al resultado de aplicar T a la vecindad con origen en (x_0, y_0) en f. Por lo general, el proceso comienza en la parte superior izquierda de la imagen de entrada y avanza píxel por píxel en un escaneo horizontal-vertical, una fila-columna a la vez.

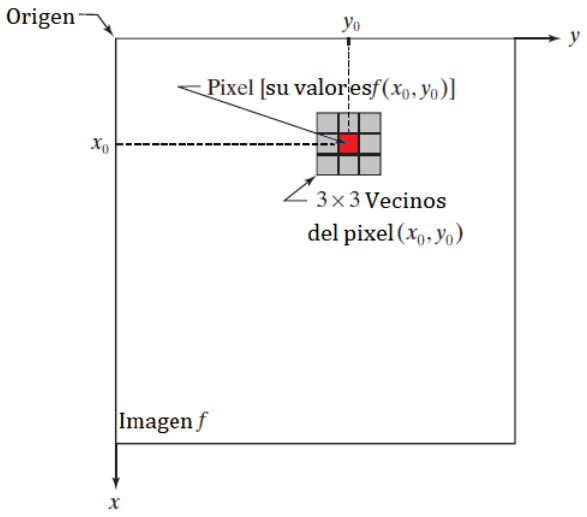


Figura 2.6: Una vecindad de 3 × 3 alrededor de un punto (x_0, y_0) en una imagen Rafael C. Gonzalez [2018].

La vecindad más pequeña posible es de tamaño 1×1. En este caso, g depende solo del valor de f en un solo punto (x, y) y T en la Ecuación 2.5 se convierte en una función de transformación de intensidad representada por la Ecuación 2.6

$$s = T(r), \tag{2.6}$$

donde s y r indican la intensidad de g y f respectivamente en cualquier punto (x, y).

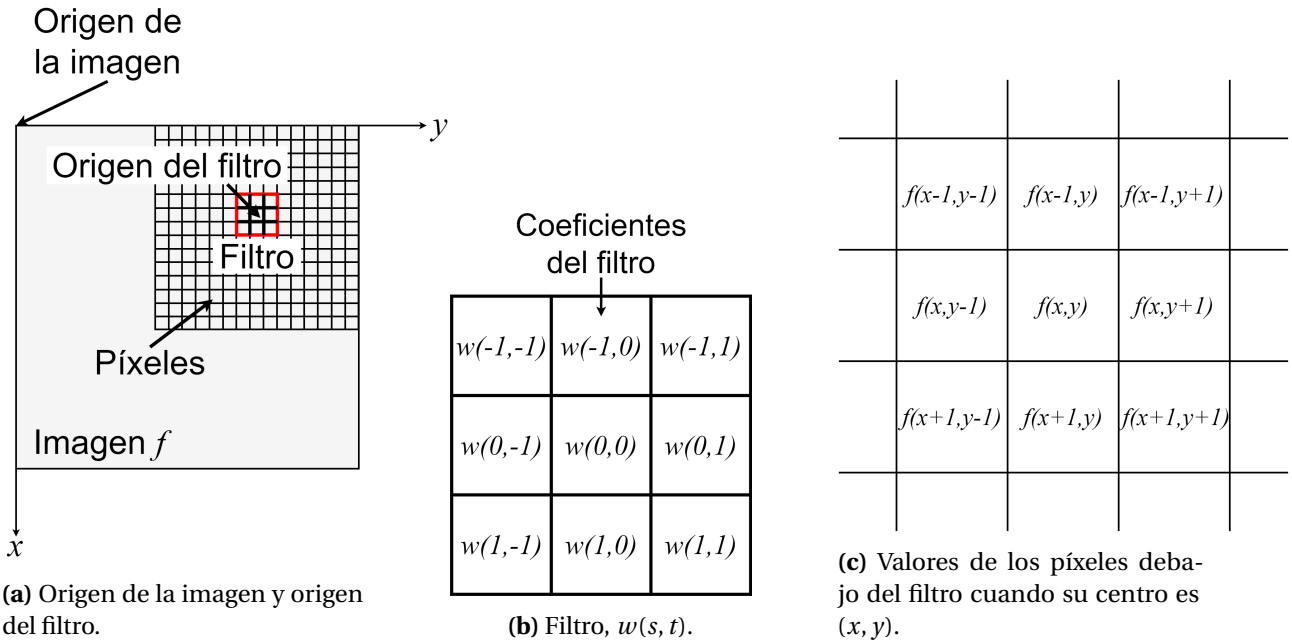
El filtrado espacial modifica una imagen reemplazando el valor de cada píxel por una función de los valores del píxel y sus vecinos. Si la operación realizada en los píxeles de la imagen es lineal, entonces el filtro se denomina filtro espacial lineal. De lo contrario, el filtro es un filtro espacial no lineal.

Un filtro espacial lineal realiza una operación de suma de productos entre una imagen f y un filtro, w. El filtro es una matriz cuyo tamaño define la vecindad de operación y cuyos coeficientes determinan la naturaleza del filtro.

La Figura 2.7a muestra el proceso del filtrado espacial lineal utilizando un filtro de 3×3. En cualquier punto (x, y) de la imagen, la respuesta, $g(x, y)$, del filtro es la suma de los productos de los coeficientes del filtro y los píxeles de la imagen abarcados por el filtro, representado por la Ecuación 2.7:

$$\begin{aligned} g(x,y) = {} & w(-1,-1)f(x-1,y-1) + w(-1,0)f(x-1,y) \\ & + \ldots + w(0,0)f(x,y) + \ldots + w(1,1)f(x+1,y+1) \end{aligned}. \tag{2.7}$$

A medida que varían las coordenadas x y y, el centro del filtro se mueve de píxel a píxel, generando la imagen filtrada, g, en el proceso. El coeficiente del centro del filtro, $w(0,0)$, se alinea con el píxel en la ubicación (x, y). Para un filtro de tamaño $m \times n$, suponiendo que $m = 2a + 1$ y $n = 2b + 1$, donde a y b son números enteros no negativos. Esto significa que el enfoque está en filtros de tamaño impar en ambas direcciones de coordenadas. En general, el filtrado espacial lineal de una imagen de tamaño $M \times N$ con un filtro de tamaño $m \times n$ viene



(a) Origen de la imagen y origen del filtro.

(b) Filtro, $w(s,t)$.

(c) Valores de los píxeles debajo del filtro cuando su centro es (x,y).

Figura 2.7: Mecánica del filtrado espacial lineal utilizando un filtro de 3 × 3 Rafael C. Gonzalez [2018].

dado por la Ecuación 2.8

$$g(x,y) = \sum_{s=-a}^{a} \sum_{s=-b}^{b} w(s,t) f(x+s, y+t), \tag{2.8}$$

donde x y y varían de modo que el centro (origen) del filtro visita cada píxel en f una vez.

Detección de bordes

Dentro del procesamiento de imágenes, la forma más común de detección de bordes es mediante la convolución de la imagen con un filtro como el que se muestra en la Figura 2.8. Al desplazar el filtro sobre toda la imagen se calcula la suma de los productos de los coeficientes del filtro por los niveles de gris de los píxeles de la imagen justo debajo de dichos coeficientes. La respuesta del filtro en un punto cualquiera de la imagen es dado por la Ecuación 2.9 Morales and Azuela [2012]:

$$R = z_1 \cdot f_1 + z_2 \cdot f_2 + \cdots + z_9 \cdot f_9 = \sum_{i=1}^{9} z_i f_i, \tag{2.9}$$

donde, f_i es el nivel de gris del píxel de la imagen justo debajo del coeficiente correspondiente z_i de la máscara utilizada. El valor R es asignado al píxel central de la región de la imagen de salida.

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

Figura 2.8: Máscara tipo de 3 × 3 para la detección de bordes.

Detección de bordes mediante el cálculo del gradiente

Los métodos para detectar puntos, bordes y líneas están basados en el cálculo del gradiente y el cálculo de la segunda derivada. El proceso de detección de bordes consiste en determinar cuáles píxeles deben ser considerados como elementos pertenecientes a bordes y cuáles no. Un borde puede ser entendido como un cambio significativo en el valor de la intensidad de los píxeles en una región de la imagen. Cuando se trabaja con imágenes, el operador gradiente toma la forma de la Ecuación 2.10:

$$\nabla f(x, y) = \frac{\partial f}{\partial x} + \frac{\partial f}{\partial y}. \tag{2.10}$$

En forma discreta este método es derivado de la diferencia horizontal y vertical entre los valores de los píxeles. Esta diferencia se realiza (ver Figura 2.9) entre las pendientes a lo largo de una línea en la imagen siguiendo las Ecuaciones 2.11 y 2.12:

$$\nabla_x f(i, j) = f(i, j) - f(i, j+1), \tag{2.11}$$

$$\nabla_y f(i,j) = f(i,j) - f(i+1,j). \tag{2.12}$$

f(i-1,j-1)	*f(i-1,j)*	*f(i-1,j+1)*
f(i,j-1)	*f(i,j)*	*f(i,j+1)*
f(i+1,j-1)	*f(i+1,j)*	*f(i+1,j+1)*

Figura 2.9: Entorno de un píxel dentro de una ventana de 3 × 3.

Detección de bordes a través de los filtros de Sobel

El método de Sobel es una de las técnicas clásicas, basadas en el cálculo del gradiente, más utilizadas para la detección de bordes. Los filtros empleados por este método (ver Figura 2.10) son derivados de las Ecuaciones 2.13 y 2.14:

$$\begin{aligned} g_x = \frac{\partial f}{\partial x} =& [f(x+1,y-1) + 2f(x+1,j) + f(x+1,y+1)] - \\ & [f(x-1,y-1) + 2f(x-1,y) + f(x-1,y-1)], \end{aligned} \tag{2.13}$$

$$\begin{aligned} g_y = \frac{\partial f}{\partial y} =& [f(x-1,y+1) + 2f(x,y+1) + f(x+1,y+1)] - \\ & [f(x-1,y-1) + 2f(x,y-1) + f(x+1,y-1)]. \end{aligned} \tag{2.14}$$

-1	*-2*	*-1*
0	*0*	*0*
1	*2*	*1*

(a) Filtro horizontal.

-1	*0*	*1*
-2	*0*	*2*
-1	*0*	*1*

(b) Filtro vertical.

Figura 2.10: Representación de los filtros de Sobel.

2.3. Aprendizaje Automático

En un nivel alto, las tareas de aprendizaje automático se pueden clasificar en tres grupos según el resultado deseado y el tipo de entrada necesaria para producirlo Swamynathan [2017] (ver figura 2.11).

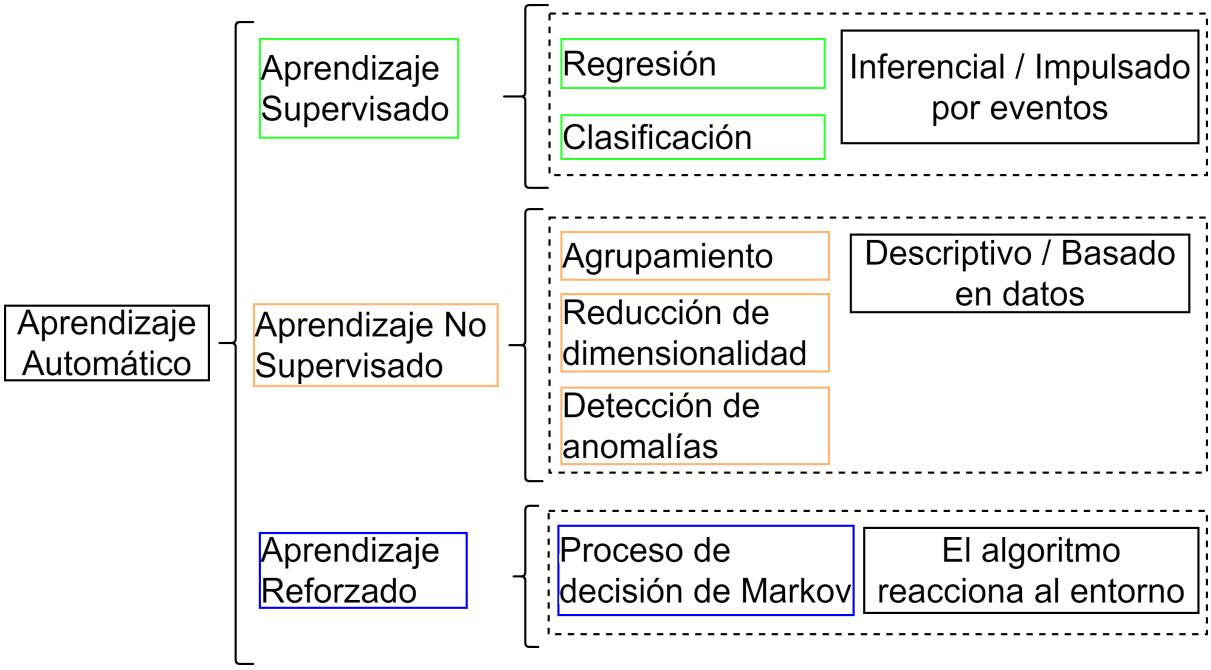


Figura 2.11: Tipos de aprendizaje automático Swamynathan [2017].

2.3.1. Aprendizaje supervisado

Los algoritmos de aprendizaje supervisado son una clase de algoritmos de aprendizaje automático que utilizan datos previamente etiquetados para aprender sus características, por lo que pueden clasificar datos similares pero sin etiquetar.

El objetivo principal del aprendizaje supervisado es aprender un modelo a partir de datos de entrenamiento etiquetados que permita hacer predicciones sobre datos no vistos o futuros. Una representación del flujo para algoritmos de aprendizaje supervisado se puede ver en la Figura 2.12. El término supervisado se refiere a un conjunto de muestras donde las señales de salida deseadas (etiquetas) ya se conocen Raschka et al. [2016].

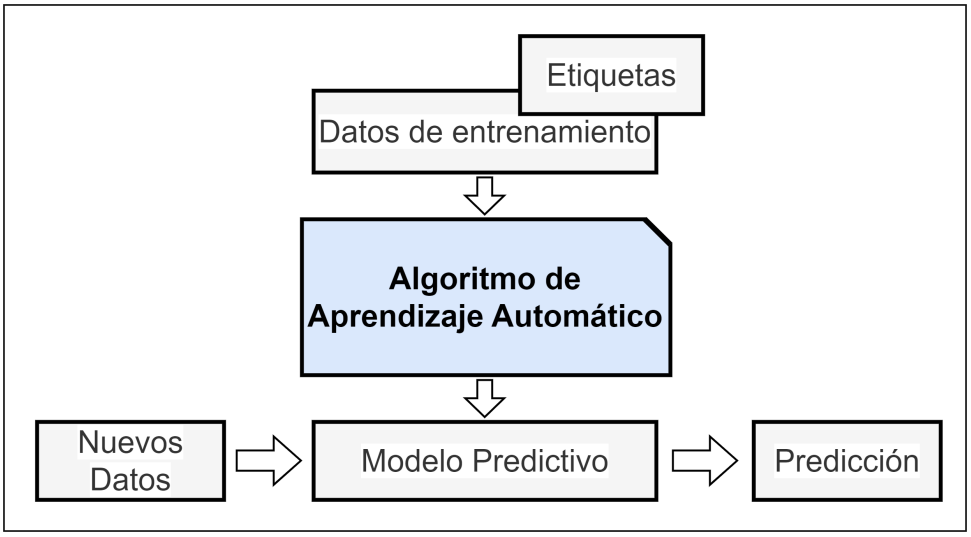


Figura 2.12: Flujo seguido por algoritmos de aprendizaje supervisado con el uso de datos de entrenamiento con etiquetas Raschka et al. [2016].

Los métodos de aprendizaje automático supervisado trabajan con datos etiquetados para entrenar modelos y luego predecir resultados para nuevas muestras de datos. Algunos procesos, como la ingeniería de características, el escalado y la selección, siempre deben permanecer constantes para que las mismas características se utilicen para entrenar el modelo y las mismas características se extraigan de nuevas muestras de datos para alimentar el modelo en la fase de predicción Sarkar et al. [2018].

2.3.2. Clasificación

La clasificación es también conocida como discriminación. En donde, cada entrada conocida como instancia pertenece a una clase, que se indica mediante el valor de un atributo llamado la clase de la instancia. Este atributo toma diversos valores discretos, correspondiendo cada uno a una clase. La clasificación busca predecir la clase desconocida de nuevas instancias o, más concretamente, clasificar de modo más preciso las nuevas instancias García et al. [2018].

Los ejemplos se presentan como un conjunto de tuplas de elementos pertenecientes a

los dos conjuntos, E y $S^{\delta} = \{\langle e, s\rangle : e \in E, s \in S\}$, siendo S el conjunto de valores de salida. Los ejemplos, representados por ϱ, suelen ir junto al valor de S, se les llama ejemplos etiquetados $\langle e, s\rangle$. En consecuencia, $\in$ será el conjunto de datos etiquetados. Esta tarea busca el aprendizaje de una función llamada clasificador, $C : E- > S^{\lambda} : E \rightarrow S$ que exponga la correspondencia presente entre los ejemplos. En otras palabras, que para cada valor de E sólo se tenga un valor de S. Esta S no toma valores numéricos ya que es un conjunto nominal. Los valores que adquiera serán los denominados clases. Si el proceso y el algoritmo se han desarrollado correctamente, la función podrá establecer la clase de nuevos ejemplos aún no etiquetados, es decir, dará un valor de S a cada valor de ϱ.

De acuerdo a Kroese et al. [2022] los métodos de clasificación son métodos de aprendizaje supervisado en los que una variable de respuesta categórica Y toma uno de c valores posibles que se va a predecir a partir de un vector X de variables explicativas utilizando una función de predicción g. En este sentido, g clasifica la entrada X en una de las clases, por ejemplo en el conjunto $\{0, ..., c-1\}$. Por esta razón, se llama a g una función de clasificación o simplemente clasificador. Al igual que con cualquier técnica de aprendizaje supervisado el objetivo es minimizar la pérdida o el riesgo esperados.

El objetivo de la clasificación es aprender un mapeo de las entradas x a las salidas y, donde $y \in \{1, ..., C\}$, siendo C el número de clases. Si $C = 2$, esto se llama clasificación binaria, en cuyo caso a menudo se asume que $y \in \{0, 1\}$; si $C > 2$, esto se denomina clasificación multiclase Murphy [2012].

Los métodos de clasificación se construyen tomando un conjunto de ejemplos etiquetados y usándolos para generar una regla que asigna una etiqueta a cualquier ejemplo nuevo. En el problema general, se tiene un conjunto de datos de entrenamiento (x_i, y_i); cada uno de los vectores de características x_i consta de medidas de las propiedades de diferentes tipos de objetos, y los y_i son etiquetas que dan el tipo de objeto que generó el ejemplo David A. Forsyth [2013].

En Han and Kamber [2001] se menciona que los métodos de clasificación y predicción se pueden comparar y evaluar de acuerdo con los siguientes criterios:

- Exactitud predictiva: se refiere a la capacidad del modelo para predecir correctamente la etiqueta de clases de datos nuevos o nunca antes vistos.
- Velocidad: se refiere al costo de cómputo involucrado.
- Robustez: se refiere a la capacidad del modelo para hacer predicciones correctas dados datos ruidosos con valores faltantes.
- Escalabilidad: se refiere a la capacidad de construir el modelo de manera eficiente con grandes cantidades de datos.
- Interpretabilidad: se refiere al nivel de comprensión que proporciona el modelo.

Los modelos de clasificación se pueden desglosar según el tipo y el número de variables de salida producidas por ellos Sarkar et al. [2018].

- Clasificación binaria: cuando se tiene un total de dos categorías para diferenciar en la variable de respuesta de salida en los datos, entonces el problema se denomina problema de clasificación binaria.
- Clasificación multiclase: se trata de una extensión del problema de clasificación binaria. En este caso se tienen más de dos categorías o clases en las que se pueden clasificar los datos.
- Clasificación de etiquetas múltiples: estos problemas de clasificación generalmente involucran datos en los que la variable de salida no siempre es un valor único sino un vector que tiene múltiples valores o etiquetas.

Los modelos de clasificación a menudo generan etiquetas de clase reales o probabilidades para cada etiqueta de clase posible que proporciona un nivel de confianza para cada clase en la predicción. Los siguientes son los principales formatos de salida de los modelos de clasificación.

- Salida de clasificación de categoría: en algunos modelos de clasificación, la salida para cualquier punto de datos desconocido es la categoría prevista o la etiqueta de clase. Estos modelos generalmente calculan las probabilidades de todas las categorías, pero informan sólo una etiqueta de clase que tiene la máxima probabilidad o confianza.
- Salida de clasificación de probabilidad de categoría: en estos modelos de clasificación, la salida es el valor de probabilidad de cada etiqueta de clase posible. Estos modelos son importantes cuando se quiere utilizar más la salida producida por el modelo de clasificación para un análisis detallado o para tomar decisiones complejas.

Árboles de Decisión

De acuerdo a Han and Kamber [2001], un árbol de decisión es una estructura de árbol similar a un diagrama de flujo, donde cada nodo interno denota una prueba en un atributo, cada rama representa un resultado de la prueba y los nodos hoja representan clases o distribuciones de clase. El nodo superior de un árbol es el nodo raíz.

Un marco general para hacer crecer un árbol de decisión según Shalev-Shwartz and Ben-David [2014] es el siguiente:

1. Se comienza con un árbol con una sola hoja (la raíz) y se asigna a esta hoja una etiqueta de acuerdo con el voto mayoritario entre todas las etiquetas del conjunto de entrenamiento.
2. Se realiza una serie de iteraciones. En cada iteración, se examina el efecto de dividir una sola hoja.

3. Se define alguna medida de "ganancia" que cuantifique la mejora debida a esta división.

4. Luego, entre todas las posibles divisiones, se elige la que maximiza la ganancia y se realiza, o se elige no dividir la hoja en absoluto.

En Mohri et al. [2012] se menciona que un árbol de decisión binario es una representación de árbol de una partición del espacio de características. La figura 2.13 muestra un ejemplo sencillo en el caso de un espacio bidimensional basado en dos características X_1 y X_2, así como la partición que representa. Cada nodo interior de un árbol de decisión corresponde a una pregunta relacionada con características. Puede ser una cuestión numérica de la forma $X_i \leq a$ para una variable de característica $X_i, i \in [1, N]$, y algún umbral $a \in R$, o una pregunta categórica como $X_i \in \{azul, blanco, rojo\}$, cuando la característica X_i toma un valor categórico como un color. Cada hoja está etiquetada de la forma $l \in Y$.

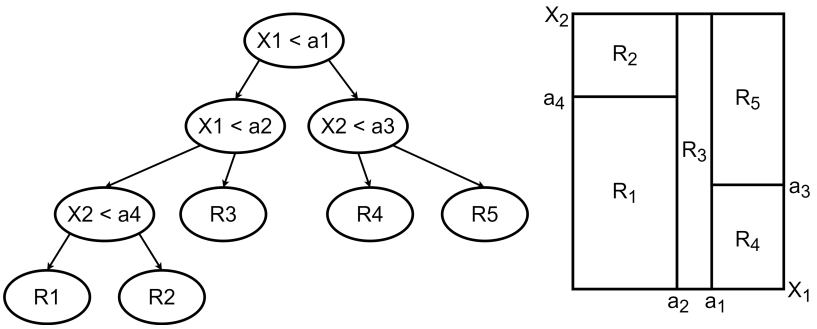


Figura 2.13: Ejemplo de árbol de decisión binario Mohri et al. [2012].

Según las características del conjunto de entrenamiento, el modelo de árbol de decisión aprende una serie de preguntas para inferir las etiquetas de clase de las muestras Raschka and Mirjalili [2017].

Maximizar la ganancia de información

Para dividir los nodos en las características más informativas, se necesita definir una función objetivo que se quiere optimizar a través del algoritmo de aprendizaje del árbol. La

función objetivo es maximizar la ganancia de información en cada división, la cual se define con la Ecuación 2.15:

$$IG(D_p, f) = I(D_p) - \sum_{j=1}^{m} \frac{N_j}{N_p} I(D_j). \tag{2.15}$$

En la ecuación 2.15, f es la función para realizar la división, D_p y D_j son el conjunto de datos del padre y el j-ésimo nodo secundario, I es la medida de impureza, N_p es el número total de muestras en el nodo principal, y N_j es el número de muestras en el nodo secundario jth. Se puede ver que la ganancia de información es simplemente la diferencia entre la impureza del nodo principal y la suma de las impurezas del nodo secundario: cuanto menor es la impureza de los nodos secundarios, mayor es la ganancia de información. Sin embargo, por simplicidad y para reducir el espacio de búsqueda combinatoria, el nodo principal se divide en dos nodos secundarios, D_{izq} y D_{der}, representado con la Ecuación 2.16:

$$IG(D_p, f) = I(D_p) - \frac{N_{izq}}{N_p} I(D_{izq}) - \frac{N_{der}}{N_p} I(D_{der}). \tag{2.16}$$

Las tres medidas de impureza o criterios de división que se usan comúnmente en los árboles de decisión binarios son la impureza de Gini (I_G), la entropía (I_H) y el error de clasificación (I_E). Comenzando con la definición de entropía para todas las clases no vacías ($p(i|t) \neq 0$) se tiene la Ecuación 2.17:

$$I_H(t) = -\sum_{i=1}^{c} p(i|t) log_2\, p(i|t). \tag{2.17}$$

Aquí, $p(i|t)$ es la proporción de las muestras que pertenecen a la clase c para un nodo particular t. La entropía es por lo tanto 0 si todas las muestras en un nodo pertenecen a la misma clase, y la entropía es máxima si se tiene una distribución de clases uniforme.

La impureza de Gini puede entenderse como un criterio para minimizar la probabilidad de clasificación errónea, y es representada por la Ecuación 2.18:

$$I_G(t) = \sum_{i=1}^{c} p(i|t)(1 - p(i|t)) = 1 - \sum_{i=1}^{c} p(i|t)^2. \tag{2.18}$$

Otra medida de impureza es el error de clasificación, que se representa con la Ecuación 2.19:

$$I_E = 1 - max\{p(i|t)\}. \tag{2.19}$$

Bosques Aleatorios

Un bosque aleatorio es un clasificador que consiste en una colección de árboles de decisión, donde cada árbol se construye aplicando un algoritmo A en el conjunto de entrenamiento S y un vector aleatorio adicional, θ, donde θ es un ejemplo de alguna distribución. La predicción del bosque aleatorio se obtiene por mayoría de votos sobre las predicciones de los árboles individuales.

Para especificar un bosque aleatorio particular, se necesita definir el algoritmo A y la distribución sobre θ. Una opción es tomar una submuestra aleatoria de S con reemplazos; es decir, se muestrea un nuevo conjunto de entrenamiento S' de tamaño M' usando la distribución uniforme sobre S. Segundo, se construye una secuencia $I_1, I_2, ...$, donde cada I_t es un subconjunto de $[d]$ de tamaño k, que se genera muestreando uniformemente elementos aleatorios de $[d]$. Todas estas variables aleatorias forman el vector θ. Luego, el algoritmo A desarrolla un árbol de decisión basado en la muestra S', donde en cada etapa de división, el algoritmo está restringido a elegir una característica que maximiza la ganancia del conjunto I_t. Intuitivamente, si k es pequeño, esta restricción puede evitar el ajuste excesivo.

Regresión Logística

El análisis de regresión intenta encontrar el valor de los parámetros para la función que mejor se ajusta a un conjunto de datos de entrada.

La regresión logística no es un algoritmo de clasificación, pero puede ser convertido en uno. Esto, al introducir una regla que determine la clase en función de la salida de una función logística.

De acuerdo a Hastie et al. [2001], el modelo de regresión logística surge del deseo de modelar las probabilidades posteriores de las clases K a través de funciones lineales en x, al mismo tiempo que se asegura que suman uno y permanecen en $[0,1]$. El modelo se representa con la Ecuación 2.20:

$$\begin{aligned} log &= \frac{Pr(G=1|X=x)}{Pr(G=K|X=x)} = \beta_{10} + \beta_1^T x \\ log &= \frac{Pr(G=2|X=x)}{Pr(G=K|X=x)} = \beta_{20} + \beta_2^T x \\ &\vdots \\ log &= \frac{Pr(G=K-1|X=x)}{Pr(G=K|X=x)} = \beta_{(K-1)0} + \beta_{K-1}^T x. \end{aligned} \tag{2.20}$$

El modelo se especifica en términos de $K-1$ transformación logit (que refleja la restricción de que las probabilidades suman uno). Aunque el modelo utiliza la última clase como denominador en las razones impares, la elección del denominador es arbitraria porque las estimaciones son equivariantes bajo esta elección. Un simple cálculo lo da la Ecuación 2.21:

$$
\begin{aligned}
Pr(G=k|X=x) &= \frac{exp(\beta_{k0}+\beta_k^T x)}{1+\sum_{l=1}^{K-1} exp(\beta_{l0}+\beta_l^T x)}, k=1,...,K-1, \\
Pr(G=K|X=x) &= \frac{1}{1+\sum_{l=1}^{K-1} exp(\beta_{l0}+\beta_l^T x)},
\end{aligned}
\tag{2.21}
$$

y claramente suman uno. Para enfatizar la dependencia de todo el conjunto de parámetros $\theta = \{\beta_{10}, \beta_1^T, ..., \beta_{(K-1)0}, \beta_{K-1}^T\}$, se denotan las probabilidades $Pr(G=k|X=x) = p_k(x;\theta)$.

Cuando $K=2$, este modelo es especialmente simple, ya que solo hay una única función lineal. Es ampliamente utilizado en aplicaciones bioestadísticas donde la respuesta binaria (dos clases) ocurre con bastante frecuencia.

Ajuste de modelos de regresión logística

Los modelos de regresión logística generalmente se ajustan por máxima verosimilitud, usando la verosimilitud condicional de G dada X. Dado que $Pr(G|X)$ especifica completamente la distribución condicional, la distribución multinomial es apropiada. La log-verosimilitud para N observaciones se representa con la Ecuación 2.22:

$$
l(\theta) = \sum_{i=1}^{N} log\, p_{g_i}(x_{i;\theta}), \tag{2.22}
$$

donde $p_k(x_i;\theta) = Pr(G=k|X=x_i;\theta)$.

Por conveniencia al codificar las dos clases g_i a través de una respuesta $0/1$ y_i, donde $y_i = 1$ cuando $g_i = 1$, y $y_i = 0$ cuando $g_i = 2$. Sea $p_1(x;\theta) = p(x;\theta)$, y $p_2(x;\theta) = 1 - p(x;\theta)$. El log-verosimilitud se puede representar como la Ecuación 2.23:

$$\begin{aligned} l(\beta) &= \sum_{i=1}^{N} \{y_i \, log \, p(x_i;\beta) + (1-y_i) \, log(1-p(x_i;\beta))\} \\ &= \sum_{i=1}^{N} \{y_i \beta^T x_i - log(1+e^{\beta^T x_i})\}. \end{aligned} \tag{2.23}$$

De la ecuación 2.23 $\beta = \{\beta_{10}, \beta_1\}$, asumiendo que el vector de entradas x_i incluye el término constante 1 para acomodar la intersección. Para maximizar la verosimilitud logarítmica, se establecen sus derivadas en cero. Estas ecuaciones de puntuación se representan con la Ecuación 2.24:

$$\frac{\partial \ell(\beta)}{\partial \beta} = \sum_{i=1}^{N} x_i (y_i - p(x_i;\beta)) = 0, \tag{2.24}$$

que son ecuaciones $p+1$ no lineales en β. Dado que el primer componente de x_i es 1, la primera ecuación de puntuación específica que $\sum_{i=1}^{N} y_i = \sum_{i=1}^{N} p(x_i;\beta)$; el número esperado de clase uno coincide con el número observado (y por lo tanto también la clase dos).

Para resolver la Ecuación 2.24, se utiliza el algoritmo de Newton-Raphson, que requiere la segunda derivada o matriz hessiana, como se muestra en la Ecuación 2.25

$$\frac{\partial^2 \ell(\beta)}{\partial \beta \partial \beta^T} = -\sum_{i=1}^{N} x_i x_i^T \, p(x_i;\beta)(1-p(x_i;\beta)). \tag{2.25}$$

Comenzando con β^{old}, se requiere una única actualización de Newton-Raphson, quedando como la Ecuación 2.26

$$\beta^{new} = \beta^{old} - \left(\frac{\partial^2 \ell(\beta)}{\partial \beta \partial \beta^T} \right)^{-1} \frac{\partial \ell(\beta)}{\partial \beta}, \tag{2.26}$$

donde los derivados se evalúan en β^{old}. Sea y el vector de valores y_i, X la matriz $N \times (p+1)$ de valores x_i, p el vector de probabilidades ajustadas con i-ésimo elemento $p(x_i;\beta^{old})$ y W una matriz diagonal de pesos $N \times N$ con i-ésimo elemento diagonal $p(x_i;\beta^{old})(1-p(x_i;\beta^{old}))$. Entonces $\frac{\partial \ell(\beta)}{\partial \beta} = X^T(y-p)$ y $\frac{\partial^2 \ell(\beta)}{\partial \beta \partial \beta^T} = -X^T W X$.

El paso de Newton-Raphson se representa en la Ecuación 2.27:

$$\begin{aligned} \beta^{new} &= \beta^{old} + (X^T W X)^{-1} X^T (y-p) \\ &= (X^T W X)^{-1} X^T W (X\beta^{old} + W^{-1}(y-p)) \\ &= (X^T W X)^{-1} X^T W z. \end{aligned} \tag{2.27}$$

En la segunda y tercera línea se ha vuelto a expresar el paso de Newton-Raphson como un paso de mínimos cuadrados ponderados, con la respuesta similar a la Ecuación 2.28

$$z = X\beta^{old} + W^{-1}(y-p). \tag{2.28}$$

Estas ecuaciones se resuelven repetidamente, ya que en cada iteración cambia p, W y z. Este algoritmo se conoce como mínimos cuadrados reponderados iterativamente o IRLS, ya que cada iteración resuelve el problema de los mínimos cuadrados ponderados.

$$\beta^{new} \leftarrow arg\,min_{\beta}(z - X\beta)^T W (z - X\beta). \tag{2.29}$$

Para el caso multiclase ($K \geq 3$), el algoritmo de Newton también se puede expresar como un algoritmo de mínimos cuadrados reponderados iterativamente, pero con un vector de respuestas $K-1$ y una matriz de peso no diagonal por observación. Este último excluye cualquier algoritmo simplificado, y en este caso es numéricamente más conveniente trabajar

directamente con el vector expandido θ.

K Vecinos más Cercanos

El algoritmo de K Vecinos más Cercanos (KNN, por sus siglas en ingles, K Nearest Neighbors) tiene como idea memorizar el conjunto de entrenamiento y luego predecir la etiqueta de cualquier instancia nueva sobre la base de las etiquetas de sus vecinos más cercanos en el conjunto de entrenamiento. La lógica detrás de este método se basa en la suposición de que las características que se utilizan para describir los puntos de dominio son relevantes para sus etiquetas de una manera que hace probable que los puntos cercanos tengan la misma etiqueta. Además, en algunas situaciones, incluso cuando el conjunto de entrenamiento es inmenso, encontrar un vecino más cercano se puede hacer rápido.

KNN pertenece a una subcategoría de modelos no paramétricos que se describe como aprendizaje basado en instancias. Los modelos basados en el aprendizaje de instancias se caracterizan por memorizar el conjunto de datos de entrenamiento.

En función de la métrica de distancia elegida, el algoritmo KNN encuentra las k muestras en el conjunto de datos de entrenamiento que son más cercanas (más similares) al punto que se quiere clasificar. La etiqueta de clase del nuevo punto de datos se determina luego por mayoría de votos entre sus k vecinos más cercanos.

De acuerdo a Han and Kamber [2001], los clasificadores de k vecinos más cercanos se basan en el aprendizaje por analogía. Las muestras de entrenamiento se describen mediante atributos numéricos n-dimensionales. Cada muestra representa un punto en un espacio n-dimensional. De esta forma, todas las muestras de entrenamiento se almacenan en un espacio de patrón n-dimensional. Cuando se le da una muestra desconocida, un clasificador k vecino más cercano busca en el espacio del patrón las muestras de entrenamiento k que están más cerca de la muestra desconocida. Estas k muestras de entrenamiento son los k "vecinos más cercanos" de la muestra desconocida. La "cercanía" se define en términos de

distancia euclidiana, donde la distancia euclidiana entre dos puntos, $X = (x_1, x_2, ..., x_n)$ y $Y = (y_1, y_2, ..., y_n)$ se presenta con la Ecuación 2.30

$$d(X, Y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}. \tag{2.30}$$

A la muestra desconocida se le asigna la clase más común entre sus k vecinos más cercanos. Los clasificadores k vecinos más cercanos al estar basados en instancias, contrastan con los métodos de aprendizaje existentes, como la inducción del árbol de decisión y la retropropagación, que construyen un modelo de generalización antes de recibir nuevas muestras para clasificar. Los clasificadores basados en instancias pueden incurrir en costos computacionales elevados cuando la cantidad de vecinos potenciales con los que comparar una muestra no etiquetada dada es grande. Por lo tanto, requieren técnicas de indexación eficientes.

De acuerdo a Kantardzic [2011], el proceso de clasificación de KNN generalmente se basa en los siguientes pasos:

- Se determina el parámetro k correspondiente al número de vecinos más cercanos.
- Se calcula la distancia entre cada muestra de prueba y todas las muestras de entrenamiento.
- Se ordena la distancia y se determinan los vecinos más cercanos en función del umbral k-ésimo.
- Se determina la categoría (clase) para cada uno de los vecinos más cercanos.
- Se utiliza la mayoría simple de la categoría de vecinos más cercanos como el valor de predicción de la clasificación de la muestra de prueba.

Máquinas de Vectores de Soporte

En el enfoque de las máquinas de vectores de soporte (SVM, por sus siglas en ingles, Support Vector Machine) la clasificación óptima de un problema separable de dos clases se logra maximizando el ancho del área vacía (margen) entre las dos clases; este ancho se define como la distancia entre las hipersuperficies de discriminación en n espacio de características dimensionales. Los vectores de cada clase que están más cerca de la superficie de discriminación se denominan vectores de soporte. Al considerar primero el caso linealmente separable de dos clases, con un conjunto de vectores de características n-dimensionales x y los identificadores de clase w asociados con ellos, $w \in \{-1, 1\}$. Para simplificar, suponiendo que los valores de las características individuales x_i se escalan de modo que $x_i \in [0, 1]$ superen la influencia desigual de las características con diferente varianza. La discriminación entre las dos clases se logra definiendo un hiperplano de separación, representado por la Ecuación 2.31 Sonka et al. [2015]

$$\mathbf{w} \cdot x + b = 0. \tag{2.31}$$

Para maximizar el margen, se definen dos hiperplanos paralelos dados por la Ecuación 2.32

$$\begin{aligned} \mathbf{w} \cdot x + b &= 1, \\ \mathbf{w} \cdot x + b &= -1, \end{aligned} \tag{2.32}$$

pasando por los vectores de soporte y sin patrones de entrenamiento entre ellos. Para garantizar que no haya patrones de entrenamiento entre estos dos hiperplanos, para todos los x_i se tiene la Ecuación 2.33

$$w_i(\mathbf{w}\cdot x_i + b) \geq 1. \tag{2.33}$$

Suponiendo que x^+ está en el plano positivo y x^- es el punto más cercano en el plano negativo. Entonces $x^+ - x^-$ es normal al(los) plano(s) y así para algunos λ dada la Ecuación 2.34

$$\begin{aligned} \lambda\mathbf{w} &= x^+ - x^-, \\ \lambda\|\mathbf{w}\|^2 &= x^+\cdot\mathbf{w} - x^-\cdot\mathbf{w}, \\ \lambda\|\mathbf{w}\|^2 &= 2 \end{aligned} \tag{2.34}$$

y por lo tanto $|x^+ - x^-| = (2/\|\mathbf{w}\|)$. Para maximizar el margen, es necesario minimizar $\|\mathbf{w}\|$. La teoría de Lagrange se emplea para reformular el problema de minimización evitando las restricciones de desigualdad y, por lo tanto, simplificando el proceso de optimización. La función lagrangiana se representa con la Ecuación 2.35

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}\|w\| - \sum_{i=1}^{N} \alpha_i w_i(\mathbf{w}_i\cdot x_i + b) + \sum_{i=1}^{N} \alpha_i, \tag{2.35}$$

donde α_i son multiplicadores de Lagrange. $L(\mathbf{w}, b, \alpha)$ se minimiza con respecto a $\mathbf{w}$ y b, mientras que α_i está restringido a $\alpha_i \geq 0$. Tomando la derivada parcial de la función Lagrangiana con respecto a $\mathbf{w}$ y b se obtienen las Ecuaciones 2.36 y 2.37

$$\frac{\partial L(\mathbf{w}, b, \alpha)}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{N} \alpha_i w_i x_i, \tag{2.36}$$

$$\frac{\partial L(\mathbf{w}, b, \alpha)}{\partial b} = \sum_{i=1}^{N} \alpha_i w_i, \tag{2.37}$$

y establecer ambas ecuaciones en cero crea las Ecuaciones 2.38 y 2.39

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i w_i x_i, \tag{2.38}$$

$$\sum_{i=1}^{N} \alpha_i w_i = 0. \tag{2.39}$$

Estas relaciones se vuelven a sustituir en la función lagrangiana original (Ecuación 2.35) para crear un problema de optimización alternativo conocido como formulación dual, teniendo en cuenta las relaciones por pares entre los patrones de entrenamiento x_i y x_j y sus respectivas etiquetas de clase w_i, w_j, obteniedo la Ecuación 2.40

$$L(\mathbf{w}, b, \alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j w_i w_j (x_i \cdot x_j), \tag{2.40}$$

donde $L(\mathbf{w}, b, \alpha)$ se maximiza con respecto a α_i, sujeto a $\sum_{i=1}^{N} w_i \alpha_i = 0$ y $\alpha_i \geq 0$. El escalar b ha desaparecido, pero se puede calcular fácilmente a partir del conjunto de datos original una vez que se optimiza la Ecuación 2.40. La clasificación de dos clases de un patrón x se puede lograr de acuerdo con la Ecuación 2.41

$$f(x) = \mathbf{w} \cdot x_i + b, \tag{2.41}$$

donde w_x se determina utilizando la Ecuación 2.42

$$\begin{aligned} w_x &= +1 \; si \; f(x) \geq 0, \\ &= -1 \; si \; f(x) < 0. \end{aligned} \tag{2.42}$$

La función de discriminación se reformula en términos de vectores de entrenamiento y multiplicadores sustituyendo w usando la Ecuación 2.38 se obtiene la Ecuación 2.43

$$f(x) = \sum_{i \in SV} \alpha_i w_i (x_i \cdot x) + b. \tag{2.43}$$

Cada uno de los multiplicadores de Lagrange α_i comparte un vector de entrenamiento correspondiente x_i. Esos vectores que contribuyen al margen maximizado tendrán α_i distinto de cero y son los vectores de soporte. El resto de los vectores de entrenamiento no contribuyen a la función de discriminación final, por lo que la suma solo se realiza sobre aquellos valores de i para los que x_i es un vector de soporte ($i \in SV$).

La Ecuación 2.42 muestra que la decisión de si un vector x pertenece a la clase $w_x = +1$ o $w_x = -1$ depende únicamente de los vectores de soporte asociados con el margen máximo identificado en la fase de entrenamiento.

Si no se puede encontrar un hiperplano de separación para dividir el espacio de características en dos clases, una solución es tolerar algunos errores de clasificación mínimos. En el espacio lineal, se determina la similitud de dos patrones x_i y x_j con una función kernel $k(x_i, x_j) = (x_i \dot{x}_j)$. Los productos punto del clasificador de vector de soporte lineal se pueden reemplazar con funciones de kernel no lineales como se muestra en la Ecuación 2.44

$$k(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j). \tag{2.44}$$

La idea es transformar los vectores en un espacio en el que se pueda determinar un hiperplano de discriminación lineal, correspondiente a una hipersuperficie no lineal en el espacio de características original.

Calcular el lado derecho de la Ecuación 2.44 puede ser costoso en general, se pueden usar varios kernel simples para este propósito, por ejemplo, polinomios homogéneos de orden d^{th}, polinomios de orden d^{th} no homogéneos, funciones de base radial, funciones de base radial gaussianas. La aplicación de la técnica del kernel a la función dual de Lagrange produce la Ecuación 2.45

$$L(\mathbf{w}, b, \alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j w_i w_j k(x_i, x_j). \tag{2.45}$$

Y la función de discriminación resultante se representa con la Ecuación 2.46

$$f(x) = \sum_{i \in sv} \alpha_i w_i k(x_i, x) + b. \tag{2.46}$$

A continuación se describen los pasos de entrenamiento y clasificación utilizando una máquina de vectores de soporte.

1. Entrenamiento: Se selecciona una función de kernel apropiada, $k(x_i, x_j)$.

2. Se minimiza $\|\mathbf{w}\|$ sujeto a la restricción dada en la Ecuación 2.33. Esto se logra maximizando la función de Lagrange modificada por el kernel, Ecuación 2.45, con respecto a α_i sujeto a las restricciones $\sum_{i=1}^{N} \alpha_i w_i = 0$ y $\alpha_i \geq 0$

3. Se almacena solo el α_i distinto de cero y los vectores de entrenamiento correspondientes x_i. Estos son los vectores de soporte.

4. Clasificación: para cada patrón x, se calcula la función de discriminación, Ecuación 2.46 usando los vectores de soporte x_i y los pesos correspondientes α_i. El signo de la función determina la clasificación de x.

2.3.3. Perceptrón multicapa

Propagación de los patrones de entrada

De acuerdo a Viñuela [2004], el perceptrón multicapa define una relación entre las variables de entrada y las variables de salida de la red. Esta relación se obtiene propagando hacia adelante los valores de las variables de entrada. A continuación, se muestran las expresiones para calcular las activaciones de las neuronas de la red.

Sea un perceptrón multicapa con C capas y n_c neuronas en las capa c, para $c = 1,2,...,C$. Sea $W^c = (w_{ij}^c)$ la matriz de pesos asociada a las conexiones de la capa c a la capa $c+1$ para $c = 1,2,...,C-1$, donde w_{ij}^c representa el peso de la conexión de la neurona i de la capa c a la neurona j de la capa $c+1$; y sea $U^c = (u_i^c)$ el vector de umbrales de las neuronas de la capa c para $c = 2,...,C$. Se denota a_i^c a la activación de la neurona i de la capa c; estas activaciones se calculan del siguiente modo:

- Activación de las neuronas de la capa de entrada (a_i^1). Las neuronas de la capa de entrada se encargan de transmitir hacia la red las señales recibidas del exterior. Por tanto:

$$a_i^1 = x_i \text{ para } i = 1,2,...,n_1, \tag{2.47}$$

  donde $X = (x_1, x_2, ..., x_{n_1})$ representa el vector o patrón de entrada a la red.

- Activación de las neuronas de la capa oculta $c(a_i^c)$. Las neuronas ocultas de la red procesan información recibida aplicando la función de activación f a la suma de los productos de las activaciones que recibe por sus correspondientes pesos, es decir:

$$\begin{gathered} a_i^1 = f\left(\sum_{j=1}^{n_{c-1}} w_{ij}^{c-1} a_j^{c-1} + u_i^c\right) \\ \text{para } i = 1,2...,n_c \text{ y } c = 2,3,...,C-1 \end{gathered} \tag{2.48}$$

  donde a_j^{c-1} son las activaciones de las neuronas de la capa $c-1$.

- Activación de las neuronas de la capa de salida (a_i^C). Al igual que en el caso anterior, la activación de estas neuronas viene dada por la función de activación f aplicada a la suma de los productos de las entradas que recibe por sus correspondientes pesos:

$$\begin{gathered} y_i = a_i^C = f\left(\sum_{j=1}^{n_{C-1}} w_{ij}^{C-1} a_j^{C-1} + u_i^C\right) \\ \text{para } i = 1,2,...,n_C \end{gathered} \tag{2.49}$$

donde $Y = (y_1, y_2, ..., y_{n_C})$ es el vector de salida de la red. La función f es llamada función de activación.

Proceso de aprendizaje del perceptrón multicapa

Los pasos que involucra el proceso completo de aprendizaje del perceptrón multicapa se describen a continuación: Dado el conjunto de muestras o patrones $(X(n), S(n)), n = 1, ...N$, que representan un problema tal, donde $X(n) = (x_i(n), ..., x_{n_1}(n))$ son los patrones de entrada a la red, $S(n) = (s_1(n), ..., s_{n_c}(n))$ son las salidas deseadas para dichas entradas y N es el número de patrones disponibles. Los pasos que componen el proceso de aprendizaje del perceptrón multicapa son los siguientes:

- Paso 1. Se inicializan los pesos y umbrales de la red. Generalmente, esta inicialización es aleatoria y con valores alrededor de cero.

- Paso 2. Se toma un patrón n del conjunto de entrenamiento, $(X(n), S(n))$, y se propaga hacia la salida de la red el vector de entrada $X(n)$ utilizando las Ecuaciones 2.47, 2.48 y 2.49, obteniéndose así la respuesta de la red para dicho vector de entrada, Y(n).

- Paso 3. Se evalúa el error cuadrático $e(n)$ cometido por la red para el patrón n utilizando la Ecuación 2.50.

  2.50:

  $$e(n) = \frac{1}{2} \sum_{i=1}^{n_c} (s_i(n) - y_i(n))^2, \tag{2.50}$$

  en donde $Y(n) = (y_1(n), ..., y_{n_C}(n))$ y $S(n) = (s_1(n), ..., s_{n_C}(n))$ los vectores de salidas de la red y salidas deseadas para el patrón n, respectivamente.

- Paso 4. Se aplica la regla delta generalizada para modificar los pesos y umbrales de la red. Para ello se siguen los siguientes pasos:

- Paso 4.1 Se calculan los valores δ para todas las neuronas de la capa de salida utilizando la Ecuación 2.51

$$\delta_i^C(n) = -(s_i(n) - y_i(n)) f'(\sum_{j=1}^{n_{C-1}} w_{ji}^{C-1} a_j^{C-1} + u_i^C). \tag{2.51}$$

- Paso 4.2 Se calculan los valores δ para el resto de las neuronas de la red utilizando la Ecuación 2.52 empezando desde la última capa oculta y retropropagando dichos valores hacia la capa de entrada.

$$\delta_j^{c+1}(n) = f'(\sum_{k=1}^{n_c} w_{kj}^c a_k^c + u_j^c) \sum_{i=1}^{n_{c+1}} \delta_i^{c+2}(n) w_{ji}^c. \tag{2.52}$$

- Paso 4.3 Se modifican pesos y umbrales de la red siguiendo las Ecuaciones 2.53 y 2.54 para los pesos y umbrales de la capa de salida y 2.55 y 2.56 para el resto de los parámetros de la red.

  Modificación de los pesos capa de salida:

$$\begin{aligned} &w_{ji}^{C-1}(n) = w_{ji}^{C-1}(n-1) + \alpha \delta_i^C(n) a_j^{C-1}(n) \\ &\text{para } j = 1,2,...,n_{C-1}\ i = 1,2,...,n_C. \end{aligned} \tag{2.53}$$

  Modificación de los umbrales capa de salida:

$$\begin{aligned} &u_i^C(n) = u_i^C(n-1) + \alpha \delta_i^C(n) \\ &\text{para } i = 1,2,...,n_C. \end{aligned} \tag{2.54}$$

  Modificación de los pesos:

$$\begin{aligned} &\delta w_{kj}^c(n) = w_{kj}^c(n-1) + \alpha \delta_j^{c+1}(n) a_k^c(n) \\ &\text{para } k = 1,2,...,n_c, j = 1,2,...,n_{c+1} \text{ y } c = 1,2,...,C-2. \end{aligned} \tag{2.55}$$

Modificación de los umbrales:

$$u_j^{c+1}(n) = u_j^{c+1}(n-1) + \alpha\delta_j^{c+1}(n)$$
$$\text{para } j = 1,2,...,n_{c+1} \text{ y } c = 1,2,...,C-2. \tag{2.56}$$

- Paso 5. Se repiten los pasos 2, 3 y 4 para todos los patrones de entrenamiento, completando así una iteración o ciclo de aprendizaje.

- Paso 6. Se evalúa el error total E (Ecuación 2.57) cometido por la red. Dicho error también recibe el nombre de error de entrenamiento, pues se calcula utilizando los patrones de entrenamiento.

$$E = \frac{1}{N}\sum_{n=1}^{N} e(n), \tag{2.57}$$

donde N es el número de patrones o muestras y $e(n)$ es el error cometido por la red para el patrón n, representado por la Ecuación 2.50:

- Paso 7. Se repiten los paso 2, 3, 4, 5 y 6 hasta alcanzar un mínimo del error de entrenamiento, para lo cual se realizan m ciclos de aprendizaje.

2.4. Fundamentos de redes neuronales convolucionales

De acuerdo a Géron [2019], las redes neuronales convolucionales (CNN, por sus siglas en inglés, Convolutional Neural Network) surgieron del estudio de la corteza visual del cerebro y se han utilizado en el reconocimiento de imágenes desde la década de 1980. Utilizan el enfoque del aprendizaje profundo para realizar tareas sin el mismo grado de instrucción explícita que necesitan muchos algoritmos de aprendizaje automático Beysolow [2017]. Con las CNN se puede realizar extracción automática de características, clasificación, detección de anomalías y muchas otras tareas de aprendizaje automático Sarkar et al. [2018].

2.4.1. Filtros

En una red neuronal convolucional los pesos de una neurona son representados como una pequeña imagen del tamaño del campo receptivo (entendiendo por campo receptivo como el grupo de neuronas cercanas que participan en la entrada Vasilev et al. [2019]). Estos son llamados filtros (o filtros de convolución). Una capa llena de neuronas que usa el mismo filtro genera un mapa de características, que resalta las áreas de una imagen que más activan el filtro. Dichos filtros no se definen manualmente: es durante el entrenamiento, cuando la capa convolucional aprenderá automáticamente los filtros más útiles para su tarea, y las capas sucesivas aprenderán a combinarlos en patrones más complejos.

Los tres tipos de capas que comúnmente están presentes en una red neuronal convolucional son convolución, agrupación y activación. Además, un conjunto final de capas a menudo está completamente conectado y se asigna de una manera específica de la aplicación a un conjunto de nodos de salida Aggarwal [2018].

2.4.2. Capa de convolución

El bloque de construcción más importante de una CNN es la capa de convolución: las neuronas en la primera capa convolucional no están conectadas a cada píxel en la imagen de entrada, sino sólo a los píxeles en sus campos receptivos. A su vez, cada neurona en la segunda capa convolucional está conectada solo a las neuronas ubicadas dentro de un pequeño rectángulo en la primera capa (ver Figura 2.14). Esta arquitectura permite que la red se concentre en características pequeñas de bajo nivel en la primera capa oculta, luego las une en características más grandes de nivel superior en la siguiente capa oculta, y así sucesivamente.

Cada neurona de entrada está asociada con un solo peso del filtro. Su propósito es resaltar una característica específica en la entrada, por ejemplo, un borde o una línea. En el contexto de la red, la salida del filtro representa el valor de activación de una neurona en la siguiente

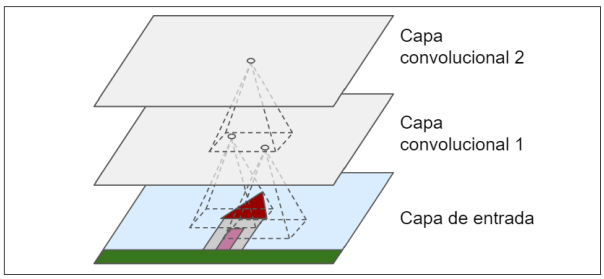


Figura 2.14: Capas convolucionales con campos receptivos locales Géron [2019].

capa. La neurona estará activa, si la característica está presente en esa ubicación espacial Vasilev et al. [2019].

A esto se le conoce como compartir parámetros, y se realiza por dos razones:

- Al reducir la cantidad de pesos, se reduce el consumo de memoria y se evita el sobreajuste.
- El filtro destaca características específicas. Al compartir pesos, se garantiza que el filtro pueda ubicar la característica en toda la imagen.

Las neuronas dispuestas espacialmente se denominan mapa de características, implicando que exista más de un mapa. El mapa puede servir como entrada para otras capas de la red. Un punto importante es que cada neurona de entrada es parte de la entrada de múltiples neuronas de salida. Los mapas de características de entrada y salida tienen diferentes dimensiones. Si se tiene una capa de entrada con tamaño $(ancho, alto)$ y un filtro con dimensiones $(filtro_w, filtro_h)$. Después de aplicar la convolución, las dimensiones de la capa de salida se calculan con la Ecuación 2.58:

$$(ancho - filtro_w + 1, altura - filtro_h + 1). \tag{2.58}$$

La combinación de mapas de características se llama volumen de entrada con profundidad de 3. Se aplica un filtro único de 3×3 a cada mapa. La activación de una neurona de salida es solo la suma ponderada de los filtros aplicados en todos los mapas de características. Es

decir, se combinan los tres filtros en un filtro grande de $3 \times 3 \times 3 + 1$ con 28 pesos (agregando profundidad y un solo sesgo). Luego, se calcula la suma ponderada aplicando los pesos relevantes a cada mapa de características. Un mapa de características de salida puede recibir entradas de:

- Todos los mapas de características de entrada.
- Un solo mapa de características de entrada.

Al denotar el ancho y la altura del filtro con F_w y F_h, la profundidad del volumen de entrada con D y la profundidad del volumen de salida con M, se puede calcular el número total de pesos W en una capa convolucional con la Ecuación 2.59

$$W = (D * F_w * F_h + 1) * M. \tag{2.59}$$

Las convoluciones operan sobre tensores 3D, con dos ejes espaciales (alto y ancho) así como un eje de profundidad (también llamado eje de canales). Para una imagen RGB, la dimensión del eje de profundidad es 3, porque la imagen tiene tres canales de color: rojo, verde y azul. Para una imagen en blanco y negro la profundidad es 1 (nivel de gris). La operación de convolución extrae regiones de su mapa de características de entrada y aplica la misma transformación a todas estas regiones, produciendo un mapa de características de salida. Este mapa de características de salida sigue siendo un tensor 3D: tiene un ancho y una altura. Su profundidad puede ser arbitraria, porque la profundidad de salida es un parámetro de la capa, y los diferentes canales en ese eje de profundidad ya no representan colores específicos como en la entrada RGB; más bien, representan filtros Chollet [2018].

La profundidad de una capa en una red neuronal convolucional no debe confundirse con la profundidad de la propia red. La palabra “profundidad” (cuando se usa en el contexto de una sola capa) se refiere a la cantidad de canales en cada capa, como la cantidad de canales

de colores primarios por ejemplo Rojo, Verde y Azul, en la imagen de entrada o el número de mapas de características en las capas ocultas Aggarwal [2018].

Las redes neuronales convolucionales tienen propiedades que brindan una solución a problemas presentes en las redes neuronales artificiales tradicionales Vasilev et al. [2019]:

- Conectan neuronas, que solo corresponden a los píxeles vecinos de la imagen. De esta manera, las neuronas se ven"forzadas" a recibir únicamente información de otras neuronas que están espacialmente cercanas.
- Una red neuronal convolucional utiliza el intercambio de parámetros. Consiguiendo compartir un número limitado de pesos entre todas las neuronas de una capa.

De este modo, una neurona ubicada en la fila i, columna j de una capa determinada se conecta a las salidas de las neuronas de la capa anterior ubicadas en las filas i a $i + f_h - 1$, columnas j a $j + f_w - 1$, donde f_h y f_w son la altura y el ancho del campo receptivo.

Un ejemplo de la operación de convolución es mostrado en la Figura 2.16a donde se puede ver que se aplica la suma ponderada entre la imagen a color y el filtro de convolución.

Paso

Es posible conectar una capa de entrada grande a una capa mucho más pequeña espaciando los campos receptivos (ver Figura 2.15a). El cambio de un campo receptivo al siguiente se llama paso (en inglés, Stride). Una neurona ubicada en la fila i, columna j en la capa superior se conecta a las salidas de las neuronas en la capa anterior ubicadas en las filas $i \times s_h$ a $i \times s_h + f_h - 1$, columnas $j \times s_w$ a $j \times s_w + f_w - 1$, donde s_h y s_w son los pasos verticales y horizontales. Se puede deslizar el filtro varias posiciones. Por lo general, el paso es el mismo en todas las dimensiones de la entrada Vasilev et al. [2019].

Al usar un paso mayor que 1, se reduce el tamaño del mapa de características de salida. De esta manera, al extender la Ecuación 2.58 para incluir el tamaño del paso, se obtiene la

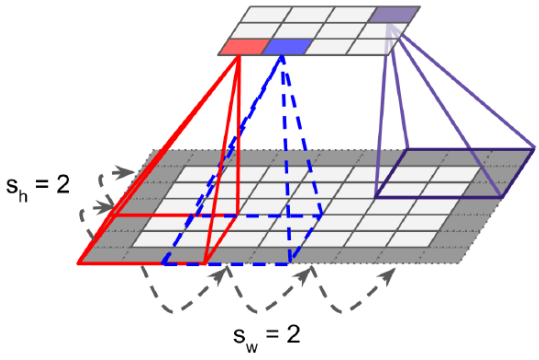
Ecuación 2.60

$$((ancho - filtro_w)/paso_w + 1, ((altura - filtro_h)/paso_h + 1). \tag{2.60}$$

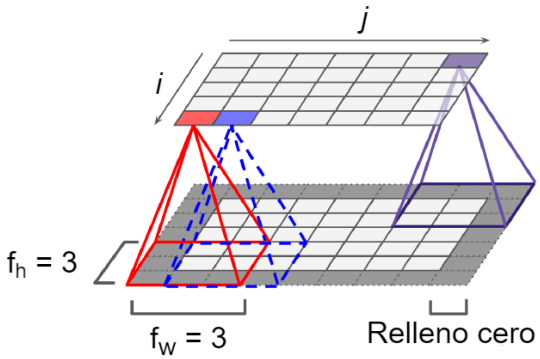
El principal efecto de un paso más grande es un aumento en el campo receptivo de las neuronas de salida. Las neuronas en las siguientes capas capturarán gradualmente la entrada de regiones más grandes de la imagen de entrada. Esto es importante porque permite detectar características más grandes y complejas de la entrada.

Relleno

Por otro lado, para controlar el tamaño de la salida se puede realizar rellenando los bordes del mapa de características de entrada con filas y columnas de ceros antes de la operación de convolución (ver Figura 2.15b). La forma más común de usar el relleno es producir una salida con las mismas dimensiones que la entrada Vasilev et al. [2019].



(a) Reduciendo la dimensionalidad usando un paso de 2



(b) Conexiones entre capas y relleno cero

Figura 2.15: Representación de las técnicas de Paso y Relleno Géron [2019].

Extendiendo la ecuación 2.60 para incluir el relleno (en inglés, Padding) y denotando el tamaño del mapa de características de entrada como $I = (I_w, I_h)$, el tamaño del filtro $F = (F_w, F_h)$, el paso $S = (S_w, S_h)$, y el relleno $P = (P_w, P_h)$. Entonces, el tamaño $O = (O_w, O_h)$ del mapa de características de salida se obtiene con la Ecuación 2.61

$$
\begin{aligned}
O_w &= \frac{I_w + 2P_w - F_w}{S_w} + 1 \\
O_h &= \frac{I_h + 2P_h - F_h}{S_h} + 1.
\end{aligned}
\tag{2.61}
$$

2.4.3. Capa de Agrupación

Su objetivo es submuestrear (es decir, reducir) la imagen de entrada para reducir la carga computacional, el uso de memoria y la cantidad de parámetros. Al igual que en las capas convolucionales, cada neurona en una capa de agrupación está conectada a las salidas de un número limitado de neuronas en la capa anterior, ubicada dentro de un pequeño campo receptivo rectangular. Se debe definir su tamaño y el paso. Sin embargo, una neurona agrupada no tiene pesos; todo lo que hace es agregar las entradas usando una función de agrupación como el máximo o la media.

La operación de máximo (en inglés, Max-Pooling) consiste en extraer regiones de los mapas de características de entrada y generar el valor máximo de cada canal. Es similar a la convolución, excepto que en lugar de transformar las regiones locales a través de una transformación lineal aprendida (el filtro de convolución), se transforman a través de una operación de tensor máximo codificada. Una gran diferencia con la convolución es que la agrupación máxima generalmente se realiza con filtros de 2×2 y paso 2, para reducir la muestra de los mapas de características en un factor de 2. La razón para usar la reducción de muestreo es reducir el número de coeficientes de los mapas de características a procesar, así como inducir jerarquías de filtros espaciales al hacer que las sucesivas capas de convolución vean regiones cada vez más grandes (en términos de la fracción de la entrada original) Chollet [2018].

Las capas de agrupación se definen mediante dos parámetros Vasilev et al. [2019]:

- Paso o Stride, que es similar al de las capas convolucionales.

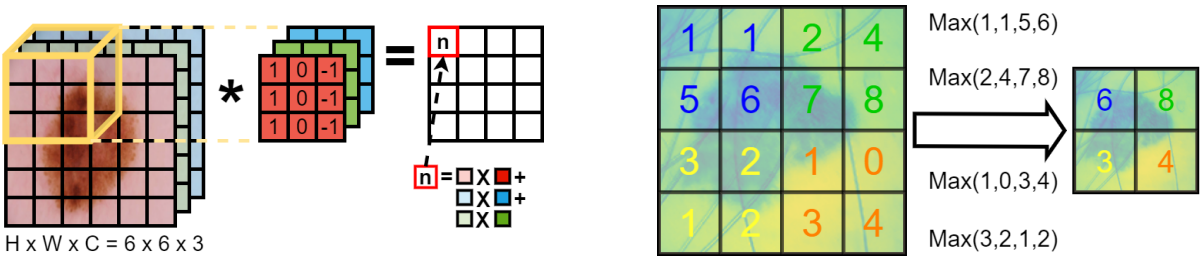
- Tamaño del campo receptivo, que es el equivalente al tamaño del filtro en capas convolucionales.

En la práctica, solo se utilizan dos combinaciones. La primera es un filtro de 2×2 con paso 2, y el segundo es un filtro de 3×3 con paso 2. Si se utiliza un valor mayor para cualquiera de los parámetros, la red pierde demasiada información. Alternativamente, si el paso es 1, el tamaño de la capa no sería menor, ni aumentaría el filtro.

Según estos parámetros, se puede calcular el tamaño de salida de una capa de agrupación. Al denotar el tamaño del segmento de entrada con I, el tamaño del filtro con F, el tamaño del paso con S y el tamaño de la salida con O. Las capas de agrupación normalmente no tienen relleno. Entonces el tamaño de salida se obtiene con la Ecuación 2.62

$$\begin{aligned} O_w &= \frac{I_w - F_w}{S_w} + 1 \\ O_h &= \frac{I_h - F_h}{S_h} + 1. \end{aligned} \tag{2.62}$$

En la Figura 2.16b se muestra un ejemplo de cómo funciona la capa de agrupación máxima, en donde, se obtiene el coeficiente máximo mediante un filtro de tamaño 2×2 y paso 2.



(a) Convolución. **(b)** Agrupación.

Figura 2.16: Ejemplos de operación de convolución y agrupación máxima.

Una vez entendidas las capas principales de una red neuronal convolucional se puede pasar a entender el proceso matemático representado por las siguientes ecuaciones. Cuando

se tiene una imagen con un tamaño de $M \times N$ y filtros con un tamaño de $m \times n$, la convolución se puede representar con la Ecuación 2.63

$$Z_{ij}^{(k)} = \sum_{s=0}^{m-1} \sum_{t=0}^{n-1} w_{st}^{(k)} x(i+s)(j+t). \tag{2.63}$$

En la Ecuación 2.63, w es el peso del filtro, es decir, el parámetro del modelo. La Ecuación 2.63, sin embargo, no es suficiente cuando se piensa en capas multiconvolucionales porque no tiene la información del canal. Por lo tanto se agrega un parámetro al filtro. La Ecuación 2.63 extendida se puede representar con la Ecuación 2.64

$$Z_{ij}^{(k)} = \sum_{c} \sum_{s=0}^{m-1} \sum_{t=0}^{n-1} w_{st}^{(k,c)} x_{(i+s)(j+t)}^{(c)}. \tag{2.64}$$

Ahora en la Ecuación 2.64, c denota el canal de la imagen. Si el número de filtros es K y el número de canales es C, se tiene $W \in \mathbb{R}^{K \times C \times m \times n}$. Luego, el tamaño de la imagen convolucionada es $(M-m+1) \times (N-n+1)$. Después de la convolución, todos los valores convolucionados serán activados por la función de activación.

Con la activación, se obtiene la Ecuación 2.65

$$a_{ij}^{(k)} = h(z_{ij}^{(k)} + b^{(k)}) = max\left(\ 0, z_{ij}^{(k)} + b^{(k)} \ \right). \tag{2.65}$$

En la Ecuación 2.65, b denota el sesgo, el otro parámetro del modelo. Se puede ver que b no tiene subíndices de i y j , es decir, se tiene $b \in \mathbb{R}^K$, una matriz unidimensional. Por lo tanto, se han propagado hacia adelante los valores de la capa convolucional. Luego viene la capa de agrupación máxima. La propagación se escribe como la Ecuación 2.66

$$y_{ij}^{(k)} = max\left(\ a_{(l_1 i+s)(l_2 j+t)}^{(k)} \ \right). \tag{2.66}$$

En la Ecuación 2.66, l_1 y l_2 son el tamaño del filtro de agrupación y $s \in [0, l_1], t \in [0, l_2]$.

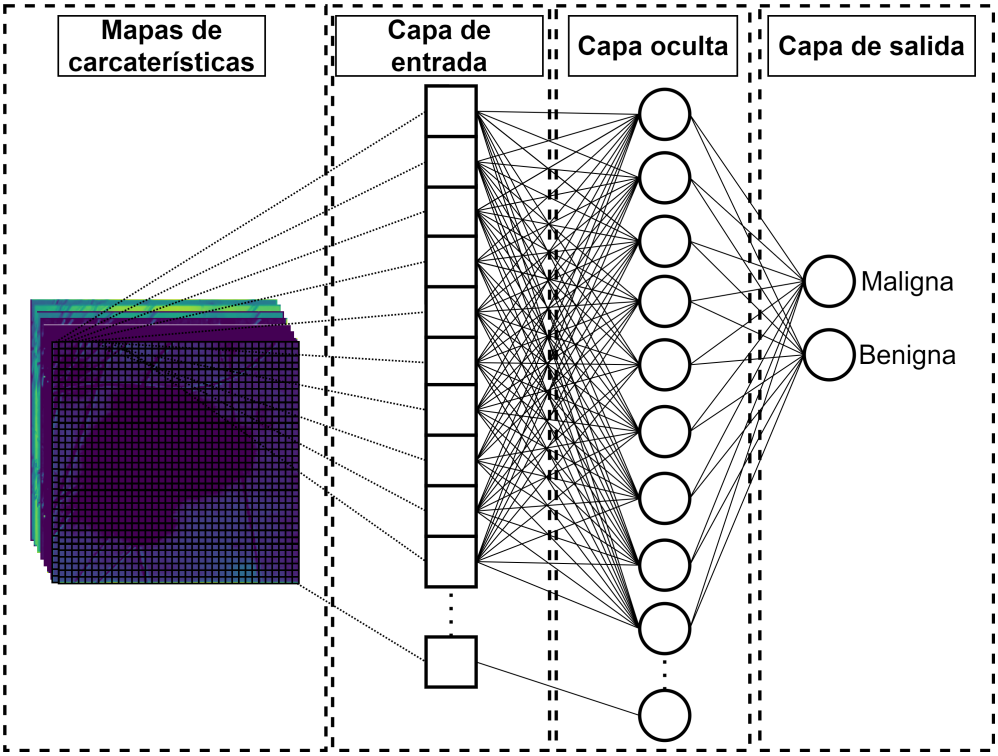


Figura 2.17: Proceso de aplanamiento y perceptrón multicapa.

Por lo general, l_1 y l_2 se establecen en el mismo valor de $2 \sim 4$.

El perceptrón multicapa simple sigue las capas convolucionales y las capas de agrupación máxima para clasificar los datos. Dado que el perceptrón multicapa solo puede aceptar datos unidimensionales, se aplanan los datos reducidos como preprocesamiento para adaptarlos a la capa de entrada de perceptrón multicapa (ver Figura 2.17). La extracción de características se completó antes del perceptrón multicapa, por lo que formatear los datos en una dimensión no será un problema. Por lo tanto, la red neuronal convolucional puede clasificar los datos de la imagen una vez que se optimiza el modelo. Para hacer esto, al igual que con otras redes neuronales, se aplica el algoritmo de retropropagación a la red neuronal convolucional para entrenar el modelo. El error de la capa de entrada del perceptrón multicapa se retropropaga a la capa de agrupación máxima, y esta vez no se aplana a dos dimensiones para adaptarse correctamente al modelo. Dado que la capa de agrupación máxima no tiene parámetros de modelo, simplemente retropropaga el error a la capa anterior.

2.5. Evaluación de un clasificador

La evaluación de un modelo se reduce a dividir los datos disponibles en tres conjuntos: entrenamiento, validación y prueba. Se entrena con los datos de entrenamiento y se evalúa el modelo con los datos de validación. Una vez que el modelo está listo, se prueba una última vez en los datos de prueba.

2.5.1. Conjuntos de entrenamiento, validación y prueba

Entrenar, validar y probar un modelo de aprendizaje automático es una tarea que consiste en dividir los datos con los que se cuenta. Es decir, se pueden dividir aleatoriamente los datos en conjuntos de entrenamiento y prueba. En donde, la división común es de 25 % a 30 % para pruebas y el restante 75 % a 70 % por ciento para entrenamiento. Pero cuando se necesita ajustar el modelo de aprendizaje los datos divididos de prueba no son una buena práctica porque provocan un sobreajuste, para evitar esto, se necesita una tercera división, llamada conjunto de validación. Una división sugerida es dividir los ejemplos en tercios: 70 % para entrenamiento, 20 % para validación y 10 % para pruebas. La división debe realizarse al azar, de lo contrario, la prueba no será confiable, y puede presentarse una sobreestimación o subestimación en la distribución de los datos.

2.5.2. Validación Cruzada

Al dividir los datos en los conjuntos de prueba y entrenamiento puede implicar el introducir sesgo en las pruebas, esto al reducir el tamaño de los datos de entrenamiento. Por otra parte, en ocasiones el conjunto de datos utilizado es escaso y no se quieren desperdiciar datos en la validación. El método de validación cruzada de k iteraciones esta diseñado para brindar una estimación precisa del error real sin desperdiciar demasiados datos.

En la validación cruzada de k iteraciones, se divide aleatoriamente el conjunto de datos de

entrenamiento en k pliegues sin reemplazo, donde $k-1$ pliegues se usan para el entrenamiento del modelo y un pliegue se usa para la evaluación del rendimiento. Este procedimiento se repite k veces para obtener k modelos y estimaciones de rendimiento. Luego, se calcula el rendimiento promedio de los modelos en función de los diferentes pliegues independientes para obtener una estimación del rendimiento que sea menos sensible a la subdivisión de los datos de entrenamiento. La validación cruzada de k iteraciones se utiliza para el ajuste del modelo, es decir, encontrar los valores de hiperparámetro óptimos que producen un rendimiento de generalización satisfactorio.

Una vez que se tienen los valores de hiperparámetro satisfactorios, se puede volver a entrenar el modelo en el conjunto de entrenamiento completo y obtener una estimación de rendimiento final utilizando el conjunto de prueba independiente. La razón detrás de ajustar un modelo a todo el conjunto de datos de entrenamiento después de la validación cruzada de k iteraciones es que proporcionar más muestras de entrenamiento a un algoritmo de aprendizaje generalmente da como resultado un modelo más preciso y sólido.

Un buen valor estándar para k en la validación cruzada de k iteraciones es 10. Sin embargo, al trabajar con conjuntos de entrenamiento relativamente pequeños, puede ser útil aumentar el número de pliegues. Al aumentar el valor de k, se usarán más datos de entrenamiento en cada iteración, lo que resulta en un menor sesgo hacia la estimación del rendimiento de la generalización promediando las estimaciones del modelo individual. Sin embargo, los valores grandes de k también aumentarán el tiempo de ejecución del algoritmo de validación cruzada y producirán estimaciones con mayor varianza, ya que los pliegues de entrenamiento serán más similares entre sí. Por otro lado, al trabajar con grandes conjuntos de datos, podemos elegir un valor más pequeño para k, por ejemplo, $k = 5$, y aún así obtener una estimación precisa del rendimiento promedio del modelo mientras se reduce el costo computacional de reacondicionamiento y evaluando el modelo en los diferentes pliegues.

Una representación gráfica del método de validación cruzada de k iteraciones se muestra

la Figura 2.18

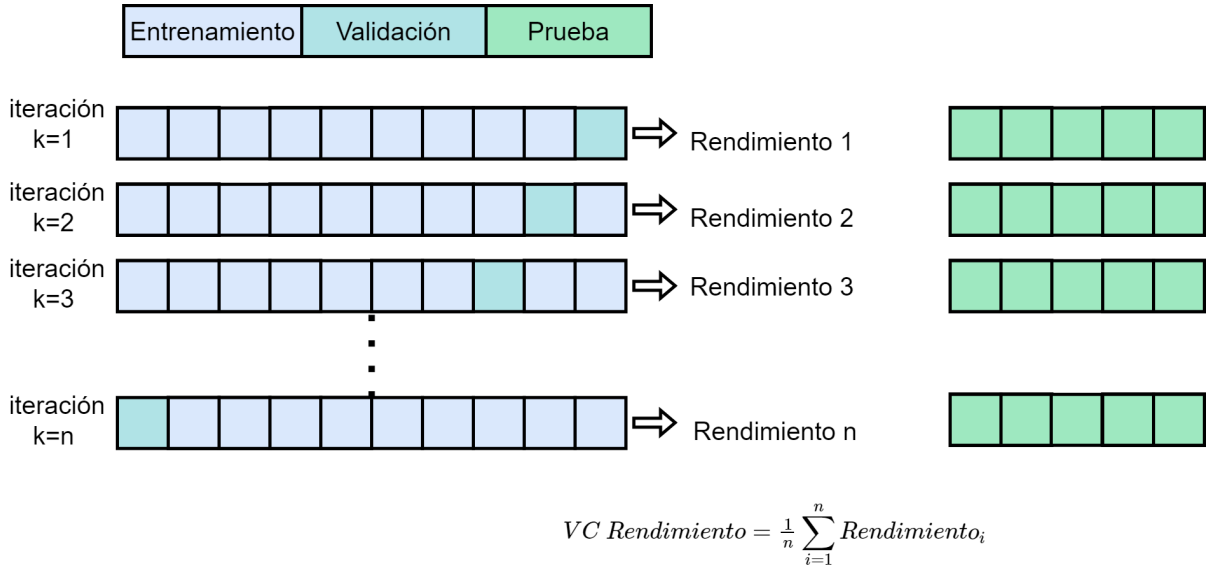


Figura 2.18: Representación del método de validación cruzada de k iteraciones.

2.5.3. Métricas de evaluación

Las métricas de evaluación sirven para evaluar qué tan bueno o qué tan "preciso" es un algoritmo para predecir la etiqueta de clase de las tuplas. Las medidas de evaluación son la exactitud (también conocida como tasa de reconocimiento), sensibilidad (o recuerdo), precisión, entre otras.

Hay cuatro términos que son los "bloques de construcción" utilizados en el cálculo de muchas medidas de evaluación.

- Verdaderos positivos (VP): se refieren a las tuplas positivas que fueron etiquetadas correctamente por el algoritmo.
- Negativos verdaderos (VN): son las tuplas negativas que el algoritmo etiquetó correctamente.
- Falsos positivos (FP): estas son las tuplas negativas que se etiquetaron incorrectamente como positivas.

- Falsos negativos (FN): estas son las tuplas positivas que se etiquetaron erróneamente como negativas.

La exactitud de un algoritmo en un conjunto de prueba dado es el porcentaje de tuplas de conjuntos de prueba que el algoritmo clasifica correctamente, ver la Ecuación 2.67

$$Exactitud = \frac{VP + VN}{VP + VN + FP + FN}. \tag{2.67}$$

También se conoce como la tasa de reconocimiento general del algoritmo, la cual, refleja qué tan bien reconoce el algoritmo las tuplas de las diversas clases. Las medidas de precisión y recuperación también se utilizan ampliamente en la clasificación. La precisión se puede considerar como una medida de exactitud, mientras que la sensibilidad (la verdadera tasa positiva) es una medida de integridad. Las Ecuaciones para la medida de precisión y recuperación se muestran en la Ecuación 2.68 y 2.69 respectivamente.

$$Precisión = \frac{VP}{VP + FP}. \tag{2.68}$$

$$Sensibilidad = \frac{VP}{VP + FN}. \tag{2.69}$$

Una puntuación de precisión perfecta de 1.0 para una clase C significa que cada tupla que el algoritmo etiquetó como perteneciente a la clase C pertenece de hecho a la clase C. Sin embargo, no dice nada sobre el número de tuplas de clase C que el algoritmo etiquetó incorrectamente. Una puntuación de sensibilidad perfecta de 1.0 para C significa que todos los elementos de la clase C fueron etiquetados como tales, pero no dice cuántas otras tuplas fueron etiquetadas incorrectamente como pertenecientes a la clase C. Tiende a haber una relación inversa entre precisión y sensibilidad, donde es posible aumentar uno a costa de reducir el otro. Las puntuaciones de precisión y sensibilidad se suelen utilizar juntas, donde los valores de precisión se comparan para obtener un valor fijo de sensibilidad, o viceversa.

Otra de las métricas disponibles es la medida F1 o puntaje F1 es la media armónica o el promedio ponderado de precisión y sensibilidad. Es una métrica de rendimiento común para evaluar clasificadores multiclase. También es una buena medida cuando hay una distribución desigual de clases. La mejor puntuación de F1 es 1, mientras que la peor puntuación es 0. Una buena medida de F1 significa que tiene pocos falsos negativos y falsos positivos. La medida F1 se define con la Ecuación 2.70

$$MedidaF1 = 2 \times \frac{Precisión \times Sensibilidad}{Precisión + Sensibilidad}. \tag{2.70}$$

Por otro lado, el área bajo la característica operativa del receptor (AUROC, por su siglas en inglés, Area Under the Receiver Operating Characteristic) es una métrica de rendimiento común para evaluar clasificadores binarios. La característica operativa del receptor (ROC, por sus siglas en inglés, Receiver Operating Characteristic Curve) es un gráfico que traza la tasa de positivos verdaderos frente a la tasa de falsos positivos. El área bajo la curva (AUC, por sus siglas en inglés, Area Under the Curve) es el área debajo de la curva ROC. El AUC se puede interpretar como la probabilidad de que el modelo clasifique un ejemplo positivo aleatorio más alto que un ejemplo negativo aleatorio. Cuanto más cerca está el AUROC de 1.0, mejor se está desempeñando el modelo. Un modelo con AUROC de 0.5 es no es bueno ya que su precisión predictiva es tan buena como la suposición aleatoria Quinto [2020] Cortes and Mohri [2003]. El AUROC se define con la Ecuación 2.71

$$A = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} 1_{x_i > y_j}}{mn}. \tag{2.71}$$

Capítulo 3

Herramientas utilizadas

En este capítulo se describen aquellas herramientas que fueron utilizadas como parte del desarrollo de esta investigación y sirvieron para construir las arquitecturas de red neuronal convolucional propuestas, dentro de las cuales, destacan el lenguaje de programación Python, el entorno de desarrollo Jupyter Notebook y varias librerías de apoyo como Scikit-learn, Numpy, Pandas, Matploitlib, Tensorflow, Keras, además del gestor de paquetes Anaconda, entre otras.

3.1. Lenguaje de programación

Python es un lenguaje de alto nivel de propósito general. Las funciones de alto nivel hacen de Python una alternativa para el desarrollo rápido de aplicaciones complejas Maruch and Maruch [2006]. Entre sus características se encuentra: Python se interpreta (escribir programas y corregir errores es rápido), Python se encarga de la gestión de la memoria, Python tiene funciones de depuración integradas, Python es un lenguaje multiparadigma, Múltiples sistemas operativos e interfaces de usuario, Tipos especiales de datos (como imágenes y sonido).

3.2. Entorno de desarrollo

Jupyter Notebook es un entorno computacional interactivo basado en IPython. A diferencia de la consola nativa de Python, el código y los datos importados se pueden reutilizar fácilmente. El código y otros contenidos se pueden separar en bloques (celdas) para una mejor organización. Jupyter Notebook funciona como una aplicación de cliente-servidor y proporciona una interfaz de navegador web ordenada donde se puede editar y ejecutar código. Puede ejecutarse localmente incluso en una computadora sin acceso a Internet Yu and Chung [2017].

3.3. Bibliotecas de Python

Scikit-learn

Scikit-learn Buitinck et al. [2013], Pedregosa et al. [2011], proporciona algoritmos para tareas de aprendizaje automático que incluyen clasificación, regresión, reducción de dimensionalidad y agrupación. También proporciona módulos para extraer características, procesar datos y evaluar modelos Hackeling [2014].

Numpy

Numpy es una biblioteca de Python para cálculos numéricos. Agrega soporte para arreglos multidimensionales. La biblioteca NumPy actual es la sucesora de la biblioteca Numeric. Se trata de un proyecto de código abierto y es una de las bibliotecas de Python más populares. Se utiliza en casi todas las bibliotecas de aprendizaje automático y computación científica Sarkar et al. [2018].

Pandas

Pandas es una biblioteca de Python para la manipulación y el análisis de datos. Funciona como un conjunto de herramientas intuitivo y fácil de usar para realizar operaciones en cualquier tipo de datos. Pandas permite trabajar tanto con datos transversales como con datos basados en series temporales. Toda la representación de datos en Pandas se realiza utilizando dos estructuras de datos principales; serie y marcos de datos Sarkar et al. [2018].

Matplotlib

Matplotlib es una extensión matemática numérica de NumPy y un paquete para ver o presentar datos en un formato pictórico o gráfico. Facilita la toma de decisiones permitiendo ver los análisis presentados visualmente, para que se puedan comprender conceptos difíciles o identificar nuevos patrones Swamynathan [2017].

TensorFlow

Tensorflow es un proyecto de código abierto de Google que tiene compatibilidad con varios tipos de arquitecturas, incluidas las redes neuronales convolucionales, los codificadores automáticos apilados, la red de creencias profundas y las redes neuronales recurrentes. En Tensorflow, una red se especifica como un gráfico simbólico de operaciones vectoriales, como suma/multiplicación de matriz o convolución, y cada capa es una composición de esas operaciones. Tensorflow usa un lenguaje de secuencias de comandos de alto nivel que es útil para la implementación rápida de modelos Vieira and Ribeiro [2018].

Keras

Keras está diseñado para ser una API de alto nivel para redes neuronales que se basa en marcos como Tensorflow, CNTK o Theano. La biblioteca permite la creación rápida de

prototipos de redes neuronales, admite una amplia variedad de arquitecturas de red y se puede ejecutar tanto en CPU como en GPU. Algunos de los módulos específicos disponibles en la biblioteca de Keras, son capas neuronales, funciones de costo, optimizadores, esquemas de inicialización, funciones de activación y esquemas de regularización. Estos módulos tienen funciones relevantes que se pueden utilizar para optimizar el rendimiento del entrenamiento de redes neuronales para tareas específicas Bhagwat et al. [2019].

Anaconda

Anaconda es una distribución gratuita de paquetes de Python distribuidos por Continuum Analytics. Esta distribución es compatible con los sistemas operativos Linux, Windows y Mac OSX. Cuando se instala la distribución Anaconda en el sistema, se tiene la oportunidad de utilizar muchas herramientas y aplicaciones, sin tener que instalar y administrar cada una de ellas por separado. La gestión de toda la distribución de Anaconda se realiza mediante una aplicación llamada conda. Este es el administrador de paquetes y el administrador de entornos de la distribución de Anaconda que maneja todos los paquetes y sus versiones. Uno de los aspectos más interesantes de esta distribución es la capacidad de gestionar múltiples entornos de desarrollo, cada uno con su propia versión de Python Nelly [2015].

Capítulo 4

Trabajos relacionados

El diagnóstico de cáncer de piel ha sido estudiado por algunos autores, los cuales, buscan aplicar las mejores técnicas para apoyar en la detección de esta enfermedad. A continuación se presentan algunos artículos relacionados.

Primero, revisando el trabajo de Ottom [2019], en donde, emplea el conjunto de datos ISIC 2017: Skin Lesion Analysis Towards Melanoma Detection Challenge. El cual contiene un conjunto de 2000 imágenes dermoscópicas. El conjunto de imágenes incluye también un conjunto de validación con 150 imágenes y un conjunto de prueba con 600 imágenes. Se utilizaron métodos de zoom de imagen y cambio de imagen para producir 3000 imágenes para cada etiqueta de clase ya que el conjunto de datos no está equilibrado y consta de 1626 imágenes benignos y 374 malignos. El trabajo consistió de tres fases de pre-procesamiento. Como primer fase, las imágenes del conjunto de datos se convirtieron del espacio de color RGB al espacio de color HSV para facilitar la extracción de característica. En la segunda fase, se aplicó el filtro bilateral a todas las imágenes, para mantener los bordes nítidos en la imagen. En la tercera fase, se convirtieron las imágenes en imágenes en escala de grises reduciendo la complejidad y la dimensión de las imágenes para posteriormente detectar automáticamente los bordes de los objetos en una imagen utilizando el método de detección de bordes de Canny. El paso final del pre-procesamiento se creó una máscara para cada imagen y se aplicó

el método Bitwise para extraer el objeto deseado en la imagen. Se introdujo el conjunto de datos de entrenamiento en un modelo de red neuronal convolucional, cuya arquitectura es de tres capas de convolución, tres capas de agrupación máxima y cuatro capas completamente conectadas. Se emplearon 25 épocas para entrenar el modelo obteniendo en la época 25 una exactitud de 0.74.

En el caso del trabajo realizado por AlShourbaji et al. [2021], el conjunto de datos de cáncer de piel utilizado cuenta con un total de 3297 imágenes, de las cuales, 2637 son de entrenamiento y 660 imágenes de prueba. Este conjunto tiene dos clases principales, que incluyen melanoma y casos benignos. La distribución del conjunto de imágenes de cáncer de piel se dividió en 2077 imágenes para entrenamiento, 560 para validación y 660 para prueba. Se empleó la técnica de data augmentation para realizar una rotación de 40 grados, un rango de zoom y un rango de corte de 0.2. Las imágenes se voltearon horizontalmente y verticalmente para aumentar la diversidad y evitar el sobreajuste del modelo. Para el entrenamiento del modelo de red neuronal convolucional se utilizaron 30 épocas, la tasa de aprendizaje fue de 0.0001 y se utilizó el optimizador de Adam. El modelo logró una exactitud de 0.824 y una pérdida de prueba de 0.381.

Por su parte, en el trabajo de Zaman et al. [2021] el conjunto de datos cuenta con 3000 imágenes RGB con una resolución de 28×28 píxeles. El conjunto de datos se divide en cuatro clases, 1000 imágenes por clase. En cada clase hay 750 imágenes (75 %) para entrenamiento y 250 imágenes (25 %) para validación. En este trabajo se experimentó con diferentes modelos de red neuronal convolucional, los cuales consistieron en: el primer experimento utilizó un modelo con una convolución, una agrupación máxima y una capa completamente conectada. Para entrenar esta red CNN, esta arquitectura usó una capa convolucional con un tamaño de filtro de 7×7 y un tamaño de canal de 8 con el mismo relleno, una capa max-pooling de 2×2 con stride 2 y una capa totalmente conectada logrando una exactitud de 86.79. En el segundo experimento el modelo consistió de dos capas convolucionales, dos max-pooling y una capa

completamente conectada. Para entrenar esta red CNN, esta arquitectura utilizó la primera capa de convolución con un tamaño de filtro de 7 × 7 y un tamaño de canal de 8 con el mismo relleno, max-pooling de 2 × 2 con stride 2. Luego otra capa convolucional con un tamaño de filtro de 5 × 5 y un tamaño de canal de 16 con el mismo relleno, max-pooling de 2 × 2 con stride 2 y una capa totalmente conectada, logrando una exactitud de clasificación de 90.03. El tercer experimento consistio de un modelo con tres convoluciones, tres capas de agrupación máxima y una capa completamente conectada. Para entrenar esta red CNN, esta arquitectura usó la primera capa convolucional con un tamaño de filtro de 7 × 7 y un tamaño de canal de 8 con el mismo relleno, max-pooling de 2 × 2 con stride 2. Luego otra una segunda capa convolucional con un tamaño de filtro de 5 × 5 y el tamaño del canal fue 16 con el mismo relleno, max-pooling de 2 × 2 con stride 2, y finalmente una tercer capa convolucional con un tamaño de filtro de 3 × 3 y el tamaño del canal es 32 con el mismo relleno, max-pooling de 2 × 2 con stride 2 y capa totalmente conectada para lograr una exactitud de 92.44.

Por otro lado, en su trabajo Gerges and Shih [2021] utiliza un conjunto de datos que consta de 170 imágenes en las que 70 imágenes representan casos de melanoma, en donde, una imagen puede representar un caso benigno o maligno. Se utilizaron una serie de pasos de preprocesamiento de imágenes para eliminar todas las características no deseadas de las imágenes, los cuales consistieron de mejorar el contraste de las imágenes para ajustar la intensidad y segmentar la imagen y extraer la región de interés en cuyo caso en los conjuntos de datos que se utilizaron constan de dos regiones la piel normal y el lunar que se encuentra en la piel. Se utilizó la segmentación basada en clusters con $k = 2$ seguida del relleno de imágenes binarias para llenar ciertos huecos. Una vez segmentada la imagen se mantuvo el lunar y se remplazaron todos los píxeles normales de la piel por píxeles negros. Una vez obtenidas las imágenes pre procesadas se utilizaron para alimentar un modelo de aprendizaje profundo basado en redes neuronales convolucionales denominado Melanoma-CNN, con el objetivo de clasificar las imágenes de lesiones cutáneas como melanoma o no. El modelo

consta de dos capas convolucionales, cada una seguida de una capa de agrupación máxima y promedio respectivamente. La última parte del modelo tiene dos capas de salida totalmente conectadas. La función de activación utilizada fue ReLU excepto en la última capa en donde se utilizó la función de activación Softmax. El modelo fue desarrollado utilizando Keras. Una vez preparadas las imágenes y el modelo se realizó una validación cruzada de 10 veces con lo que se obtuvo una exactitud del 97 %, además de una desviación estándar baja de 0.01. Un segundo experimento realizado en este artículo fue probar el modelo utilizando las imágenes sin el paso de ajuste de contraste, únicamente se segmenta y extrae la región de interés obteniendo una exactitud de 93 %.

Finalmente, el trabajo de Nasr Esfahani et al. [2016] propone un sistema que consta de una etapa de preprocesamiento. Las imágenes pre procesadas se introducen en la segunda etapa, que es un modelo de red neuronal convolucional. En la etapa de pre procesamiento se corrigen los efectos de iluminación detectando los cambios bruscos en los canales de saturación y valor del espacio de color HSV. Esto se realizó sin destruir los bordes reales de la imagen original. Se produjo una máscara de segmentación aplicando un clasificador de k-medias con $k = 2$ sobre la imagen preprocesada para extraer la región de la lesión. Esta máscara fue reforzada con algunas operaciones morfológicas. Para reducir los efectos de la textura de la piel normal en el proceso de clasificación, se utilizó la máscara de segmentación para alisar el área exterior de la lesión. Esto se realizó aplicando un filtro gaussiano con $\sigma = 2$ sobre las partes normales de la piel en base a la información de la máscara de segmentación. El modelo de red neuronal utilizado consiste de dos capas convolucionales con un filtro de 5×5. Obteniendo 20 mapas de características en la primera capa de convolución y 50 mapas de características en la segunda capa de convolución. Hay una capa de agrupación después de cada capa de convolución. Las salidas de estas cuatro capas se alimentan a una etapa de 2 capas totalmente conectadas que tiene respectivamente 100 y 2 neuronas. El conjunto de entrenamiento de 170 imágenes se incrementó a 6120 imágenes originales y

sintetizadas. Para realizar el entrenamiento y la prueba, el conjunto de datos se dividió en dos grupos seleccionados al azar. Se utiliza una proporción del 80 % al 20 % en la que el 80 % de las imágenes del conjunto de datos se seleccionan aleatoriamente para el entrenamiento y el resto se usa para la prueba. Las imágenes de entrenamiento se envían a una red con un tamaño de lote de 64. La red convolucional se entrenó a través de 20000 iteraciones. Para que los resultados sean independientes de los datos de prueba y entrenamiento seleccionados, el procedimiento de aprendizaje y prueba se repitió 50 veces. El modelo propuesto se evaluó con un conjunto de imágenes, el cual, consta de 170 imágenes 70 melanomas y 100 nevus. El modelo propuesto fue implementado en un servidor con procesador Intel Core i7-4790K, 32GB de RAM y dos tarjetas GPU NVIDIA GeForce GTX Titan X con interfaz de enlace escalable (SLI). Obteniendo con esto una exactitud de 0.81.

En la Tabla 4.1, se muestra un resumen de los trabajos antes mencionados.

Artículo	Año	Origen de imágenes	No. de imágenes procesadas	Método de clasificación	Desempeño (Exactitud)
Ottom [2019]	2019	Archivo ISIC 2017.	6000 Imágenes.	- Red neuronal convolucional: tres capas de convolución, tres capas de agrupación máxima y cuatro capas completamente conectadas.	74.0 %
AlShourbaji et al. [2021]	2021	Archivo ISIC (Año no especificado).	3297 Imágenes.	- Red neuronal convolucional.	82.4 %
Zaman et al. [2021]	2021	Base de datos extraída de GitHub.	3000 Imágenes.	- Red neuronal convolucional 1: una capa de convolución, una capa de agrupación máxima y una capa completamente conectada. - Red neuronal convolucional 2: dos capas de convolución, dos capas de agrupación máxima y una capa completamente conectada. - Red neuronal convolucional 3: tres capas de convolución, tres capas de agrupación y una capa completamente conectada.	86.79 % 90.03 % 92.44 %
Gerges and Shih [2021]	2021	Base de datos MED-NODE.	170 Imágenes.	- Red neuronal convolucional: dos capas de convolución, dos capas de agrupación máxima y dos capas completamente conectadas.	97.0 %
Nasr Esfahani et al. [2016]	2016	Base de datos MED-NODE.	6120 Imágenes.	- Red neuronal convolucional: dos capas de convolución, dos capas de agrupación y dos capas completamente conectadas.	81.0 %

Tabla 4.1: Resumen de los trabajos relacionados. Elaboración propia.

Capítulo 5

Metodología de investigación

La secuencia de actividades para desarrollar este proyecto de investigación pueden ser vistas en 3 fases, las cuales, consistieron en lo siguiente:

Fase 1: Consiste en observar el desempeño de algunos algoritmos de aprendizaje automático clásicos, tales como Árboles de Decisión (AD), Bosques Aleatorios (BA), Regresión Logística (RL), K Vecinos más Cercanos (KVC), Máquinas de Vectores de Soporte (MVS) y Perceptrón Multicapa (PM), utilizando Imágenes Sin Segmentar y Con Artefactos (ISSCA) e Imágenes Segmentadas y Sin Artefactos (ISSA). Ésto, con el fin de determinar si hay alguna mejora en el desempeño de estos algoritmos cuando se utilizan imágenes ISSA en comparación con imágenes ISSCA, es decir, tomadas directamente del Archivo ISIC 2019.

El proceso ejecutado en esta Fase consiste en crear dos subconjuntos de 100 imágenes bajo los siguientes términos:

- El primer subconjunto consta de imágenes ISSCA de las cuales 50 imágenes pertenecen a clase Benigna y 50 a clase Maligna.
- El segundo subconjunto consta de imágenes ISSA de las cuales 50 imágenes pertenecen a clase Benigna y 50 a clase Maligna.

Con cada subconjunto se entrenan los algoritmos mencionados anteriormente, y se mide

el desempeño de cada uno con las métricas de Exactitud (EXA), Precisión (PRE), Sensibilidad (SEN), Puntuación F1 (F1) y Área Bajo la Curva (AUC).

Los resultados obtenidos durante esta fase pueden ser consultados en el Capítulo 6, en las Tablas 6.1, para el caso de imágenes ISSA y Tabla 6.2 para imágenes ISSCA. Analizando el desempeño de cada algoritmo por la métrica de exactitud se observa que los algoritmos BA, kVC y MVS tuvieron mejor desempeño dando una exactitud de 78.99 %, 78.0 % y 75.5 %, respectivamente, para realizar un diagnóstico correcto. Esto al clasificar las imágenes segmentadas y sin artefactos (ISSA). Para mayor comprensión, se puede ver la Figura 6.1 donde se muestra la gráfica de exactitud por algoritmo.

Por otro lado, en el caso de las imágenes ISSCA los algoritmos que tuvieron mejor desempeño fueron BA, kVC y MVS dando una exactitud de 82.0 %, 81.5 % y 74.5 %, respectivamente, pero en este caso, se observa un incremento de 3.01 % y 3.5 % en la exactitud de los algoritmos BA y kVC. Por su parte, el algoritmo MVS tuvo un decremento del 1 %. Para mayor detalle se puede ver la Figura 6.1.

Fase 2: Derivado de los resultados obtenidos en la Fase 1, en donde se visualiza que la exactitud de diagnosticar una lesión como Benigna tuvo mejor probabilidad al trabajar con imágenes ISSCA, se decide trabajar con estas imágenes, pero en esta Fase 2 se optó por incrementar la cantidad de imágenes a 1200, de la cuales, 600 son lesiones de piel de clase Benigna y 600 lesiones de piel de clase Maligna. De igual forma, se utilizan los mismos algoritmos de la Fase 1 y además se emplea una arquitectura de red neuronal convolucional (RNC) con el fin de comparar su desempeño contra los algoritmos de aprendizaje automático clásicos utilizados en la Fase 1. La arquitectura empleada para la red neuronal convolucional consiste de 3 capas convolucionales y 3 capas de agrupación máxima. En las capas convolucionales se utilizan filtros de tamaño 3×3 y se utilizan 16, 32 y 64 filtros en la primera, segunda y tercera capa convolucional, respectivamente; para el desplazamiento del filtro en la imagen y los mapas de características de entrada se utiliza un paso igual a 1 y el relleno utilizado fue el

conocido como relleno cero, permitiendo con esto que una capa tenga el mismo alto y ancho que la capa anterior. En el caso de las capas de agrupación, se utilizan filtros de tamaño 2×2 y el paso fue igual a 2 para el desplazamiento del filtro. La arquitectura de red neuronal convolucional se puede ver en la Figura 5.1.

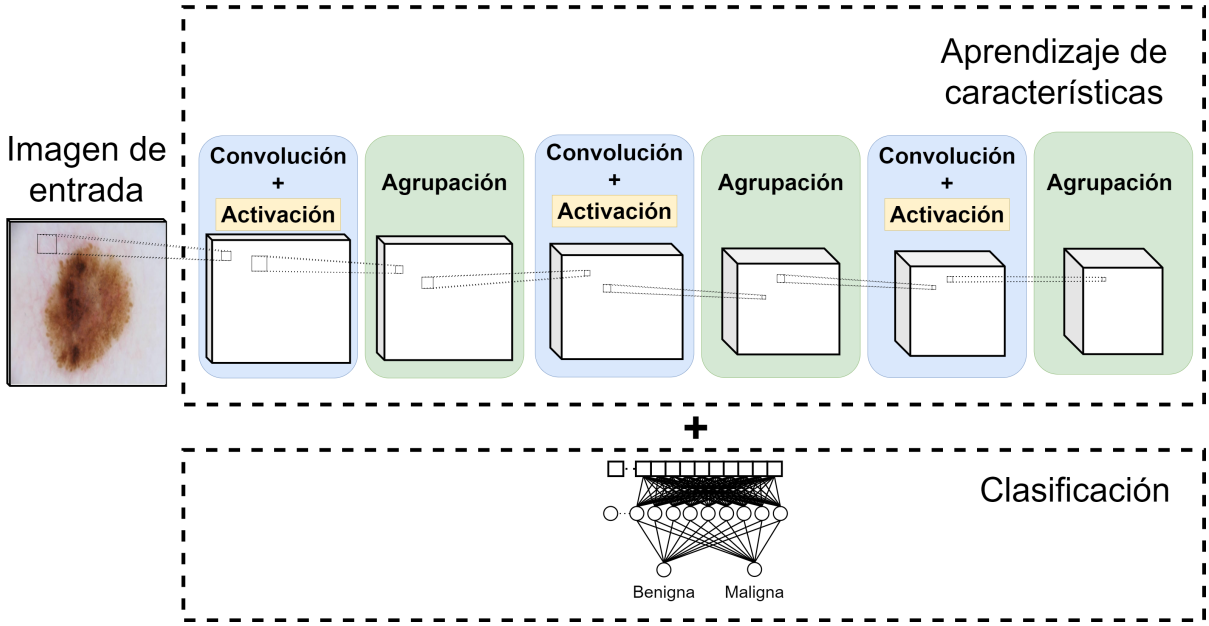


Figura 5.1: Estructura de capas de la Arquitectura 1.

La secuencia de pasos seguida para elaborar esta fase puede ser vista en la Figura 5.4. A continuación se realiza una descripción:

- Paso 1. Del archivo ISIC 2019 se toman dos muestras de 1200 y 9000 imágenes. En cada muestra 50 % de imágenes son lesiones de piel de clase Benigna y 50 % imágenes de lesiones de piel de clase Maligna. Además a cada muestra se realiza lo siguiente:
  - Paso 1.1. Se cambia el tamaño de las imágenes, quedando con dimensión de 180 de ancho y 180 de alto.
  - Se normalizan los valores de intensidad de los píxeles de las imágenes, quedando en un rango de $(0,1)$.

- Paso 2. Para realizar una mejor estimación del desempeño de los algoritmos con cada métrica calculada, se emplea el método de validación cruzada de K iteraciones. De esta manera se define $K = 10$ y en cada iteración se realiza lo siguiente:
    - Paso 2.1. Mediante un proceso aleatorio, cada muestra es dividida en 70 % de imágenes de entrenamiento, 10 % de imágenes de validación y 20 % de imágenes de prueba.
    - Paso 2.2. Se entrenan los algoritmos clásicos y la arquitectura de red neuronal convolucional, utilizando los conjuntos de imágenes de entrenamiento y validación.
    - Paso 2.3. Se ajustan los parámetros mediante un proceso iterativo definido a 10 épocas.
    - Paso 2.4. Se evalúa el desempeño parcial de los algoritmos utilizando el conjunto de imágenes de prueba.
- Paso 3. Se calcula el desempeño global de los algoritmos clásicos y la arquitectura de red neuronal convolucional.

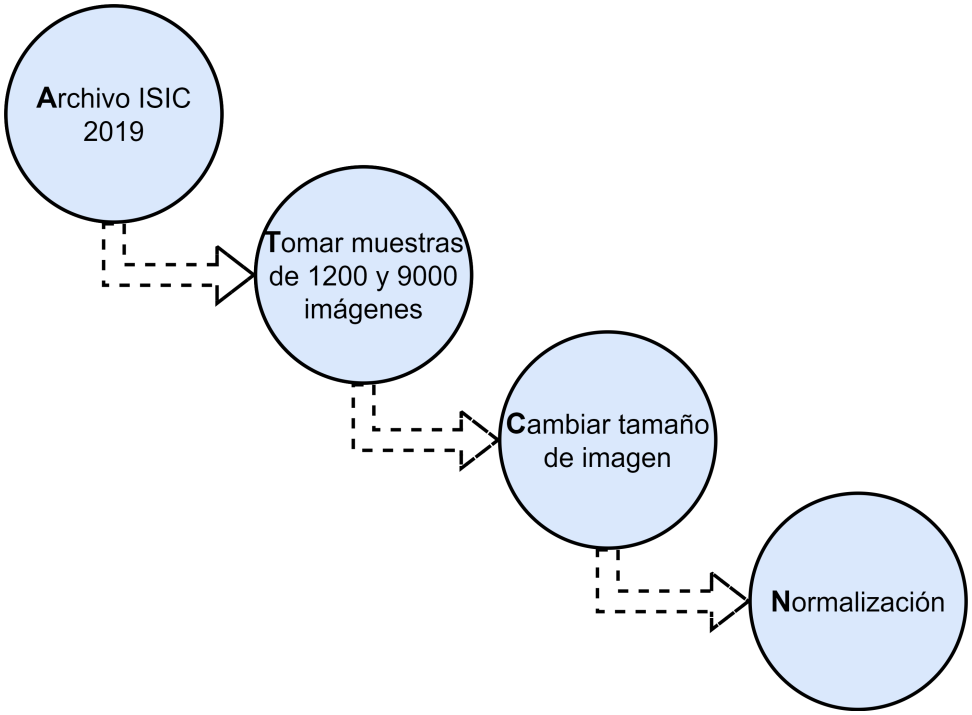


Figura 5.2: Secuencia de actividades realizadas en las imágenes dermoscópicas antes del entrenamiento del algoritmo.

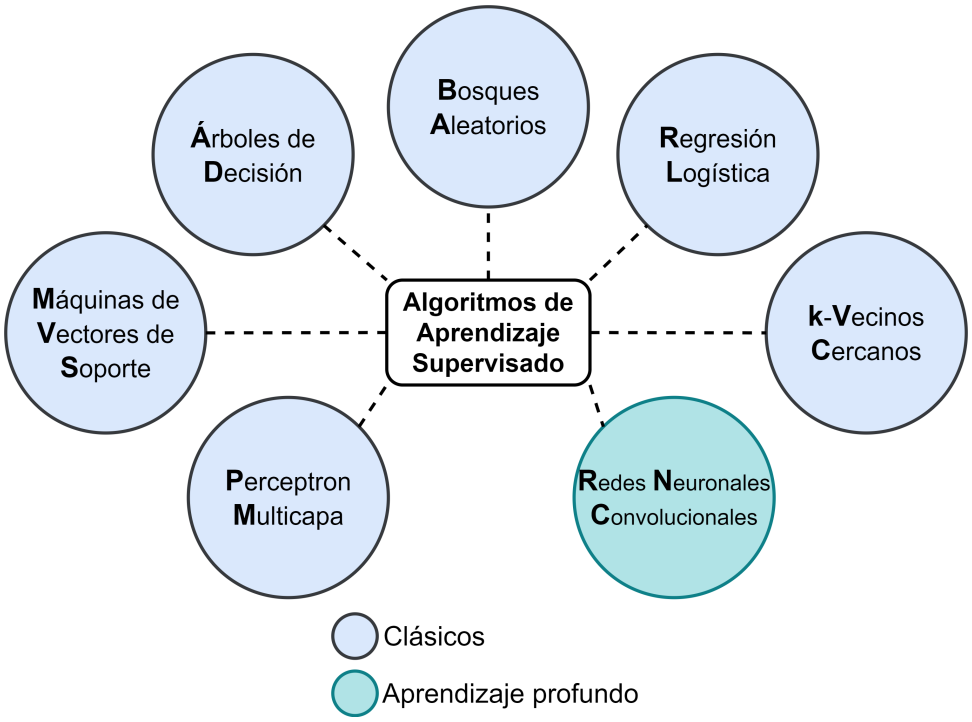


Figura 5.3: Representación de los algoritmos clásicos y de aprendizaje profundo empleados para clasificar las imágenes dermoscópicas.

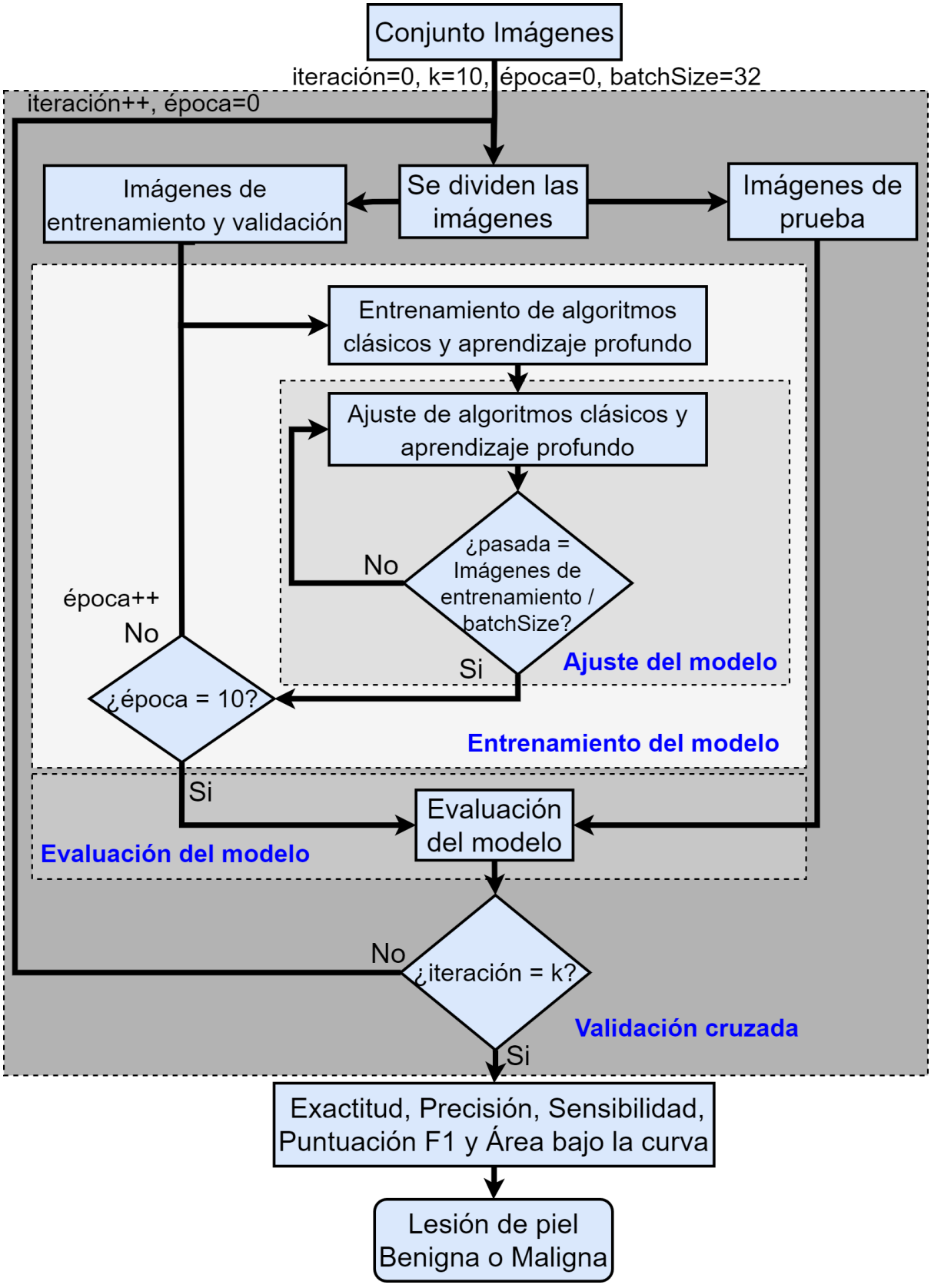


Figura 5.4: Diagrama de flujo que muestra el ajuste, entrenamiento y evaluación de los algoritmos clásicos y de aprendizaje profundo.

Para entender de manera más clara el flujo seguido en la arquitectura de red neuronal convolucional al trabajar con imágenes que tienen varios canales (Rojo, Verde y Azul) se puede ver la Figura 5.5. En este caso, los filtros son adaptados por separado para cada canal. Por ejemplo, al tomar como referencia que se tienen 8 filtros de convolución, estos filtros serán aplicados a cada canal de la imagen, por lo cual, se tendrá primero un total de 24 imágenes convolucionadas, compuestas por ocho imágenes mapeadas para cada uno de los 3 canales. Luego, todas las imágenes convolucionadas originalmente de la misma imagen se combinan en un mapa de características. Como resultado, se tendrán ocho mapas de características. Es decir, las imágenes se descomponen en diferentes datos canalizados, se aplican los filtros y luego se combinan nuevamente en imágenes de canales mixtos.

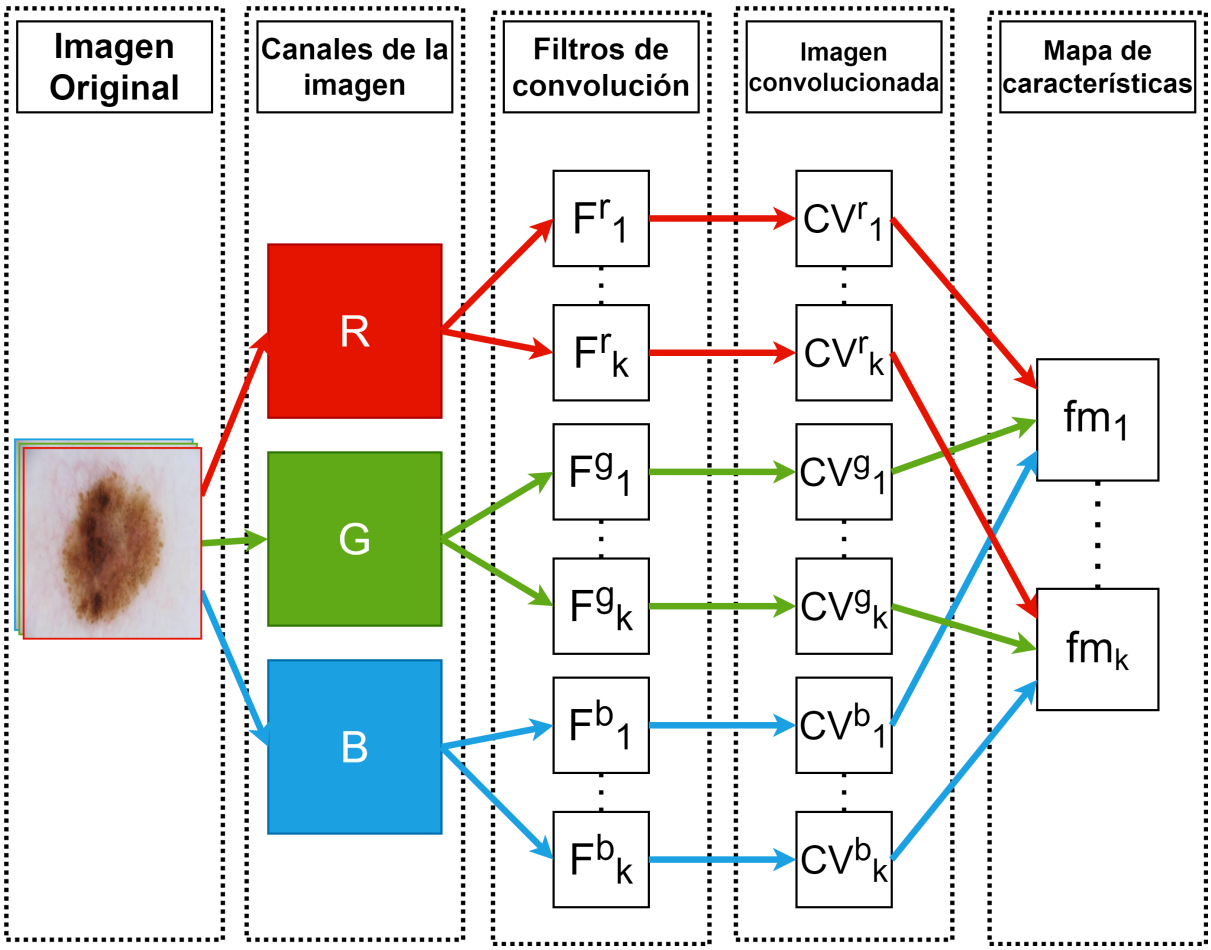


Figura 5.5: Proceso aplicado a imágenes RGB en la capa de convolución.

Los resultados obtenidos en esta Fase se muestran en la Tabla 6.3, en donde, nuevamente basándose en la métrica de exactitud, se puede ver que los algoritmos que mejor desempeño tuvieron fueron RNC, MVS y RL obteniendo una probabilidad de 91.75 %, 79.66 % y 77.66 %, respectivamente. Para mayor claridad, se puede ver la Figura 6.2, en la cual, se muestra que la arquitectura de red neuronal convolucional obtuvo mejor desempeño en comparación con los algoritmos clásicos.

Fase 3: En la Fase 2, se visualiza que la probabilidad de diagnosticar una lesión como Benigna, mejora al emplear la arquitectura de red neuronal convolucional. Es por ello, que para esta Fase 3, se decide experimentar con tres diferentes arquitecturas de RNC variando en cada una ellas la secuencia de capas de convolución y agrupación, así como, la cantidad de filtros y el tamaño de los mismos. Cada arquitectura se puede ver en las Figuras 5.1, 5.7 y 5.8. La primer arquitectura es similar a la empleada en la Fase 2, es decir, consta de 3 capas de convolución y 3 capas de agrupación; la segunda arquitectura consta de 2 capas de convolución y 1 capa de agrupación; por último, la tercer arquitectura consta de 4 capas de convolución y 1 capa de agrupación. Para las 3 arquitecturas, en las capas de convolución los parámetros de paso y relleno se establecen como 1 y cero. En cambio, el número de filtros varía entre 8, 16 y 32; y el tamaño del filtro variá entre 3×3, 4×4 y 5×5. Un ejemplo de esto puede ser al tomar como referencia la arquitectura 1, la cual, consta de 3 capas de convolución y en cada capa se emplean 8 filtros de tamaño 3×3. Por otro lado, en la capa de agrupación se utilizan filtros de tamaño 2×2 con paso o stride de 2. En la Figura 5.6 se muestra con mayor claridad el flujo de experimentación contemplado en esta Fase, los cuales son un total de 27 experimentos. Cabe mencionar que el conjunto de imágenes utilizadas fueron las mismas 1200 de la Fase 2 y además se volvió a incrementar la cantidad de imágenes a 9000, de las cuales, 4500 son lesiones de piel de clase Benigna y 4500 lesiones de piel de clase Maligna.

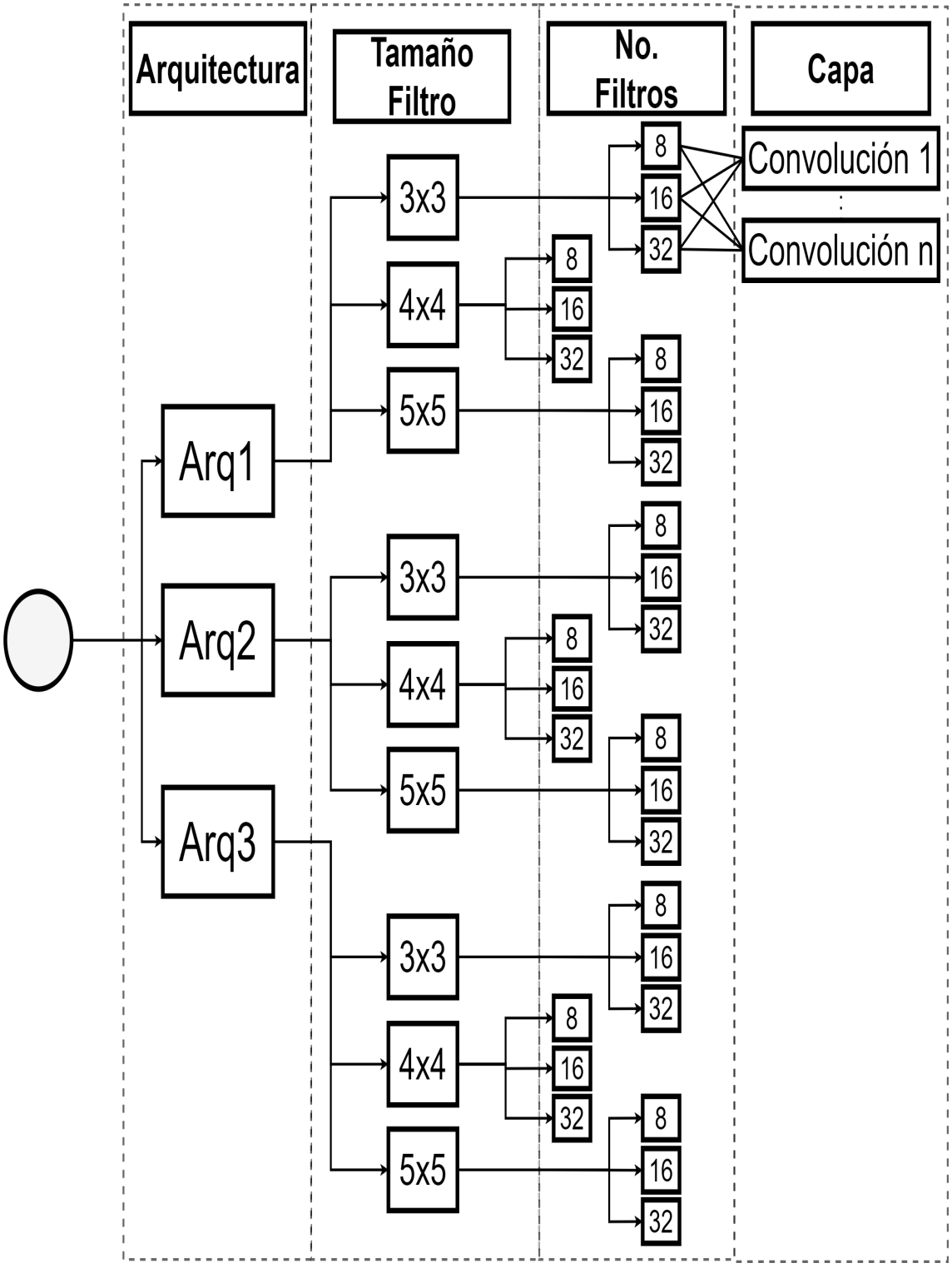


Figura 5.6: Diagrama de árbol que representa el flujo de experimentos realizados con las arquitecturas de red neuronal convolucional.

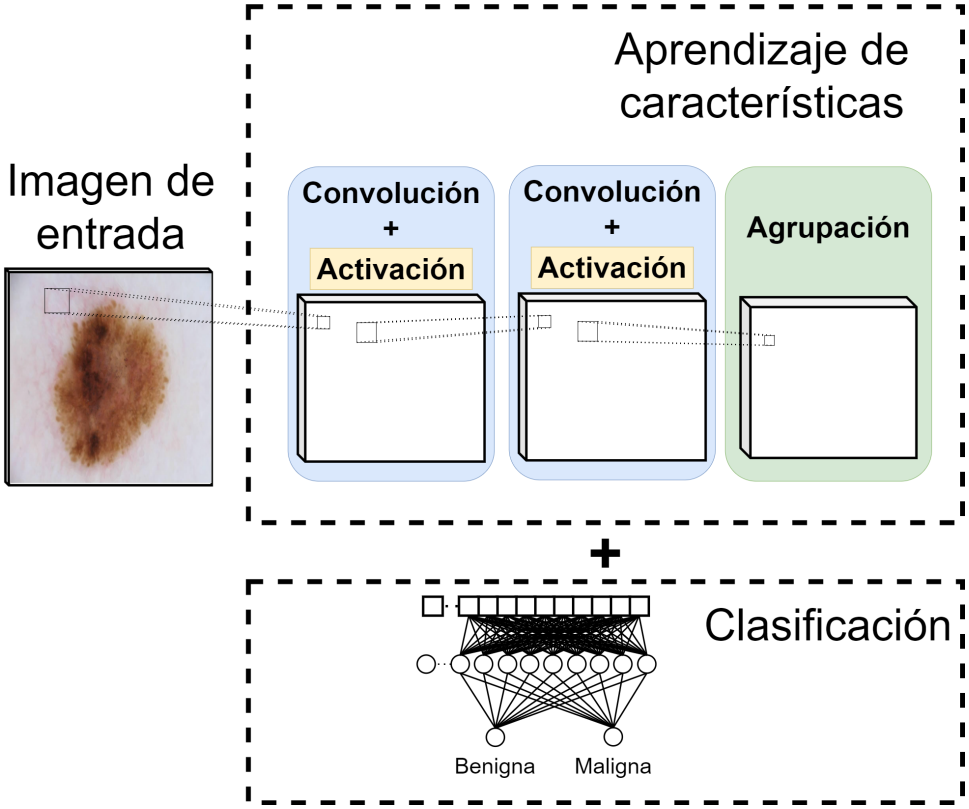


Figura 5.7: Estructura de capas de la Arquitectura 2.

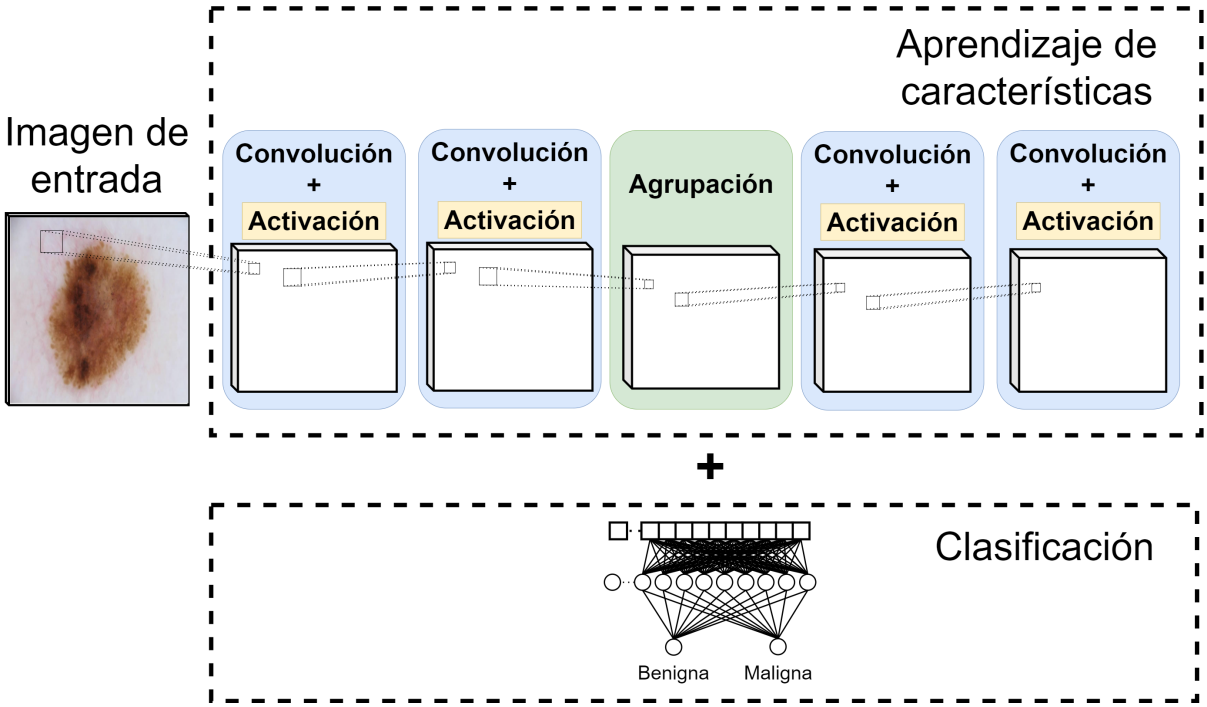


Figura 5.8: Estructura de capas de la Arquitectura 3.

Analizando los resultados en la Tabla 6.4 obtenidos para el caso del conjunto de 1200 imágenes empleando la arquitectura 1, se visualiza que la mejor probabilidad de diagnóstico de una lesión Benigna es de 94.15 % obtenida utilizando 16 filtros de tamaño 3×3. Para la arquitectura 2 (ver Tabla 6.5) la mejor probabilidad es de 95.55 % y nuevamente es obtenida utilizando 16 filtros de tamaño 3×3. Por su parte, en la arquitectura 3 (ver Tabla 6.6) la mejor probabilidad es también de 95.55 %, pero esta vez se obtuvo utilizando 8 filtros de tamaño 4×4.

Por otro lado, analizando el caso del conjunto de 9000 imágenes, al emplear la arquitectura 1 (ver Tabla 6.7) la mejor probabilidad es de 95.44 % obtenida utilizando 16 filtros de tamaño 3×3. Para la arquitectura 2 (ver Tabla 6.8) la mejor probabilidad es de 95.96 % obtenida utilizando 16 filtros de tamaño 3×3. En la arquitectura 3 (ver Tabla 6.9) la mejor probabilidad es de 96.37 %, pero esta vez se obtuvo utilizando 8 filtros de tamaño 3×3.

Cabe mencionar que, para los experimentos realizados se emplea una máquina con sistema operativo Windows 7 Professional de la marca HP con las siguientes características: Modelo HP Z230 SFF Workstation, Procesador Intel (R) Core (TM) i5-4590 CPU @ 3.30 GHz 3.30 GHz, RAM: 16.0 GB y Tipo de sistema de 64 bits.

A continuación, se presenta la serie de pasos a nivel código para desarrollar cada una de las arquitecturas de red neuronal convolucional empleadas. Se inicia por definir una estructura de carpetas similar a la Figura 5.9, en donde, la muestra de imágenes de lesiones de piel es alojada en una carpeta temporal, la cual, contiene dos subcarpetas con el nombre de cada clase de lesión de piel, es decir, benigna y maligna. También se crean tres subcarpetas con el nombre de cada subconjunto utilizado para entrenar y evaluar los modelos de red convolucional, es decir, train, valid y test. Al igual que la carpeta temporal, cada subcarpeta contiene dos carpetas con el nombre de cada clase de lesión de piel. Cabe mencionar que, las imágenes almacenadas en la carpeta temporal son movidas durante el proceso de validación cruzada y repartidas entre cada subconjunto considerando las ponderaciones de 70 % para

entrenamiento, 10 % para validación y 20 % para prueba. Como se mostrará más adelante, estas ponderaciones se definen como variables.

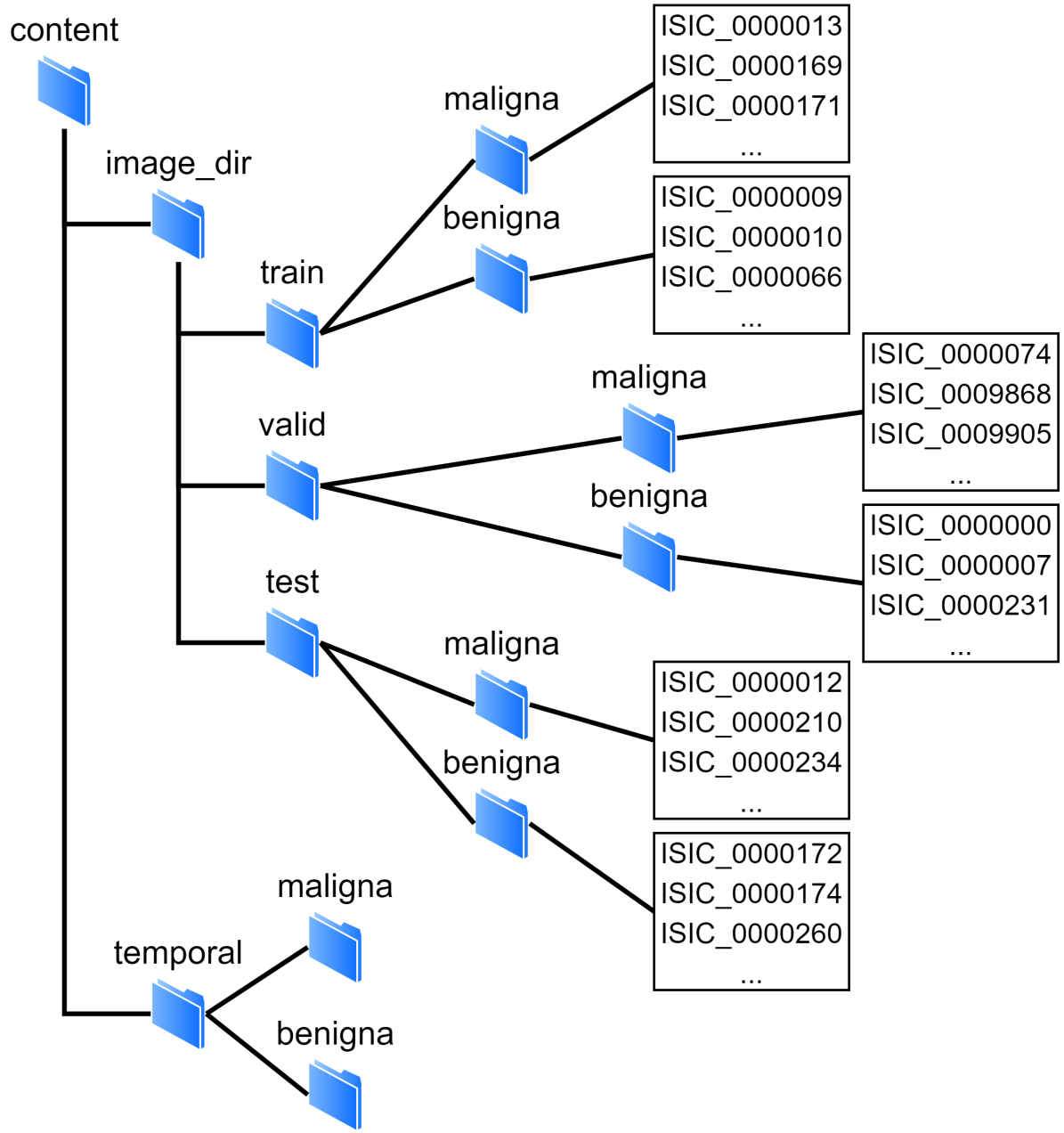


Figura 5.9: Estructura de carpetas para las imágenes de entrenamiento, validación y prueba.

En las Figuras 5.10a y 5.10b se muestran las librerías que son utilizadas, y además, se muestra la definición de cada variable que almacena la ruta a donde se encuentran alojadas las imágenes dermoscópicas. También se definen otras variables que son descritas

a continuación: CLASS_MODE (Es el tipo de clasificación en este caso binaria), HEIGTH, WIDTH y CHANNELS (Son el alto, ancho y cantidad de canales de las imágenes), P_TRAIN, P_TEST, P_VALID (Son las ponderaciones para los subconjuntos de entrenamiento, validación y pruebas), N_FILTROS (Es el número de filtros a utilizar en las capas de convolución), TAMANIO (Es el tamaño de los filtros en las capas de convolución), ARQUITECTURA (Define la arquitectura a emplear 1- Arquitectura 1, 2 - Arquitectura 2 o 3- Arquitectura 3), DATASET (Es un arreglo de cada subconjunto) y CLASES (Es un arreglo con el nombre de las clases de lesiones de piel).

En las Figuras 5.10c y 5.10d se muestran las funciones moverImagenes y cargaImagenes, las cuales, se utilizan para el proceso de validación cruzada y mueven las imágenes entre las carpetas train, valid, test y temporal. La función moverImagenes mueve las imágenes que se encuentren en las carpetas train, valid y test para alojarlas en la carpeta temporal. La función cargarImagenes mueve de forma aleatoria las imágenes de la carpeta temporal a las carpetas train, valid y test, esto, cumpliendo con las ponderaciones asignadas para los subconjuntos de entrenamiento, validación y prueba asignados a las variables P_TRAIN, P_VALID y P_TEST.

En las Figuras 5.10e, 5.10f y 5.10g se muestra la configuración de cada arquitectura de red convolucional. Como se puede observar, se utiliza la clase Conv2D del API Keras para configurar cada capa de convolución mediante los argumentos filters (cantidad de filtros asignados en la variable N_FILTROS), kernel_size (tamaño del filtro asignado en la variable TAMANIO), padding (relleno=same, indicando que el tamaño de salida en la capa de convolución sea el mismo que la entrada) y activation (función de activación=relu indicando que los valores en la salida de la capa de convolución estén en el rango $[0,\infty]$). Por otro lado, para la capa de agrupación se utiliza la clase MaxPooling2D. Para el caso de la etapa de clasificación mediante el perceptrón multicapa, se utilizan las clases Flatten y Dense.

En las Figuras 5.10h y 5.10i se crean tres funciones para calcular la métricas de precisión, sensibilidad y puntuación F1, la cuales, se guardan en un arreglo de métricas en donde ade-

más, también se indica las métricas de exactitud y área bajo la curva, calculadas utilizando las clases Accuracy y AUC del API de Keras. Posteriormente, se compila la red convolucional pasando como argumentos el optimizador, la función de pérdida y el arreglo de métricas mencionado anteriormente. Como se puede observar, el optimizador utilizado es el nombrado en Keras como Adam, el cual, utiliza el método de gradiente descendente. En el caso de la función de pérdida se utiliza la clase BinaryCrossentropy ya que el tipo de clasificación es binaria.

En las Figuras 5.10j, 5.10k y 5.10l se muestra el proceso de entrenamiento y evaluación de la red convolucional, esto, utilizando los métodos fit y evaluate, respectivamente. Destacan las variables K_FOLD indicando el número de iteraciones utilizadas para la validación cruzada, EPOCH indicando el número de veces que el conjunto de entrenamiento realizará la pasada hacia delante y hacia atrás de la red convolucional, y por último, BATCH_SIZE indicando el número de imágenes a procesar antes de actualizar los parámetros de la red convolucional. Finalmente, se imprimen los resultados parciales (Figura 5.10m) para las métricas de desempeño ocupadas y posteriormente, se imprime el desempeño promedio (Figura 5.10n) de la arquitectura de red neuronal convolucional.

```python
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import os
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
from keras_preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Conv2D, MaxPool2D, Dense, Flatten
from keras.preprocessing import image
from pathlib import Path
import shutil
import random
import time
```

(a) Bibliotecas utilizadas.

```python
SLASH = "\\"
BASE = "D:\\"
SUBCARPETA1 = "content"
SUBCARPETA2 = "image_dir"
DATASET = ["train", "test", "valid"]
PATH_TRAIN   = BASE+SUBCARPETA1+SLASH+SUBCARPETA2+SLASH+DATASET[0]
PATH_TEST    = BASE+SUBCARPETA1+SLASH+SUBCARPETA2+SLASH+DATASET[1]
PATH_VALID   = BASE+SUBCARPETA1+SLASH+SUBCARPETA2+SLASH+DATASET[2]
PATH_ORIGEN  = BASE+SUBCARPETA1+SLASH+SUBCARPETA2
PATH_DESTINO = BASE+SUBCARPETA1+SLASH+"temporal"
CLASS_MODE = "binary"
HEIGTH = 180
WIDTH = 180
CHANNELS = 3
P_TRAIN = 0.8
P_TEST  = 0.1
P_VALID = 0.1
N_FILTROS = 8
TAMANIO_FILTRO = 3
ARQUITECTURA = 1
CLASES = ["benigna", "maligna"]
```

(b) Definición de variables.

Figura 5.10: Código desarrollado para la ejecución del algoritmo de aprendizaje profundo.

```python
# FUNCIÓN 1 UTLIZADA PARA VALIDACIÓN CRUZADA: MUEVE
# LAS IMÁGENES DE LAS CARPETAS TRAIN, VALID Y TEST
# A LAS CARPETA TEMPORAL
def moverImagenes():
    for i in range(0,3):
        folderB = PATH_ORIGEN+SLASH+DATASET[i]+SLASH+CLASES[0]
        folderM = PATH_ORIGEN+SLASH+DATASET[i]+SLASH+CLASES[1]
        onlyfilesB = [b for b in os.listdir(folderB)
                      if os.path.isfile(os.path.join(folderB, b))]
        onlyfilesM = [m for m in os.listdir(folderM)
                      if os.path.isfile(os.path.join(folderM, m))]
        for _fileB in onlyfilesB:
            shutil.move(folderB + SLASH + _fileB,
                        PATH_DESTINO + SLASH + CLASES[0] + SLASH + _fileB)
        for _fileM in onlyfilesM:
            shutil.move(folderM + SLASH + _fileM,
                        PATH_DESTINO + SLASH + CLASES[1] + SLASH + _fileM)
```

(c) Función para mover imágenes a la carpeta temporal.

```python
# FUNCIÓN 2 UTLIZADA PARA VALIDACIÓN CRUZADA: CARGA DE
# MANERA ALEATORIA LAS IMÁGENES DE LA CARPETA TEMPORAL
# A LAS CARPETAS TRAIN, VALID Y TEST.
def cargarImagenes(TOTAL, nBenignas, nMalignas, SET):
    for i in range(0, TOTAL):
        CLASE = random.randint(0,1)
        if CLASE == 0 and nBenignas > 0:
            rB = random.randint(0, np.size(imgsBenigna)-1)
            nameB = imgsBenigna.pop(rB)
            shutil.move(pathBenigna+SLASH+nameB,
                        PATH_ORIGEN+SLASH+DATASET[SET]+SLASH+CLASES[0]+SLASH+nameB)
            nBenignas = nBenignas-1
        elif CLASE == 1 and nMalignas > 0:
            rM = random.randint(0, np.size(imgsMaligna)-1)
            nameM = imgsMaligna.pop(rM)
            shutil.move(pathMaligna+SLASH+nameM,
                        PATH_ORIGEN+SLASH+DATASET[SET]+SLASH+CLASES[1]+SLASH+nameM)
            nMalignas = nMalignas-1
        elif CLASE == 1 and nMalignas == 0 and nBenignas > 0:
            rB = random.randint(0, np.size(imgsBenigna)-1)
            nameB = imgsBenigna.pop(rB)
            shutil.move(pathBenigna+SLASH+nameB,
                        PATH_ORIGEN+SLASH+DATASET[SET]+SLASH+CLASES[0]+SLASH+nameB)
            nBenignas = nBenignas-1
        elif CLASE == 0 and nBenignas == 0 and nMalignas > 0:
            rM = random.randint(0, np.size(imgsMaligna)-1)
            nameM = imgsMaligna.pop(rM)
            shutil.move(pathMaligna+SLASH+nameM,
                        PATH_ORIGEN+SLASH+DATASET[SET]+SLASH+CLASES[1]+SLASH+nameM)
            nMalignas = nMalignas-1
    return nBenignas, nMalignas
```

(d) Función para mover imágenes a las carpetas train, valid y test.

Figura 5.10: Código desarrollado para la ejecución del algoritmo de aprendizaje profundo (cont.).

```
# SE CONFIGURA LA ARQUITECTURA 1
if ARQUITECTURA == 1:
    model = Sequential([
      layers.Conv2D(N_FILTROS, TAMANIO_FILTRO, padding='same',
                    activation='relu', input_shape=(HEIGTH,
                                                    WIDTH,
                                                    CHANNELS)),
      layers.MaxPooling2D(),
      layers.Conv2D(N_FILTROS, TAMANIO_FILTRO,
                    padding='same', activation='relu'),
      layers.MaxPooling2D(),
      layers.Conv2D(N_FILTROS, TAMANIO_FILTRO,
                    padding='same', activation='relu'),
      layers.MaxPooling2D(),
      layers.Flatten(),
      layers.Dense(128, activation='relu'),
      layers.Dense(1,activation='sigmoid')
    ])
```

(e) Configuración arquitectura 1.

```
# SE CONFIGURA LA ARQUITECTURA 2
if ARQUITECTURA == 2:
    model = Sequential([
      layers.Conv2D(N_FILTROS, TAMANIO_FILTRO, padding='same',
                    activation='relu', input_shape=(HEIGTH,
                                                    WIDTH,
                                                    CHANNELS)),
      layers.Conv2D(N_FILTROS, TAMANIO_FILTRO,
                    padding='same', activation='relu'),
      layers.MaxPooling2D(),
      layers.Flatten(),
      layers.Dense(128, activation='relu'),
      layers.Dense(1,activation='sigmoid')
    ])
```

(f) Configuración arquitectura 2.

Figura 5.10: Código desarrollado para la ejecución del algoritmo de aprendizaje profundo (cont.).

```
# SE CONFIGURA LA ARQUITECTURA 3
if ARQUITECTURA == 3:
    model = Sequential([
      layers.Conv2D(N_FILTROS, TAMANIO_FILTRO, padding='same',
                    activation='relu', input_shape=(HEIGTH,
                                                    WIDTH,
                                                    CHANNELS)),
      layers.Conv2D(N_FILTROS, TAMANIO_FILTRO,
                    padding='same', activation='relu'),
      layers.MaxPooling2D(),
      layers.Conv2D(N_FILTROS, TAMANIO_FILTRO,
                    padding='same', activation='relu'),
      layers.Conv2D(N_FILTROS, TAMANIO_FILTRO,
                    padding='same', activation='relu'),
      layers.Flatten(),
      layers.Dense(128, activation='relu'),
      layers.Dense(1,activation='sigmoid')
    ])
```

(g) Configuración arquitectura 3.

```
from keras import backend as K

# FUNCIONES PARA CALCULAR LAS MÉTRICAS
# SENSIBILIDAD, PRECISION Y PUNTUACIÓN F1
def recall_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
    recall = true_positives / (possible_positives)
    return recall

def precision_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
    precision = true_positives / (predicted_positives)
    return precision

def f1_m(y_true, y_pred):
    precision = precision_m(y_true, y_pred)
    recall = recall_m(y_true, y_pred)
    return 2*((precision*recall)/(precision+recall))
```

(h) Funciones para calcular métricas de sensibilidad, precisión y puntuación F1.

Figura 5.10: Código desarrollado para la ejecución del algoritmo de aprendizaje profundo (cont.).

```
# SE DEFINEN LAS MÉTRICAS A CALCULAR
METRICAS = [
    'accuracy',
    precision_m,
    recall_m,
    f1_m,
    keras.metrics.AUC(name='auc', from_logits=True)
]

# SE COMPILA EL MODELO ESPECIFICANDO EL OPTIMIZADOR,
# FUNCIÓN DE PÉRDIDA Y LAS MÉTRICAS
model.compile(optimizer='adam',
              loss=tf.keras.losses
              .BinaryCrossentropy(from_logits = False),
              metrics=METRICAS)

# SE NORMALIZAN LOS VALORES DE LOS PIXELES
# EN EL RANGO [0,1]
datagen = ImageDataGenerator(rescale=1./255)
```

(i) Definición de métricas, compilación del modelo y normalización de imágenes.

```
INICIO = time.time()
K_FOLD = 10
EPOCHS = 20
BATCH_SIZE = 32
loss = []
accuracy = []
precision = []
recall = []
f1 = []
auc = []

for f in range(0, K_FOLD):
    print(str(f+1) + "-Fold * * * * * * * * * * * * * * * * * * * * ")

    # PROCESO PARA ALEATORIZAR IMÁGENES (TRAIN, VALID AND TEST)
    moverImagenes()
    pathBenigna = PATH_DESTINO+SLASH+CLASES[0]
    pathMaligna = PATH_DESTINO+SLASH+CLASES[1]
```

(j) Proceso de entrenamiento y evaluación Part.1.

Figura 5.10: Código desarrollado para la ejecución del algoritmo de aprendizaje profundo (cont.).

```
    imgsBenigna = [b for b in os.listdir(pathBenigna)
                   if os.path.isfile(os.path.join(pathBenigna, b))]
    imgsMaligna = [b for b in os.listdir(pathMaligna)
                   if os.path.isfile(os.path.join(pathMaligna, b))]
    nBenignas = np.size(imgsBenigna)
    nMalignas = np.size(imgsMaligna)
    N_TOTAL = nBenignas + nMalignas
    N_TRAIN = int(N_TOTAL * P_TRAIN)
    N_TEST  = int(N_TOTAL * P_TEST)
    N_VALID = int(N_TOTAL * P_VALID)
    nBenignas, nMalignas = cargarImagenes(N_VALID, nBenignas,
                                          nMalignas, 2)# Valid
    nBenignas, nMalignas = cargarImagenes(N_TEST, nBenignas,
                                          nMalignas, 1)# Test
    nBenignas, nMalignas = cargarImagenes(N_TRAIN, nBenignas,
                                          nMalignas, 0)# Train

    # PROCESO PARA CARGA DE SETS
    train_generator = datagen.flow_from_directory(
        PATH_TRAIN,
        target_size = (HEIGTH, WIDTH),
        batch_size = BATCH_SIZE,
        class_mode = CLASS_MODE)

    valid_generator = datagen.flow_from_directory(
```

(k) Proceso de entrenamiento y evaluación Part.2.

```
        PATH_VALID,
        target_size = (HEIGTH, WIDTH),
        batch_size = BATCH_SIZE,
        class_mode = CLASS_MODE)

     test_generator = datagen.flow_from_directory(
        PATH_TEST,
        target_size = (HEIGTH, WIDTH),
        batch_size = BATCH_SIZE,
        class_mode = CLASS_MODE)

     # PROCESO DE ENTRENAMIENTO (TRAIN AND VALID SETS)
     history = model.fit(train_generator,
                         steps_per_epoch=N_TRAIN//BATCH_SIZE,
                         epochs=EPOCHS,
```

(l) Proceso de entrenamiento y evaluación Part.3.

Figura 5.10: Código desarrollado para la ejecución del algoritmo de aprendizaje profundo (cont.).

```
                    validation_data=valid_generator,
                    validation_steps=N_VALID//BATCH_SIZE,
                    callbacks=[tensorboard_callback])

    # PROCESO DE EVALUACIÓN (TEST SET)
    model_CNN_score = model.evaluate(test_generator)

    # DESEMPEÑO PARCIAL EN CADA FOLD
    print("Rendimiento Fold-" + str(f+1))
    print("CNN Pérdida:",           model_CNN_score[0])
    print("CNN Exactitud:",       model_CNN_score[1])
    print("CNN Precisión:",      model_CNN_score[2])
    print("CNN Sensibilidad:",      model_CNN_score[3])
    print("CNN F1:",      model_CNN_score[4])
    print("CNN AUC:", model_CNN_score[5])

    loss.append(model_CNN_score[0]*100)
    accuracy.append(model_CNN_score[1]*100)
    precision.append(model_CNN_score[2]*100)
    recall.append(model_CNN_score[3]*100)
    f1.append(model_CNN_score[4]*100)
    auc.append(model_CNN_score[5]*100)
    print("- - - - - - - - - - - - - - - - - - - - - - - - - -")
FIN = time.time()
```

(m) Proceso de entrenamiento y evaluación Part.4.

```
# DESEMPEÑO GENERAL DEL MODELO
print("Duración: " , FIN-INICIO, " segundos")
print("Rendimiento Total:")
print("CNN Exactitud:",    sum(accuracy)/K_FOLD)
print("CNN Precisión:",    sum(precision)/K_FOLD)
print("CNN Sensibilidad:", sum(recall)/K_FOLD)
print("CNN F1:",           sum(f1)/K_FOLD)
print("CNN AUC:",          sum(auc)/K_FOLD)
```

(n) Impresión del desempeño del modelo.

Figura 5.10: Código desarrollado para la ejecución del algoritmo de aprendizaje profundo (cont.).

Capítulo 6

Resultados

En este capítulo se muestran los resultados obtenidos en las tres fases mencionadas en el Capítulo 5.

Respecto a los resultados de la Fase 1, el desempeño de los algoritmos clásicos utilizando imágenes ISSA se muestra en la Tabla 6.1, en este caso, se puede ver que la exactitud obtenida por el algoritmo de bosques aleatorios fue la mejor con 78.99 % de diagnosticar una lesión como Maligna.

Métrica / Algoritmo	Exa (%)	Pre (%)	Sen (%)	Auc (%)	F1 (%)
Árboles de Decisión	63.99	63.99	63.99	47.03	47.03
Bosques Aleatorios	78.99	78.99	78.99	67.82	78.99
Regresión Logística	68.49	68.49	68.49	53.41	68.49
K Vecinos más Cercanos	78.00	78.00	78.00	63.88	78.00
Máquinas de Vectores de Soporte	75.50	75.50	75.50	45.73	75.50
Perceptrón Multicapa	72.00	72.00	72.00	50.36	72.00

Tabla 6.1: Desempeño de Exactitud, Precisión, Sensibilidad, Área bajo la curva y Puntuación F-1 de los algoritmos clásicos utilizando Imágenes Segmentadas y Sin Artefactos (ISSA).

Por otro lado, el desempeño de los algoritmos clásicos utilizando imágenes ISSCA se muestra en la Tabla 6.2, en donde, se puede ver que nuevamente la exactitud obtenida por el algoritmo de bosques aleatorios fue la mejor con 82.00 % de diagnosticar una lesión como Maligna.

Métrica / Algoritmo	Exa (%)	Pre (%)	Sen (%)	Auc (%)	F1 (%)
Árboles de Decisión	72.50	72.50	72.50	56.65	72.50
Bosques Aleatorios	82.00	82.00	82.00	64.78	82.00
Regresión Logística	72.50	72.50	72.50	48.62	72.50
K Vecinos más Cercanos	81.50	81.50	81.50	59.96	81.50
Máquinas de Vectores de Soporte	74.50	74.50	74.50	50.41	74.50
Perceptrón multicapa	65.50	65.50	65.50	48.98	65.50

Tabla 6.2: Desempeño de Exactitud, Precisión, Sensibilidad, Área bajo la curva y Puntuación F-1 de los algoritmos clásicos utilizando Imágenes Sin Segmentar y Con Artefactos (ISSCA).

En la Figura 6.1, se muestra la gráfica de la exactitud obtenida por cada algoritmo clásico empleado. Aquí, se puede comparar el desempeño de cada algoritmo clásico al utilizar imágenes ISSA e imágenes ISSCA. Se puede ver que en 4 de los 6 algoritmos clásicos (Árboles de Decisión, Bosques Aleatorios, Regresión Logística y K Vecinos más Cercanos) el valor de la exactitud incremento, indicando que para el caso de utilizar imágenes ISSA existe la posibilidad de pérdida de información cuando se segmentan y se eliminan los artefactos en las imágenes, impactando en el desempeño de los algoritmos clásicos. Es por este motivo, y como se menciono en el Capítulo 5, que se optó por utilizar imágenes ISSCA en la Fase 2 y Fase 3.

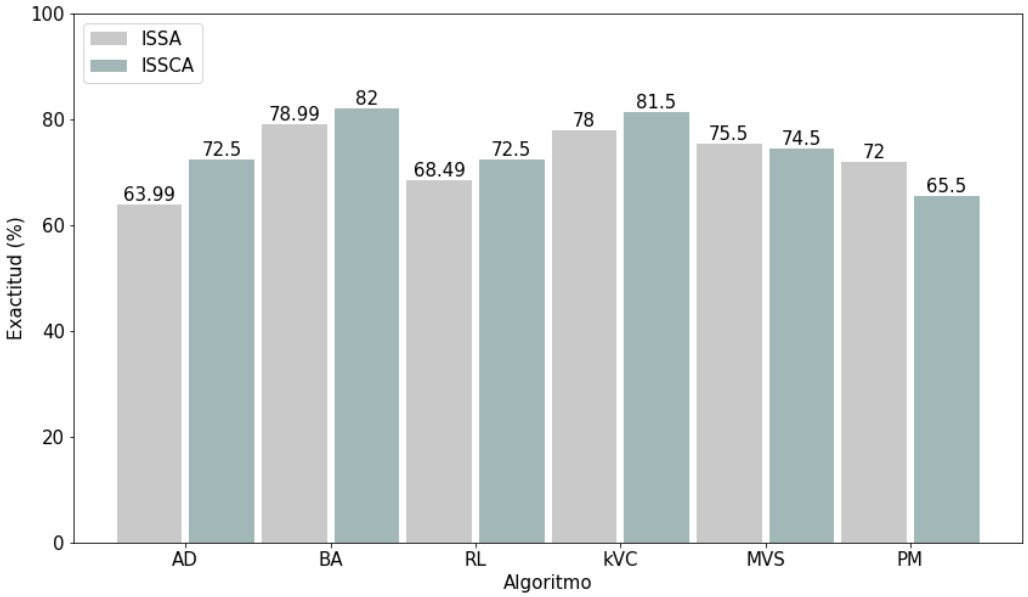


Figura 6.1: Desempeño de Exactitud de los algoritmos clásicos utilizando imágenes ISSA e ISSCA.

Por otra parte, respecto a los resultados de la Fase 2, en donde, se incremento la cantidad de imágenes a 1200. En la Tabla 6.3, se muestra el desempeño de los algoritmos clásicos y la red neuronal convolucional (RNC). Esta vez, se puede ver que la RNC obtuvo mejor desempeño con 91.75% de diagnosticar una lesión de piel como Maligna. Cabe destacar que las RNC´s se basan en el enfoque del aprendizaje profundo, por lo tanto, tienen implícito un proceso de extracción de características realizado mediante las capas de convolución y agrupación lo que ayuda de forma considerable a mejorar su desempeño.

Métrica / Algoritmo	Exa (%)	Pre (%)	Sen (%)	Auc (%)	F1 (%)
Árboles de decisión	70.70	70.70	70.70	70.53	70.70
Regresión logística	72.04	72.04	72.04	78.67	72.04
Bosques aleatorios	77.66	77.66	77.66	87.34	77.66
k-Vecinos más cercanos	71.33	71.33	71.33	80.15	71.33
Máquinas de vectores de soporte	79.66	79.66	79.66	88.32	79.66
Perceptrón Multicapa	66.87	66.87	66.87	69.40	66.87
Red neuronal convolucional	91.75	90.63	93.70	95.82	91.48

Tabla 6.3: Desempeño de Exactitud, Precisión, Sensibilidad, Área bajo la curva y Puntuación F-1 de los algoritmos clásicos y de aprendizaje profundo utilizando 1200 imágenes ISSCA.

En la Figura 6.2, se muestra la gráfica de exactitud obtenida por los algoritmos clásicos y la red neuronal convolucional. En este caso, se puede ver que el desempeño de la red neuronal convolucional tiene una amplia ventaja con el algoritmo clásico más cercano con exactitud de 79.66% obtenida por Máquinas de Vectores de Soporte.

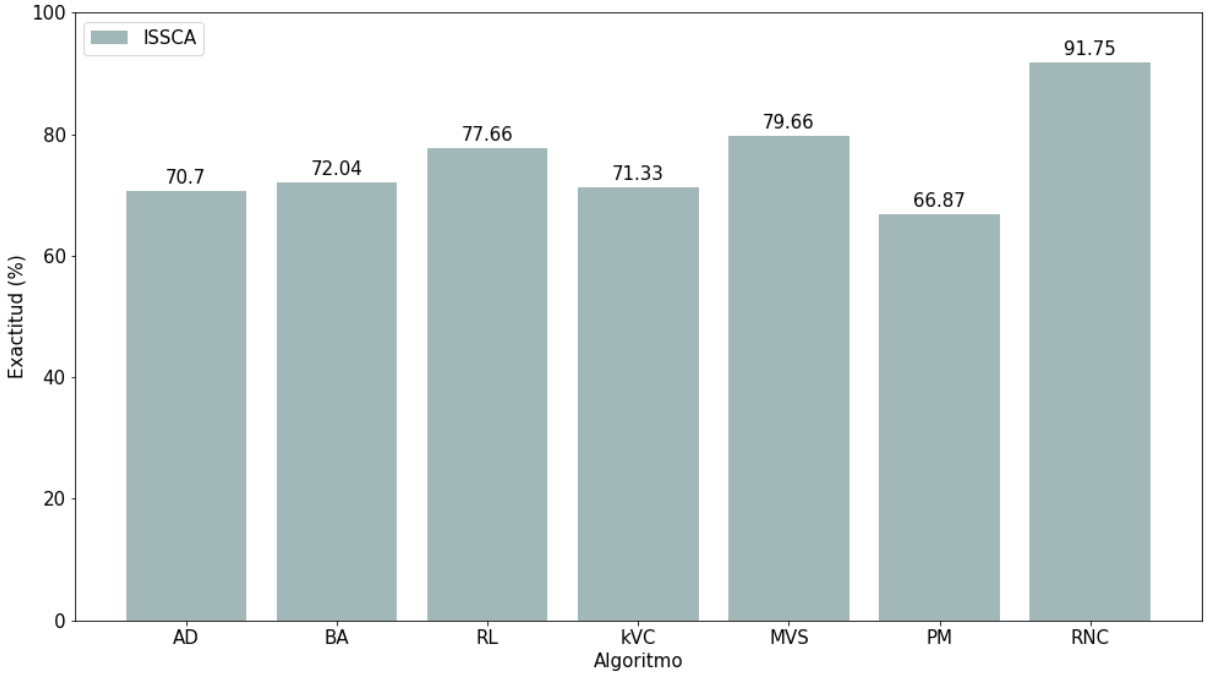


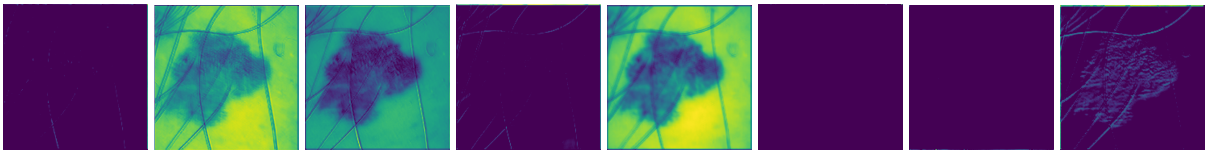
Figura 6.2: Desempeño de Exactitud de los algoritmos clásicos y la red neuronal convolucional.

Finalmente, respecto a la Fase 3, en donde, se trabajo con muestras de 1200 y 9000 imágenes, pero, esta vez empleando únicamente la red neuronal convolucional y variando el tamaño y la cantidad de los filtros de convolución, así como, el número de capas de convolución y agrupación. En las Figuras, 6.4, 6.5, 6.6 y 6.7 se muestran algunos ejemplos de mapas de características generados para la lesión de piel pigmentada que se muestra en la Figura 6.3 utilizando la Arquitectura 3 (ver Figura 5.8) con 8 filtros de tamaño 3×3, la cual, como se verá más adelante, obtuvo el mejor desempeño ocupando la muestra de 9000 imágenes.

Figura 6.3: Imagen original.

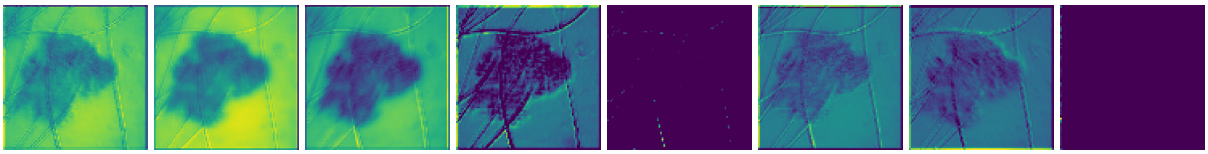
(a) (b) (c) (d) (e) (f) (g) (h)

Figura 6.4: Mapa de características Capa de Convolución 1 Arquitectura 3.



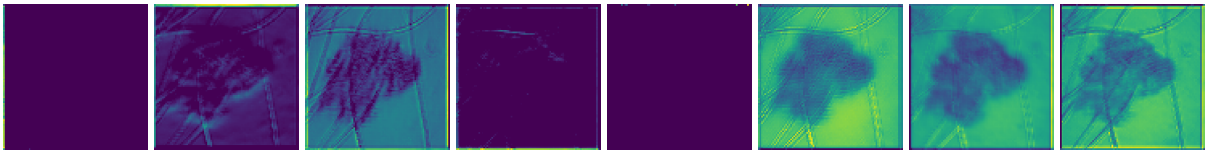
(a) (b) (c) (d) (e) (f) (g) (h)

Figura 6.5: Mapa de características Capa de Convolución 2 Arquitectura 3.



(a) (b) (c) (d) (e) (f) (g) (h)

Figura 6.6: Mapa de características Capa de Convolución 3 Arquitectura 3.



(a) (b) (c) (d) (e) (f) (g) (h)

Figura 6.7: Mapa de características Capa de Convolución 4 Arquitectura 3.

En las imágenes de los mapas de características de cada arquitectura se puede observar como las capas de convolución fueron identificando los bellos presentes en la lesión de piel, pero, en algunos casos también se puede observar que se identifica mayormente la región de la lesión. Se aprecia que en la salida de las capas de convolución 1 y 2 (Figuras 6.4 y 6.5) los bellos se identifican con mayor intensidad, pero, en la salida de las capas de convolución 3 y 4 (Figuras 6.6 y 6.7) se va desvaneciendo ligeramente la presencia de los bellos y se identifica mayormente la región de la lesión de piel.

Por otro lado, continuado con los resultados obtenidos en la Fase 3, en la Tabla 6.4, se muestra el desempeño obtenido utilizando 1200 imágenes y la Arquitectura 1 (ver Figura 5.1) con 8, 16 y 32 filtros de tamaño 3×3, 4×4 y 5×5. Se puede observar que empleando 16 filtros de tamaño 3×3 se obtiene la mejor exactitud siendo de 94.15 % de diagnosticar una lesión como Maligna.

ARQUITECTURA [1]							
Tamaño	**# Filtros**	**Exa (%)**	**Pre (%)**	**Sen (%)**	**Auc (%)**	**F1 (%)**	**Tiempo (m)**
5 x 5	8	93.14	93.97	92.10	96.56	94.10	30.80
5 x 5	16	86.75	85.70	89.10	91.88	87.20	38.26
5 x 5	32	91.85	91.62	92.40	95.35	92.25	64.10
4 x 4	8	90.19	92.89	86.99	95.45	90.80	27.17
4 x 4	16	90.69	90.87	91.20	92.80	91.15	30.34
4 x 4	32	92.00	91.35	93.10	95.81	92.40	51.51
3 x 3	8	91.70	91.50	92.80	96.38	92.10	26.99
3 x 3	16	94.15	93.72	94.70	96.44	94.50	27.62
3 x 3	32	92.39	93.43	91.20	96.13	92.81	36.94

Tabla 6.4: Desempeño de Exactitud, Precisión, Sensibilidad, Área bajo la curva y Puntuación F-1 obtenido por el algoritmo de aprendizaje profundo utilizando la Arquitectura 1 (tres capas de convolución, tres capas de activación y tres capas de agrupación máxima) con 1200 imágenes ISSCA.

En la Figura 6.8, se muestra la gráfica de exactitud obtenida con la Arquitectura 1 y ocupando 1200 imágenes.

En la Figura 6.9, se muestra la gráfica del tiempo que duró cada experimento con la Arquitectura 1 y ocupando 1200 imágenes. En este caso, se puede observar que el experimento que duró más tiempo fue empleando 32 Filtros de tamaño 5×5 tardando 64.10 min.

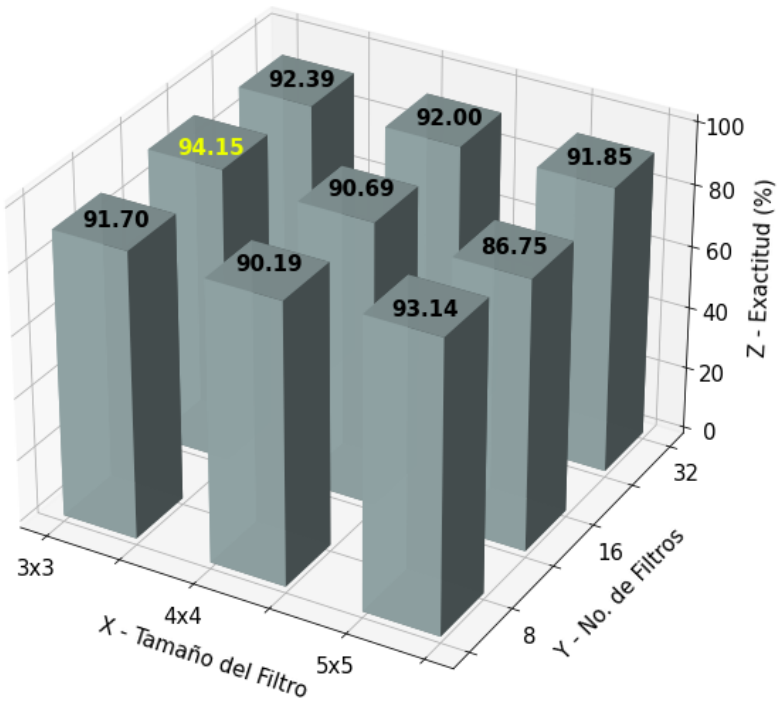


Figura 6.8: Gráfica de Exactitud utilizando la Arquitectura 1 (tres capas de convolución y tres capas de agrupación máxima) con 1200 imágenes.

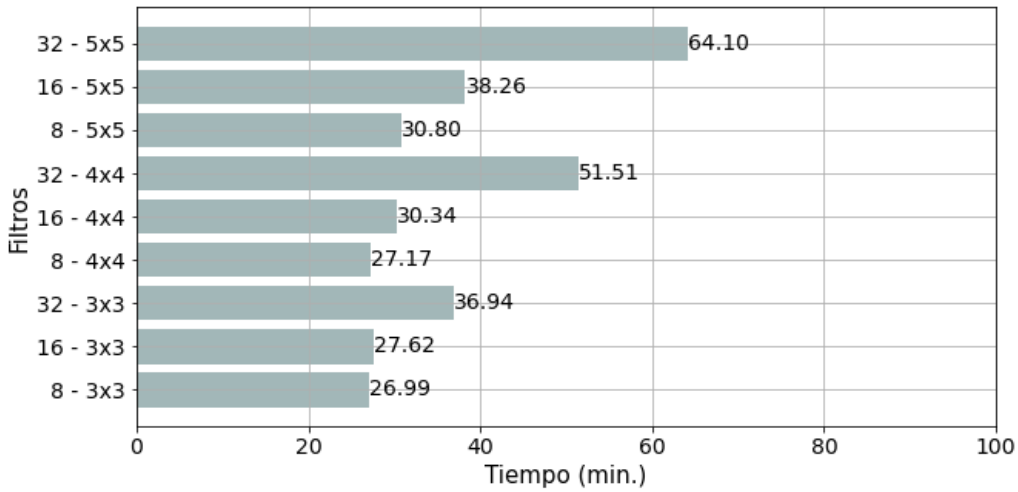


Figura 6.9: Gráfica de tiempo utilizando la Arquitectura 1 con 1200 imágenes.

En la Tabla 6.5, se muestra el desempeño obtenido utilizando 1200 imágenes y la Arquitectura 2 (ver Figura 5.7) con 8, 16 y 32 filtros de tamaño 3×3, 4×4 y 5×5. Se puede ver que utilizando 16 filtros de tamaño 3×3 se obtiene la mejor exactitud con 95.55 % de diagnosticar

una lesión como Maligna.

ARQUITECTURA [2]							
Tamaño	**# Filtros**	**Exa (%)**	**Pre (%)**	**Sen (%)**	**Auc (%)**	**F1 (%)**	**Tiempo (m)**
5 x 5	8	95.0	94.89	95.19	97.41	95.40	29.95
5 x 5	16	91.15	91.80	91.30	95.83	91.50	41.30
5 x 5	32	89.55	89.97	89.80	93.73	89.10	96.04
4 x 4	8	95.29	96.52	93.70	97.94	94.60	29.30
4 x 4	16	94.00	94.67	93.70	96.52	93.55	34.75
4 x 4	32	90.40	90.40	90.29	94.72	91.08	87.28
3 x 3	8	94.60	94.81	94.60	97.04	95.13	28.28
3 x 3	16	95.55	96.07	94.89	97.39	96.15	31.95
3 x 3	32	94.90	94.45	95.80	97.42	95.40	78.15

Tabla 6.5: Desempeño de Exactitud, Precisión, Sensibilidad, Área bajo la curva y Puntuación F-1 obtenido por el algoritmo de aprendizaje profundo utilizando la Arquitectura 2 (dos capas de convolución, dos capas de activación y una capa de agrupación máxima) con 1200 imágenes ISSCA.

En la Figura 6.10, se muestra la gráfica de exactitud obtenida con la Arquitectura 1 y ocupando 1200 imágenes.

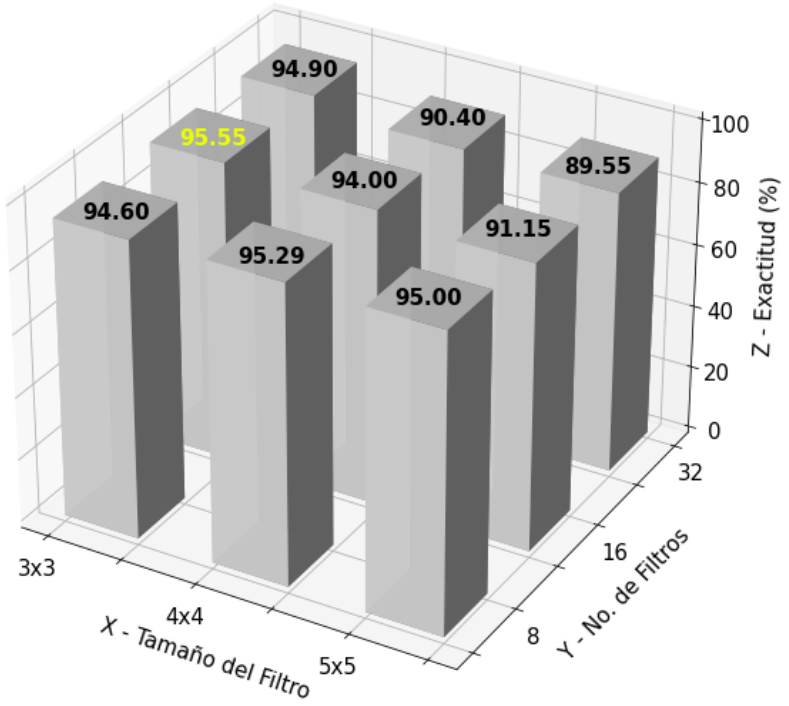


Figura 6.10: Gráfica de Exactitud utilizando la Arquitectura 2 (dos capas de convolución y una capa de agrupación máxima) con 1200 imágenes.

En la Figura 6.11, se muestra la gráfica del tiempo que duró cada experimento con la Arquitectura 2 y ocupando 1200 imágenes. En este caso, se puede observar que el experimento que duró más tiempo fue empleando 32 Filtros de tamaño 5 × 5 tardando 96.04 min.

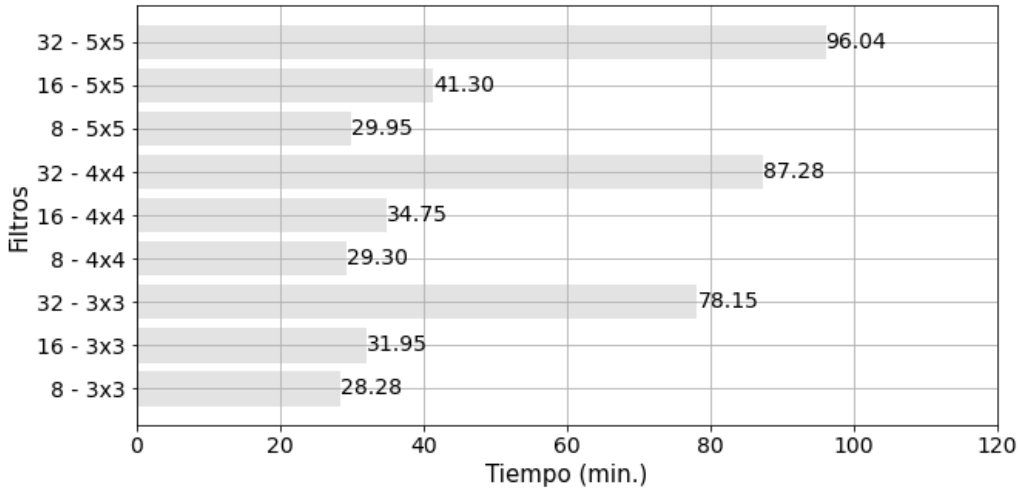


Figura 6.11: Gráfica de tiempo utilizando la Arquitectura 2 con 1200 imágenes.

En la Tabla 6.6, se muestra el desempeño obtenido utilizando 1200 imágenes y la Arquitectura 3 (ver Figura 5.8) con 8, 16 y 32 filtros de tamaño 3 × 3, 4 × 4 y 5 × 5. Se puede ver que utilizando 8 filtros de tamaño 4 × 4 se obtiene la mejor exactitud con 95.55 % de diagnosticar una lesión como Maligna

ARQUITECTURA [3]							
Tamaño	**# Filtros**	**Exa (%)**	**Pre (%)**	**Sen (%)**	**Auc (%)**	**F1 (%)**	**Tiempo (m)**
5 x 5	8	95.04	95.49	94.80	96.99	95.55	36.95
5 x 5	16	67.74	66.52	83.90	74.28	68.03	41.26
5 x 5	32	49.95	20.0	40.0	50.0	48.13	86.11
4 x 4	8	95.55	96.21	94.50	97.44	96.10	43.63
4 x 4	16	94.35	93.91	95.10	96.52	95.12	51.67
4 x 4	32	78.24	80.03	78.30	82.65	78.64	72.42
3 x 3	8	94.59	94.77	94.19	96.41	94.80	38.07
3 x 3	16	93.50	93.04	94.70	96.50	93.67	46.07
3 x 3	32	85.50	86.73	85.79	91.63	85.74	61.02

Tabla 6.6: Desempeño de Exactitud, Precisión, Sensibilidad, Área bajo la curva y Puntuación F-1 obtenido por el algoritmo de aprendizaje profundo utilizando la Arquitectura 3 (cuatro capas de convolución, cuatro capas de activación y una capa de agrupación máxima) con 1200 imágenes ISSCA.

En la Figura 6.12, se muestra la gráfica de exactitud obtenida con la Arquitectura 3 y ocupando 1200 imágenes.

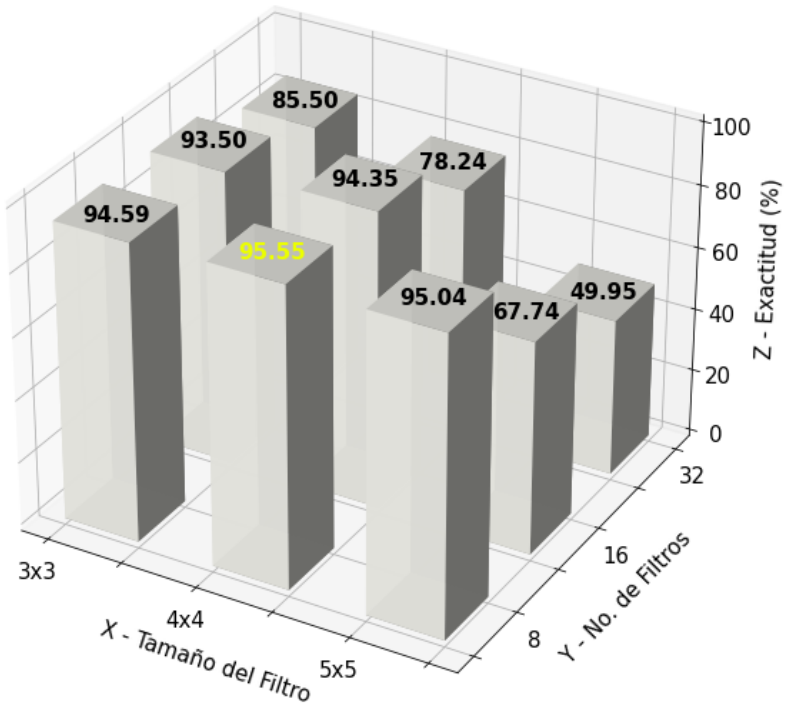


Figura 6.12: Gráfica de Exactitud utilizando la Arquitectura 3 con 1200 imágenes.

En la Figura 6.13, se muestra la gráfica del tiempo que duró cada experimento con la Arquitectura 3 y ocupando 1200 imágenes. En este caso, se puede observar que el experimento que duró más tiempo fue empleando 32 Filtros de tamaño 5×5 tardando 86.11 min.

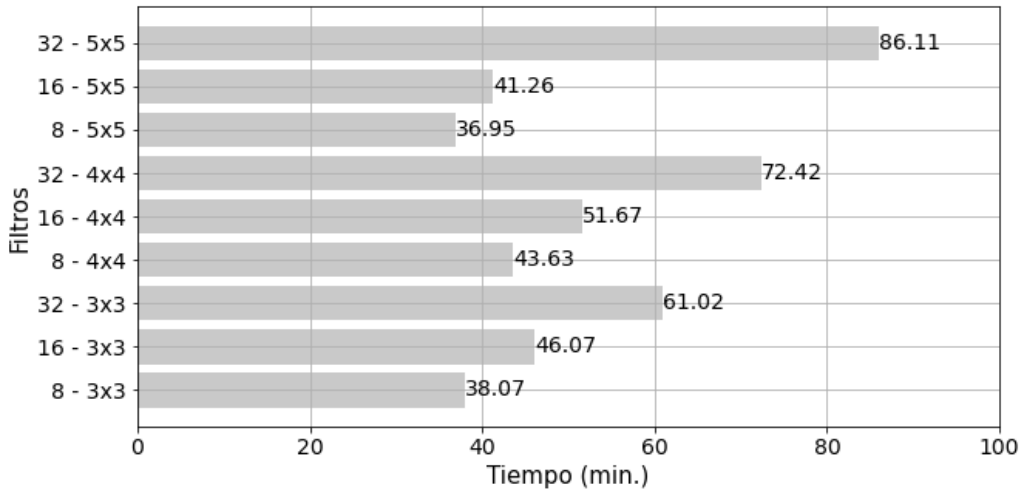


Figura 6.13: Gráfica de tiempo utilizando la Arquitectura 3 con 1200 imágenes.

En la Tabla 6.7, se muestra el desempeño obtenido utilizando 9000 imágenes y la Arquitectura 1 (ver Figura 5.1) con 8, 16 y 32 filtros de tamaño 3×3, 4×4 y 5×5. Se puede observar que empleando 16 filtros de tamaño 3×3 se obtiene la mejor exactitud siendo de 95.44 % de diagnosticar una lesión como Maligna.

ARQUITECTURA [1]							
Tamaño	**# Filtros**	**Exa (%)**	**Pre (%)**	**Sen (%)**	**Auc (%)**	**F1 (%)**	**Tiempo (hrs)**
5 x 5	8	94.24	95.14	93.44	97.15	94.05	2.76
5 x 5	16	94.34	94.42	94.20	97.16	94.10	2.72
5 x 5	32	94.41	95.03	93.62	97.31	94.10	4.20
4 x 4	8	94.38	95.13	93.41	97.32	93.96	2.90
4 x 4	16	93.54	95.27	91.38	96.83	93.00	2.68
4 x 4	32	94.37	95.32	93.36	97.20	94.11	3.75
3 x 3	8	94.32	94.12	94.71	97.40	94.22	2.40
3 x 3	16	95.44	96.77	94.09	97.90	95.10	2.68
3 x 3	32	94.53	95.38	93.55	97.41	94.21	2.78

Tabla 6.7: Desempeño de Exactitud, Precisión, Sensibilidad, Área bajo la curva y Puntuación F-1 obtenido por el algoritmo de aprendizaje profundo utilizando la Arquitectura 1 (tres capas de convolución, tres capas de activación y tres capas de agrupación máxima) con 9000 imágenes ISSCA.

En la Figura 6.14, se muestra la gráfica de exactitud obtenida con la Arquitectura 1 y ocupando 9000 imágenes.

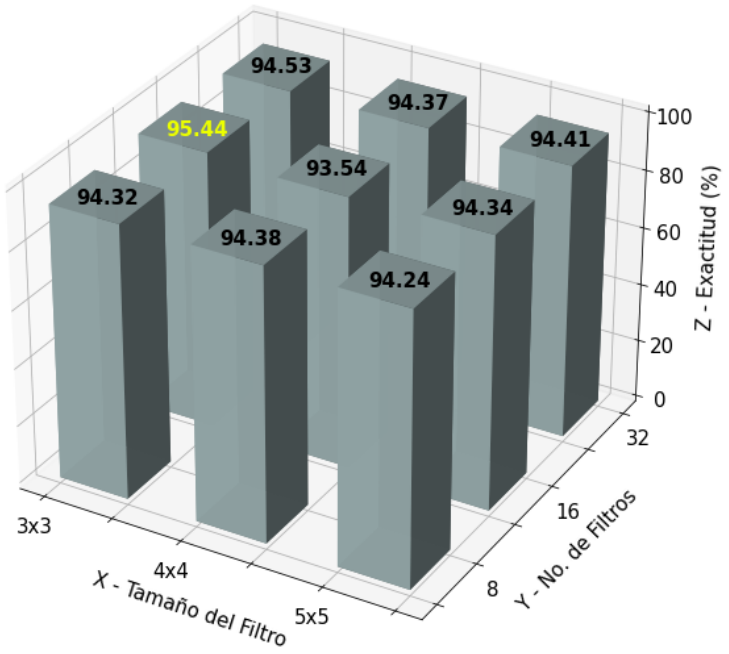


Figura 6.14: Gráfica de Exactitud utilizando la Arquitectura 1 con 9000 imágenes.

En la Figura 6.15, se muestra la gráfica del tiempo que duró cada experimento con la Arquitectura 1 y ocupando 9000 imágenes. En este caso, se puede observar que el experimento que duró más tiempo fue empleando 32 Filtros de tamaño 5×5 tardando 4.20 hrs.

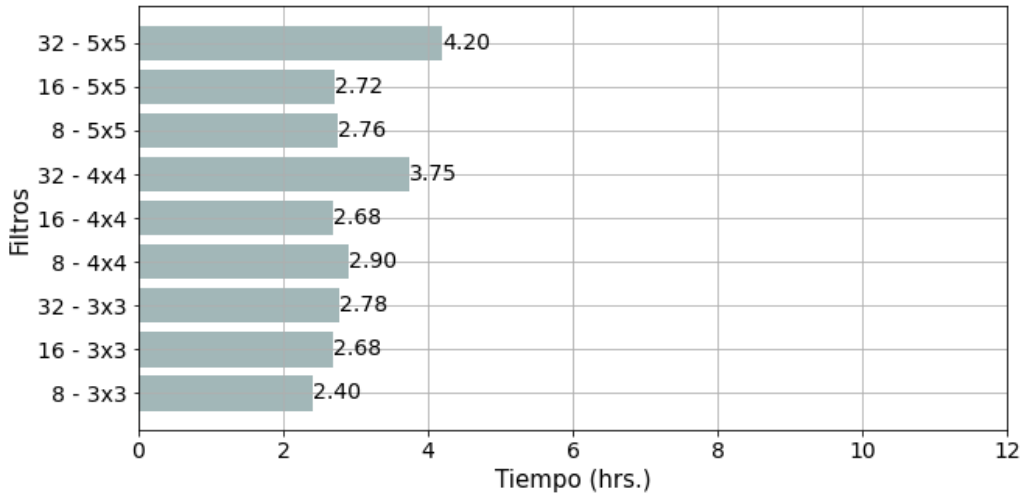


Figura 6.15: Gráfica de tiempo utilizando la Arquitectura 1 con 9000 imágenes.

En la Tabla 6.7, se muestra el desempeño obtenido utilizando 9000 imágenes y la Arquitectura 2 (ver Figura 5.7) con 8, 16 y 32 filtros de tamaño 3×3, 4×4 y 5×5. Se puede observar

que empleando 16 filtros de tamaño 3 × 3 se obtiene la mejor exactitud, siendo de 95.96% de diagnosticar una lesión como Maligna.

ARQUITECTURA [2]							
Tamaño	**# Filtros**	**Exa (%)**	**Pre (%)**	**Sen (%)**	**Auc (%)**	**F1 (%)**	**Tiempo (hrs)**
5 x 5	8	95.88	96.83	94.80	97.80	95.60	3.54
5 x 5	16	93.16	94.17	91.58	95.86	92.65	4.05
5 x 5	32	92.82	92.96	92.44	95.75	92.52	7.56
4 x 4	8	95.91	97.18	94.17	97.88	95.55	2.74
4 x 4	16	95.78	95.94	95.26	97.74	95.46	3.47
4 x 4	32	94.35	95.50	92.93	96.64	93.92	6.58
3 x 3	8	95.80	95.32	97.14	97.57	96.01	2.79
3 x 3	16	95.96	96.05	96.18	97.89	96.02	3.00
3 x 3	32	95.34	94.70	96.58	97.43	95.52	5.30

Tabla 6.8: Desempeño de Exactitud, Precisión, Sensibilidad, Área bajo la curva y Puntuación F-1 obtenido por el algoritmo de aprendizaje profundo utilizando la Arquitectura 2 (dos capas de convolución, dos capas de activación y una capa de agrupación máxima) con 9000 imágenes ISSCA.

En la Figura 6.16, se muestra la gráfica de exactitud obtenida con la Arquitectura 2 y ocupando 9000 imágenes.

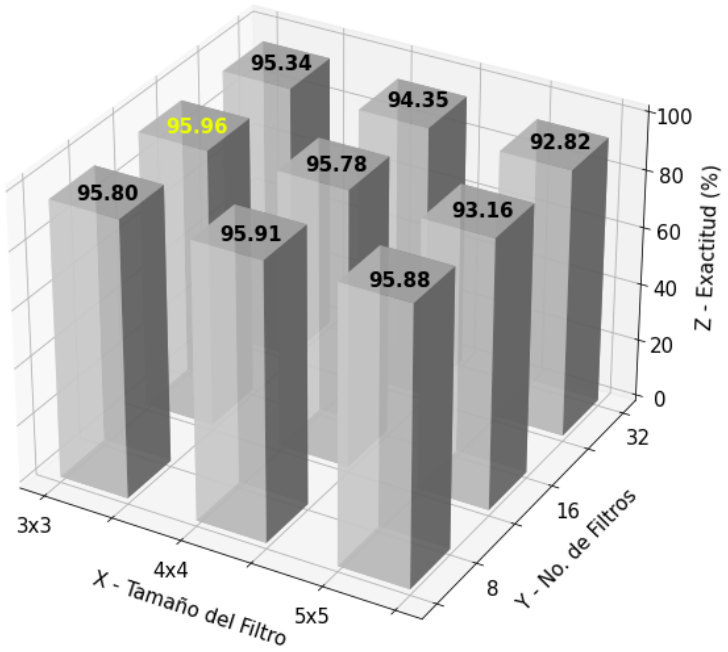


Figura 6.16: Gráfica de Exactitud utilizando la Arquitectura 2 con 9000 imágenes.

En la Figura 6.17, se muestra la gráfica del tiempo que duró cada experimento con la Arquitectura 2 y ocupando 9000 imágenes. En este caso, se puede observar que el experimento que duró más tiempo fue empleando 32 Filtros de tamaño 5 × 5 tardando 7.56 hrs.

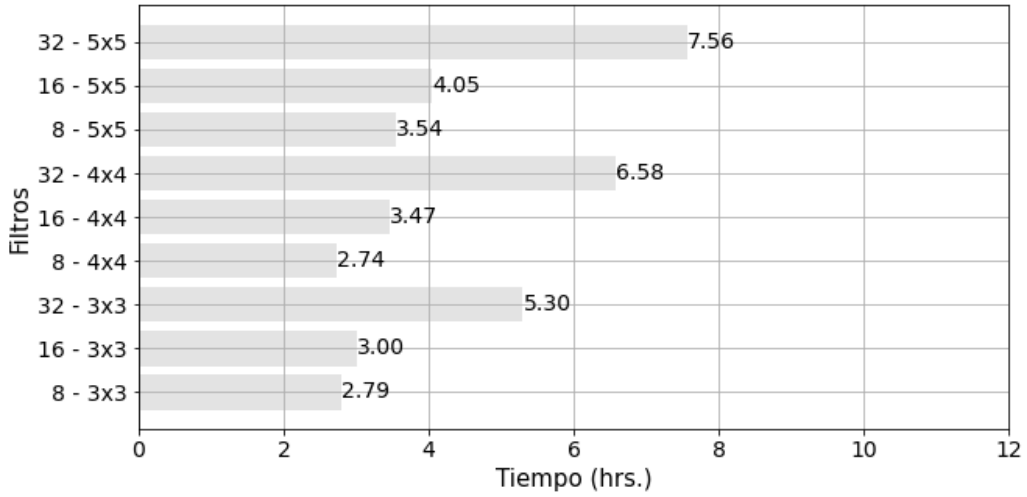


Figura 6.17: Gráfica de tiempo utilizando la Arquitectura 2 con 9000 imágenes.

En la Tabla 6.9, se muestra el desempeño obtenido utilizando 9000 imágenes y la Arquitectura 3 (ver Figura 5.8) con 8, 16 y 32 filtros de tamaño 3 × 3, 4 × 4 y 5 × 5. Se puede observar que empleando 8 filtros de tamaño 3 × 3 se obtiene la mejor exactitud, siendo de 96.37 % de diagnosticar una lesión como Maligna.

ARQUITECTURA [3]							
Tamaño	**# Filtros**	**Exa (%)**	**Pre (%)**	**Sen (%)**	**Auc (%)**	**F1 (%)**	**Tiempo (hrs)**
5 x 5	8	95.43	95.85	94.93	97.13	95.29	6.52
5 x 5	16	94.85	95.48	94.06	96.93	94.57	4.75
5 x 5	32	89.23	93.40	84.26	93.50	87.82	10.22
4 x 4	8	96.31	96.83	95.72	97.54	96.16	6.15
4 x 4	16	94.05	94.67	93.30	96.75	93.76	4.05
4 x 4	32	84.86	88.00	81.67	90.00	83.25	8.21
3 x 3	8	96.37	96.54	96.40	97.73	96.42	6.65
3 x 3	16	95.76	96.30	95.06	97.45	95.56	3.54
3 x 3	32	95.10	95.55	94.57	97.20	94.93	6.88

Tabla 6.9: Desempeño de Exactitud, Precisión, Sensibilidad, Área bajo la curva y Puntuación F-1 obtenido por el algoritmo de aprendizaje profundo utilizando la Arquitectura 3 (cuatro capas de convolución, cuatro capas de activación y una capa de agrupación máxima) con 9000 imágenes ISSCA.

En la Figura 6.18, se muestra la gráfica de exactitud obtenida con la Arquitectura 3 y ocupando 9000 imágenes.

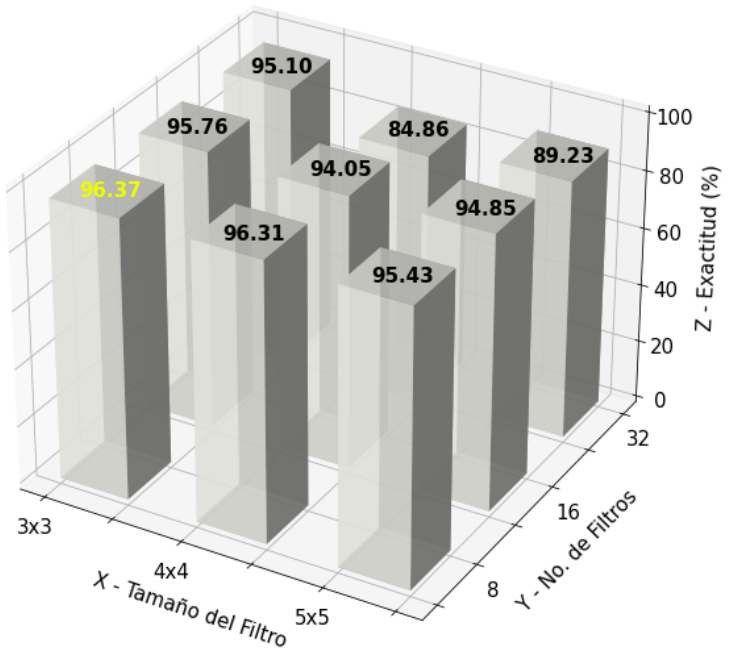


Figura 6.18: Gráfica de Exactitud utilizando la Arquitectura 3 con 9000 imágenes.

En la Figura 6.19, se muestra la gráfica del tiempo que duró cada experimento con la Arquitectura 3 y ocupando 9000 imágenes. En este caso, se puede observar que el experimento que duró más tiempo fue empleando 32 Filtros de tamaño 5×5 tardando 10.22 hrs.

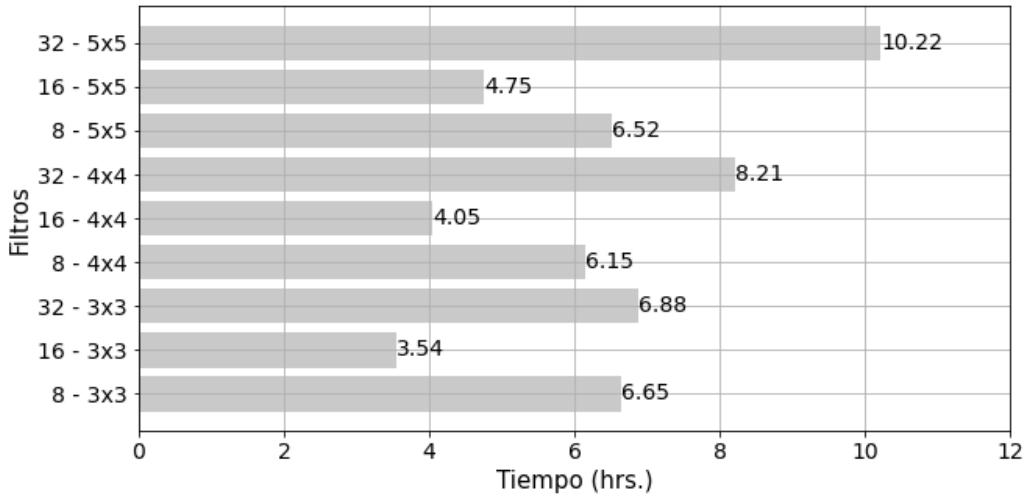


Figura 6.19: Gráfica de tiempo utilizando la Arquitectura 3 con 9000 imágenes.

Capítulo 7

Conclusiones generales

En este proyecto se construyeron y evaluaron tres arquitecturas de red neuronal convolucional, con distinta secuencia de capas de convolución y agrupación, de la cuales, se encontró que la compuesta por dos capas de convolución seguidas de una capa de agrupación más dos capas de convolución, obtiene el mejor desempeño en la clasificación de imágenes dermoscópicas de lesiones de piel pigmentada. Por otro lado, también se contrastó con el desempeño obtenido con los algoritmos de aprendizaje automático clásicos de Árboles de Decisión, Bosques Aleatorios, Regresión Logística, K Vecinos más Cercanos, Máquinas de Vectores de Soporte y Perceptrón Multicapa. A continuación, se resumen los resultados obtenidos en cada fase de experimentación.

En la Fase 1 se observó que utilizando Imágenes Segmentadas y Sin Artefactos (ISSA) puede existir cierta pérdida de información que impacta en la probabilidad para realizar un diagnóstico correcto utilizando los algoritmos clásicos. En contraparte, al utilizar Imágenes Sin Segmentar y Con Artefactos (ISSCA) se observó que el desempeño mejora en cuatro de los algoritmos clásicos, siendo estos, Árboles de Decisión, Bosques Aleatorios, Regresión Logística y K Vecinos más Cercanos.

Por otra parte, en la Fase 2 se observó un aumento de 12.09 % en la probabilidad de realizar un diagnóstico correcto al emplear la arquitectura de red neuronal convolucional compuesta

por tres capas de convolución y tres capas de agrupación, obteniendo 91.71 % de probabilidad, esto, en comparación con el algoritmo de Máquinas de Vectores de Soporte, siendo éste, el que obtuvo el mejor desempeño entre los algoritmos clásicos, con una probabilidad de 79.66 %. Derivado de lo anterior y siguiendo la teoría vista sobre aprendizaje profundo, en esta Fase se puede concluir que, para el caso de la arquitectura de red neuronal convolucional empleada, está presenta mayores capacidades para clasificar una lesión de piel pigmentada, debido al proceso que realizan las capas de convolución y agrupación; permitiéndole, extraer y aprender características de manera automática.

Finalmente en la Fase 3, donde se utilizaron las tres arquitecturas de red neuronal convolucional y se encontró aquella con el mejor desempeño para clasificar una lesión como Maligna o Benigna. Al comparar los resultados obtenidos entre las arquitecturas empleadas, se puede concluir que al variar el tamaño del filtro en las capas de convolución, el desempeño tiende a mejorar utilizando filtros de tamaño 3×3. En el caso de la cantidad de filtros, analizando exclusivamente la probabilidad obtenida por filtros de tamaño 3×3, se observó que para la arquitectura 1 y 2 el desempeño mejora utilizando 16 filtros, sin embargo, en la arquitectura 3 al utilizar 8 filtros es mejor el desempeño. En lo que respecta al número de capas, se observó que utilizando la arquitectura 3, compuesta por 4 capas de convolución y 1 capa de agrupación, se obtiene una probabilidad de 96.37 %, siendo esta, la mejor obtenida entre las tres arquitecturas. Al ser analizados los resultados obtenidos única y exclusivamente con las tres arquitecturas propuestas, utilizando las muestras de 1200 y 9000 imágenes, se puede observar que aumenta el desempeño de estas arquitecturas empleadas en los casos siguientes: la cantidad de capas de convolución incrementa, tal es el caso de la arquitectura 3 compuesta por cuatro capas de convolución; cuando la cantidad de filtros disminuye, en este caso al utilizar 32, 16 y 8 filtros; y cuando el tamaño del filtro disminuye, en este caso al utilizar 5×5, 4×4 y 3×3. Por otro lado, analizando el tiempo que tarda en realizar su entrenamiento cada arquitectura propuesta, se puede observar que aumenta la duración del

entrenamiento de estas arquitecturas empleadas en los casos siguientes: al utilizar filtros de mayor tamaño, en este caso 5×5; la cantidad de filtros que se utilizó es mayor, en este caso 32; y la cantidad de capas de convolución que es utilizada se incrementa.

7.1. Trabajo futuro

Una vez que se realizaron los experimentos y se analizaron los resultados obtenidos, y dadas las condiciones para entrenar la arquitectura que obtuvo el mejor desempeño, la cual, requirió un tiempo en promedio de 6.33 horas para procesar 9000 imágenes correspondientes a 423MB. Se propone que la arquitectura se despliegue en una plataforma de aprendizaje máquina en la nube, como, Amazon SageMaker de Amazon Web Service (AWS). El motivo de elegir una plataforma en la nube es que permite mayor capacidad de procesamiento y almacenamiento que el entorno local con el que se cuenta. Por su parte, SageMaker presenta algunas ventajas en comparación con otras plataformas como Azure Machine Learning y Google Cloud Platform Vertex AI, además de que, también es posible automatizar y estandarizar los procesos al implementar el ciclo de vida de las operaciones de aprendizaje máquina, que abarcan el despliegue, evaluación, monitoreo y operación de la arquitectura de red neuronal convolucional. Por su parte, el costo de utilizar SageMaker es flexible, por lo tanto, soló se paga por los servicios que se ocupen. Un costo aproximado de utilizar SageMaker, con sus utilerías de menor precio, es de $31.58 por hora, de acuerdo a los precios encontrados en el sitio web de AWS. Por otro lado, se va a empaquetar la arquitectura en una API Rest, buscando con esto, operar en tiempo real para generar al usuario predicciones en el menor tiempo posible. Por otra parte, para gestionar de mejor manera las dependencias utilizadas, será empleado un contenedor Docker. De esta manera, se espera desarrollar e integrar un sistema computacional que sirva como apoyo a los médicos especialistas en el diagnóstico de cáncer de piel.

Referencias

Aggarwal, C. C. (2018). *Neural Networks and Deep Learning*. Springer, IBM T. J. Watson Research Center International Business Machines Yorktown Heights, NY, USA.

AlShourbaji, I., Samara, G., Zogaan, W., Alam, S., and Aliero, M. (2021). Early detection of skin cancer using deep learning approach. *İlköğretim Online*, 20:3880–3884.

Beysolow, T. (2017). *Introduction to Deep Learning Using R*. Apress.

Bhagwat, R., Abdolahnejad, M., and Moocarme, M. (2019). *Applied Deep Learning with Keras: Solve complex real-life problems with the simplicity of Keras*. Packt.

Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B., and Varoquaux, G. (2013). API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122.

Chollet, F. (2018). *Deep Learning with Python*. Manning Publications Co., 20 Baldwin Road PO Box 761 Shelter Island, NY 11964.

Codella, N. C. F., Gutman, D. A., Celebi, M. E., Helba, B., Marchetti, M. A., Dusza, S. W., Kalloo, A., Liopyris, K., Mishra, N. K., Kittler, H., and Halpern, A. (2017). Skin lesion analysis toward melanoma detection: A challenge at the 2017 international symposium on biome-

dical imaging (isbi), hosted by the international skin imaging collaboration (ISIC). *CoRR*, abs/1710.05006.

Codella, N. C. F., Rotemberg, V., Tschandl, P., Celebi, M. E., Dusza, S. W., Gutman, D. A., Helba, B., Kalloo, A., Liopyris, K., Marchetti, M. A., Kittler, H., and Halpern, A. (2019). Skin lesion analysis toward melanoma detection 2018: A challenge hosted by the international skin imaging collaboration (ISIC). *CoRR*, abs/1902.03368.

Combalia, M., Codella, N. C. F., Rotemberg, V., Helba, B., Vilaplana, V., Reiter, O., Carrera, C., Barreiro, A., Halpern, A. C., Puig, S., and Malvehy, J. (2019). Bcn20000: Dermoscopic lesions in the wild.

Cortes, C. and Mohri, M. (2003). Auc optimization vs. error rate minimization. In Thrun, S., Saul, L., and Schölkopf, B., editors, *Advances in Neural Information Processing Systems*, volume 16. MIT Press.

David A. Forsyth, J. P. (2013). *Computer Vision a Modern Approach*. Pearson, second edition.

Falabella, R. F., Chaparro, J. V., and Cabal, M. I. B. (2017). *Fundamentos de medicina*. CIB Fondo Editorial.

Foundation, S. C. (2022). Cancer de piel en personas de piel oscura. [Web; accedido el 07-04-2022].

García, J., Molina, J. M., Berlanga, A., Patricio, M. A., Álvaro L. Bustamante, and Padilla., W. R. (2018). *Ciencia de datos Técnicas analíticas y aprendizaje estadístico*. Alfaomega.

Gatti, J. C. and Cardama, J. E. (1975). *Manual de dermatología*. El ateneo.

Gerges, F. and Shih, F. (2021). A convolutional deep neural network approach for skin cancer detection using skin lesion images. volume 15, pages 475 – 478.

Géron, A. (2019). *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly, second edition.

Hackeling, G. (2014). *Mastering Machine Learning with scikit-learn*. Packt Publishing, Livery Place 35 Livery Street Birmingham B3 2PB, UK.

Han, J. and Kamber, M. (2001). *Data Mining Concepts and Techniques*. Academic Press.

Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The Elements of Statistical Learning Data Mining, Inference, and Prediction*. Springer.

Kantardzic, M. (2011). *Data Mining Concepts, Models, Methods, and Algorithm*. IEEE Press, second edition.

Kroese, D. P., Botev, Z. I., Taimre, T., and Vaisman, R. (2022). *Data Science and Machine Learning Mathematical and Statistical Methods*. CRC Press.

Manzur, J., Almeida, J. D., and Cortés, M. (2002). *Dermatología*. Ciencia Médicas.

Maruch, S. and Maruch, A. (2006). *Python for dummies*. Wiley Publishing, Inc.

Mohri, M., Rostamizadeh, A., and Talwalkar, A. (2012). *Foundations of Machine Learning*. MIT Press.

Morales, R. R. and Azuela, J. H. S. (2012). *Procesamiento y Análisis Digital de Imágenes*. Alfaomega.

Murphy, K. P. (2012). *Machine Learning A Probabilistic Perspective*. The MIT Press Cambridge, Massachusetts London, England.

Nasr Esfahani, E., Samavi, S., Karimi, N., Soroushmehr, S., Jafari, M., Ward, K., and Najarian, K. (2016). Melanoma detection by analysis of clinical images using convolutional neural network. volume 2016.

Nelly, F. (2015). *Python Data Analytics: Data analysis and science using pandas, matplotlib and the python programming language*. Apress.

Nouri, K. (2008). *Skin Cancer*. McGraw-Hill.

Ottom, M. A. (2019). Convolutional neural network for diagnosing skin cancer. *International Journal of Advanced Computer Science and Applications*, 10(7).

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Quinto, B. (2020). *Next-Generation Machine Learning with Spark*. Apress.

Rafael C. Gonzalez, R. E. W. (2018). *Digital Image Processing*. Pearson, 330 Hudson Street, New York, NY 10013, fourth edition.

Raschka, S., Julian, D., and Hearty, J. (2016). *Python: Deeper Insights into Machine Learning*. Packt Publishing.

Raschka, S. and Mirjalili, V. (2017). *Python Machine Learning*. Packt Publishing, second edition.

Reichrath, J. (2006). *Molecular Mechanisms of Basal Cell and Squamous Cell Carcinomas*. Springer Science and Business Media.

Rigel, D. S., Robinson, J. K., Ross, M., Friedman, R. J., Cockerell, C. J., Lim, H. W., Stockfleth, E., and Kirkwood, J. M. (2011). *Cancer of the skin*. Elsevier, second edition.

Roldan, R. (2019). Unam dirección general de comunicación social. [Web; accedido el 07-04-2022].

Salvaris, M., Dean, D., and Tok, W. H. (2018). *Deep Learning with Azure*. Apress.

Sarkar, D., Bali, R., and Sharma, T. (2018). *Practical Machine Learning with Python*. Apress.

Shalev-Shwartz, S. and Ben-David, S. (2014). *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press.

Society, A. C. (2022). Estadísticas importantes sobre cáncer de piel tipo melanoma. [Web; accedido el 07-04-2022].

Sonka, M., Hlavac, V., and Boyle, R. (2015). *Image Processing, Analysis, and Machine Vision*. Cengage Learning, fourth edition.

Steigleder, G. K. (1985). *Atlas de dermatología*. Cientifica PLM.

Stipaničev, D. (1994). *Introduction to Digital Image Processing and Analysis*. Faculty of Electrical Engineering, Naval Architecture UNIVERSITY OF SPLIT.

Swamynathan, M. (2017). *Mastering Machine Learning with Python in Six Steps*. Apress.

Tschandl, P., Rosendahl, C., and Kittler, H. (2018). The HAM10000 dataset: A large collection of multi-source dermatoscopic images of common pigmented skin lesions. *CoRR*, abs/1803.10417.

Vasilev, I., Daniel Slater, G. S., Roelants, P., and Zocca, V. (2019). *Python Deep Learning*. Packt Publishing Ltd, second edition.

Vieira, A. and Ribeiro, B. (2018). *Introduction to Deep Learning Business Applications for Developers From Conversational Bots in Customer Service to Medical Image Processing*. Apress.

Viñuela, P. I. (2004). *Redes de Neuronas Artificiales Un Enfoque Práctico*. Prentice Hall.

Yu, A. C. S. and Chung, C. Y. L. (2017). *Matplotlib 2.x By Example: Multidimensional charts, graphs, and plots.* Packt Publishing.

Zaman, K., Sozan, S., and Maghdid, S. (2021). Medical images classification for skin cancer using convolutional neural network algorithms. *Advances in Applied Mechanics*, 09:526–541.