



CENTRO DE INVESTIGACIONES
EN OPTICA, A.C.

“DESARROLLO DE UN SISTEMA DE DETECCIÓN DE CARCINOMA DUCTAL INVASIVO (CDI) VÍA IMÁGENES HISTOPATOLÓGICAS”

Tesis que para obtener el grado de Maestro en Optomecatrónica

Presenta: Juan Marcos Chávez Medina

Director de Tesis: Dr. Sebastián Salazar Colores

Co – Director de Tesis: Dr. Edgar Efrén Lozada Hernández

Versión Definitiva. Incluye cambios sugeridos por revisores.

Vo.Bo.

*16 de junio del 2023
León · Guanajuato · México
Junio de 2023*

Índice general

1. Introducción	4
1.1. Inteligencia Artificial (IA)	5
1.2. Machine Learning (ML)	5
1.2.1. Aprendizaje automático supervisado	6
1.2.2. Aprendizaje automático no supervisado	6
1.3. Aprendizaje profundo y las Redes Neuronales Artificiales (RNA)	7
1.4. Sistemas de detección de carcinoma ductal invasivo vía imágenes histopatológicas	9
1.5. Planteamiento del problema	10
1.6. Justificación	11
1.7. Alcance y objetivos del proyecto	11
1.7.1. Alcance	11
1.7.2. Objetivo	11
1.7.3. Objetivos específicos	12
1.8. Hipótesis	12
1.9. Estado del arte	15
1.9.1. Keras	15
1.9.2. Red Neuronal Convolutiva (RNC)	16
1.9.3. Funciones de activación	17
1.9.4. Aplanamiento o Flatten	36
1.9.5. Retropropagación o Backpropagation	36
1.9.6. RNC para detección de CDI con AlexNet	37
1.9.7. Predictor CDI en cáncer de pecho con imágenes histopatológicas	39
1.9.8. RNC comparando varios modelos	40
2. Metodología	45
2.1. Selección de librerías y datos	45
2.2. TensorFlow	45
2.3. Base de datos o Dataset	46
2.4. Modelos neuronales	47
2.4.1. Diferentes arquitecturas de los modelos	48
2.4.2. LeNet	49
2.4.3. AlexNet	49
2.4.4. VGG16	51
2.4.5. ResNet	52
2.4.6. Inception (GoogleNet)	53
2.4.7. Xception	54

2.4.8.	EfficientNet	55
2.4.9.	MobileNet	56
2.4.10.	DenseNet	58
2.5.	Métricas de clasificación	59
2.5.1.	Precisión	59
2.5.2.	Sensitividad	60
2.5.3.	Puntaje F1	61
2.5.4.	Soporte	61
2.5.5.	Exactitud	62
2.6.	Desarrollo de plataforma	62
2.6.1.	Raspberry Pi4	63
2.6.2.	Instalación de TensorFlow en Raspberry Pi 4	63
2.6.3.	Diseño de GUI	64
2.6.4.	Adaptación del sistema embebido con el detector de CDI en un diseño portable	65
3.	Resultados	66
3.1.	Manejo de datos para los parámetros de entrada y de entrenamiento, validación y prueba	66
3.2.	Entrenamiento del sistema con los modelos	66
3.3.	Desarrollo del sistema embebido	68
3.4.	Desarrollo de interfaz y aplicación al sistema embebido	70
3.5.	Resultados del entrenamiento	72
3.6.	Imágenes del detector en el sistema embebido	82
3.7.	Diseño de carcasa (case)	84
3.8.	Algoritmo de inferencia	85
4.	Conclusiones	86
A.	Instalación del sistema operativo de Raspberry Pi Os	92
B.	Lista de acrónimos	99

Índice de figuras

1.1. Función del aprendizaje automático supervisado.	7
1.2. Función del aprendizaje automático no supervisado.	7
1.3. Diagrama de la estructura básica de las RNA (ejemplo con tres capas, cuatro neuronas de entrada, dos neuronas ocultas y una neurona de salida).	8
1.4. Diagrama con la estructura de una neurona con tres entradas y sus respectivos pesos (Perceptrón), se asemeja a una neurona biológica.	8
1.5. Diagrama de la estructura biológica de una neurona [5].	9
1.6. Pasos para aplicar la capa de agrupación con Maxpooling.	17
1.7. Función sigmoideal o sigmoide	18
1.8. Función tangente hiperbólica	19
1.9. Función ReLU.	19
1.10. Función de activación Softmax.	20
1.11. Función Leaky ReLU, con valor α de 0.1.	21
1.12. Función ELU.	22
1.13. Función GELU.	23
1.14. Función SELU.	24
1.15. Función Swish.	26
1.16. Aplanamiento (Flatten).	36
1.17. Orden de tareas de prioridad para el entrenamiento del modelo [7].	38
1.18. Diseño de modelo propio utilizado en artículo de Mooney.	39
1.19. Resumen de la RNC del modelo 1.	41
1.20. Resumen de la RCN del modelo 2 con el triple de capas convolucionales.	42
1.21. Resumen de la RCN del modelo 3.	43
2.1. Ejemplo del uso de la microscopía virtual en imágenes histopatológicas [12].	47
2.2. Diagrama de la arquitectura de LeNet5.	49
2.3. Diagrama de la arquitectura de AlexNet.	50
2.4. Arquitectura del modelo VGG16.	52
2.5. Arquitectura del modelo ResNet.	53
2.6. Arquitectura del modelo Inception V3 o GoogleNet.	54
2.7. Arquitectura del modelo Xception.	55
2.8. Arquitectura de EfficientNet.	56
2.9. Arquitectura de MobileNet.	58
2.10. Arquitectura de DenseNet.	59
2.11. Interfaz inicial para la instalación del sistema operativo en la Raspberry Pi 3 y 4.	64

3.1.	Interfaz de comunicación del detector CDI.	70
3.2.	Elección de carpeta a analizar.	70
3.3.	Selección y aceptación de la carpeta dentro de la PC.	71
3.4.	Carpeta con imágenes histopatológicas seleccionada.	71
3.5.	Se muestra progreso de generación de imagen Draw slice mask prediction.	72
3.6.	Imagen histopatológica de Slide.	82
3.7.	Imagen histopatológica de fondo con máscara marcada en color.	83
3.8.	Slice_Mask.	83
3.9.	Slice_Mask_Pred.	83
3.10.	Mask_Predict.	84
3.11.	Prototipo de carcasa para pantalla táctil de 7 pulgadas, parte frontal.	84
3.12.	Adaptación del case de RaspBerry Pi 4 a la carcasa de la pantalla táctil de 7 pulgadas, parte trasera.	85
4.1.	Resultados de la exactitud respecto a tiempo de entrenamiento de los modelos.	87
A.1.	Selección de sistema operativo.	92
A.2.	Selección de puerto USB.	93
A.3.	Selección de algún otro sistema operativo como Ubuntu en diferentes versiones.	93
A.4.	Paso final de la instalación.	94
A.5.	Pasar la microSD con el sistema operativo ya instalado a la Raspberry Pi 4.	94
A.6.	Interfaz de cambio de contraseña.	95
A.7.	Pantalla para configuración de pantalla.	95
A.8.	Selección de redes Wifi.	96
A.9.	Configurar contraseña para WiFi.	96
A.10.	Confirmación de carga del sistema operativo.	97
A.11.	Lectura para verificación de carga de lista del sistema operativo.	97
A.12.	Selección de país y región horaria.	97
A.13.	Confirmación de configuración completa.	98

Agradecimientos

Quiero agradecer primeramente a Dios, por permitirme realizar y finalizar mis estudios en el Centro de Investigaciones en Óptica.

Agradezco también a mi asesor de tesis, el Dr. Sebastián Salazar Colores, por su paciencia, dedicación, apoyo y conocimiento ofrecido en estos dos años de maestría. De igual manera, también agradezco a mis sinodales por el esfuerzo en revisar y corregir este trabajo.

De manera especial, quiero agradecer a mi esposa Kelly y a mis hijas Alessia y Ariadna, por ser el motivo y soporte durante todos mis estudios de posgrado y en la vida, las amo.

Gracias a mi papá Guadalupe y a mi mamá Concepción, por tanto apoyo y amor ofrecido durante toda mi vida, sin ser la excepción en esta etapa de maestría.

Gracias a mis hermanos Manuel, Keila, Josias y Adrián, por su compañía y consejos. A mis sobrinos, por que me hacen feliz.

A mis compañeros de generación, por el apoyo y camaradería durante mi estancia en el CIO.

A mis profesores, gracias por su dedicación y esfuerzo, y por el aprendizaje que me dejan.

Y finalmente, agradezco al Centro de Investigaciones en Óptica (CIO) y al Consejo Nacional de Ciencia y Tecnología (CONACYT) por el apoyo económico y la confianza, que tiene en la juventud de México y el mundo.

Resumen

En esta tesis, se ha abordado el desafío de desarrollar un detector de Carcinoma Ductal Invasivo basado en imágenes histopatológicas. Para lograrlo, se emplearon redes neuronales convolucionales (RNC) y múltiples clasificadores, aprovechando la adquisición de imágenes de grandes dimensiones a través del uso de parches.

Los resultados obtenidos son altamente prometedores, superando el estado del arte en términos de precisión, sensibilidad, exactitud, tiempo de entrenamiento, puntaje F1 y soporte. Además, se destaca que el modelo seleccionado de entre 19 redes neuronales de Keras applications, con tres diferentes tasas de aprendizaje, tiene la capacidad de inferir en un sistema embebido, presentando características específicas que lo hacen ideal para su implementación en dispositivos con recursos limitados.

Un aspecto clave de este trabajo de investigación, es el diseño y desarrollo de un sistema embebido con una interfaz sencilla de operar en un dispositivo portátil y ligero. Esto permite que el detector de CDI sea accesible en entornos clínicos, proporcionando a los profesionales de la salud una herramienta confiable y de fácil uso para la detección temprana de esta enfermedad.

Así, este trabajo de investigación presenta un enfoque innovador para la detección de Carcinoma Ductal Invasivo, aprovechando el poder de las redes neuronales convolucionales y clasificadores múltiples. Los resultados obtenidos respaldan la eficacia y viabilidad de este enfoque, además de su capacidad para ser implementado en sistemas embebidos, lo que lo convierte en una herramienta potencialmente invaluable para la detección temprana de esta forma común de cáncer de mama.

Abstract

In this thesis, the challenge of developing an invasive ductal carcinoma detector based on histopathological images has been addressed. To achieve this, convolutional neural networks (CNN) and multiple classifiers were employed, taking advantage of the acquisition of high-dimensional images through the use of patches.

The results obtained are highly promising, surpassing the state of the art in terms of precision, sensitivity, accuracy, training time, F1 score and support. In addition, it is highlighted that the model selected from among 19 Keras applications neural networks, with three different learning rates, has the ability to infer in an embedded system, presenting specific characteristics that make it ideal for implementation in devices with limited resources.

A key aspect of this research work is the design and development of an embedded system with a simple to operate interface in a portable and lightweight device. This allows the CDI detector to be accessible in clinical settings, providing healthcare professionals with a reliable and easy-to-use tool for the early detection of this disease.

Thus, this research work presents an innovative approach for the detection of Invasive Ductal Invasive Carcinoma, leveraging the power of convolutional neural networks and multiple classifiers. The results obtained support the efficacy and feasibility of this approach, in addition to its ability to be implemented in embedded systems, making it a potentially invaluable tool for the early detection of this common form of breast cancer.

Capítulo 1

Introducción

En este capítulo, se presenta una visión general del tema de investigación, destacando su importancia y relevancia en el contexto actual. Se expondrá el planteamiento del problema que se aborda en la tesis, así como los objetivos específicos que se pretenden alcanzar. Además, se proporcionará una descripción de los principales conceptos y teorías relacionados, sentando las bases para el desarrollo de la investigación. Por último, se revisará el estado del arte actual, identificando las investigaciones previas relevantes y resaltando la brecha de conocimiento que esta tesis busca abordar.

Actualmente, el cáncer se ha convertido en la principal causa de muerte en el mundo, causando más de 10 millones de muertes a nivel mundial en el 2020; [16] ya que lamentablemente esta enfermedad puede desarrollarse en cualquier parte del cuerpo, a cualquier edad y sin importar el género.

Desde el 2018, el cáncer de mama se convirtió en la primera causa de muerte en mujeres de 30 a 59 años en México, superando al cáncer cervicouterino y al cáncer de pulmón. También existen subtipos de cáncer de mama y el nombre de la enfermedad se toma según la sección del seno o el tejido en dónde se forma el cáncer [1]. De todos los subtipos de cáncer de mama, el CDI es el más común y uno de los que evoluciona más rápidamente, cuando se detecta a través de una mamografía o un ultrasonido el cáncer se estuvo desarrollando alrededor de 5 a 7 años [2].

El Carcinoma Ductal Invasivo (CDI), a veces denominado carcinoma ductal infiltrante, que es el tipo más común de cáncer de mama. Alrededor del 80% de todos los casos de cáncer de mama son carcinomas ductales invasivos. Con el tiempo, el carcinoma ductal invasivo puede propagarse hacia los ganglios linfáticos y posiblemente a otras áreas del cuerpo. Cada año aumenta el número de pacientes que sufren esta enfermedad y un gran porcentaje mueren en un corto o mediano plazo [1]. Por lo anterior, se han realizado diversos trabajos de investigación enfocados en ayudar a los médicos a detectar lo antes posible este padecimiento, apoyándose de la Microscopía virtual, que es una técnica que consiste en tomar pequeñas fotografías de alta definición y en unión forman una imagen de mayor dimensión.

En el ámbito de la detección temprana y precisa de enfermedades, especialmente en el campo del cáncer, el uso de tecnologías de inteligencia artificial se ha convertido en un área de investigación activa y prometedora. En particular, el Carcinoma Ductal Invasivo (CDI), una forma común de cáncer de mama, requiere de métodos efectivos

de detección para un diagnóstico temprano y un tratamiento adecuado.

Con estas imágenes se entrena la red neuronal convolucional, y actualmente los sistemas logran detectar el CDI o el tejido sano con un poco más del 93 % de confiabilidad. Por eso, cada modelo busca aumentar el porcentaje de precisión, mejorando cada vez más el resultado de los detectores.

Este proyecto de investigación, se enfoca en desarrollar un sistema que procesa imágenes extraídas en biopsias de 179 pacientes, para entrenar una red neuronal convolucional que detecta si el tejido tiene CDI o no. Además, se implementa en un sistema embebido portátil, teniendo un mayor alcance en centros médicos con difícil concentración de especialistas en el área, con ello se ayudará a eficientar los diagnósticos y darles mayor certeza a los médicos para que tomen decisiones oportunas [3].

1.1. Inteligencia Artificial (IA)

Durante toda la historia, el hombre ha buscado entender e imitar tanto la anatomía de los seres vivos, como el comportamiento y el funcionamiento de la naturaleza; sin embargo, el cerebro humano es tan complejo que, a pesar de que lleva décadas de ser estudiado, no se ha podido comprender su funcionamiento por completo.

Desde inicios de los años 50, con la creación y evolución de las computadoras, se han buscado generar algoritmos que sean capaces de hacer que una máquina se comporte de una manera similar al cerebro humano y pueda realizar operaciones básicas como contar y operar números, pero no fue hasta mediados de los 70 que estos dispositivos pudieron realizar actividades más complejas como procesar información y recordar resultados que obtuvieron para usarlos más adelante con algún objetivo, buscando siempre imitar a nuestro cerebro.

Precisamente esta última línea describe básicamente la definición de la IA. Actualmente, vemos a la IA en todas partes en nuestra vida cotidiana, ya que se puede aplicar en la industria, la educación, la salud e inclusive en aparatos del hogar y uso diario como un celular o una TV.

1.2. Machine Learning (ML)

Como ya se mencionó, la IA busca imitar el funcionamiento del cerebro humano, y de esta necesidad surge el Machine Learning (ML) o aprendizaje automático en español, es un método que busca aprender del procesamiento de datos con el cual es alimentado y con el aprendizaje que obtiene actualiza su algoritmo para retener esa información y pueda tomar decisiones sin la necesidad de que un humano intervenga en este proceso. Dicho sea de paso, una de sus aplicaciones más comunes es el de procesamiento de imágenes, ya que puede reconocer patrones y clasificarlas según las características que pudo extraer.

Generalmente, el ML se puede clasificar en dos tipos:

- Entrenamiento automático supervisado
- Entrenamiento automático no supervisado

1.2.1. Aprendizaje automático supervisado

En el aprendizaje supervisado, cada punto de datos de entrenamiento está asociado con varias características de entrada, generalmente un vector de características de entrada (como el valor de la característica, el tamaño, etcétera) y su etiqueta correspondiente (como el nombre de la variable o el tipo). Un modelo se construye con varios parámetros que intentan predecir la etiqueta de salida dado el vector de características de entrada. Los parámetros del modelo se obtienen optimizando alguna forma de función de costo que se basa en el error de predicción; es decir, la discrepancia entre las etiquetas reales y las etiquetas previstas para los puntos de datos de entrenamiento. Alternativamente, maximizar la probabilidad de los datos de entrenamiento también nos proporcionaría los parámetros del modelo.

Uno de los métodos de entrenamiento automático supervisado más utilizado es el método de regresión lineal. Podríamos tener un conjunto de datos que tenga el precio de las casas como variable objetivo o etiqueta de salida, mientras que características como el área de la casa, el número de dormitorios, el número de baños, etc., son su vector de características de entrada. Sea el vector de características de entrada representado por x' y el valor predicho sea y_p . Denote el valor real del precio de la vivienda, es decir, la etiqueta de salida, mediante y . Se puede definir un modelo en el que la etiqueta de salida se exprese como una función del vector de características de entrada, como se muestra en la siguiente ecuación. El modelo está parametrizado por varias constantes que queremos aprender a través del proceso de entrenamiento.

$$\hat{y}/x' = \theta'^T x' + b + \epsilon \quad (1.1)$$

Donde ϵ es la variación aleatoria en la predicción, b el sesgo y T el nivel del valor θ [5].

1.2.2. Aprendizaje automático no supervisado

Los algoritmos de aprendizaje automático no supervisados tienen como objetivo encontrar patrones o estructuras internas dentro de conjuntos de datos que contienen puntos de datos de entrada sin etiquetas ni objetivos. El agrupamiento de K-medias, la mezcla de Gaussianas, etc., son métodos de aprendizaje no supervisado. Incluso las técnicas de reducción de datos como el análisis de componentes principales, la descomposición de valores singulares, los codificadores automáticos, etc., son métodos de aprendizaje no supervisados [5].

La figura 1.1 muestra la función del aprendizaje automático supervisado y la figura 1.2 muestra la forma en que trabaja el aprendizaje automático no supervisado.

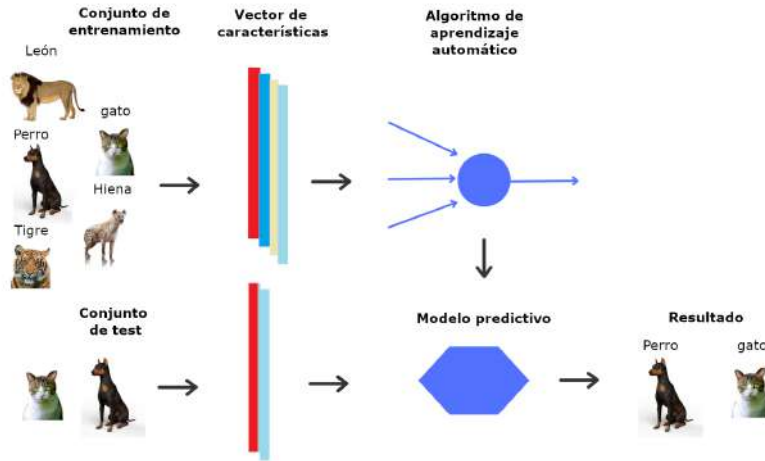


Figura 1.1: Función del aprendizaje automático supervisado.

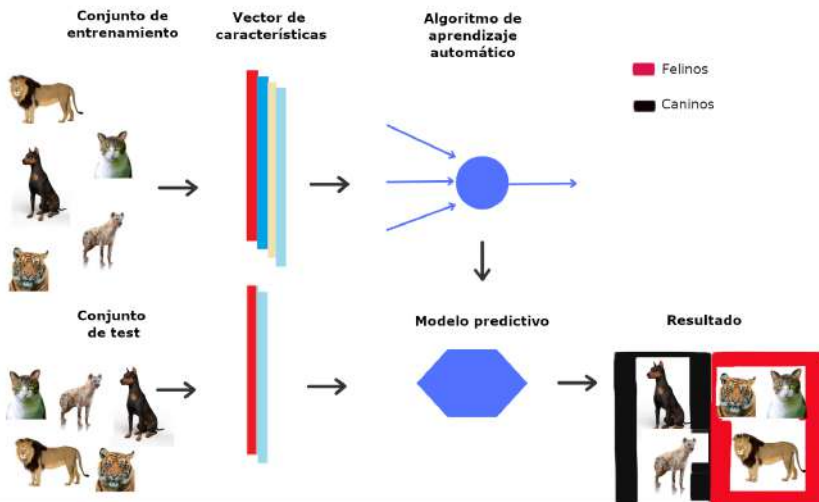


Figura 1.2: Función del aprendizaje automático no supervisado.

1.3. Aprendizaje profundo y las Redes Neuronales Artificiales (RNA)

A finales del siglo XIX un neurobiólogo Santiago Ramón y Cajal (1836-1921) determinó que el cerebro humano contenía millones de células cerebrales llamadas neuronas, y que cada una de estas neuronas aprendían y procesaban información muy en específico, es decir, por ejemplo, mientras unas neuronas se enfocaban en aprender el color de algún objeto, otras neuronas se enfocaban en detectar la forma del objeto y algunas otras más aprendían de su tamaño, etc. Al final, todas se pasaban la información obtenida y procesada para que el individuo humano comprendiera y le diera sentido a lo que sus ojos estaban viendo [6].

De la misma manera, un subconjunto dentro del ML, el Deep Learning (DL) o Aprendizaje Profundo (AP) en español, busca interconectar una red de neuronas

(llamada precisamente red neuronal) en donde cada neurona realiza diferentes procesamientos de esos datos y que la salida de cada neurona se interconecta con la entrada de otra capa de neuronas, es importante mencionar que cada entrada inicial x_n de la red neuronal, debe traer un peso w_n que indica "la importancia" de incidencia. En la Figura 1.3 se muestra un diagrama de la estructura de una red neuronal artificial.

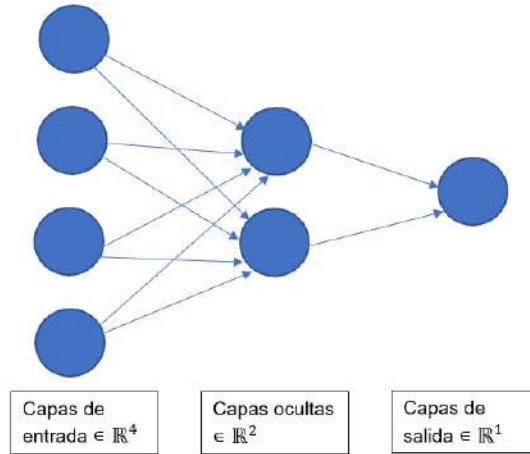


Figura 1.3: Diagrama de la estructura básica de las RNA (ejemplo con tres capas, cuatro neuronas de entrada, dos neuronas ocultas y una neurona de salida).

Ahora, si se representa la salida de la RN como y , la entrada como x y el peso como w la ecuación para representar este sistema que aparece en la Figura 1.4, quedaría de la siguiente manera:

$$y^{(i)} = w_1x_1^{(i)} + w_2x_2^{(i)} + w_3x_3^{(i)} \quad (1.2)$$

Donde: i es el número de la capa de la red neuronal y los subíndices representan las neuronas a las cuales están conectadas. A esta estructura se le conoce como Perceptrón multicapa.

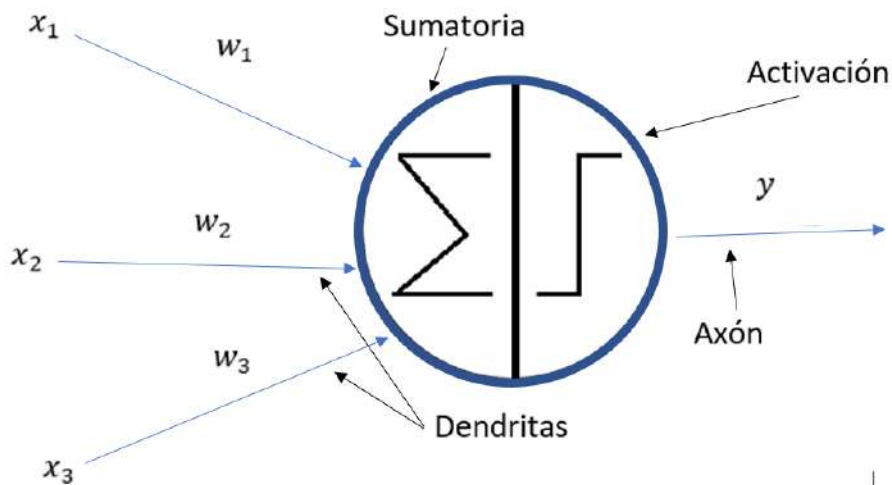


Figura 1.4: Diagrama con la estructura de una neurona con tres entradas y sus respectivos pesos (Perceptrón), se asemeja a una neurona biológica.

Es decir, los Perceptrones son clasificadores binarios lineales que usan un hiperplano para separar las dos clases. Se garantiza que el algoritmo de aprendizaje del Perceptrón obtenga un conjunto de pesos y sesgos que clasifique todas las entradas correctamente, siempre que exista un conjunto factible de pesos y sesgos.

Como se puede observar, estas RNA son una interpretación muy vaga de los axones de las redes neuronales biológicas que se encuentran en el cerebro humano, en la Figura 1.5 se observa un diagrama de una neurona biológica.

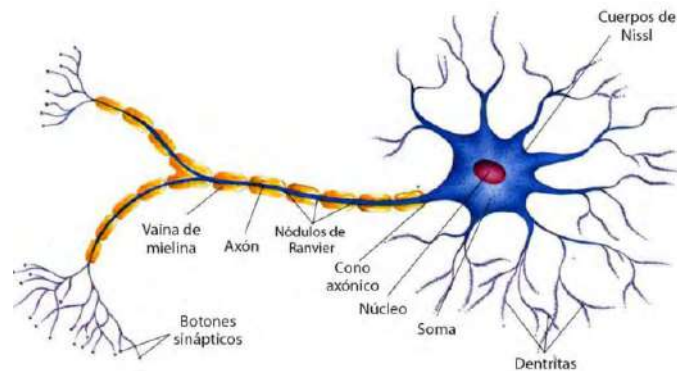


Figura 1.5: Diagrama de la estructura biológica de una neurona [5].

1.4. Sistemas de detección de carcinoma ductal invasivo vía imágenes histopatológicas

Los sistemas de detección de carcinoma ductal invasivo vía imágenes histopatológicas son métodos y técnicas utilizados para identificar y diagnosticar la presencia de carcinoma ductal invasivo en muestras de tejido obtenidas de biopsias o resecciones quirúrgicas.

La detección temprana y precisa del carcinoma ductal invasivo es fundamental para un diagnóstico y tratamiento adecuados, lo que mejora las posibilidades de supervivencia y pronóstico para las pacientes.

El proceso de detección de carcinoma ductal invasivo utilizando imágenes histopatológicas implica el examen microscópico de las muestras de tejido obtenidas mediante biopsias o resecciones quirúrgicas. Estas muestras son procesadas, fijadas, cortadas en secciones delgadas y teñidas con colorantes especiales. Luego, se examinan bajo el microscopio para identificar las características morfológicas y estructurales del carcinoma ductal invasivo.

Los sistemas de detección emplean técnicas de procesamiento de imágenes y análisis computacional para automatizar y asistir en la detección del carcinoma ductal invasivo. Estos sistemas utilizan algoritmos y modelos de aprendizaje automático, como Redes Neuronales Convolucionales (RNC), para analizar las imágenes histopatológicas y detectar patrones o características específicas asociadas con el carcinoma ductal invasivo. Estas características pueden incluir la presencia de células atípicas, la invasión de los tejidos circundantes y la arquitectura anormal de los conductos mamarios.

Al utilizar sistemas de detección de carcinoma ductal invasivo vía imágenes histopatológicas, se busca mejorar la precisión y la eficiencia en el diagnóstico, reducir la subjetividad y el error humano, y agilizar el proceso de detección. Estos sistemas pueden ser una herramienta valiosa para los patólogos y clínicos en la detección temprana, el pronóstico y la toma de decisiones clínicas relacionadas con el carcinoma ductal invasivo de mama.

Estos detectores se basan en algoritmos y modelos de aprendizaje automático para identificar y clasificar la presencia de CDI en muestras de tejido. A continuación, se mencionan algunos ejemplos de detectores de CDI.

DetectNet: Es un detector de objetos desarrollado por NVIDIA que utiliza RNC para la detección de diversas clases de objetos, incluido el carcinoma ductal invasivo. Utiliza la arquitectura de red Deep Convolutional Neural Network y puede ser entrenado en imágenes histopatológicas etiquetadas para identificar regiones que contienen CDI [29].

CAMELYON: Es un proyecto de investigación que se centra en el desarrollo de algoritmos para la detección automática de CDI en imágenes histopatológicas de muestras de tejido mamario. El proyecto ha organizado competencias y desafíos en los que se evalúan diferentes enfoques y algoritmos para la detección de CDI [30].

ACDC-Net: Es un detector de CDI basado en redes neuronales convolucionales profundas. Este modelo utiliza una red residual profunda para extraer características relevantes de las imágenes histopatológicas y realizar la detección y clasificación del CDI. Se ha demostrado que ACDC-Net logra un alto rendimiento en la detección precisa de CDI [31].

DeepSlide: Es un sistema de detección de CDI que utiliza el aprendizaje profundo y la segmentación de imágenes para identificar y cuantificar el CDI en imágenes histopatológicas digitales. El sistema utiliza una red neuronal convolucional para realizar la detección y segmentación de las regiones con CDI, lo que facilita la evaluación y el análisis de las muestras de tejido [32].

Estos son solo algunos ejemplos de detectores de carcinoma ductal invasivo basados en imágenes histopatológicas. Cada uno de ellos utiliza diferentes técnicas y enfoques de aprendizaje automático para la detección y clasificación del CDI. Es importante tener en cuenta que la investigación en este campo está en constante evolución, y se están desarrollando y mejorando continuamente nuevos algoritmos y modelos para mejorar la detección y el diagnóstico del carcinoma ductal invasivo.

1.5. Planteamiento del problema

El carcinoma ductal invasivo es el tipo más común de cáncer de mama y representa aproximadamente el 80 % de todos los casos. Detectar este tipo de cáncer de manera temprana es fundamental para un tratamiento exitoso y mejores resultados para los pacientes.

Las imágenes histopatológicas son uno de los métodos utilizados para diagnosticar el carcinoma ductal invasivo. Estas imágenes son capturadas a través de la microscopía

de tejidos y muestran la estructura y las características de las células cancerosas en una muestra de tejido mamario.

El uso de redes neuronales profundas, como las redes neuronales convolucionales (CNN), ha demostrado ser efectivo en el análisis de imágenes médicas. Estas redes neuronales pueden aprender a reconocer patrones y características específicas en las imágenes histopatológicas que son indicativas del carcinoma ductal invasivo.

El uso de redes neuronales profundas y técnicas de aprendizaje automático para crear un detector de carcinoma ductal invasivo a partir de imágenes histopatológicas ofrece un gran potencial para mejorar el diagnóstico y el tratamiento de esta enfermedad. Con un enfoque cuidadoso en la recopilación de datos, el entrenamiento del modelo y la validación clínica, se puede lograr un sistema de detección preciso y confiable.

1.6. Justificación

La creación de un detector de carcinoma ductal invasivo mediante el uso de redes neuronales profundas utilizando imágenes histopatológicas tiene varias justificaciones importantes, como mejorar la precisión del diagnóstico, ayudar en la toma de decisiones clínicas, optimizar los recursos y el tiempo, proporcionar apoyo en entornos con recursos limitados y la contribución a la investigación y el desarrollo de tratamientos, además, implementar el detector de CDI en un prototipo funcional portable, facilita la movilidad del dispositivo a cualquier área demográfica y reduce el costo del transporte.

En resumen, la creación de un detector de carcinoma ductal invasivo basado en redes neuronales profundas e imágenes histopatológicas tiene el potencial de mejorar la precisión del diagnóstico, agilizar el proceso de evaluación, optimizar los recursos médicos y contribuir al avance de la investigación y el tratamiento del cáncer de mama.

1.7. Alcance y objetivos del proyecto

1.7.1. Alcance

Comparación de varios modelos del estado del arte y aplicación del modelo con mayor exactitud en el menor tiempo de entrenamiento.

Desarrollo y ejecución de un prototipo funcional de detección para CDI en un sistema embebido.

1.7.2. Objetivo

Desarrollar un sistema de detección de Carcinoma Ductal Invasivo (CDI) vía imágenes histopatológicas e implementado en una plataforma embebida con un case impreso en 3D.

1.7.3. Objetivos específicos

- Obtener y adecuar bases de datos utilizadas en la literatura para la detección de Carcinoma Ductal Invasivo (CDI) vía imágenes histopatológicas.
- Estudio e implementación de los más representativos modelos neuronales de aprendizaje profundo del estado del arte.
- Basados en el conocimiento y experiencia obtenidos en el objetivo específico anterior, elección y de ser posible mejora de algún modelo neuronal del estado del arte.
- Estudio y elección de alguna plataforma embebida de inferencia neuronal adecuado para el modelo neuronal utilizado.
- Implementación de un sistema que permita hacer inferencias de CDI basado en la red neuronal desarrollada sobre la plataforma embebida.

1.8. Hipótesis

Existen tres aspectos importantes para la hipótesis de este trabajo.

1. Segmentación de imágenes con resoluciones muy grandes.
2. Comparación de modelos del estado del arte.
3. Diseño e implementación del detector en un sistema embebido y con una interfaz amigable.

A continuación, se describe cada una de ellas.

1. Segmentación de imágenes con resoluciones muy grandes

Segmentar imágenes con resoluciones gigantes es muy complicado por requerimientos de hardware, si se reduce la imagen se pierden detalles, entonces, al utilizar múltiples clasificadores adquiridos por parches en un enfoque de detección para identificar el CDI en imágenes histopatológicas, se espera que la combinación de las predicciones de los clasificadores a nivel de parche mejore la precisión de detección en comparación con el uso de un solo clasificador a nivel de una imagen completa. Las cualidades para usar múltiples clasificadores se presentan a continuación.

- Mayor capacidad de localización: el CDI puede presentar una distribución no uniforme en una imagen histopatológica. Al utilizar clasificadores adquiridos por parches, se puede lograr una mayor capacidad de localización al detectar regiones específicas de células cancerosas en lugar de depender de una única clasificación a nivel de imagen completa. Esto permite una detección más precisa y localizada.
- Captura de características locales: los clasificadores adquiridos por parches permiten capturar características locales y sutiles que pueden ser indicativas de la presencia del CDI. Al analizar los parches individuales, los clasificadores pueden enfocarse en detalles específicos y aprender características locales relevantes para la detección del cáncer de mama.

- **Diversidad de información:** se puede aprovechar la diversidad de información proporcionada por diferentes regiones de la imagen histopatológica. Cada parche puede contener información única y complementaria, lo que permite una mejor comprensión global de la presencia del CDI.
- **Reducción del efecto de ruido:** las imágenes histopatológicas pueden contener artefactos o regiones de ruido que pueden afectar la precisión de detección a nivel de imagen completa. Al utilizar clasificadores adquiridos por parches, se puede reducir el impacto del ruido al enfocarse en regiones más específicas y minimizar la influencia de áreas no relacionadas con el carcinoma ductal invasivo.
- **Mayor confianza en la detección:** al combinar las predicciones de múltiples clasificadores a nivel de parche, se puede obtener una mayor confianza en la detección. Y puede ayudar a evaluar de manera más confiable la presencia del cáncer de mama y reducir el riesgo de falsos positivos o falsos negativos.

2. Comparación de modelos del estado del arte

Comparar diferentes modelos de Keras Applications para el detector de CDI, se espera que algunos modelos más pequeños en uso de memoria, como MobileNet o EfficientNet, demuestren buenos resultados en términos de detección precisa y eficiencia computacional. Las ventajas son las siguientes.

- **Tamaño de memoria reducido:** los modelos de Keras Applications más pequeños, como MobileNet o EfficientNet, están diseñados para ser más livianos en términos de uso de memoria. Estos modelos logran un tamaño reducido mediante técnicas como compresión de parámetros, factorización de convoluciones y uso de bloques más simples. Al utilizar modelos más pequeños, se espera que se reduzca el uso de memoria durante la inferencia, lo que permite un despliegue más eficiente y escalable en entornos con recursos limitados como lo son los sistemas embebidos.
- **Buen rendimiento en detección:** aunque los modelos más pequeños pueden tener menos capacidad de representación que los modelos más grandes, todavía son capaces de capturar características relevantes para la detección precisa del CDI. Estos modelos han sido diseñados y entrenados de manera eficiente para optimizar el equilibrio entre rendimiento y tamaño. A través de técnicas como la utilización de capas convolucionales profundas y la aplicación de regularización, estos modelos pueden proporcionar buenos resultados de detección sin comprometer significativamente la precisión.
- **Eficiencia computacional:** los modelos más pequeños en uso de memoria suelen requerir menos tiempo de procesamiento durante la inferencia, lo que conduce a una mayor eficiencia computacional. Esto es especialmente importante en aplicaciones clínicas donde se requiere una detección en tiempo real o en grandes volúmenes de imágenes histopatológicas. Al utilizar modelos más pequeños, se puede lograr un procesamiento más rápido, permitiendo una detección más ágil y un flujo de trabajo clínico eficiente.
- **Generalización y adaptabilidad:** aunque los modelos más pequeños pueden tener una capacidad de representación ligeramente reducida en comparación

con los modelos más grandes, se espera que sigan siendo capaces de generalizar y adaptarse a diferentes conjuntos de datos de imágenes histopatológicas. Estos modelos han sido entrenados en grandes conjuntos de datos y han demostrado su capacidad para reconocer patrones y características relevantes en imágenes médicas. Su capacidad de adaptabilidad es esencial para garantizar que el detector pueda funcionar de manera efectiva en diferentes escenarios clínicos.

3. Diseño e implementación del detector en un sistema embebido y con una interfaz amigable

Al diseñar el detector con una interfaz sencilla y fácil de operar, se espera que los usuarios, como médicos y profesionales de la salud, puedan utilizarlo de manera efectiva y eficiente, lo que mejorará la detección temprana y la toma de decisiones clínicas. Las ventajas son las siguientes.

- Usabilidad: una interfaz sencilla y fácil de operar reduce la curva de aprendizaje y la carga cognitiva asociada al uso del detector. Los usuarios podrán navegar y utilizar las funciones del detector de manera intuitiva, sin requerir una formación técnica extensa. Esto facilitará la adopción y el uso regular del detector en la práctica clínica.
- Eficiencia en la interacción: una interfaz sencilla y bien diseñada minimiza la cantidad de pasos y acciones requeridos para completar una tarea. Los usuarios podrán realizar rápidamente las tareas de carga de imágenes, inicio del proceso de detección y visualización de resultados sin complicaciones innecesarias. Esto optimiza el flujo de trabajo y ahorra tiempo valioso a los profesionales de la salud.
- Claridad y legibilidad: una interfaz sencilla y fácil de operar se caracteriza por una presentación clara y una organización lógica de la información. Los resultados de detección deben ser presentados de manera comprensible, con indicadores claros sobre la presencia o ausencia de carcinoma ductal invasivo. Esto permite a los usuarios interpretar y comprender rápidamente los resultados, facilitando la toma de decisiones clínicas informadas.
- Eficiencia computacional: los sistemas embebidos están diseñados para operar con recursos limitados, como capacidad de procesamiento, memoria y energía. Al aplicar el detector de carcinoma ductal invasivo en un sistema embebido, se espera que se logre un uso eficiente de estos recursos, lo que permitirá una detección en tiempo real sin requerir una infraestructura computacional compleja y costosa.
- Portabilidad y accesibilidad: los sistemas embebidos son compactos, portátiles y pueden ser fácilmente integrados en dispositivos médicos o equipos clínicos existentes. Esto facilita el acceso a la detección de carcinoma ductal invasivo en diferentes entornos clínicos, como clínicas rurales, centros de atención primaria o áreas con recursos limitados. Al llevar el detector a través de un sistema embebido, se pueden superar barreras geográficas y mejorar el acceso a la detección temprana del carcinoma ductal invasivo.
- Tiempo de respuesta rápido: la detección temprana del carcinoma ductal invasivo es crucial para un tratamiento efectivo y mejores resultados clínicos. Al

aplicar el detector en un sistema embebido, se espera lograr una detección en tiempo real, lo que permitirá una respuesta rápida y oportuna a los casos positivos. Esto agilizará el proceso de diagnóstico y la toma de decisiones clínicas, mejorando la atención y la calidad de vida de los pacientes.

En resumen, la hipótesis plantea que el uso de múltiples clasificadores adquiridos por parches puede mejorar la precisión y la localización en la detección del carcinoma ductal invasivo en imágenes histopatológicas. La capacidad de capturar características locales y la diversidad de información.

Comparar diferentes modelos de Keras Applications, aquellos que sean más pequeños en uso de memoria, como MobileNet o EfficientNet, pueden ofrecer buenos resultados en términos de detección precisa y eficiencia computacional. Estos modelos más pequeños pueden proporcionar un equilibrio adecuado entre rendimiento y recursos computacionales, lo que los hace adecuados para su implementación en entornos con limitaciones de memoria y necesidades de detección eficientes.

Y plantea que una interfaz sencilla y fácil de operar, mejora la efectividad y la eficiencia en la detección temprana, al permitir a los usuarios utilizar el detector de manera intuitiva, comprender rápidamente los resultados y tomar decisiones clínicas informadas. Así como aplicarlo en un sistema embebido logra una detección eficiente y en tiempo real, permitiendo su implementación en entornos clínicos con recursos limitados y facilitando la detección temprana y el tratamiento oportuno del carcinoma ductal invasivo. La portabilidad, eficiencia computacional y adaptabilidad de los sistemas embebidos proporcionan una solución práctica y accesible para la detección de esta enfermedad.

1.9. Estado del arte

La IA dentro del área de la salud ofrece un diagnóstico de enfermedades en etapas tempranas y ayuda a predecir su avance e inclusive proponer un tratamiento adecuado a los pacientes. En esta sección se estudiarán otros trabajos hechos utilizando métodos y herramientas iguales a las realizadas en la presente tesis.

1.9.1. Keras

Keras son un conjunto de RN en formato para Python, diseñado por la empresa Google, y es eficiente para trabajar con entrenamientos en poco tiempo (aunque podría tardarse horas en algunos casos, según el procesamiento de los datos). Se ejecuta con bibliotecas de aprendizaje automático tales como TensorFlow y utilizando Redes Neuronales de Aprendizaje Profundo (RNAP) [20].

A continuación, se muestran algunos trabajos recopilados del estado del arte, como son RNC para detección de CDI con AlexNet”, ”Predictor CDI en cáncer de pecho con imágenes histopatológicas” ”Boosting Breast Cancer Detection Using Convolutional Neural Network (2021)”, en los tres trabajos, se muestran los resultados obtenidos y son expuestos como referencia.

1.9.2. Red Neuronal Convolutacional (RNC)

Ya que se ha comprendido la estructura de las RN, ahora se define lo que es una Red Neuronal Convolutacional (RNC), básicamente es necesario el Perceptrón como anteriormente se vio, pero ahora se agregarán una serie de capas convolucionales, es decir, capas de neuronas que realizan diferentes filtros para que al final, la información de entrada sea procesada más detalladamente y se tengan más características como patrones, formas, direcciones de líneas, diferentes tamaños de la entrada, etc. y tener un panorama más completo de los datos de análisis.

A continuación, se muestra la definición de la acción que realiza cada capa en la red neuronal:

Capas de entradas: al ser una RNA necesariamente se requiere ingresar una serie de información de entrada como pueden ser datos numéricos de ingresos de una empresa o la matriz de una imagen dada en píxeles.

Capas ocultas: Las capas ocultas de una red neuronal contienen unidades no observables. El valor de cada unidad oculta es alguna función de los predictores; la forma exacta de la función depende en parte del tipo de red. Los perceptrones multicapa pueden tener una o más capas ocultas.

Capas convolucionales: Estas capas se encargan de extraer características generales de la imagen, por ejemplo, extrae líneas y bordes manteniendo el tamaño original de la imagen de entrada. Internamente, lo que hace es recibir una matriz filtro o kernel de $L \times L$ (por lo general es 3×3) y se hace un barrido de cada píxel de la imagen superponiendo el kernel sobre el píxel inicial y se multiplica (de aquí el nombre de convolución) el píxel del kernel por el valor del píxel que se encuentra en la misma casilla superpuesta. Al final el resultado se coloca en otra casilla de una matriz de salida vacía llamada mapa de características.

En una imagen que está en escala de grises la salida solo tendrá una matriz de salida, pero en las imágenes a colores se extraerán tres matrices de salida con diferentes valores, debido a que estas imágenes tienen tres canales de entrada (R, G y B) y no solo uno como es el caso de las imágenes en "blanco y negro".

Cabe mencionar, que dependiendo el número de núcleos o filtros, será el número de imágenes procesadas que se obtendrán al final de este proceso y además, el movimiento de este proceso es de píxel a píxel, como ya se mencionó.

Capas de agrupación ver Figura 1.6: También conocidas como Pooling en inglés, al igual que las capas de convolución, este tipo de capas son utilizadas para extraer características, aunque serán más específicas y tienen dos objetivos principalmente:

- 1. Reducir el tamaño de la imagen.
- 2. Resaltar las características más importantes de las imágenes.

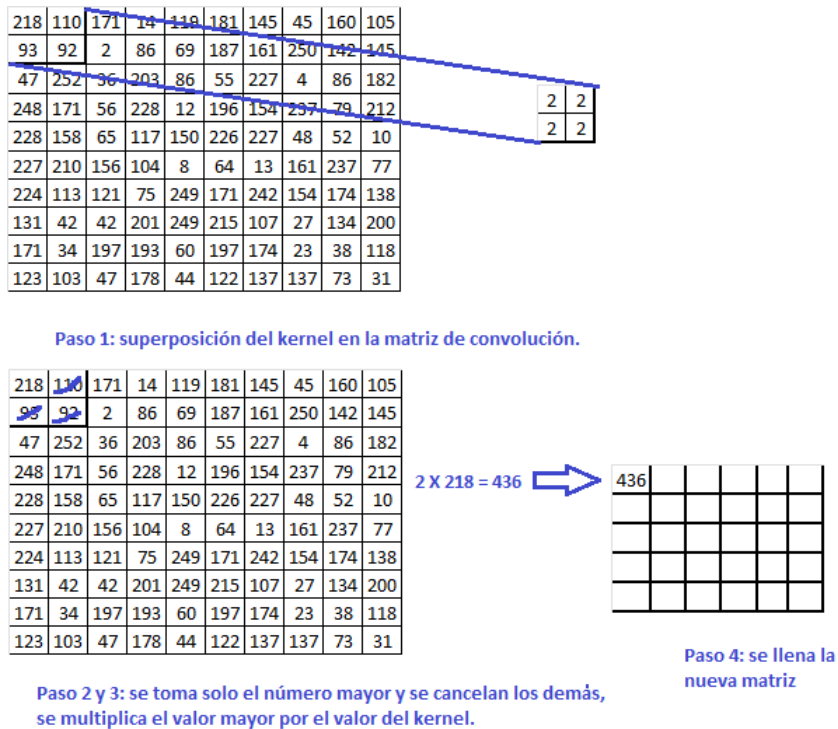


Figura 1.6: Pasos para aplicar la capa de agrupación con Maxpooling.

Es importante mencionar que, a la salida de las capas de convolución se conectan las capas de agrupación para que sean alimentadas por las primeras. Esto hace que a la matriz de salida de la capa de convolución se le apliquen más filtros o kernels.

Como ya se mencionó, la imagen se reducirá aplicando el filtro o kernel del pooling, en donde el kernel debe ser $L \times L$, un kernel clásico y común es el de agrupación máxima o Max Pooling y consta de un kernel de 2×2 .

Esto hará que el kernel se superponga en la matriz de convolución y realice la siguiente operación:

- Paso 1: se superpone el kernel en la matriz a la salida de la capa convolucional.
- Paso 2: se toma solo el valor mayor del conjunto del kernel superpuesto y se eliminan el resto.
- Paso 3: se multiplica el valor de las casillas superpuestas que corresponden y el resultado se coloca en una nueva matriz vacía.
- Paso 4. se recorre el kernel según el tamaño del filtro, a diferencia de la matriz de convolución que se recorre píxel a píxel.

Los pasos anteriores se muestran en la Figura 1.6. Nótese que la matriz de convolución se reduce en proporción según el valor del kernel.

1.9.3. Funciones de activación

La función de activación se encarga de devolver una salida a partir de un valor de entrada, normalmente el conjunto de valores de salida están en un rango determinado como $(0,1)$ o $(-1,1)$.

Se buscan funciones que las derivadas sean simples, para minimizar con ello el coste computacional.

Existen varias funciones de activación, las más comunes son:

- **Función de activación sigmoideal o sigmoide:** La función sigmoide transforma los valores introducidos a una escala (0,1), donde los valores altos tienen de manera asintótica a 1 y los valores muy bajos tienden de manera asintótica a 0. En la Figura 1.7 se muestra gráficamente la función sigmoide [28]. Características de la función sigmoide:

- Satura y desvanece al gradiente.
- Está acotada entre 0 y 1.
- Alenta la convergencia.
- No está centrada en el cero.
- Tiene un buen rendimiento en la última capa.

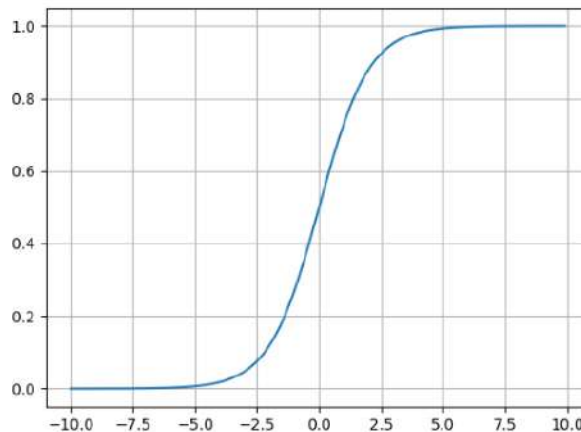


Figura 1.7: Función sigmoideal o sigmoide

Su ecuación es:

$$f(x) = \frac{1}{1 + e^{-x}} \quad (1.3)$$

- **Función de activación de tangente hiperbólica:** Transforma los valores introducidos a una escala (-1,1), donde los valores altos tienen de manera asintótica a 1 y los valores muy bajos tienden de manera asintótica a -1. En la Figura 1.8 se muestra la gráfica que corresponde a la función de tangente hiperbólica [19].

Características de la función tangente hiperbólica:

- Muy similar a la sigmoide.
- Satura y mata el gradiente.
- Lenta convergencia.

- Centrada en 0.
- Está acotada entre -1 y 1.
- Se utiliza para decidir entre una opción y la contraria.
- Buen desempeño en redes recurrentes.

Su ecuación es:

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (1.4)$$

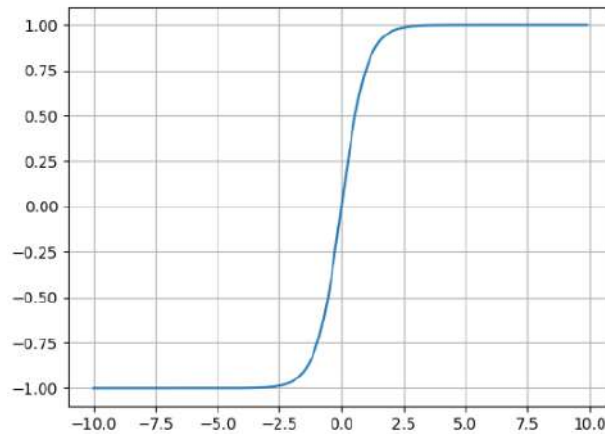


Figura 1.8: Función tangente hiperbólica

- **ReLU:** La función ReLU (del inglés Rectified Linear Unit y en español, unidad lineal rectificada) transforma los valores introducidos, anulando los valores negativos y dejando los positivos tal y como entran. En la Figura 1.9 se puede apreciar el comportamiento gráfico de la función ReLU.

Su ecuación es:

$$f(x) = \max(0,x) \begin{cases} 0 & \text{for } x < 0, \\ x & \text{for } x \geq 0 \end{cases} \quad (1.5)$$

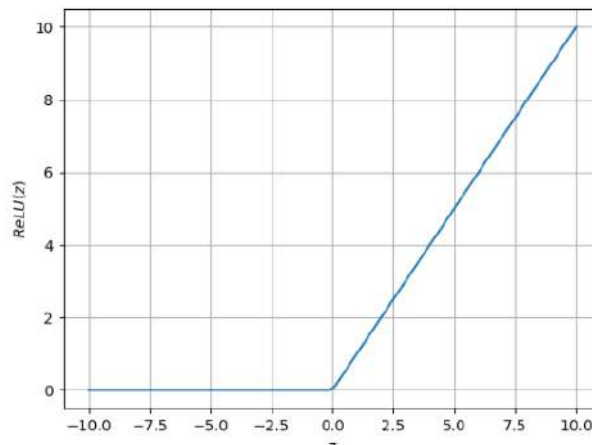


Figura 1.9: Función ReLU.

- **La función Softmax:** La función Softmax transforma las salidas a una representación en forma de probabilidades, de tal manera que la sumatoria de todas las probabilidades de las salidas de 1 [19].

Características de la función Softmax:

- Es muy útil para representar probabilidades.
- Está acotada entre 0 y 1.
- Tiene un buen rendimiento en las últimas capas.
- Se utiliza para normalizar tipo multiclase.
- Es muy diferenciable.

Su ecuación se representa a continuación y la Figura 1.10 muestra el diagrama de la red de Softmax.

$$f(z)_j = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}}$$

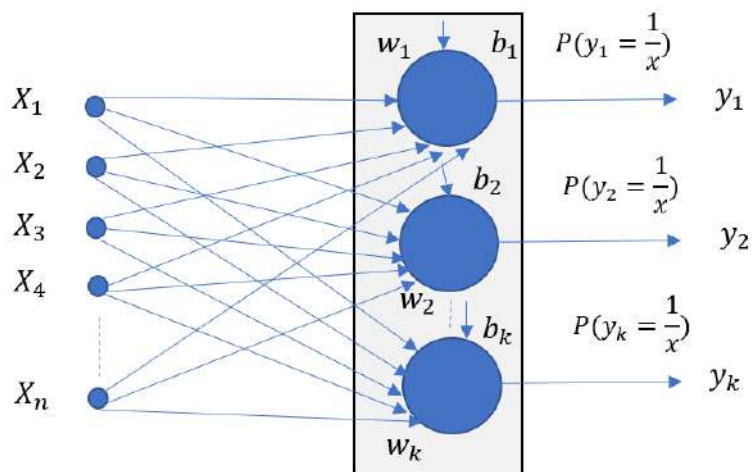


Figura 1.10: Función de activación Softmax.

- **Función de activación Leaky ReLU:** la función Leaky ReLU (Rectified Linear Unit) es una función de activación utilizada en redes neuronales para introducir no-linealidad en la salida de una neurona. A diferencia de la función ReLU estándar, que establece en cero todos los valores negativos de la entrada, la función Leaky ReLU permite un pequeño valor de pendiente (por ejemplo, 0.1) para los valores negativos [?]. Su ecuación es la 1.6 y su gráfica se presenta en la Figura 1.11.

$$\begin{aligned} f(x) &= \alpha x, & \text{si } x < 0 \\ f(x) &= x, & \text{si } x \geq 0 \end{aligned} \tag{1.7}$$

Dónde α , es un valor pequeño que se puede ajustar.

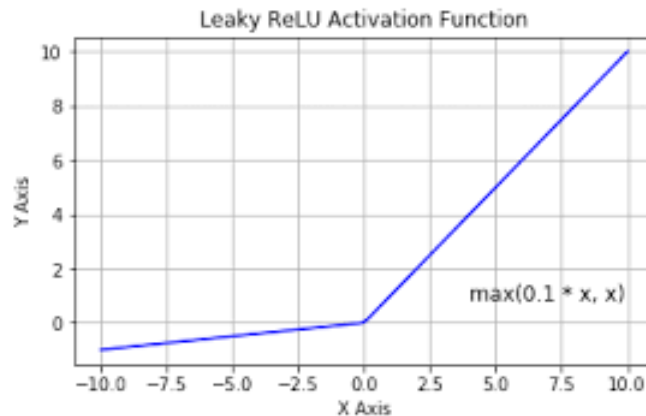


Figura 1.11: Función Leaky ReLU, con valor α de 0.1.

En la gráfica, la línea recta para valores negativos tiene una pendiente pequeña positiva, en este caso, 0.1. La línea recta para valores positivos es la misma que en la función ReLU estándar. Esta pendiente positiva para valores negativos ayuda a evitar el problema de "neuronas muertas" mejora el rendimiento de la red neuronal en ciertos tipos de problemas [24].

- **Función de activación ELU (Exponential Linear Units):** es una función de activación no lineal utilizada en redes neuronales. Fue propuesta en 2015 por Djork-Arne Clevert, Thomas Unterthiner y Sepp Hochreiter como una alternativa a la función ReLU y sus variantes.

La función ELU, también es similar a la función ReLU al igual que Leaky ReLU, ya que es una función lineal rectificadora que activa una neurona solo si la entrada es positiva, y se define como:

$$f(x) = \alpha e^{x-1}, \quad \text{si } x < 0$$

$$f(x) = x, \quad \text{si } x \geq 0 \tag{1.8}$$

Donde α es un parámetro que determina la pendiente de la función para valores negativos.

A diferencia de la función ReLU, la función ELU tiene una respuesta suave y continua tanto para valores positivos como para valores negativos, lo que ayuda a evitar el problema del "neurona muerta". Además, la función ELU puede converger más rápidamente que la función ReLU en ciertos tipos de redes neuronales [25].

La gráfica de la función ELU es una curva suave que tiene una pendiente cercana a cero para valores negativos y una pendiente de 1 para valores positivos, como se muestra a continuación en la Figura 1.12 [26].

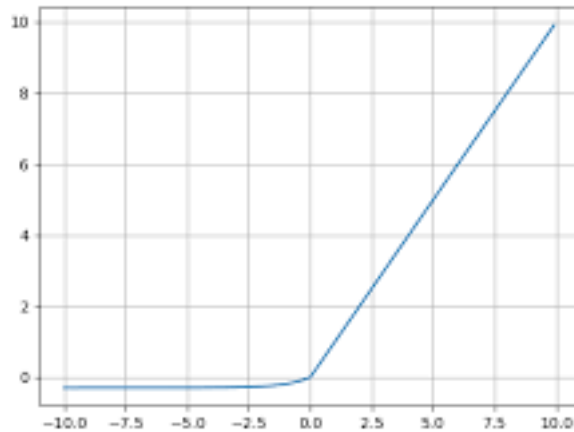


Figura 1.12: Función ELU.

Existen otras funciones menos comunes o utilizadas, a continuación se mencionan algunas de ellas.

- Función de activación Gaussian Error Linear Units (GELUs):** es una función de activación propuesta en 2018 por Dan Hendrycks y Kevin Gimpel. GELUs es una función no lineal que ha demostrado ser eficaz en redes neuronales profundas.

La función GELUs se define como una función sigmoidea escalada y sesgada por una distribución normal estándar. Matemáticamente, la función GELUs se define como:

$$f(x) = \phi x \tag{1.9}$$

donde ϕ es la función de distribución acumulativa de la distribución normal estándar. La función GELUs es diferenciable y su derivada también tiene una fórmula cerrada.

La idea detrás de la función GELUs es que combina la no linealidad de la función sigmoidea con la simetría y la escalabilidad de la distribución normal estándar. Se ha demostrado que esta función funciona bien en redes neuronales profundas, especialmente en tareas de clasificación y procesamiento de lenguaje natural.

La función GELUs ha sido implementada en algunas bibliotecas de aprendizaje profundo, como PyTorch y TensorFlow, lo que facilita su uso en la práctica [24].

La gráfica de la función de activación GELU es similar a la de una función sigmoidea, pero escalada y sesgada por una distribución normal estándar, se muestra en la Figura 1.13.

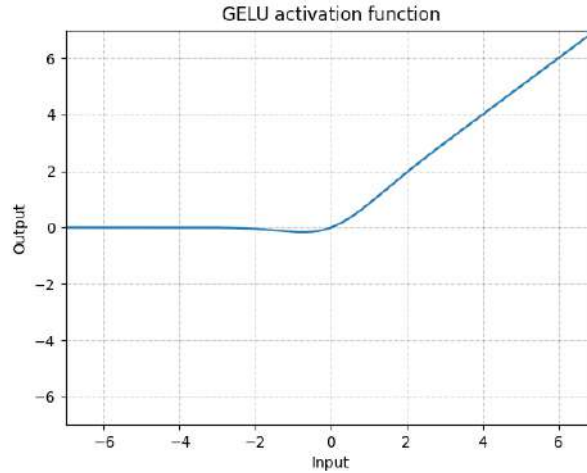


Figura 1.13: Función GELU.

- Función de activación SELU (Scaled Exponential Linear Unit):** es una función de activación propuesta en 2017 por Günter Klambauer, Thomas Unterthiner, Andreas Mayr y Sepp Hochreiter. SELU es una función no lineal que ha demostrado ser muy efectiva en redes neuronales profundas.

La función SELU se define como:

$$\begin{aligned}
 f(x) &= \lambda e^{x-1}, & \text{si } x < 0 \\
 f(x) &= x, & \text{si } x \geq 0
 \end{aligned}
 \tag{1.10}$$

donde λ es un parámetro ajustado automáticamente por la red neuronal. La función SELU es una función continua, diferenciable y no saturada, lo que la hace ideal para su uso en redes neuronales profundas.

La idea detrás de la función SELU es que, cuando se combina con un inicializador adecuado para los pesos de la red neuronal, la función puede garantizar que las activaciones se mantengan estables a medida que se propagan a través de las capas de la red, evitando así el problema del desvanecimiento o la explosión del gradiente [25].

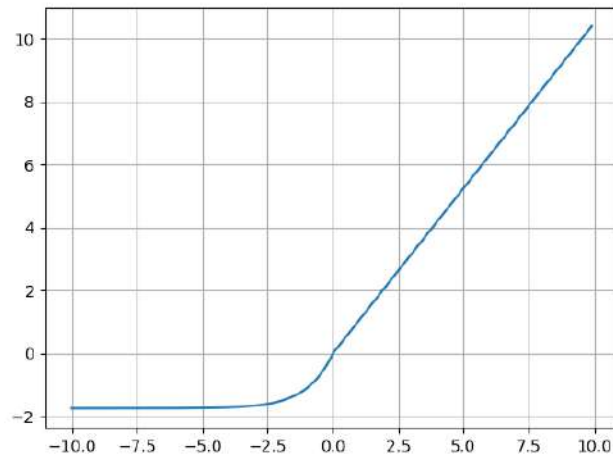


Figura 1.14: Función SELU.

La función SELU es una función no lineal que se aproxima a la función identidad para valores de entrada cercanos a cero, pero se vuelve no lineal para valores más grandes o más pequeños. Además, la función SELU es una función escalada y sesgada de tal manera que las activaciones se mantienen estables a medida que se propagan a través de las capas de una red neuronal profunda (Figura 1.14).

- **Función de activación Maxout:** es una función de activación propuesta en 2013 por Ian Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville y Yoshua Bengio.

La función Maxout se define como:

$$f(x) = \max(w_1x + b_1, w_2x + b_2) \quad (1.11)$$

donde w_1, w_2, b_1 y b_2 son parámetros ajustables de la red neuronal.

La idea detrás de la función Maxout es que, en lugar de utilizar una función de activación no lineal como la ReLU o la sigmoideal, se utiliza una función lineal con múltiples ramas. Cada rama de la función Maxout es una función lineal de la entrada x , que se combina mediante la operación \max para producir la salida final. Cada rama puede aprender una función diferente de la entrada, lo que la hace muy útil en situaciones donde las características relevantes para una tarea pueden ser diferentes en diferentes partes del espacio de entrada.

La función Maxout ha demostrado ser muy efectiva en redes neuronales profundas, especialmente en tareas de visión por computadora y reconocimiento de voz. También es muy útil en situaciones donde se requiere la identificación de características importantes en diferentes partes del espacio de entrada.

La función Maxout ha sido implementada en algunas bibliotecas de aprendizaje profundo, como TensorFlow y Keras, lo que facilita su uso en la práctica.

Nota. La función \max es una función matemática que toma dos o más números como entrada y devuelve el número más grande de ellos. Formalmente, si a y b son dos números, entonces $\max(a, b)$ es el número más grande entre a y b .

Por ejemplo, $\max(3, 6)$ devolvería 6, ya que 6 es el número más grande entre 3 y 6.

En el contexto de las redes neuronales, la función \max se utiliza a menudo en la función de activación Maxout, como se mencionó anteriormente. En la función Maxout, la función \max se utiliza para combinar dos o más ramas lineales, eligiendo el valor más grande entre ellas en cada punto de entrada.

La función Maxout es una función que no tiene una gráfica simple como otras funciones de activación, como la ReLU o la sigmoide. En cambio, la función Maxout se define como la operación \max de dos o más ramas lineales, lo que significa que su forma depende del número de ramas y de los parámetros específicos de cada una.

En general, se puede decir que la función Maxout es una función no lineal que se puede aproximar a una función lineal en algunos puntos de entrada y a una función constante en otros puntos de entrada. Además, debido a su capacidad para aprender múltiples funciones lineales, la función Maxout es capaz de modelar relaciones más complejas entre las entradas y las salidas de una red neuronal.

- **Función de activación Swish:** es una función de activación propuesta por los investigadores de Google, Prajit Ramachandran, Barret Zoph y Quoc V. Le, en un artículo publicado en 2017.

La función de activación Swish se define como:

$$f(x) = x \cdot \text{sigmoid}(\beta x) \quad (1.12)$$

donde x es la entrada a la función, y β es un parámetro ajustable de la red neuronal.

La función de activación Swish es una combinación de la función identidad x y la función sigmoide $\text{sigmoid}(x)$. La función Swish es una función no lineal y diferenciable que tiene una forma de “S” suave y monótona. La función Swish también tiene una ventaja sobre otras funciones de activación populares, como la ReLU, en términos de suavidad y continuidad, lo que puede ayudar a prevenir problemas de gradiente explotando o desvanecimiento en redes neuronales profundas.

La función Swish ha demostrado ser muy efectiva en redes neuronales profundas, especialmente en tareas de visión por computadora y procesamiento del lenguaje natural. Además, la función Swish puede ser fácilmente implementada en bibliotecas de aprendizaje profundo como TensorFlow y PyTorch.

En general, la función de activación Swish es una buena opción para aquellos que buscan una función de activación no lineal suave y diferenciable que puede ayudar a mejorar el rendimiento de una red neuronal profunda.

En la gráfica, a medida que la entrada a la función de activación aumenta, la salida también aumenta, pero con una curva suave y monótona. En comparación con la función de activación ReLU, que tiene una línea recta en el rango de valores positivos y una pendiente abrupta en el rango de valores negativos, la función de activación Swish se considera más suave y continua [26] (Figura 1.15).

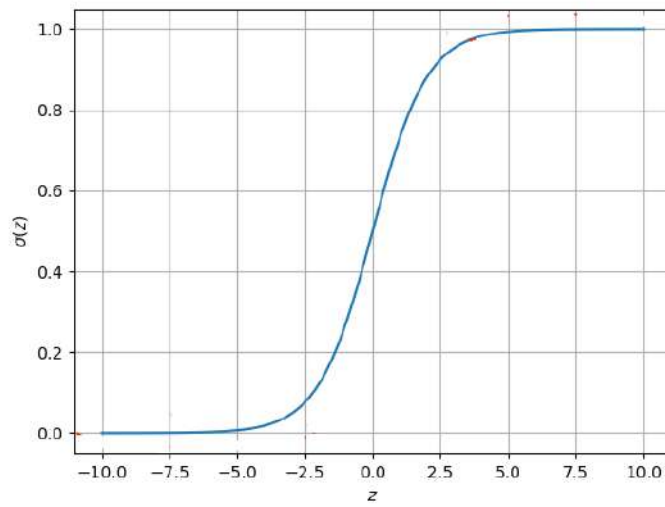


Figura 1.15: Función Swish.

A continuación, en la tabla se muestra un resumen de las funciones de activación mencionadas anteriormente.

Tabla 1.1: Resumen de las funciones de activación

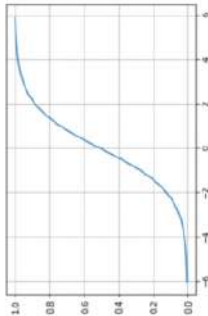
Función de activación	Ecuación	Características	Gráfica	Páginas
Sigmoide	$f(x) = \frac{1}{1 + e^{-x}}$	<p>Se utiliza con frecuencia en problemas de clasificación binaria, donde la salida deseada es una de dos categorías posibles, como "sí." "no", "verdadero." "falso", "1." "0", etc.</p> <p>En general, la función de activación sigmoide es una opción útil y popular para las redes neuronales que se utilizan en problemas de clasificación binaria.</p>		18 y 19

Tabla 1.1: Resumen de las funciones de activación

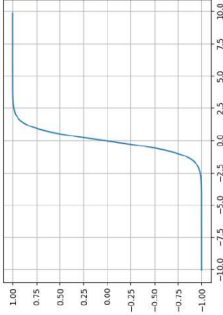
Función de activación	Ecuación	Características	Gráfica	Páginas
Tangente hiperbólica	$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	<p>Se utiliza a menudo en problemas de clasificación binaria, en los que se desea predecir si una entrada pertenece a una de dos clases posibles. También se utiliza en problemas de regresión, en los que se desea predecir un valor numérico continuo.</p> <p>Generalmente, se usa en lugar de la función de activación sigmoide porque tiene una salida centrada en cero y es simétrica alrededor de ese punto, por lo que puede ayudar a reducir el problema de gradiente desvaneciente. Sin embargo, a diferencia de la función sigmoide, tiene una salida en el rango $[-1, 1]$, lo que puede ser problemático si las entradas no están normalizadas.</p>		19 y 20

Tabla 1.1: Resumen de las funciones de activación

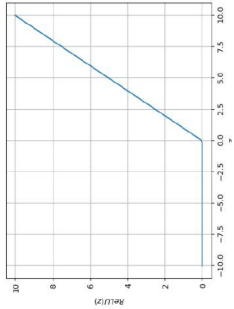
Función de activación	Ecuación	Características	Gráfica	Páginas
ReLU	$\max(0,x) \begin{cases} 0 & \text{for } x < 0, \\ x & \text{for } x \geq 0 \end{cases}$	<p>Se utiliza comúnmente en redes neuronales artificiales como función de activación para las capas ocultas. Esta función tiene un rango de valores entre 0 y infinito y es una función no lineal que puede ayudar a la red neuronal a aprender relaciones no lineales en los datos de entrada. También, se usa en problemas de clasificación y regresión en los que se desea predecir una salida positiva o en problemas de detección de objetos y reconocimiento de patrones.</p>		20

Tabla 1.1: Resumen de las funciones de activación

Función de activación	Ecuación	Características	Gráfica	Páginas
Softmax	$f(z)_j = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}}$	<p>Se utiliza en redes neuronales para la clasificación multiclase, y se utiliza en la última capa de la red neuronal para producir una distribución de probabilidades para cada clase posible. Es una función no lineal que toma como entrada un vector de valores y los normaliza para producir un vector de probabilidades que suman 1.</p>	No aplica	20 y 21

Tabla 1.1: Resumen de las funciones de activación

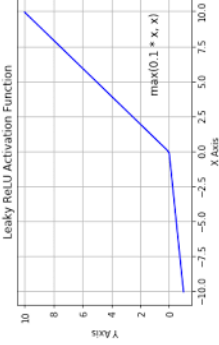
Función de activación	Ecuación	Características	Gráfica	Páginas
Leaky ReLU	$f(x) = \alpha x, \quad \text{si } x < 0$ $f(x) = x, \quad \text{si } x \geq 0$	<p>Se utiliza en problemas de clasificación y regresión. Sin embargo, puede ser especialmente útil cuando se entrena una red neuronal profunda con muchas capas ocultas, donde la "muerte" de ReLU puede ser un problema. Por lo tanto, Leaky ReLU puede ayudar a evitar este problema al permitir que los valores negativos también tengan una pequeña contribución a la activación.</p>		21 y 22

Tabla 1.1: Resumen de las funciones de activación

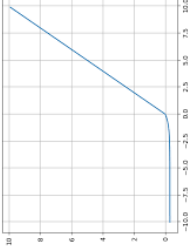
Función de activación	Ecuación	Características	Gráfica	Páginas
ELU	$f(x) = \alpha e^{x-1}, \quad \text{si } x < 0$ $f(x) = x, \quad \text{si } x \geq 0$	<p>Es similar a la función Leaky ReLU, pero en lugar de tener una pendiente lineal para los valores negativos, tiene una función exponencial. Esto significa que la función ELU tiene una curva suave en lugar de una discontinuidad en $x = 0$, lo que puede ayudar a la red neuronal a converger más rápidamente y generalizar mejor.</p> <p>Se utiliza en problemas de clasificación y regresión, pero puede ser especialmente útil cuando se entrena una red neuronal profunda con muchas capas ocultas, ya que puede mejorar la tasa de convergencia y precisión de la red neuronal y evitar la "muerte" de ReLU.</p>		22 y 23

Tabla 1.1: Resumen de las funciones de activación

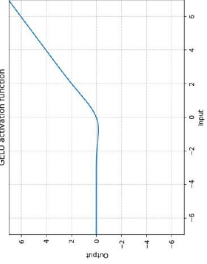
Función de activación	Ecuación	Características	Gráfica	Páginas
GELUs	$f(x) = \phi x$	<p>Se utiliza en redes neuronales profundas para mejorar la precisión y reducir el problema de la muerte de ReLU, y se utiliza en las capas ocultas para problemas de clasificación y regresión en conjuntos de datos grandes y complejos.</p> <p>Tiene una curva suave tanto para los valores positivos como negativos de entrada, lo que puede ayudar a la red neuronal a converger más rápidamente.</p>		23 y 24

Tabla 1.1: Resumen de las funciones de activación

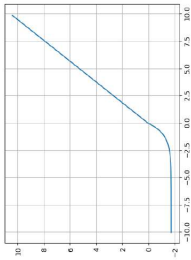
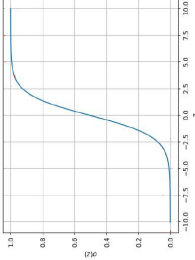
Función de activación	Ecuación	Características	Gráfica	Páginas
SELU	$f(x) = \lambda e^{x-1}, \quad \text{si } x < 0$ $f(x) = x, \quad \text{si } x \geq 0$	<p>Tiene una propiedad adicional que la hace especialmente útil en redes neuronales profundas: la propiedad de autonormalización.</p> <p>Es decir, que ayuda a la red neuronal a mantener la media y la desviación estándar de la activación de cada capa oculta cercana a 0 y 1, respectivamente, durante el entrenamiento.</p> <p>Se utiliza en redes neuronales profundas para mejorar la precisión y la estabilidad de la red neuronal, y se utiliza en capas ocultas para problemas de clasificación y regresión en grandes conjuntos de datos especialmente en tareas de reconocimiento de lenguaje natural y visión por computadora.</p>		24 y 25

Tabla 1.1: Resumen de las funciones de activación

Función de activación	Ecuación	Características	Gráfica	Páginas
Maxout	$f(x) = \max(w_1x + b_1, w_2x + b_2)$	<p>Se utiliza en tareas de clasificación y regresión, especialmente en problemas de reconocimiento de imágenes y procesamiento de señales. Se ha demostrado que la función Maxout puede mejorar la precisión de la red neuronal en comparación con otras funciones de activación en algunas tareas.</p>	<p>No aplica</p>	<p>25 y 26</p>
Swish	$f(x) = x \cdot \text{sigmoid}(\beta x)$	<p>Se utiliza en redes neuronales como una alternativa efectiva a la función de activación ReLU en capas ocultas, y se utiliza en tareas de clasificación y regresión, especialmente en tareas de reconocimiento de imágenes y procesamiento del lenguaje natural.</p>		<p>26 y 27</p>

En el caso de este proyecto, la función de activación Softmax se utilizó en el detector de carcinoma ductal invasivo para asignar probabilidades a diferentes clases, incluyendo la presencia o ausencia de CDI. Proporciona una forma de interpretar las probabilidades de clasificación, es compatible con el entrenamiento y la optimización del modelo, y garantiza una salida numéricamente estable y normalizada.

1.9.4. Aplanamiento o Flatten

En esta capa se crea una nueva capa oculta, que se conecta casi al final de la red neuronal, tiene la tarea de reducir el número de dimensiones de las imágenes de 3D a 1D (queda un vector que se conecta a la red neuronal).

Entonces, a esta nueva capa oculta, se le aplica la función de Softmax que conecta contra la capa de salida final que tendrá la cantidad de neuronas correspondientes con las clases que se clasifican. Si se clasifican perros y gatos, serán 2 neuronas. Si clasificamos coches, aviones o barcos serán 3 neuronas, etcétera.

Las salidas al momento del entrenamiento tendrán el formato conocido como “one hot encoding” en el que para perros y gatos será: [1,0] y [0,1], para coches, aviones o barcos será [1,0,0]; [0,1,0]; [0,0,1].

Y la función de Softmax se encarga de pasar a probabilidad (entre 0 y 1) a las neuronas de salida. Por ejemplo, una salida [0,2 0,8] indica 20 % probabilidades de que sea perro y 80 % de que sea gato, según este ejemplo [17].

Gráficamente, se observa como en la Figura 1.16.

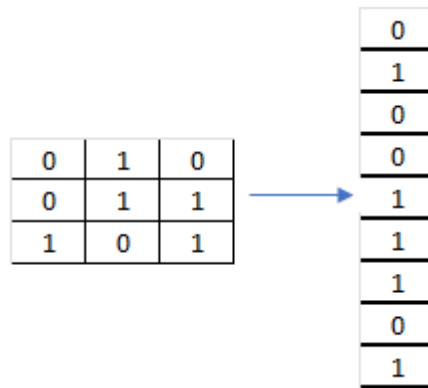


Figura 1.16: Aplanamiento (Flatten).

1.9.5. Retropropagación o Backpropagation

La retropropagación es un método útil para propagar el error en la capa de salida hacia atrás, de modo que los gradientes en las capas anteriores se puedan calcular fácilmente mediante la regla de la cadena de las derivadas.

Considerando un ejemplo de entrenamiento y se trabaja con la retropropagación, teniendo en cuenta la estructura de la red XOR. Sea la entrada $x = [x_1, x_2]^T$ y la clase correspondiente sea y . Entonces, la función de costo para el registro único se convierte en la siguiente ecuación:

$$C = -y \log z_3 - (1 - y) \log(1 - z_3) \quad (1.13)$$

Así:

$$\frac{\partial(C)}{\partial(w_1)} = \frac{dC}{dz_3}$$

$$\frac{\partial(C)}{\partial(z_3)} = \frac{(z_3 - y)}{z_3(1 - z_3)}$$

Si ahora: $Z_3 = \frac{1}{(1+e^{-z_3})}$

$$\frac{dz_3}{di_3} = z_3(1 - z_3)$$

$$\frac{dC}{di_3} = \frac{dC}{dz_3} \frac{dz_3}{di_3} = \frac{(z_3 - y)}{z_3(1 - z_3)} z_3(1 - z_3) = (z_3 - y)$$

Entonces, se puede observar que la derivada de la función de costo con respecto a la entrada neta en la capa final es nada más que el error en la estimación de la salida $(z_3 - y)$ [5].

1.9.6. RNC para detección de CDI con AlexNet

Dentro de la detección de CDI utilizando RNC se encuentran:

Deep learning for digital pathology image analysis: A comprehensive tutorial with selected use cases (2016), de los autores Andrew Janowczyk, Anant Madabhushi y se trata de un algoritmo utilizando el modelo AlexNet modificando un par de capas (capa 7 y capa 8) combinándolas con capas de activación ReLU y realizando variaciones a las capas de entrada, el trabajo realiza el entrenamiento priorizando tareas de características como muestra la Figura 1.17, extrayendo peculiaridades para segmentación detección y clasificación [7].

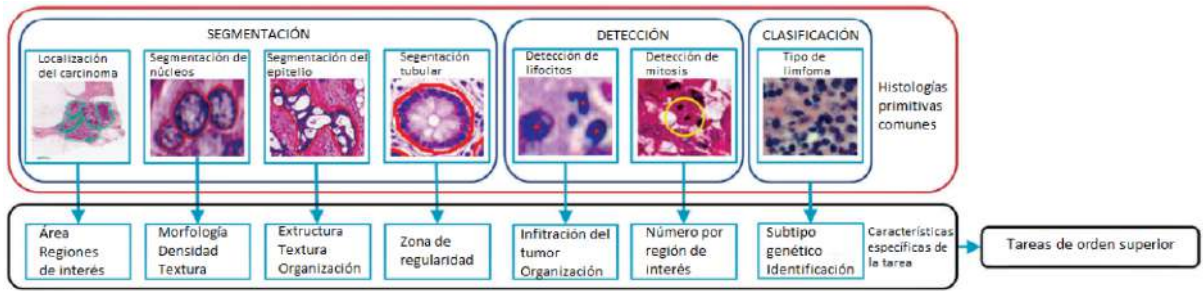


Figura 1.17: Orden de tareas de prioridad para el entrenamiento del modelo [7].

Para el entrenamiento de cada modelo en las diferentes tareas, se utilizaron diferentes datasets, los cuales, se muestran en la Tabla 1.2, especificando la cantidad y magnificación de imágenes, además de dónde fueron extraídas.

Tabla 1.2: Resultados de entrenamiento de las variaciones del modelo AlexNet.

Tarea	Dataset utilizado
Segmentación de núcleos	141 2,000 x 2,000 @40x ROIs de estrógeno receptor positivo (ER+) cáncer de pecho (BCa), contenedores de subconjuntos de 12,000 con un núcleo mostrado.
Segmentación del epitelio	42 1,000 x 1,000 @20x ROIs de ER + BCa, contiene 1735 regiones.
Segmentación tubular	85 775 x 522 @40x ROIs de cáncer colorectal , contiene 795 túbulos delineados 162 en toda la diapositiva @40x de BCa pacientes.
Detección de linfocitos para CDI	100 100 x 100 @40x BCa ROIs, contiene 3, 064 linfocitos centrales.
Detección de mitosis	311 2,000 x 2,000 @40x ROI from 12 BCa, contiene 550 de mitosis centrales.
Clasificación de subtipos de linfomas	374 1,388 x 1,040 @40x of 3 subtipos de linfoma (CLL, FL, MCL).

Al modificar el tamaño del modelo se probaron varios enfoques dando los resultados de la Tabla 1.3:

Así, la Tabla 1.3 muestra los resultados observando que cuando el modelo cambia el tamaño de la entrada, los resultados del puntaje F1-score y la precisión son los más altos comparados a los demás métodos utilizados y con el modelo sin modificación que es extraído del artículo original [7].

Tabla 1.3: Tabla de dataset utilizados según la tarea prioritaria.

Método	Puntaje F1	Precisión
AlexNet, cambio de tamaño	0.764	0.8468
AlexNet, cambio de tamaño omitiendo neuronas	0.757	0.842
AlexNet, recorte	0.753	0.841
AlexNet, recorte y rotaciones adicionales	0.755	0.836
AlexNet sin modificaciones (extraído del artículo original)	0.718	0.842

1.9.7. Predictor CDI en cáncer de pecho con imágenes histopatológicas

Para poder comparar de mejor manera el resultado de la precisión y el puntaje F1 será necesario hablar del trabajo titulado "Predicting IDC in Breast Cancer Histology Images" del autor Paul Mooney, que es un trabajo que entrena con un modelo propio; así, el modelo diseñado por Mooney, es el que se muestra en la Figura 1.18.

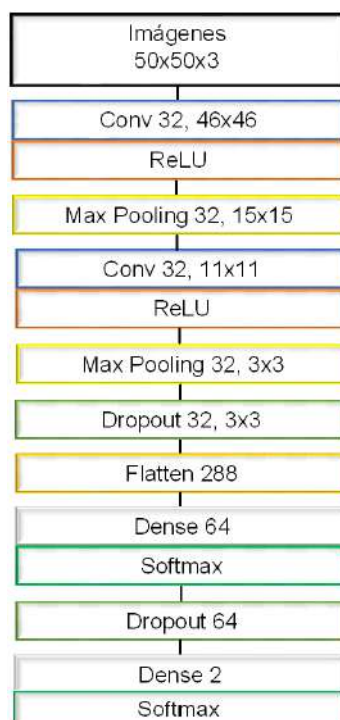


Figura 1.18: Diseño de modelo propio utilizado en artículo de Mooney.

También, el algoritmo de Mooney está entrenado con el mismo dataset de la presente tesis (Histopatology Cancer Images), sin embargo, el trabajo de Mooney solo entrena en el modelo propio y también utiliza las librerías de keras. Los parámetros de entrada como el Learning Rate y el tamaño de las imágenes siempre son constantes.

El resultado de la precisión y del puntaje F1 se muestran en la Tabla 1.4 lo interesante de este trabajo es la extracción de las coordenadas de las secciones con CDI en cada una de las imágenes y su gráfica binarizada.

Después de doce épocas, el resultado de la precisión es de **0.7054**.

Tabla 1.4: Resumen de la RNC del modelo 1.

	Precisión	Sensitividad	Puntaje F1	Soporte
IDC(-)	0.74	0.82	0.78	566
IDC(+)	0.79	0.70	0.74	544
avg/total	0.76	0.76	0.76	1110

Después de realizar el trabajo anterior, Paul Mooney realizó algunos ajustes para mejorar los resultados, los principales cambios que realizó fueron:

- Balanceo del dataset utilizado.
- Automatizar los parámetros para la reducción del error.
- Automatizar la selección del Learning Rate.

Después de estos ajustes, se entrenó y evaluó el mismo modelo dando los resultados de la Tabla 1.5. Los resultados fueron mejores, esta vez con una precisión total de **0.824430558355** [8].

Tabla 1.5: Mejora del entrenamiento y evaluación de los datos, utilizando el mismo modelo propio.

	Precisión	Sensitividad	Puntaje F1	Soporte
IDC(-)	0.90	0.73	0.81	4961
IDC(+)	0.77	0.92	0.84	4961
avg/total	0.84	0.82	0.82	9922

1.9.8. RNC comparando varios modelos

De igual manera, un trabajo que ha hecho una comparación de modelos (tres modelos propios de los autores), es el titulado "Boosting Breast Cancer Detection Using Convolutional Neural Network (2021)" de la Universidad Kakaha de Arabia Saudita [9].

Este trabajo utiliza el modelo propio 1 con varios kernel, con dropout y ReLU como función de activación, a excepción de la capa de salida que utiliza Softmax. El resumen de la RNC del modelo 1 se presenta en la Figura 1.19.

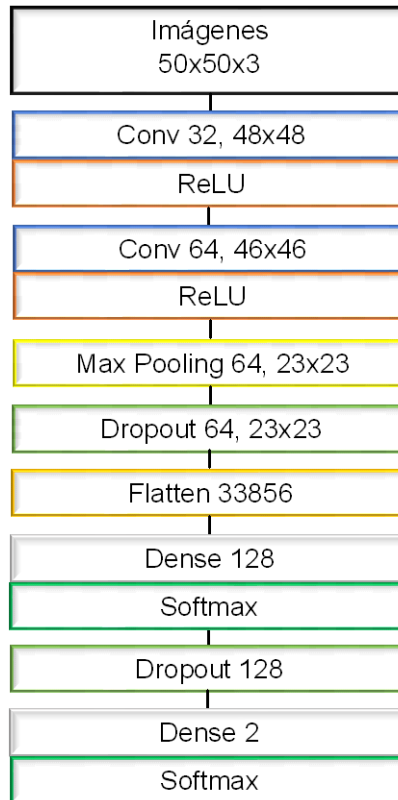


Figura 1.19: Resumen de la RNC del modelo 1.

Este modelo tiene los siguientes resultados de la Tabla 1.6:

Tabla 1.6: Resultados del modelo 1.

	Precisión	Sensitividad	Puntaje F1	Soporte	Exactitud
IDC(-)	0.59	0.80	0.68	596	0.59
IDC(+)	0.60	0.36	0.45	514	0.59

Para aumentar los valores anteriores, en el modelo 2 se triplicaron las capas convolucionales, dando como resultado el resumen de la Figura 1.20:

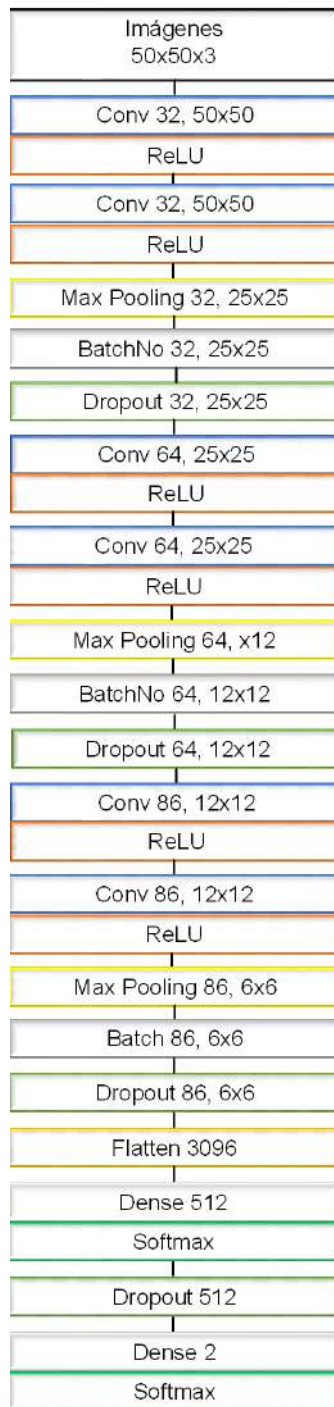


Figura 1.20: Resumen de la RCN del modelo 2 con el triple de capas convolucionales.

Con el modelo 2 los resultados mejoran de la siguiente manera según la Tabla 1.7:

Tabla 1.7: Resultados del modelo 2.

	Precisión	Sensitividad	Puntaje F1	Soporte	Exactitud
IDC(-)	0.72	0.91	0.81	596	0.76
IDC(+)	0.85	0.60	0.70	514	0.76

El modelo 3 es menos profundo que el modelo 1 y modelo 2 con cinco capas para detección de CDI y es el modelo que mejores resultados arroja en precisión y F1-score, tal como se muestra en la Tabla 1.8 Y el resumen del modelo se muestra en la Figura 1.21.

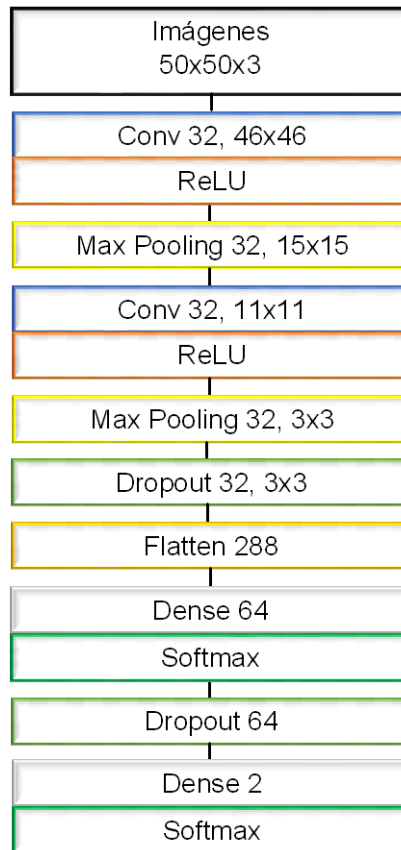


Figura 1.21: Resumen de la RCN del modelo 3.

Tabla 1.8: Resultados del modelo 3.

	Precisión	Sensitividad	Puntaje F1	Soporte	Exactitud
IDC(-)	0.82	0.92	0.91	596	0.87
IDC(+)	0.86	0.76	0.85	514	0.87

La estructura de la tesis es la siguiente.

Capítulo 2 Metodología. Describe detalladamente los métodos y técnicas utilizados en la investigación, como la microscopia virtual, la descripción de los modelos entrenados y comparados, así como la versión de TensorFlow utilizada y su sistema operativo, además de la configuración realizada para trabajar con la Raspberry PI4.

Capítulo 3 Implementación del algoritmo en el sistema embebido. Muestra los pasos a seguir para la instalación del sistema operativo y el modelo CAD utilizado para el diseño de la carcasa, también muestra como se realizó el manejo de datos para los parámetros y el desarrollo de la elección del sistema embebido. Por último, muestra la interfaz y uso para introducir las imágenes al detector.

Capítulo 4 Resultados. Presenta los hallazgos y los resultados obtenidos a partir del análisis de los datos recopilados, así como las tablas de las características de los equipos utilizados para el entrenamiento y los valores de tiempo de entrenamiento, métricas (precisión, sensibilidad, exactitud, etc.) y las imágenes obtenidas despues del entrenamiento, así como la comparativa entre las imágenes de los cortes biológicos, imágenes con máscaras y las imágenes con diagnóstico de CDI.

Capítulo 5 Conclusiones. En esta sección se muestra la gráfica de los modelos con los resultados de su exactitud respecto a su tiempo de entrenamiento, así como la descripción de si la hipótesis fue comprobada o no.

Capítulo 2

Metodología

En este capítulo, se describe detalladamente la metodología utilizada en el desarrollo del sistema de detección de carcinoma ductal invasivo vía imágenes histopatológicas. Se expondrán los pasos y procesos seguidos para alcanzar los objetivos planteados, incluyendo la adquisición de datos, el preprocesamiento, la extracción de características y la implementación del modelo de detección seleccionado. Además, se presentarán las herramientas y tecnologías utilizadas, así como las métricas de evaluación empleadas para medir el rendimiento del sistema.

2.1. Selección de librerías y datos

Existen varias librerías de aprendizaje automático que podrían ser útiles para crear un detector de carcinoma ductal invasivo. Algunas opciones populares son TensorFlow, Keras, PyTorch y scikit-learn. Estas librerías ofrecen una amplia gama de herramientas y algoritmos de aprendizaje automático que puedes utilizar para entrenar y evaluar modelos. Para este trabajo elegimos trabajar con TensorFlow.

2.2. TensorFlow

TensorFlow de Google es una librería de fuente abierta (Open Source) la cual tiene como principal objetivo es el entrenamiento profundo.

Utiliza gráficos de flujo de datos computacionales para representar una arquitectura de red neuronal complicada. Los nodos en el gráfico denotan cálculos matemáticos, también llamados operaciones, mientras que los bordes denotan los tensores de datos transferidos entre ellos. Además, los gradientes relevantes se almacenan en cada nodo del gráfico computacional y, durante la retropropagación, estos se combinan para obtener los gradientes con respecto a cada peso. Los tensores son matrices de datos multidimensionales que utiliza TensorFlow [4].

Hay varias razones por las que TensorFlow es una opción popular y ampliamente utilizada para desarrollar modelos de aprendizaje automático. Aquí hay algunas ventajas clave de utilizar TensorFlow:

1. Flexibilidad y escalabilidad: TensorFlow proporciona una interfaz flexible que

permite a los desarrolladores construir y entrenar una amplia variedad de modelos de aprendizaje automático. Puedes crear modelos desde redes neuronales básicas hasta arquitecturas de aprendizaje profundo más complejas. Además, TensorFlow es altamente escalable y puede aprovechar recursos de hardware como CPUs, GPUs o incluso TPUs (Tensor Processing Units) para acelerar el entrenamiento y la inferencia.

Ecosistema y comunidad activa: TensorFlow cuenta con un ecosistema sólido que incluye una amplia gama de herramientas y recursos complementarios. Esto incluye bibliotecas adicionales como TensorFlow Extended (TFX) para el desarrollo de flujos de trabajo de aprendizaje automático a gran escala, TensorFlow.js para el desarrollo de modelos en JavaScript y TensorFlow Lite para la implementación en dispositivos móviles y embebidos. Además, TensorFlow cuenta con una comunidad activa de desarrolladores que comparten conocimientos, tutoriales y contribuyen con nuevas mejoras y modelos pre-entrenados [33].

Soporte multiplataforma: TensorFlow es compatible con diferentes plataformas, lo que permite desarrollar y desplegar modelos en una variedad de entornos. Puede ejecutar modelos en computadoras personales, servidores, dispositivos móviles, en la nube e incluso en dispositivos IoT (Internet of Things). Esto facilita la implementación y adaptación de los modelos a diferentes escenarios y requisitos.

Integración con otras herramientas y librerías: TensorFlow se integra bien con otras bibliotecas y herramientas populares de aprendizaje automático y análisis de datos. Se puede combinar TensorFlow con librerías como Keras, que proporciona una interfaz de alto nivel para construir y entrenar redes neuronales de manera más sencilla. También se puede utilizar TensorFlow con herramientas de visualización de datos como TensorBoard, que facilita el seguimiento y la visualización del rendimiento de modelos.

En resumen, TensorFlow se destaca por su flexibilidad, escalabilidad, soporte multiplataforma y su comunidad activa. Estas características hacen de TensorFlow una opción sólida para el desarrollo de modelos de aprendizaje automático en aplicaciones de detección de características de imágenes [34].

2.3. Base de datos o Dataset

Para obtener un conjunto de datos que contenga imágenes o datos relevantes para el carcinoma ductal invasivo. Se debe buscar en bases de datos médicas, como el Cancer Imaging Archive (TCIA) o Kaggle, o solicitar acceso a conjuntos de datos de investigación específicos.

Una vez que se obtengan los datos, se deben realizar tareas de preprocesamiento, como normalización, limpieza de datos y etiquetado. A continuación, se menciona el procedimiento seguido en esta tesis.

Para el entrenamiento de los modelos es necesario un dataset en alta definición, con tamaño de entrada de, al menos, 50 x 50 píxeles y una cantidad de imágenes superior a las 2000.

Un conjunto de imágenes de alta definición que se puede descargar libremente de

plataformas como Kaggle llamado “Histopatology Cancer Images”, está compuesto de 179 carpetas numeradas (ID), cada carpeta corresponde a un paciente diagnosticado con CDI por expertos patólogos, y usan una técnica excelente para mantener la alta definición llamada “Microscopía Virtual” (esta técnica es también utilizada para fotografías en el espacio) ver Figura 2.1. Consiste en tomar imágenes de rebanadas (Slice) de biopsias y en vez de tomar una sola fotografía se toman muchas imágenes pequeñas de alta definición de ese slice escaneadas a 40x. De ahí se extrajeron 277.524 parches de tamaño 50 x 50 (198.738 IDC negativos y 78.786 IDC positivos).

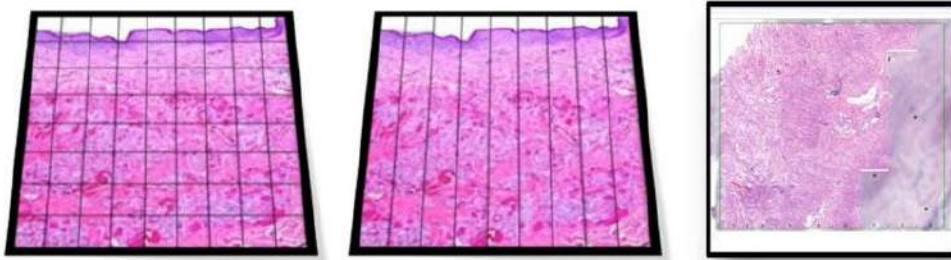


Figura 2.1: Ejemplo del uso de la microscopía virtual en imágenes histopatológicas [12].

El nombre de archivo de cada parche tiene el formato: $u\ xX\ yY\ classC.png$. Por ejemplo 10253 idx5 x1351 y1101 class0.png. Donde u es el ID del paciente (10253 idx5), X es la coordenada x de donde se recortó este parche, Y es la coordenada y de donde se recortó este parche y C indica la clase, donde 0 es sin CDI y 1 es con CDI.

Entonces, las carpetas de estos 179 pacientes contienen esas fotografías de alta definición que según sea el tamaño de la muestra de tejido extraída contendrá diferentes cantidades de imágenes por paciente.

2.4. Modelos neuronales

La aplicación de los modelos pre entrenados de Keras, cumplen una función muy importante en esta tesis, ya que se pueden utilizar para la predicción, la extracción de características y el ajuste fino.

Keras Applications tiene actualmente 33 modelos; sin embargo, debido a que el tiempo de entrenamiento y el coste de procesamiento es muy alto, solo se toman en cuenta 19, por ejemplo Xception, VGG16, VGG19, variaciones de ResNet y EfficientNet entre otros; las características de cada uno de los modelos se presentan en la Tabla 2.1.

Tabla 2.1: Características principales de los modelos disponibles de Aplicaciones de Keras utilizados en esta tesis, extraído de la página oficial de Keras[13]

Modelo	Tamaño (MB)	Top 1 Exactitud	Top 5 Exactitud	Parámetros	Profundidad	Tiempo (ms) por inferencia en CPU	Tiempo (ms) por inferencia en GPU
VGG16	528	71.3 %	90.1 %	138.4M	16	69.5	4.2
VGG19	549	71.3 %	90.0 %	143.7M	19	4.4	4.4
ResNet50	98	74.9 %	92.1 %	25.6M	107	58.2	4.6
ResNet50V2	98	76.0 %	93.0 %	25.6M	103	45.6	4.4
ResNet101	171	76.4 %	92.8 %	44.7M	209	89.6	5.2
ResNet101V2	171	77.2 %	93.8 %	44.7M	205	72.7	5.4
ResNet152	232	76.6 %	93.1 %	60.4M	311	127.4	6.5
ResNet152V2	232	78.0 %	94.2 %	60.4M	307	107.5	6.6
InceptionV3	92	77.9 %	93.7 %	23.9M	189	42.2	6.9
Inception-ResNetV2	215	80.3 %	95.3 %	55.9M	449	130.2	10.0
MobileNet	16	70.4 %	89.5 %	4.3M	55	22.6	3.4
MobileNetV2	14	71.3 %	90.1 %	3.5M	105	25.9	3.8
DenseNet121	33	75.0 %	92.3 %	8.1M	242	77.1	5.4
DenseNet169	57	76.2 %	93.2 %	14.3M	338	96.4	6.3
DenseNet201	80	77.3 %	93.6 %	20.2M	402	127.2	6.7
NasNetMobile	23	74.4 %	91.9 %	5.3M	389	27.0	6.7
EfficientNet B0	29	77.1 %	93.3 %	5.3M	132	46.0	4.9
EfficientNet B2	36	80.1 %	94.9 %	9.2M	186	80.8	6.5
EfficientNet V2B0	29	78.7 %	94.3 %	7.2M	-	-	-

El modelo se entrena con 60 % entrenamiento (Train), 20 % testeo (Test) y 20 % validación (Validation) y se propone entrenar el modelo con al menos 200 épocas.

Debido a que, un objetivo del entrenamiento de una RNA, es asignar valores iniciales en el vector de pesos para que al evaluar los ejemplos de entrenamiento, el error sea mínimo.

Así, la prueba con otros patrones que no se han utilizado para el entrenamiento, se pueden utilizar para el testeo; y al realizar varias épocas, se puede llegar a un buen resultado. Aunque, no necesariamente tiene que entrenar la cantidad total de las épocas programadas, ya que, el algoritmo es capaz de detenerse al no encontrar mejoras en el proceso.

2.4.1. Diferentes arquitecturas de los modelos

Las arquitecturas de los modelos son utilizadas no solo para clasificación, sino que realizando modificaciones menores pueden ser usadas también para detección, localización y segmentación.

Además, existen versiones pre entrenadas de cada una de estas redes que permiten a la comunidad transferir el aprendizaje o ajustar los modelos. Excepto LeNet, casi todos los modelos de CNN han ganado el concurso ImageNet para la clasificación de mil clases [5].

A continuación, se muestran todos los modelos y sus características seleccionados para realizar el entrenamiento, y posteriormente, su comparación.

2.4.2. LeNet

La primera RNC exitosa fue diseñada por Yann LeCunn en 1990 para clasificar dígitos escritos a mano, funciona con imágenes de entradas con un tamaño de 32×32 que pasan por una primera capa convolucional que produce seis mapas de características de 28×28 . De estos seis mapas de características se extrae un sub muestreo para tener seis imágenes de salida de 14×14 . El sub muestreo se puede considerar como una operación de agrupación. La segunda capa de convolución tiene 16 mapas de características de 10×10 y se tiene un segundo sub muestreo que reduce cada mapa de característica a 5×5 . Posteriormente, esto es seguido por dos capas completamente conectadas de 120 y 84 unidades respectivamente, seguidas de la capa de salida de diez clases correspondientes a diez dígitos. La Figura 2.2 muestra la arquitectura de LeNet5, una versión más reciente a LeNet.

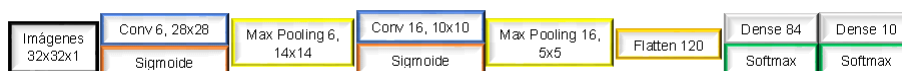


Figura 2.2: Diagrama de la arquitectura de LeNet5.

La agrupación a través del sub muestreo toma parches de vecindario de 2×2 , sumando los valores de intensidad de cuatro píxeles. La suma se escala mediante un peso entrenable con un sesgo, y luego se alimenta a través de una función de activación sigmoidea. [18]

Para este trabajo, el modelo de LeNet solo se consideró como una representación histórica de las RNA.

2.4.3. AlexNet

La arquitectura AlexNet RNC fue desarrollada por Alex Krizhevsky, Ilya Sutskever y Geoffrey Hinton en 2012 para ganar el ImageNet 2012 (ImageNet Large-Scale Visual Recognition Challenge). Fue la primera vez que una arquitectura RNC venció a otros métodos por un amplio margen. Su red logró una tasa de error del 15,4 por ciento en sus cinco predicciones principales en comparación con la tasa de error del 26,2 por ciento para la segunda mejor entrada.

En la Figura 2.3 se observa el diagrama de la arquitectura de AlexNet. Esta RNC consiste en cinco capas convolucionales, capa de maxpooling, una capa de dropout y tres capas totalmente conectadas, además de la capa de entrada y salida de mil clases. El tamaño de la entrada de la red es de $224 \times 224 \times 3$. La primera capa convolucional produce 96 mapas de características correspondientes a 96 kernel como filtros de tamaño $11 \times 11 \times 3$ con pasos de cuatro píxeles.

La segunda capa convolucional produce 256 mapas de características que corresponden a los kernel de $5 \times 5 \times 48$. Las primeras dos capas convolucionales están seguidas de una capa de maxpooling y las tres capas convolucionales posteriores contienen una capa de maxpooling intermedia. La quinta capa convolucional sigue una maxpooling y seguidas de dos capas completamente conectadas de 4096 unidades, y finalmente la capa de salida está compuesta por mil clases de Softmax.

Las características clave de AlexNet son:

- Dropout se utilizan para reducir el ajuste en el modelo.
- ReLU es utilizada como no linealidad.
- Se utiliza agrupación superpuesta en lugar de agrupación no superpuesta.
- El tamaño del conjunto de datos se incrementa mediante técnicas de aumento de datos, como traducción de imágenes, reflejos horizontales y extracciones de parches.

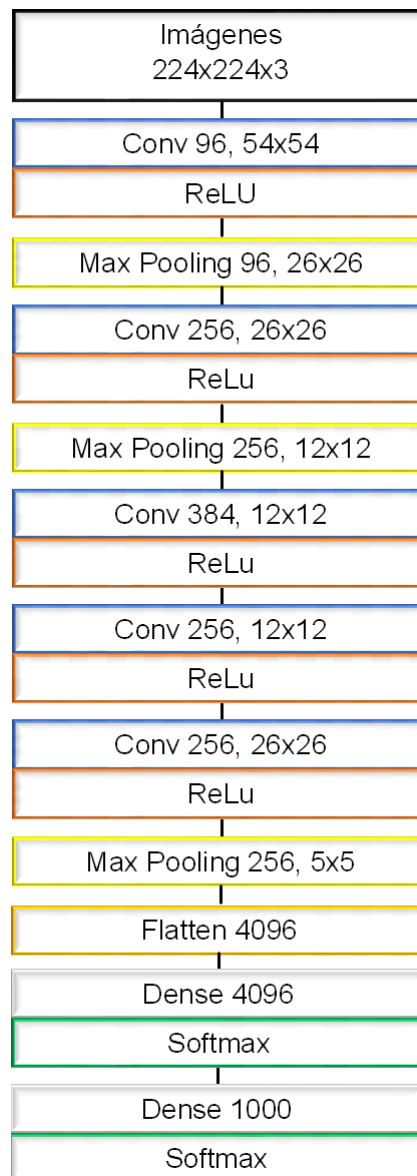


Figura 2.3: Diagrama de la arquitectura de AlexNet.

En la subsección 1.8.2, Mooney realizó su investigación con este modelo, en esta tesis se busca comparar los resultados de otros modelos con AlexNet, por lo tanto, el modelo de AlexNet no está entrenado en este trabajo.

2.4.4. VGG16

El grupo VGG en 2014 fue subcampeón en ILSVRC-2014, una competencia con una arquitectura de 16 capas llamada VGG16. Utiliza una arquitectura profunda pero simple que ha ganado mucha popularidad desde entonces. El artículo perteneciente a la red VGG se titula "Very Deep Convolutional Networks for Large-Scale Image Recognition" está escrito por Karen Simonyan y Andrew Zisserman.

En lugar de usar un tamaño de filtro de kernel grande para la convolución, la arquitectura VGG16 usó filtros de 3×3 y siguió con activaciones de ReLU y agrupación máxima con un campo receptivo de 2×2 .

El razonamiento de los inventores fue que el uso de dos capas de convoluciones de 3×3 es equivalente a tener una convolución de 5×5 conservando las ventajas de un tamaño de filtro de kernel más pequeño; es decir, realizar una reducción en el número de parámetros y realizar una mayor no linealidad debido a dos pares de convolución-ReLU en lugar de uno. Una propiedad especial de esta red es que a medida que las dimensiones espaciales del volumen de entrada se reducen debido a la convolución y la agrupación máxima, la cantidad de mapas de características aumenta debido al aumento en la cantidad de filtros a medida que nos adentramos en la red.

Las imágenes de entrada de la red son de tamaño 224×224 . Las dos primeras capas convolucionales producen 64 mapas de características, cada una seguida por Max pooling. Los filtros de convolución son de tamaño 3×3 con un paso de 1 en 1. El max pooling es de tamaño 2×2 con paso de 2 para toda la red. La tercera y cuarta capa convolucional producen 128 mapas de características, cada una seguida por una capa de Max pooling. La red VGG16 está representada en la Figura 2.4, con la capa de dropout completamente conectada y toda se activa con ReLU [5].

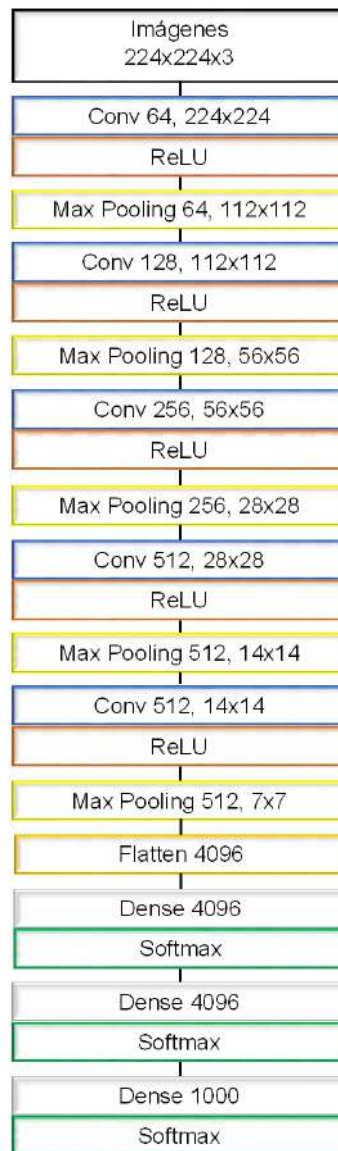


Figura 2.4: Arquitectura del modelo VGG16.

2.4.5. ResNet

ResNet es una RNC de 152 capas de profundidad de Microsoft ganadora de la competencia ILSVRC 2015 con una tasa de error de solo 3.6 por ciento, la cual se percibe como una mejor tasa de error que la humana entre un 10 a 15 por ciento. El trabajo fue hecho por Kaiming He, Xiangyu Zhang, Shaoqing Ren y Jian Sun titulado "Deep Residual Learning for Image Recognition".

Además de ser profundo, ResNet es una de las primeras en implementar una idea única de bloque residual. Después de cada serie de operaciones de convolución-ReLUs-convolución, la entrada a la operación se retroalimenta a la salida de la operación. En los métodos tradicionales, mientras hacemos la convolución y otras transformaciones, intentamos ajustar un mapeo subyacente a los datos originales para resolver la tarea de clasificación.

Sin embargo, con el concepto de bloque residual de ResNet, tratamos de aprender

un mapeo residual y no un mapeo directo de la entrada a la salida. Otras versiones de ResNet son, ResNet50, ResNet50V2, ResNet101, ResNet101V2, entre otras.

A continuación en la Figura 2.5 se muestra la arquitectura de la red ResNet.

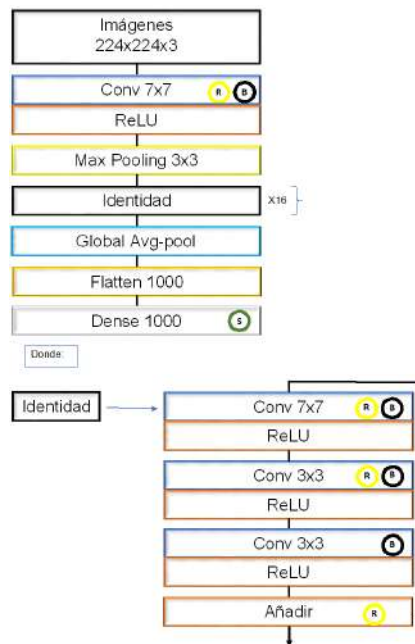


Figura 2.5: Arquitectura del modelo ResNet.

2.4.6. Inception (GoogleNet)

Creada por Google, Inception V3 ganó el concurso ILSVRC 2014 y 2016 y en su versión Inception V4 pudo obtener una tasa de error de 3.08 por ciento, produciendo así el error más bajo en el conjunto de datos de clasificación de ImageNet, pero hay algunos puntos en los que se pueden realizar mejoras para mejorar la precisión y disminuir la complejidad del modelo. El modelo se representa en la siguiente Figura 2.6.

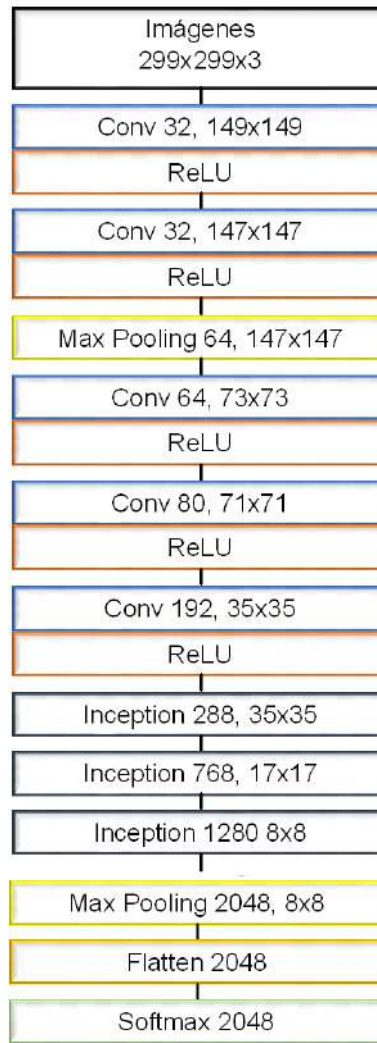


Figura 2.6: Arquitectura del modelo Inception V3 o GoogleNet.

2.4.7. Xception

Una de las arquitecturas de redes de neuronas profundas que ha demostrado funcionar bastante bien para el reconocimiento de imágenes es Xception. Xception fue presentada por Google en la principal conferencia sobre visión por computador y reconocimiento de patrones (CVPR, por sus siglas en inglés), en el año 2017.

La arquitectura Xception se inspira en un resultado anterior, denominado Inception. Inception es resultado de un importante trabajo de ingeniería para lograr capas más anchas y menos profundas, introduciendo en una misma capa varios operadores de convolución y luego concatenando sus resultados, algo parecido a la Figura 2.7:

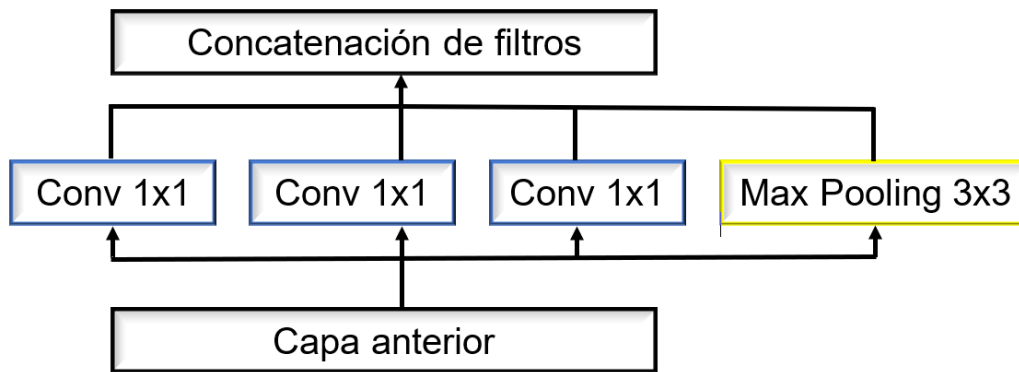


Figura 2.7: Arquitectura del modelo Xception.

De este modo, se consiguen modelos que son capaces de tener un alto poder predictivo, pero más eficientes que redes más profundas. Este modelo ha sido revisado en varias ocasiones con algunas mejoras, dando lugar a diferentes versiones de la arquitectura.

Xception, en cambio, introduce un elemento novedoso en la arquitectura: las convoluciones separables en profundidad. Este operador realiza una convolución de forma separada en cada canal de la entrada, y luego aplica una convolución 1×1 para proyectar los canales generados como salida en un nuevo espacio de canales [10].

2.4.8. EfficientNet

EfficientNet es una arquitectura de red neuronal convolucional y un método de escalado que escala uniformemente todas las dimensiones de profundidad/anchura/resolución mediante un coeficiente compuesto. A diferencia de la práctica convencional que escala arbitrariamente estos factores, el método de escala EfficientNet escala uniformemente el ancho, la profundidad y la resolución de la red con un conjunto de coeficientes de escala fijos. Por ejemplo, si queremos utilizar 2^N veces más recursos computacionales, entonces simplemente podemos aumentar la profundidad de la red, el ancho por el tamaño de la imagen, donde son coeficientes constantes determinados por una búsqueda de cuadrícula pequeña en el modelo pequeño original. EfficientNet utiliza un coeficiente compuesto para escalar uniformemente el ancho, la profundidad y la resolución de la red de una manera basada en principios.

El método de escalado compuesto se justifica por la intuición de que si la imagen de entrada es más grande, entonces la red necesita más capas para aumentar el campo receptivo y más canales para capturar patrones más detallados en la imagen más grande.

La red base EfficientNet-B0 se basa en los bloques residuales de cuello de botella invertidos de MobileNetV2, además de los bloques de compresión y excitación.

EfficientNets también se transfiere bien y logra una precisión de vanguardia en CIFAR-100 (91,7%), Flowers (98,8%) y otros 3 conjuntos de datos de aprendizaje de transferencia, con un orden de magnitud menos de parámetros.

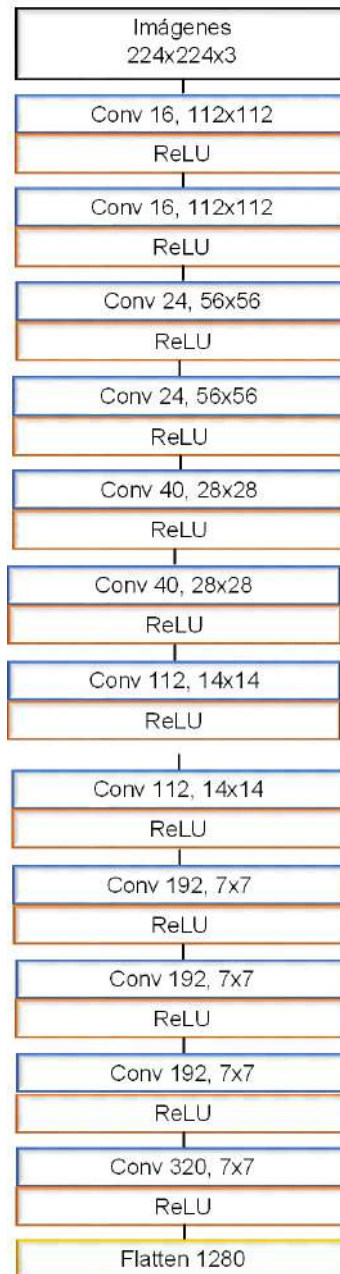


Figura 2.8: Arquitectura de EfficientNet.

2.4.9. MobileNet

MobileNet es el primer modelo de visión por computadora de TensorFlow diseñado para usarse en aplicaciones móviles. MobileNet utiliza circunvoluciones separables en profundidad. Reduce significativamente el número de parámetros en comparación con la red con convoluciones regulares con la misma profundidad en las redes. Esto da como resultado redes neuronales profundas ligeras [14].

MobileNet es una arquitectura general y se puede utilizar para múltiples casos de uso. Dependiendo del caso de uso, puede usar diferentes tamaños de capa de entrada y diferentes factores de ancho. Esto permite que diferentes modelos de ancho reduzcan el número de adiciones múltiples y, por lo tanto, reduzcan el costo de inferencia en

los dispositivos móviles.

MobileNets admite cualquier tamaño de entrada superior a 32×32 , con tamaños de imagen más grandes que ofrecen un mejor rendimiento. El número de parámetros y el número de sumas múltiples se pueden modificar utilizando el alpha parámetro, que aumenta/disminuye el número de filtros en cada capa. Al alterar el tamaño y el alpha parámetro de la imagen, se pueden construir los 16 modelos del artículo, con los pesos de ImageNet provistos, la Figura 2.9 muestra la arquitectura de este modelo.

El documento demuestra el desempeño de MobileNets usando alpha valores de 1.0 (también llamado 100 % MobileNet), 0.75, 0.5 y 0.25. Para cada uno de estos alpha valores, se proporcionan pesos para 4 tamaños de imagen de entrada diferentes (224, 192, 160, 128) [13].

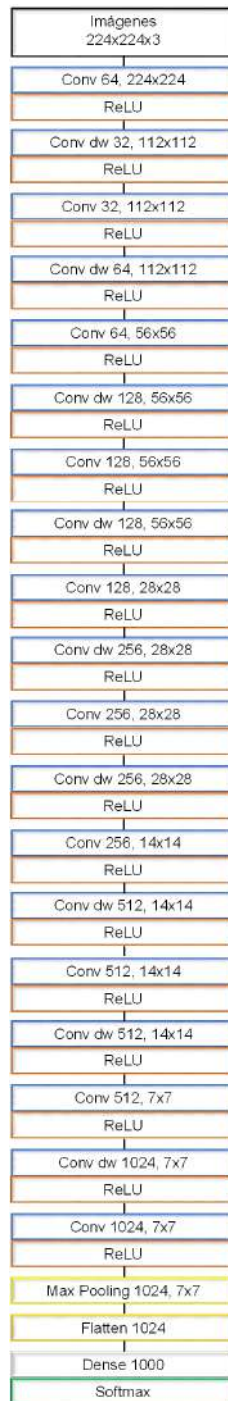


Figura 2.9: Arquitectura de MobileNet.

2.4.10. DenseNet

DenseNet es uno de los nuevos descubrimientos en redes neuronales para el reconocimiento visual de objetos. DenseNet es bastante similar a ResNet con algunas diferencias fundamentales. ResNet utiliza un método aditivo (+) que fusiona la capa anterior (identidad) con la capa futura, mientras que DenseNet concatena (*) la salida de la capa anterior con la capa futura.

DenseNet se desarrolló específicamente para mejorar la disminución de la precisión

provocada por el gradiente de desaparición en las redes neuronales de alto nivel. En términos más simples, debido al camino más largo entre la capa de entrada y la capa de salida, la información se desvanece antes de llegar a su destino, la Figura 2.10 muestra la arquitectura de este modelo [15].

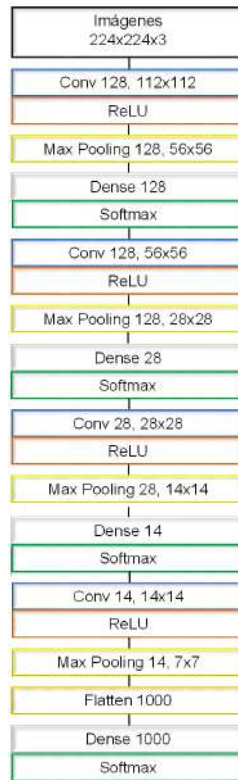


Figura 2.10: Arquitectura de DenseNet.

2.5. Métricas de clasificación

Las métricas de clasificación, permiten evaluar y comparar el rendimiento de los modelos de clasificación en diferentes aspectos, como la exactitud general, la precisión en la clasificación de positivos, la capacidad de detectar correctamente los positivos, la especificidad en la clasificación de negativos y la capacidad de distinguir entre clases. Su elección depende del contexto y los objetivos específicos del problema de clasificación. A continuación, se describen las métricas de clasificación utilizadas en ésta tesis.

2.5.1. Precisión

En el contexto de las RNC, la precisión se refiere a una medida de desempeño que evalúa la exactitud de las predicciones realizadas por el modelo. Es una métrica comúnmente utilizada para evaluar el rendimiento de un modelo de clasificación en problemas de reconocimiento de imágenes [36].

La precisión se calcula dividiendo el número de predicciones correctas realizadas por el modelo entre el número total de muestras o ejemplos en el conjunto de datos.

Se calcula dividiendo el número de verdaderos positivos (TP) entre la suma de los verdaderos positivos y los falsos positivos (FP) (ecuación 2.1).

$$Precisión = \frac{TP}{TP + FP} \quad (2.1)$$

En el caso de las RNC, la precisión se puede calcular tanto en el conjunto de datos de entrenamiento como en el conjunto de datos de prueba o validación. La precisión en el conjunto de entrenamiento proporciona una medida del rendimiento del modelo en los datos utilizados para su entrenamiento, mientras que la precisión en el conjunto de prueba o validación es una medida más relevante del rendimiento del modelo en datos no vistos durante el entrenamiento, lo que indica su capacidad de generalización.

Es importante tener en cuenta que la precisión por sí sola no proporciona una imagen completa del rendimiento de un modelo de RNC. Es posible que un modelo tenga una alta precisión pero aún tenga dificultades para clasificar correctamente ciertos ejemplos o clases específicas. Por lo tanto, es común evaluar otras métricas, como la matriz de confusión o el F1-score, para obtener una comprensión más completa del rendimiento del modelo [37]. En este trabajo, también se evaluará el F1-score o Puntaje F1.

2.5.2. Sensitividad

La sensibilidad, también conocida como recall o tasa de verdaderos positivos, es una medida de desempeño que evalúa la capacidad del modelo para detectar correctamente los casos positivos en un problema de clasificación.

La sensibilidad se calcula dividiendo el número de casos positivos correctamente identificados (verdaderos positivos) entre el número total de casos positivos en el conjunto de datos [38]. Matemáticamente, se expresa de la siguiente manera:

$$Sensitividad = \frac{\text{Número de verdaderos positivos}}{\text{Número de verdaderos positivos} + \text{Número de falsos negativos}} \quad (2.2)$$

En términos de clasificación de imágenes, la sensibilidad mide la proporción de imágenes positivas (por ejemplo, imágenes que contienen un objeto de interés) que son correctamente identificadas como positivas por el modelo. Una sensibilidad alta indica que el modelo tiene una capacidad sólida para detectar casos positivos.

La sensibilidad es particularmente importante en problemas donde los falsos negativos (casos positivos clasificados incorrectamente como negativos) tienen un impacto significativo, como en la detección de enfermedades o anomalías médicas. En tales casos, se prioriza la capacidad del modelo para minimizar los falsos negativos, lo que se refleja en una alta sensibilidad [39].

Al evaluar un modelo de CNN, es común considerar otras métricas además de la sensibilidad, como la precisión, el puntaje F1, la especificidad y la exactitud global, para tener una imagen completa del rendimiento del modelo y comprender su capacidad de detección y clasificación precisa.

2.5.3. Puntaje F1

El puntaje F1 es una métrica comúnmente utilizada en problemas de clasificación para evaluar el rendimiento de un modelo, especialmente cuando se desea encontrar un equilibrio entre la precisión y la sensibilidad.

Combina la precisión y el recall en una sola medida para proporcionar una evaluación más completa del modelo. Se calcula como la media armónica de la precisión y el recall, y se expresa matemáticamente de la siguiente manera:

$$\text{Puntaje F1} = \frac{2(\text{Precisión} * \text{sensitividad})}{\text{precisión} + \text{sensitividad}} \quad (2.3)$$

El puntaje F1 oscila entre 0 y 1, donde 1 representa el mejor rendimiento. Un puntaje F1 alto indica un equilibrio entre la capacidad del modelo para realizar clasificaciones precisas (alta precisión) y su capacidad para identificar correctamente los casos positivos (alto recall).

El puntaje F1 es particularmente útil cuando hay un desequilibrio en la distribución de clases o cuando los falsos negativos y los falsos positivos tienen un impacto desigual. En tales casos, se busca un modelo que pueda lograr un equilibrio óptimo entre la precisión y el recall, y el puntaje F1 proporciona una medida de esa capacidad.

Es importante tener en cuenta que el puntaje F1 no considera la tasa de verdaderos negativos (especificidad), por lo que su interpretación puede ser limitada en algunos contextos. Dependiendo del problema y los requisitos específicos, puede ser necesario evaluar y comparar múltiples métricas para obtener una comprensión completa del rendimiento del modelo [40].

2.5.4. Soporte

El concepto de soporte está relacionado con la cantidad de ejemplos o instancias que pertenecen a una clase específica en un conjunto de datos.

El soporte se refiere al número de muestras o ejemplos en un conjunto de datos que pertenecen a una clase determinada. Por ejemplo, si estás trabajando en un problema de clasificación de imágenes con dos clases (A y B), el soporte de la clase A sería la cantidad de imágenes en el conjunto de datos que se clasifican como clase A [40].

El soporte es relevante en el contexto de la evaluación del rendimiento de un modelo de RNC, ya que puede ayudar a comprender la distribución de las clases y la representación de los datos. Un soporte desequilibrado, donde hay una gran diferencia en el número de ejemplos entre clases, puede afectar el rendimiento del modelo y la capacidad para clasificar correctamente las clases minoritarias.

Es importante considerar el soporte al interpretar las métricas de desempeño, como la precisión, la sensibilidad o el puntaje F1, ya que estas métricas pueden ser influenciadas por la distribución del soporte de las clases. En algunos casos, puede ser útil ponderar o ajustar las métricas para tener en cuenta el desequilibrio en el soporte de las clases y obtener una evaluación más precisa del rendimiento del modelo [41].

En resumen, el soporte en el contexto de las redes neuronales convolucionales se refiere al número de ejemplos o muestras que pertenecen a una clase específica en un conjunto de datos y es relevante para comprender la distribución de las clases y evaluar el rendimiento del modelo.

2.5.5. Exactitud

Tanto la precisión como la exactitud son métricas utilizadas para evaluar el rendimiento de un modelo. Sin embargo, se diferencian en los aspectos que miden y cómo se calculan.

La precisión se refiere a la proporción de predicciones positivas correctas realizadas por el modelo con respecto al total de predicciones positivas que ha realizado. En otras palabras, la precisión mide la capacidad del modelo para identificar correctamente los casos positivos. Se calcula como se muestra en la ecuación 2.

Por otro lado, la exactitud se refiere a la proporción de predicciones totales correctas realizadas por el modelo con respecto al total de muestras evaluadas. La exactitud mide la capacidad general del modelo para clasificar correctamente los casos, ya sean positivos o negativos[42]. Se calcula dividiendo la suma de verdaderos positivos y verdaderos negativos (TN) entre el total de muestras evaluadas, ecuación 2.4.

$$Exactitud = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.4)$$

2.6. Desarrollo de plataforma

Implementar un detector de carcinoma ductal invasivo utilizando redes neuronales convolucionales en un sistema embebido ofrece portabilidad, tiempos de respuesta rápidos, privacidad de datos, eficiencia en el consumo de energía y facilidad de uso. Estas ventajas son fundamentales para mejorar la detección temprana y el diagnóstico preciso del CDI, lo que puede tener un impacto significativo en el tratamiento y el pronóstico de los pacientes.

Un sistema o plataforma embebida es un sistema de computación basado en un microprocesador o un microcontrolador, diseñado para realizar una o algunas pocas funciones dedicadas, frecuentemente en un sistema de computación en tiempo real.

Para este proyecto se presenta la implementación en una tarjeta que funciona como una computadora llamada Raspberry Pi, versión 4.

Este ordenador es de bajo costo, y eso se busca para este proyecto, a demás que puede ser portátil, por lo cual también se usará una pantalla táctil o touchscreen de 7 pulgadas, un teclado, un ratón o mouse, una batería y como red neuronal se ocupará una movidius neural compute stick de Intel.

Es importante mencionar que debido a que los gadgets anteriores son muy lentos, se ocuparán hasta el final del proyecto y solo para las inferencias, por lo pronto se puede programar, probar y entrenar en plataformas online como “Google Colab” o compiladores como “Spyder de Anaconda 3” o cualquiera en lenguaje Python.

Todos los componentes físicos deben adaptarse a una carcasa o case, en internet se pueden conseguir o diseñar en algún software CAD para impresión 3D o ser manufacturados en algún CNC.

Para el caso de este proyecto se diseñó un modelo en 3D, que posteriormente se proceso en impresión 3D con PLA, para adaptar la raspberri PI4 y implementarse en una pantalla táctil con una GUI de uso sencillo (también diseñada en esta tesis), para que pueda ser portátil y operado por algún usuario sin tener una capacitación especialida.

También es necesario instalar un sistema operativo a la Raspbery Pi 4, se recomienda usar el “Raspberri Pi 64-bit Bullseye” debido a que para el entrenamiento se trabajará con TensorFlow 2.8 y este sistema operativo es compatible con esa versión, según la información de la Tabla 2.2:

Tabla 2.2: Relación entre versión de TensorFlow y sistema operativo de RaspBerry Pi.

Sistema operativo	TF 2.9.1	TF 2.8.0	TF 2.7.0	TF 2.6.0	TF 2.5.1	TF 2.5.0	TF 2.4.1	TF 2.4.0	TF 2.3.1	TF 2.3.0
Raspberri Pi 64-bit Buster	-	-	-	X	X	X	X	X	X	X
Raspberri Pi 64-bit Bullseye	-	X	X	X	-	-	-	-	-	-
Raspberri Pi Ubuntu 18.04	X	-	-	-	-	-	X	X	X	X
Raspberri Pi Ubuntu 20.04	-	-	-	X	-	X	X	X	X	X
Jetson Nano JetPack 4.6	-	-	-	-	-	-	X	X	X	-

2.6.1. Raspberri Pi4

La raspberri Pi 4 es ideal para el trabajo con RNC debido a que el procesador incorporado es un quad-core de 64 bits a 1.4 GHz de frecuencia y puede utilizar WiFi o conexión vía cable a internet.

Tiene salida HDMI, LAN y Bluetooth así como ”shields” para expansión de memoria, también otra característica importante es que tiene puerto para tarjetas microSD.

Sin embargo, algunas limitaciones del dispositivo es que la potencia de su GPU es escasa y su memoria RAM es muy pequeña que lo hace muy lento. Por eso es importante realizar el entrenamiento fuera de él y posteriormente agregarlo externamente.

2.6.2. Instalación de TensorFlow en Raspberri Pi 4

Como se mencionó en el capítulo anterior, la Raspberri Pi 4 cuenta con módulo de memoria microSD y es necesario asegurarse de que está formateada y libre.

Una vez hecho el paso anterior, se instalará el software para Windows, Linux y MacOS llamado "Raspberry Pi Imager", puede ser obtenido de la liga que se encuentra en la referencia [35].

Una vez instalada, aparecerá la siguiente interfaz (ver Figura 2.11), en donde se muestra la interfaz inicial, para ver la configuración ver anexos.



Figura 2.11: Interfaz inicial para la instalación del sistema operativo en la Raspberry Pi 3 y 4.

2.6.3. Diseño de GUI

La interfaz se diseñó pensando en que el usuario debe interactuar de forma intuitiva y sin tantas complicaciones. Solo contiene ocho botones y una ventana de salida.

A continuación, se describe los elementos que componen la interfaz de comunicación, en la sección 4.4 Desarrollo de interfaz y aplicación al sistema embebido, se explica de forma detallada la función de cada elemento.

- Botón "Elegir carpeta": permite al usuario seleccionar un conjunto de imágenes para su análisis.
- Botón "Analizar carpeta": después de la carga de la carpeta de imágenes, permite al usuario al cambio de carpeta o el análisis o si así se desea.
- Botones "Slice", "Mask", "Slice_Mask", "Slice_Mask_Pred" y "Mask_Pred": selecciona y muestra en la ventana la imagen correspondiente, después del análisis de la carpeta.
- Ventana de salida: muestra al usuario todas las imágenes seleccionadas con los botones antes mencionados.
- Icono del CIO: es una imagen del lugar en donde se desarrollo el detector, el Centro de Investigaciones en Óptica.

2.6.4. Adaptación del sistema embebido con el detector de CDI en un diseño portable

Es fundamental diseñar una carcasa que tenga dimensiones compactas y un peso liviano. Esto facilitará el transporte del sistema y hará que sea más cómodo de llevar.

Debe tener un diseño ergonómico que se adapte cómodamente a la mano o que se pueda llevar fácilmente en una mochila u otro dispositivo de transporte. Los bordes y las esquinas deben ser suaves y redondeados para evitar lesiones y molestias al usuario,

Es esencial que permita un acceso conveniente a los puertos, conectores y controles del sistema como ratón, teclado, etcétera, de manera que no sea necesario quitarla o desmontarla para utilizar las funciones principales. Esto implica el diseño de recortes o aberturas estratégicas en la carcasa. El material utilizado es PLA por ser muy económico y resistente, impreso en una máquina convencional de manufactura aditiva.

Cabe mencionar, que actualmente no se tienen registros de aplicaciones de detectores de CDI vía imágenes histopatológicas aplicados en sistemas embebidos, que tienen la cualidad de ser portátiles y con buenos resultados en la inferencia.

Para la implementación del algoritmo, el dispositivo embebido debe tener algunos ajustes como la instalación del sistema operativo, verificar e instalar la versión correcta de TensorFlow, entre otras configuraciones.

Capítulo 3

Resultados

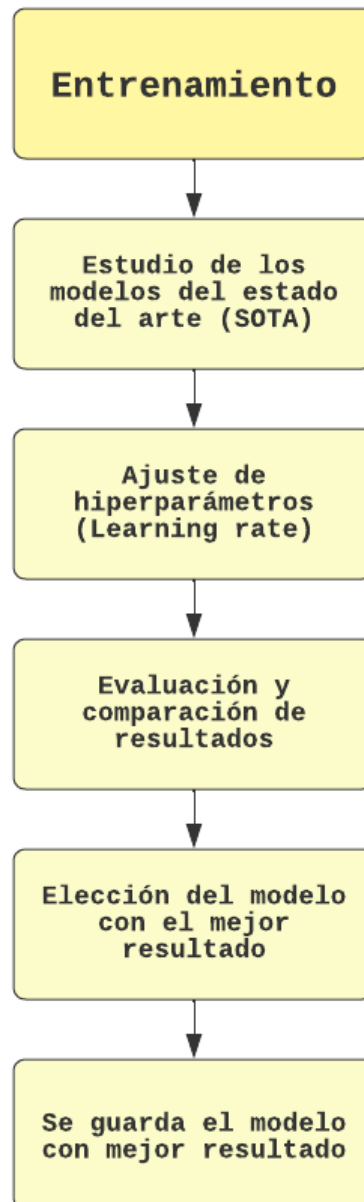
En este capítulo, se presentan los resultados obtenidos en el desarrollo del sistema de detección de carcinoma ductal invasivo vía imágenes histopatológicas. Se mostrarán los hallazgos y métricas de evaluación que permiten evaluar el desempeño y la efectividad del sistema. Además, se realizará un análisis detallado de los resultados, comparándolos con los objetivos planteados en la investigación y discutiendo su relevancia en el contexto médico.

3.1. Manejo de datos para los parámetros de entrada y de entrenamiento, validación y prueba

Para facilitar el manejo de los datos, se realizó una función para crear tres carpetas, una para entrenamiento con el 60% , una para validación con 20% y otra para prueba con el 20% restante de las imágenes del dataset (277566 imágenes en total), quedando en tamaños de 166,540 imágenes, 55,513 imágenes y 55,513 imágenes respectivamente.

3.2. Entrenamiento del sistema con los modelos

Para encontrar el mejor modelo para el entrenamiento, se compararon los más representativos modelos, en el siguiente diagrama se muestran los pasos que se siguió para su elección.



Estudio de los modelos del estado del arte (SOTA): Se llevó a cabo la lectura y evaluación de los resultados de diferentes modelos ya entrenados, y además, se buscó que las aplicaciones en que los modelos fueron utilizados, fueran similares al tema de esta tesis.

Ajuste de hiperparámetros (Learning Rate): Utilizando tres diferentes valores de Learning Rate o tasa de aprendizaje, se entrenaron todos los modelos, para tomar en cuenta el resultado más significativo.

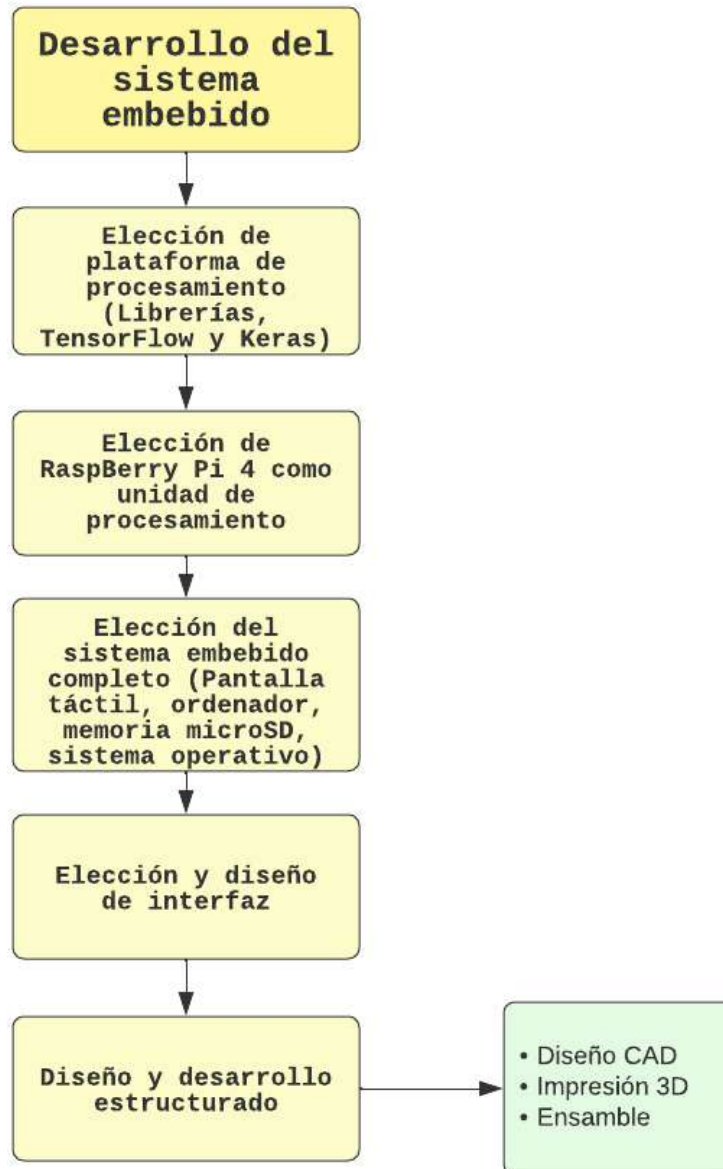
Evaluación y comparación de resultados: Se generó una gráfica con los resultados obtenidos, tomando en cuenta principalmente el tiempo que tardó en entrenarse el sistema y el valor de la exactitud.

Elección del modelo con el mejor resultado: Con base a los resultados del punto anterior, se eligió el mejor.

Se guarda el modelo con mejor resultado: Después de la elección, el modelo se guarda con la intención de utilizar en el detector de CDI.

3.3. Desarrollo del sistema embebido

En esta etapa del proyecto se buscó utilizar una unidad de procesamiento de fácil acceso y bajo costo. En el siguiente diagrama se muestra la secuencia seguida para cumplir con el objetivo.



Elección de plataforma de procesamiento (librerías, TensorFlow y Keras): En este paso, se buscó la compatibilidad de las versiones de TensorFlow y Keras, a demás de que las librerías estuvieran actualizadas.

Elección de RaspBerry Pi 4 como unidad de procesamiento: La elección de este dispositivo, fue debido al precio y capacidad de procesamiento, así como su tamaño y calidad de portabilidad.

Elección del sistema embebido completo (Pantalla táctil, ordenador, memoria microSD, sistema operativo): Los periféricos también fueron elegidos por el bajo costo y en el caso de la pantalla, se busca tener portabilidad, facilidad de uso y nitidez en la imagen.

Elección y diseño de interfaz: Para que el usuario pueda tener una buena interacción con la pantalla, se diseñó utilizando botones grandes y se creó un instructivo de uso.

Diseño y desarrollo estructurado: Para el prototipo, se diseñó una carcasa de filamento PLA (ácido poliláctico), ligera y con manejabilidad, con un par de patas para soporte a un ángulo de 50 grados aproximadamente y un soporte trasero para la RaspBerry Pi 4. Ensambla perfectamente con la pantalla táctil y tiene las aberturas para conexión de cables.

3.4. Desarrollo de interfaz y aplicación al sistema embebido

Después de ser analizado, el sistema operativo fue instalado en la Raspberry Pi 4, se diseñó una interfaz de interacción táctil y de fácil operación (ver Figura 3.1).

El sistema de inferencia requiere una carpeta con imágenes histopatológicas y que el operador pulse el botón “Elegir carpeta”, se abrirá una ventana (ver Figura 3.2).



Figura 3.1: Interfaz de comunicación del detector CDI.

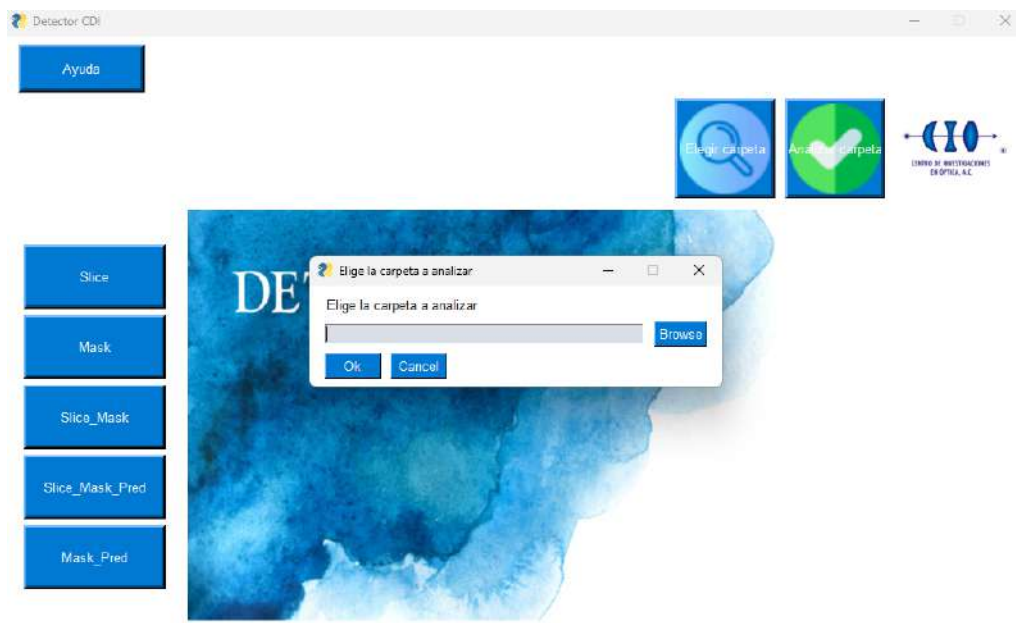


Figura 3.2: Elección de carpeta a analizar.

Al presionar el botón “Browse”, aparecerá el buscador de archivos de la PC, se debe

elegir la carpeta con imágenes que requiere ser analizada, tal como se muestra en la Figura 3.3. Se debe presionar en “Seleccionar carpeta”.

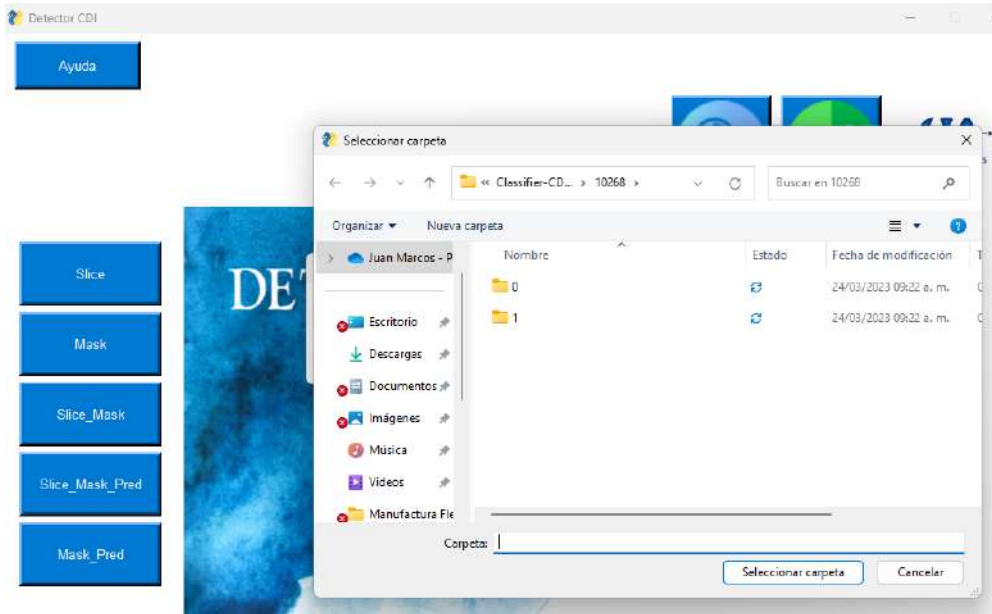


Figura 3.3: Selección y aceptación de la carpeta dentro de la PC.

Posteriormente, aparecerá la carpeta mostrada en la Figura 3.4 nuevamente, pero en esta ocasión, debe aparecer la ruta de la carpeta, en este punto se debe pulsar el botón “Ok”.

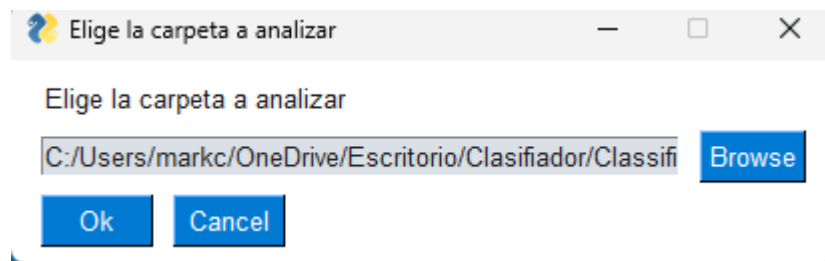


Figura 3.4: Carpeta con imágenes histopatológicas seleccionada.

Después, se debe presionar el botón “Analizar carpeta”, y el sistema comenzará a procesar el modelo, generando cinco imágenes, y antes de mostrarlas, aparece en pantalla una barra circular de progreso para indicar que las imágenes ya se están cargando (Figura 3.5).

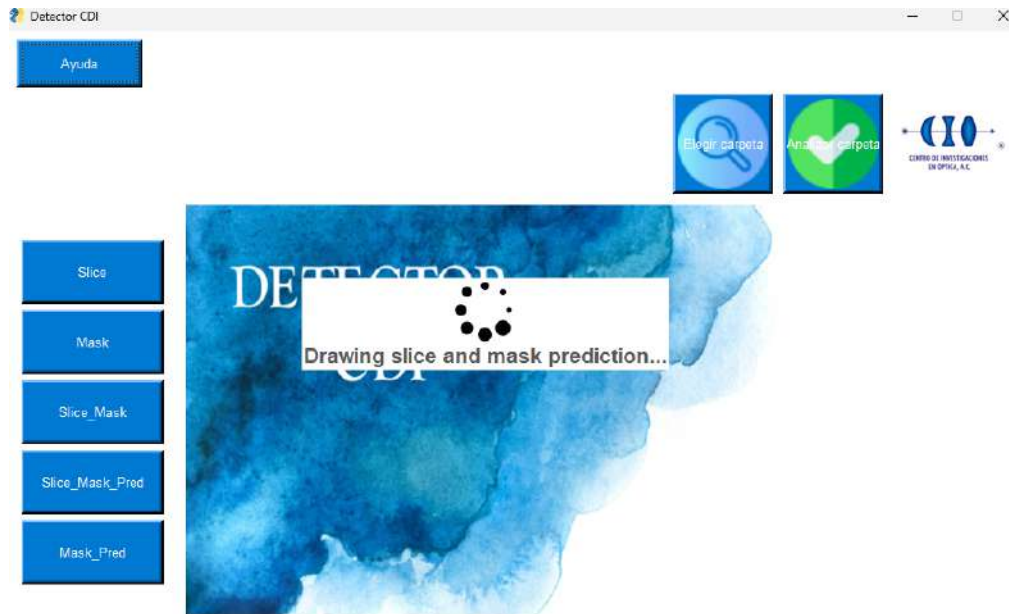


Figura 3.5: Se muestra progreso de generación de imagen Draw slice mask prediction.

Al terminar el proceso anterior, el usuario es capaz de analizar cinco imágenes definidas como:

- Slice: muestra la imagen del corte biológico tomada en alta definición.
- Mask: se observa la máscara de la imagen que el patólogo definió donde se encuentra el CDI en el corte biológico, de fondo está la imagen del corte biológico (Slice).
- Slice_Mask: se grafica en forma binarizada tanto la imagen del corte biológico, como la máscara definida.
- Slice_Mask_Predict: muestra la máscara de predicción del sistema en el corte biológico.
- Mask_Predict: genera una máscara de predicción binarizada sobre el corte biológico. Se busca que esta máscara sea semejante a Slice_Mask.

El botón superior “Ayuda” (Figura 3.2), muestra al usuario el instructivo mencionado anteriormente.

El algoritmo está diseñado para entrenar tres veces cada uno de los modelos, y en cada entrenamiento cambia las tasas de entrenamiento en la entrada, siendo 0.001, 0.0001 y 5.00E-05 sus valores cambiantes. El tamaño de lote en todos los casos es de 32 y se entrenan todas las capas de los modelos.

3.5. Resultados del entrenamiento

Los resultados del entrenamiento de los modelos de aprendizaje automático, como los modelos de Keras Applications, son fundamentales para evaluar el rendimiento y la capacidad de generalización del modelo en tareas específicas. Estos resultados son utilizados para una variedad de propósitos, incluyendo la evaluación del rendimiento,

la comparación de los modelos, el ajuste de hiperparámetros y evaluar que tan buenos son los modelos o que tan capaces son de predecir con buenos resultados los nuevos datos de entrada.

La tabla 3.1, muestra las características de los equipos de computo utilizados para el entrenamiento de los modelos.

Individualmente, los modelos trabajaron en un tiempo bastante largo debido al tamaño de los datos de entrada, ya que al ser casi 280 mil imágenes, el proceso se llevó bastante tiempo (semanas), sin embargo, los resultados fueron muy interesantes.

Las Tablas 3.2 a 3.3 muestran los resultados obtenidos, como el mejor resultado de exactitud (accuracy), y el tiempo que tardaron para terminar de entrenar con las tres diferentes Tasas de Aprendizaje (TA), y el equipo de computo utilizado para el entrenamiento.

Es importante mencionar que, todos los modelos fueron entrenados sin ninguna transferencia de aprendizaje, porque esto hace que los modelos entrenen con todas las capas y de forma aleatoria, además que se utilizó el optimizador de Adam debido a que en temas relacionados con detectores en imágenes histopatológicas funciona muy bien [20].

Tabla 3.1: Características de equipos computacionales utilizados para entrenar.

Computadora 1 y 2	Computadora 3 y 4
Ryzen 5 5600G	Ryzen 7 1700x
Asus A520	AIO 240mm NZXT
nvme 256 team group	nvme 512 GB
HDD 1TB Toshiba	HDD 2TB Toshiba
RTX 3060 zotac	RTX 3080ti y 3090
Fuente 550w Corsair bronze	MotherBoard
2 ventiladores	4 ventiladores

El tiempo de entrenamiento de la Tabla 3.2, muestra que algunos modelos tuvieron un tiempo de entrenamiento muy grande, desde 21.5 y 21.6 horas, como los modelos DenseNet169 y EfficientNetB2 respectivamente. Y otros modelos como ResNet50V2 con 2.7 horas y MobileNet con 3 horas, que son relativamente muy rápidos.

Recordemos que los modelos de Keras Applications son redes neuronales preentrenadas que se pueden utilizar para tareas de visión por computadora, como la clasificación de imágenes.

Estos modelos son arquitecturas de redes neuronales profundas complejas que han sido entrenadas previamente en conjuntos de datos masivos, como ImageNet, para aprender a reconocer características visuales comunes.

El tiempo de entrenamiento de los modelos de Keras Applications puede variar por varias razones:

1. Complejidad de la arquitectura: Los modelos de Keras Applications varían en complejidad según la arquitectura de la red neuronal utilizada. Algunos modelos, como VGG16 o ResNet, tienen arquitecturas más profundas y complejas que otros,

como MobileNet o DenseNet. Las arquitecturas más complejas generalmente requieren más tiempo de entrenamiento debido al mayor número de capas y parámetros que deben ajustarse durante el proceso de entrenamiento. Sin embargo, existen modelos en sus segundas versiones como DenseNet169.

DenseNet169 es una variante específica de DenseNet que se caracteriza por tener 169 capas en total, incluyendo capas convolucionales, capas de agrupación (pooling) y capas totalmente conectadas. Esta arquitectura es más profunda que la versión original de DenseNet, que tiene 121 capas (DenseNet121). Al tener más capas, DenseNet169 tiene la capacidad de aprender representaciones más complejas y puede capturar características más sutiles en las imágenes. Sin embargo, debido a su mayor complejidad, DenseNet169 también puede requerir más tiempo y recursos para entrenarse.

2. Tamaño del conjunto de datos de entrenamiento: El tiempo de entrenamiento también depende del tamaño del conjunto de datos utilizado para entrenar el modelo. Si se utiliza un conjunto de datos grande con muchas imágenes, el modelo puede requerir más tiempo para procesar y aprender de todos los ejemplos de entrenamiento. Además, si el conjunto de datos es desafiante y contiene una gran variabilidad de imágenes, el modelo puede necesitar más tiempo para ajustarse a esta variabilidad y aprender a generalizar correctamente.

3. Recursos computacionales disponibles: El tiempo de entrenamiento de los modelos también está influenciado por los recursos computacionales disponibles. Las arquitecturas más grandes y complejas pueden requerir más memoria y capacidad de procesamiento para entrenar eficientemente. Si se dispone de recursos limitados, como una GPU de menor capacidad o una menor cantidad de memoria, inclusive el modelo de RTX, hace que el tiempo de entrenamiento se prolongue debido a la limitación de los recursos.

4. Hiperparámetros y configuración del entrenamiento: Los hiperparámetros y la configuración utilizados durante el entrenamiento, como la tasa de aprendizaje, el tamaño del lote (batch size) y el número de épocas de entrenamiento, pueden afectar el tiempo de entrenamiento de los modelos. Al ajustar estos hiperparámetros, es posible acelerar o ralentizar el proceso de entrenamiento.

Así que, el tiempo de entrenamiento de los modelos de Keras Applications puede variar debido a la complejidad de la arquitectura, el tamaño del conjunto de datos, los recursos computacionales disponibles y la configuración del entrenamiento. Es importante tener en cuenta estos factores al seleccionar un modelo preentrenado y planificar el tiempo necesario para entrenarlo según las necesidades del proyecto.

Cuando se entrena un modelo de aprendizaje automático, es común observar variaciones en los resultados a lo largo de diferentes épocas. Esto se debe a varias razones.

1. Aprendizaje inicial: al comienzo del entrenamiento, el modelo está inicializado con pesos aleatorios y no tiene conocimiento sobre los datos de entrenamiento. En las primeras épocas, el modelo comienza a aprender y ajustar sus parámetros para capturar patrones en los datos. Durante esta fase inicial, es posible que los resultados mejoren significativamente a medida que el modelo se ajusta mejor a los ejemplos de entrenamiento.

2. Proceso de optimización: durante el entrenamiento, el modelo ajusta sus parámetros utilizando algoritmos de optimización, como el descenso del gradiente. Estos algoritmos buscan minimizar una función de pérdida que mide la discrepancia entre las predicciones del modelo y las etiquetas verdaderas de los ejemplos de entrenamiento. En cada época, el modelo actualiza sus pesos para reducir la pérdida. Sin embargo, el proceso de optimización no es perfecto y puede haber variaciones en la mejora de la función de pérdida de una época a otra.

4. Variabilidad en los datos: los conjuntos de datos de entrenamiento pueden contener variabilidad intrínseca, como diferentes clases de objetos, condiciones de iluminación, perspectivas o ruido. esta variabilidad puede influir en los resultados del entrenamiento. En algunas épocas, el modelo puede encontrarse con ejemplos de entrenamiento que son más representativos y fáciles de aprender, lo que resulta en un mejor rendimiento.

4. Hiperparámetros y configuración: los hiperparámetros y la configuración utilizados durante el entrenamiento, como la tasa de aprendizaje, el tamaño del lote (batch size) o la regularización, también pueden influir en los resultados de cada época. Ajustar estos hiperparámetros de manera óptima puede requerir experimentación y ajustes a lo largo del entrenamiento, lo que puede conducir a variaciones en el rendimiento en diferentes épocas.

En resumen, las variaciones en los resultados de diferentes épocas del entrenamiento son comunes y pueden ser el resultado de la inicialización aleatoria, el proceso de optimización, la variabilidad en los datos y la configuración del entrenamiento.

Es importante evaluar y analizar los resultados a lo largo de múltiples épocas para tener una comprensión completa del rendimiento del modelo y tomar decisiones informadas sobre su mejora y ajuste. En la Tabla 3.3, se muestra el mayor puntaje de exactitud (Accuracy), en diferentes épocas del entrenamiento, aunque es interesante saber que estos valores más altos, siempre se obtuvieron en las primeras 20 épocas. También se muestra el equipo de cómputo utilizado, y sus características se especifican en la Tabla 4.1: Características de equipos computacionales utilizados para entrenar.

Las columnas de las Tablas 3.2 y 3.3, aparece como Entrenamiento 1, Entrenamiento 2 y Entrenamiento 3, lo cual indica que:

- Entrenamiento 1 tiene una tasa de aprendizaje de 0.001
- Entrenamiento 2 tiene una tasa de aprendizaje de 0.0001
- Entrenamiento 3 tiene una tasa de aprendizaje de 0.00005

Tabla 3.2: Tiempo de entrenamiento de los modelos en horas.

Modelo	Entrenamiento 1	Entrenamiento 2	Entrenamiento 3
VGG16	3.05	3.93	4.9
VGG19	3.6	6.8	8.4
ResNet50	4.8	3.2	4.2
ResNet50V2	4.2	3	2.7
ResNet101	6.5	6.5	6
ResNet101V2	7	7	5
ResNet152	12.5	8.8	8.7
ResNet152V2	11	7.5	7.6
InceptionV3	5	4.7	4.7
InceptionResNetV2	15.5	10.6	9.5
MobileNet	3	6	4.5
MobileNetV2	6.3	-	-
DenseNet121	8.6	7	9
DenseNet169	21.5	7.7	7.3
DenseNet201	12	13	10
NasNetMobile	11.5	8.2	13
EfficientNetB0	11.5	9	10.8
EfficientNetB2	21.6	-	-
EfficientNetV2B0	4	-	-

Tabla 3.3: Resultados obtenidos de los entrenamientos y computadora utilizada para entrenamiento.

Modelo	Entrenamiento 1	Entrenamiento 2	Entrenamiento 3	Número de computadora y RTX
VGG16	0.687	0.847	0.847	4, 3090
VGG19	0.687	0.859	0.853	4, 3090
ResNet50	0.860	0.862	0.869	4, 3090
ResNet50V2	0.852	0.855	0.849	4, 3090
ResNet101	0.859	0.866	0.868	4, 3090
ResNet101V2	0.859	0.853	0.864	4, 3090
ResNet152	0.852	0.855	0.863	4, 3090
ResNet152V2	0.852	0.860	0.853	4, 3090
InceptionV3	0.862	0.860	0.864	4, 3090
InceptionResNetV2	0.864	0.861	0.858	3, 3080ti
MobileNet	0.860	0.870	0.846	3, 3080ti
MobileNetV2	0.868	-	-	1, 3060a
DenseNet121	0.8537	0.8535	0.855	1, 3060a
DenseNet169	0.855	0.848	0.842	1, 3060a
DenseNet201	0.8522	0.859	0.8521	1, 3060a
NasNetMobile	0.865	0.850	0.845	2, 3060b
EfficientNetB0	0.873	0.869	0.866	2, 3060b
EfficientNetB2	0.878	-	-	2, 3060b
EfficientNetV2B0	0.861	-	-	2, 3060b

Y para poder realizar la comparación de los resultados con los trabajos expuestos en la sección 1.8 de estado del arte, se muestran las Tablas de la 3.2 y 3.3. Debido al tiempo que tardaron entrenando, fue necesario trabajar con cuatro diferentes computadoras para optimizar el tiempo, esto también es importante de tomar en cuenta, cada ordenador estarán identificados como:

- Computadora 1 = 3060a.
- Computadora 2 = 3060b.
- Computadora 3 = 3080ti.
- Computadora 4 = 3090.

Las características del hardware, se encuentran en la tabla 3.1:

Con la finalidad de optimizar del tiempo, el algoritmo está diseñado para detener el proceso de entrenamiento cada que un modelo deja de subir su exactitud en cincuenta épocas consecutivas.

Así, las Tablas de la 3.5 al 3.23 muestran los resultados de cuatro métricas importantes para la inferencia en cada uno de los modelos, que son:

- Precisión.
- Sensitividad.
- Puntaje F1.
- Soporte.
- Exactitud.

Para facilitar la gráfica de la Figura 4.1, los modelos tienen una clave de etiqueta para poder observar en la gráfica su valor. Por ejemplo, El modelo MobileNetV2, con una tasa de aprendizaje de 0.001, tiene la etiqueta H2, cuyo valor de exactitud es de 0.865. La etiqueta se clasifico de tal manera que indique alfabéticamente una letra y numericamente un número que crece ascendentemente, similar a la clasificación que hace excel en sus celdas. En el cuadro 3.4, se muestran todos los modelos etiquetados desde A1 hasta Y2, con sus respectivas tasas de aprendizaje.

Como se puede observar, la precisión y la exactitud no depende del tiempo de entrenamiento en todos los casos. Hay modelos que tardaron casi 22 horas en realizar el entrenamiento, y otros apenas sobrepasaron las 2 horas, parece indicar que mientras más profundos sean los modelos, el tiempo es mayor, mientras que la tasa de aprendizaje esta relacionado con los resultados de la exactitud.

Los modelos MobileNetV2 y EfficientNetB2 están muy cercanos en sus valores de exactitud, siendo 0.865 y 0.867 respectivamente; sin embargo, mientras el primero se entreno en un tiempo de 6.3 horas, y el segundo lo hizo en 21.6 horas, se puede considerar que el modelo MobileNetV2 es más conveniente que EfficientNetB2.

El tercer mejor valor de exactitud, lo obtuvo MobileNet con 0.858 entrenado en un tiempo de tres horas, pero en este caso, el valor ya está relativamente más alejado de EfficientNetB2. Nótese que los tres valores tienen una tasa de aprendizaje de 0.001.

Por otro lado, los valores de exactitud más bajos los obtuvieron NasNetMobile con 0.829 entrenado en 13 horas y DenseNet169 con 0.831 con un tiempo de entrenamiento de 7.3 horas, ambos con una tasa de entrenamiento de 0.00005

Tabla 3.4: Etiquetas de los modelos con su respectivo valor de entrenamiento de tasa de aprendizaje.

Modelo	Etiqueta según la tasa de aprendizaje del modelo		
	Tasa de aprendizaje 0.001	Tasa de aprendizaje 0.0001	Tasa de aprendizaje 0.00005
VGG16	A1	B1	C1
VGG19	D1	E1	F1
ResNet50	G1	H1	I1
ResNet50V2	J1	K1	L1
ResNet101	M1	N1	O1
ResNet101V2	P1	Q1	R1
ResNet152	S1	T1	U1
ResNet152V2	V1	W1	X1
InceptionV3	Y1	Z1	A2
Inception-ResNetV2	B2	C2	D2
MobileNet	E2	F2	G2
MobileNet V2	H2	-	-
DenseNet 121	I2	J2	K2
DenseNet 169	L2	M2	N2
DenseNet 201	O2	P2	Q2
NasNet-Mobile	R2	S2	T2
EfficientNet B0	U2	V2	W2
EfficientNet B2	X2	-	-
EfficientNet V2B0	Y2	-	-

Tabla 3.5: Métricas del modelo VGG16.

Etiqueta del modelo	Precisión	Sensitividad	Puntaje F1	Soporte	Exactitud
A1	0.361	0.5	0.419	60490	0.723
B1	0.814	0.828	0.820	60490	0.853
C1	0.817	0.820	0.818	60490	0.854

Tabla 3.6: Métricas del modelo VGG19.

Etiqueta del modelo	Precisión	Sensitividad	Puntaje F1	Soporte	Exactitud
D1	0.361	0.5	0.419	60490	0.723
E1	0.797	0.781	0.788	60490	0.835
F1	0.817	0.838	0.826	60490	0.856

Tabla 3.7: Métricas del modelo ResNet50.

Etiqueta del modelo	Precisión	Sensitividad	Puntaje F1	Soporte	Exactitud
G1	0.805	0.801	0.803	60490	0.843
H1	0.817	0.799	0.807	60490	0.850
I1	0.815	0.811	0.813	60490	0.852

Tabla 3.8: Métricas del modelo ResNet50V2.

Etiqueta del modelo	Precisión	Sensitividad	Puntaje F1	Soporte	Exactitud
J1	0.816	0.793	0.807	60490	0.848
k1	0.810	0.796	0.802	60490	0.846
L1	0.823	0.780	0.797	60490	0.844

Tabla 3.9: Métricas del modelo ResNet101.

Etiqueta del modelo	Precisión	Sensitividad	Puntaje F1	Soporte	Exactitud
M1	0.819	0.797	0.807	60490	0.850
N1	0.820	0.808	0.814	60490	0.854
O1	0.814	0.807	0.810	60490	0.850

Tabla 3.10: Métricas del modelo ResNet101V2.

Etiqueta del modelo	Precisión	Sensitividad	Puntaje F1	Soporte	Exactitud
P1	0.818	0.797	0.806	60490	0.850
Q1	0.826	0.781	0.799	60490	0.849
R1	0.824	0.813	0.818	60490	0.857

Tabla 3.11: Métricas del modelo ResNet152.

Etiqueta del modelo	Precisión	Sensitividad	Puntaje F1	Soporte	Exactitud
S1	0.822	0.784	0.799	60490	0.849
T1	0.831	0.791	0.807	60490	0.854
U1	0.821	0.816	0.819	60490	0.856

Tabla 3.12: Métricas del modelo ResNet152V2.

Etiqueta del modelo	Precisión	Sensitividad	Puntaje F1	Soporte	Exactitud
V1	0.816	0.798	0.806	60490	0.849
W1	0.821	0.810	0.815	60490	0.855
X1	0.808	0.794	0.800	60490	0.844

Tabla 3.13: Métricas del modelo InceptionV3.

Etiqueta del modelo	Precisión	Sensitividad	Puntaje F1	Soporte	Exactitud
Y1	0.821	0.803	0.811	60490	0.853
Z1	0.823	0.805	0.813	60490	0.855
A2	0.813	0.812	0.813	60490	0.851

Tabla 3.14: Métricas del modelo InceptionResNetV2.

Etiqueta del modelo	Precisión	Sensitividad	Puntaje F1	Soporte	Exactitud
B2	0.818	0.798	0.807	60490	0.850
C2	0.815	0.793	0.802	60490	0.847
D2	0.805	0.795	0.799	60490	0.842

Tabla 3.15: Métricas del modelo MobileNet.

Etiqueta del modelo	Precisión	Sensitividad	Puntaje F1	Soporte	Exactitud
E2	0.826	0.813	0.819	60490	0.858
F2	0.8152	0.825	0.819	60490	0.853
G2	0.807	0.785	0.795	60490	0.841

Tabla 3.16: Métricas del modelo MobileNetV2.

Etiqueta del modelo	Precisión	Sensitividad	Puntaje F1	Soporte	Exactitud
H2	0.839	0.813	0.824	60490	0.865

Tabla 3.17: Métricas del modelo DenseNet121.

Etiqueta del modelo	Precisión	Sensitividad	Puntaje F1	Soporte	Exactitud
I2	0.818	0.797	0.807	60490	0.850
J2	0.826	0.799	0.810	60490	0.854
K2	0.808	0.759	0.778	60490	0.835

Tabla 3.18: Métricas del modelo DenseNet169.

Etiqueta del modelo	Precisión	Sensitividad	Puntaje F1	Soporte	Exactitud
L2	0.805	0.790	0.797	60490	0.841
M2	0.817	0.780	0.795	60490	0.845
N2	0.803	0.753	0.771	60490	0.831

Tabla 3.19: Métricas del modelo DenseNet201.

Etiqueta del modelo	Precisión	Sensitividad	Puntaje F1	Soporte	Exactitud
O2	0.821	0.793	0.805	60490	0.851
P2	0.825	0.797	0.809	60490	0.854
Q2	0.815	0.791	0.801	60490	0.847

Tabla 3.20: Métricas del modelo NasNetMobile.

Etiqueta del modelo	Precisión	Sensitividad	Puntaje F1	Soporte	Exactitud
R2	0.823	0.808	0.815	60490	0.855
S2	0.815	0.784	0.797	60490	0.846
T2	0.793	0.763	0.775	60490	0.829

Tabla 3.21: Métricas del modelo EfficientNetB0.

Etiqueta del modelo	Precisión	Sensitividad	Puntaje F1	Soporte	Exactitud
U2	0.817	0.820	0.819	60490	0.854
V2	0.816	0.822	0.819	60490	0.854
W2	0.817	0.800	0.808	60490	0.850

Tabla 3.22: Métricas del modelo EfficientNetB2.

Etiqueta del modelo	Precisión	Sensitividad	Puntaje F1	Soporte	Exactitud
X2	0.833	0.836	0.835	60490	0.867

Tabla 3.23: Métricas del modelo EfficientNetV2B0.

Etiqueta del modelo	Precisión	Sensitividad	Puntaje F1	Soporte	Exactitud
Y2	0.815	0.798	0.806	60490	0.849

3.6. Imágenes del detector en el sistema embebido

El sistema arrojó las siguientes imágenes, entrenadas con el modelo MobileNetV2, con una tasa de aprendizaje de 0.001.

Figura 3.6, muestra la imagen extraída de biopsia de una paciente tomada en alta definición.

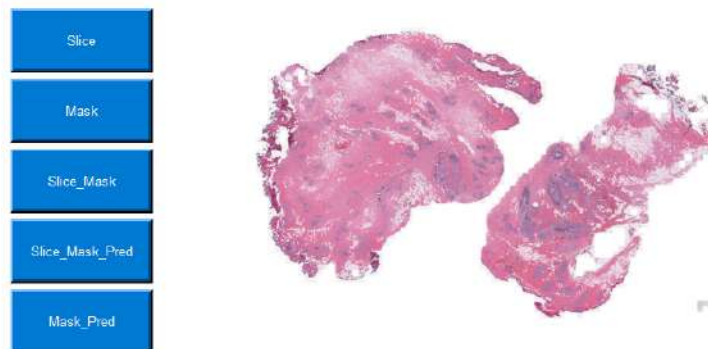


Figura 3.6: Imagen histopatológica de Slide.

Figura 3.7, se observa una máscara de color amarillo, de fondo se observa Slide.

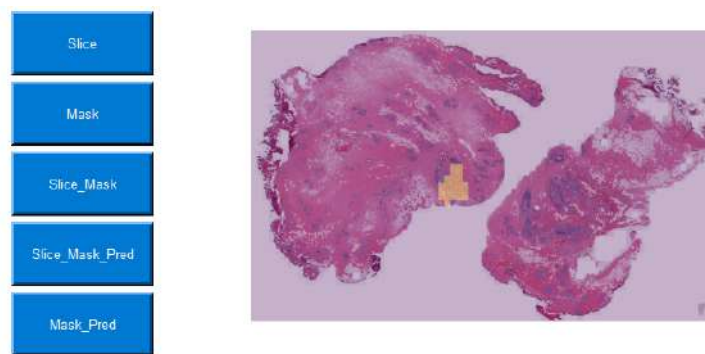


Figura 3.7: Imagen histopatológica de fondo con máscara marcada en color.

Figura 3.8, imagen binarizada de Slice_Mask, el color azul oscuro es el corte biológico y el amarillo la máscara.

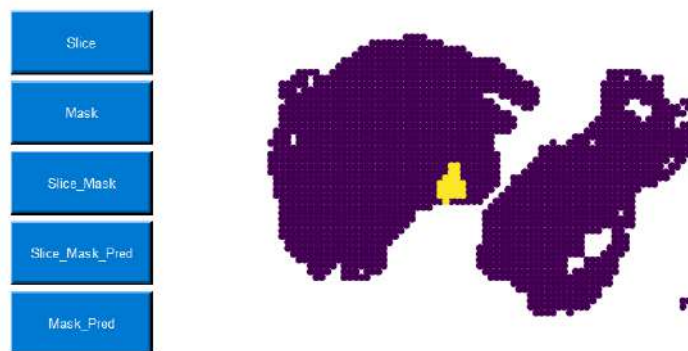


Figura 3.8: Slice_Mask.

Figura 3.9, muestra la figura con máscara de la predicción del sistema y de fondo la imagen histopatológica.

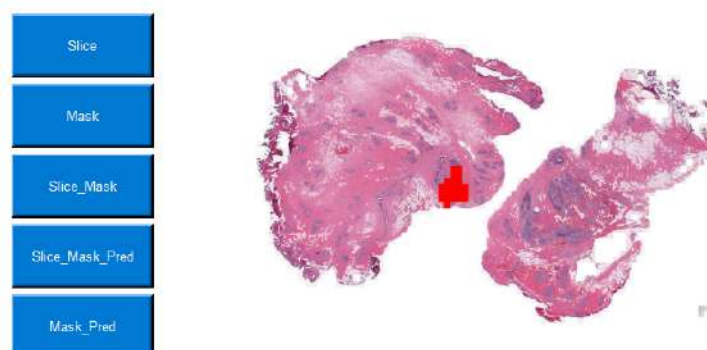


Figura 3.9: Slice_Mask_Pred.

Figura 3.10, imagen binarizada con máscara de la predicción y el corte biológico.

Como se puede observar, es exactamente igual a Slice_Mask, lo cuál, es un buen resultado.

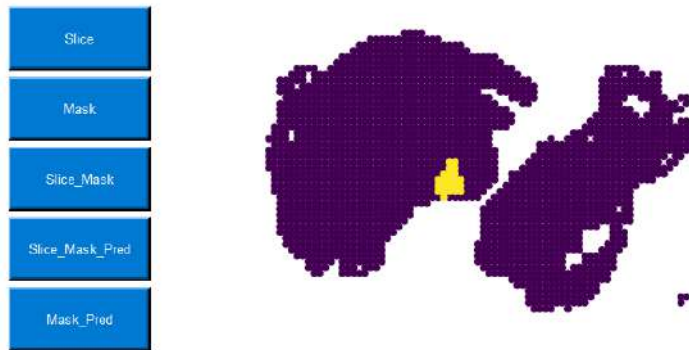


Figura 3.10: Mask_Predict.

3.7. Diseño de carcasa (case)

Una de las razones principales de realizar la inferencia del detector en un dispositivo embebido tiene que ver con el aumento de portabilidad y bajo costo del sistema; La Raspberry Pi 4 es un pequeño ordenador adaptado al case que el fabricante incluye en su compra más equipada, y consiste en una carcasa metálica de color negro que tiene aperturas en las conexiones para los cables USB, tarjeta de memoria SD, entrada de audio y para función de mando de pantalla táctil así como rendijas para facilitar la entrada de aire del ventilador interno. Para el diseño de la carcasa principal, que tiene la función de contener una pantalla táctil de 7 pulgadas, se buscó que pudiera soportar en la parte trasera a la carcasa de la Raspberry Pi 4 para que todo el sistema pareciera uno solo. Por lo anterior, se diseñó un prototipo y se procesó en impresión 3D ver Figuras 3.11 y 3.12 con un par de patas inclinadas 50° y entradas para las conexiones HDMI, micro USB, alimentación energética y habilitación de la pantalla táctil. Al sistema se le puede adaptar un teclado, un ratón y bocinas, sin embargo, al tener la cualidad táctil, el uso de estos periféricos es opcional.



Figura 3.11: Prototipo de carcasa para pantalla táctil de 7 pulgadas, parte frontal.

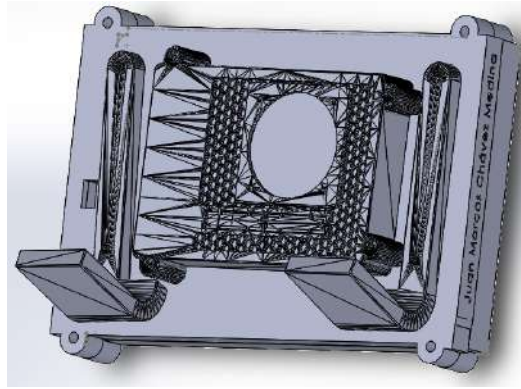


Figura 3.12: Adaptación del case de RaspBerry Pi 4 a la carcasa de la pantalla táctil de 7 pulgadas, parte trasera.

3.8. Algoritmo de inferencia

El algoritmo diseñado para que en su etapa final pueda realizar la inferencia de imágenes requirió algunas etapas:

- Manejo de datos para los parámetros de entrada y de entrenamiento, validación y prueba.
- Entrenamiento del sistema con los modelos.
- Desarrollo del sistema embebido.
- Desarrollo de interfaz y aplicación al sistema embebido.

Capítulo 4

Conclusiones

En este proyecto, se ha realizado un prototipo funcional de un detector de carcinoma ductal invasivo, utilizando imágenes histopatológicas de alta definición extraídas de biopsias de pacientes diagnosticados con dicha enfermedad. Se logró aplicar el algoritmo en un dispositivo de bajo costo, con características portables y de fácil utilización por médicos generales.

Se entrenaron varios modelos con cuatro equipos de diferentes características, y se comparó el tiempo de entrenamiento con respecto a la exactitud que arrojaron los modelos, encontrando el modelo más eficiente hablando del menor tiempo en que realizó los entrenamientos con respecto a la exactitud mayor que se obtuvo.

Tomando en cuenta el resultado mostrado en las Tablas de la 3.5 a la 3.23, respecto al tiempo de entrenamiento mostrado en la Tabla 3.2, se ha realizado la gráfica de la Figura 4.1.

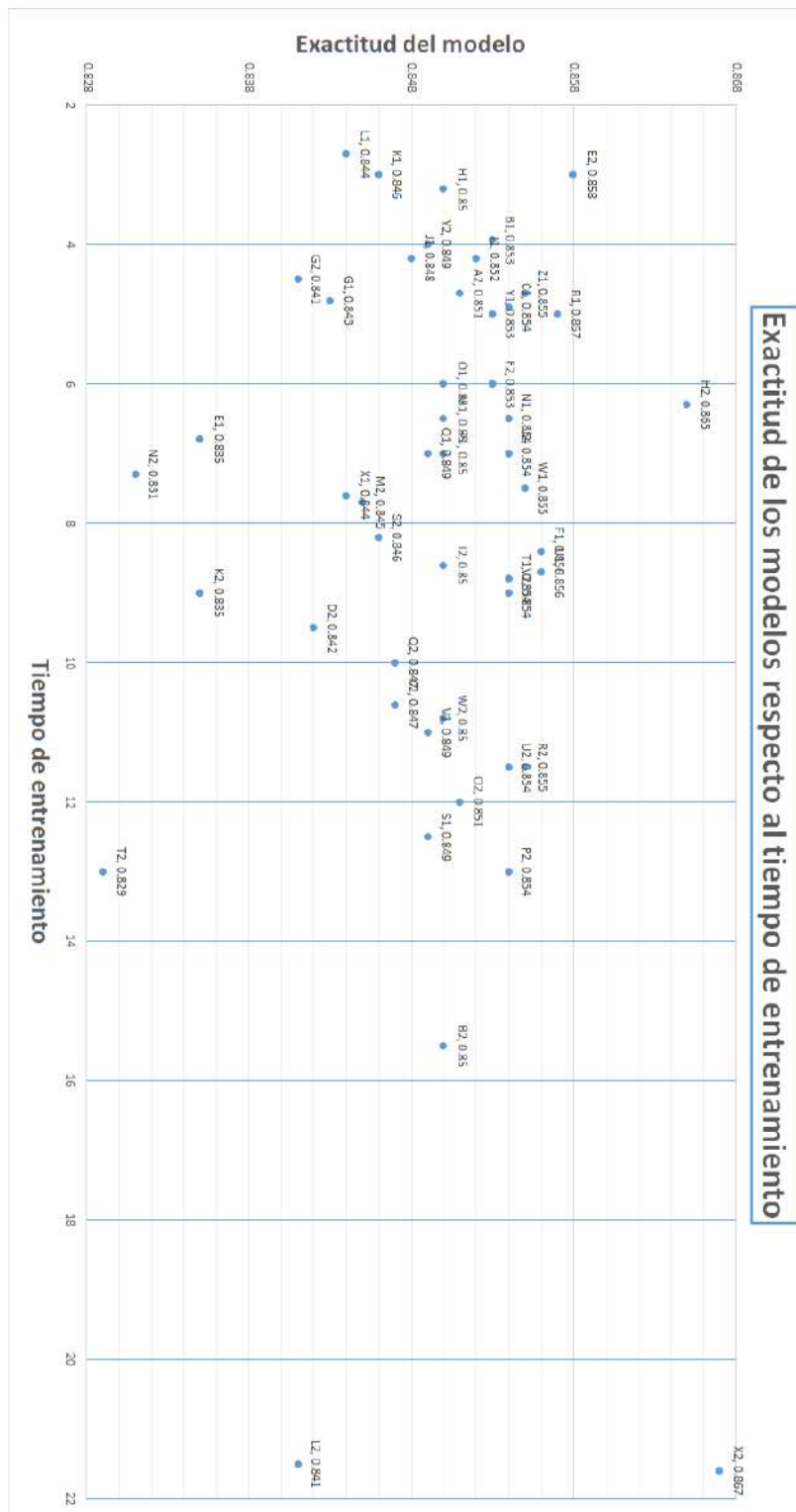


Figura 4.1: Resultados de la exactitud respecto a tiempo de entrenamiento de los modelos.

Se puede observar que el modelo MobileNetV2 con una tasa de aprendizaje de 0.001 predice en un tiempo de 6.3 horas a los demás modelos y con un valor de 0.865 en exactitud, convirtiendolo en el más adecuado para nuestro sistema, tomando en cuenta a demás que, este modelo esta diseñado para dispositivos móviles como la Raspberry e inclusive celulares. Con estos valores, el detector realiza una muy buena

predicción, aunque con la deficiencia aún del tiempo que tarda en inferir.

El costo del sistema, adaptabilidad de los elementos físicos (periféricos) del mismo, son de bajo costo y fácil disponibilidad, el diseño de la carcasa es muy flexible y al alcance de cualquier diseñador no experimentado.

Este proyecto tiene una gran ventaja, con la integración del detector y el sistema embebido, además de su aplicación en una pantalla portátil, es una excelente opción para su uso dentro de cualquier clínica médica.

El sistema muestra al usuario cinco imágenes para su interpretación, dando buenos resultados ya que Slice_Mask y Mask_Prediction son iguales.

Los resultados de la inferencia, en la detección de sí una imagen tiene CDI o no, es muy buena y con una interpretación muy sencilla para el usuario. Sin embargo, el tiempo que tarda en analizar un conjunto de imágenes, dentro del sistema embebido Raspberry Pi 4, sigue siendo bastante lento.

Bibliografía

- [1] B.R. Fernández, Características clínicas patológicas del cáncer de mama en una población de mujeres en México Universitat Oberta Catalunya., 2-3. Mañuz Aziz, 2020.
- [2] A. Labastida Almendaro, A. Espejo Fonseca, S. Rodríguez Cuevas, Características clinico patológicas del cáncer de mama en una población de mujeres en México, Cirugía cirujanos, 201-207, 2016.
- [3] P. M. Paucar, Características histopatológicas de las pacientes con signos mamográficos sospechosos de cáncer de mama. Universidad Veritas, 27-35, 2014.
- [4] J. A. Anderson, Redes Neuronales, Alfaomega, 2007.
- [5] S. Pattanayak, Pro Deep Learning with TensorFlow, Apress, 2017.
- [6] J. Klain, Santiago Ramón y Cajal, el hombre que dibujó los secretos del cerebro, New York Times, 2017.
- [7] A. Janowezyk, A. Madabhusi, Deep for digital pathology image analysis: A comprehensive tutorial with selected use case, 2016.
- [8] P. Mooney, Predicting IDC in Breast Cancer Histology Images, 2020.
- [9] Universidad Kakaha de Arabia Saudita, Boosting Breast Cancer Detection Using Convolutional Neural Network, 2021.
- [10] A. Baldominos, Deep Learning: sobre hombros de gigantes [online], 2022.
- [11] J. Hernández, Instalar y configurar Raspberry Pi Os (Raspbian) [Online], 2018.
- [12] N. Farahani, A. Parwani, L. Pantanowitz, Whole slide imaging in pathology: advantages, limitations, and emerging perspectives, 2014.
- [13] TensorFlow, Module: tf.keras.applications.mobilenet [online], citado septiembre 2022.
- [14] A. Pujara, Image Classification With MobileNet [online], 2020.
- [15] G. Huang, Z. Liu, K. Weinberger, L. Van Der Maaten, Densely Connected Convolutional Networks, 2016.
- [16] Organización Mundial de la Salud (OMS). Cáncer. [online], 2022.
- [17] J. Berrios, Redes Neuronales Convolucionales, Health Big Data, 2021.

- [18] Y. LeCun and Y. Bengio. Convolutional networks for images, speech, and time-series. In M. A. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*. MIT Press, 1995.
- [19] D. Calvo, *Función de activación-Redes neuronales*, [online], 2018.
- [20] A. G. Suijit, *Deep Learning with Keras*, Packt, 2017.
- [21] S. Bock and M. Weid, A proof of local convergence for the adam optimizer, *International Joint Conference on Neural Networks (IJCNN)*, pp. 1-8, 2019.
- [22] A. L. Maas, A. Y. Hannun, A. Y. NG,. Rectifier nonlinearities improve neural network acoustic models. En *Proc. ICML (Vol. 30, pp. 3-11)*, 2013.
- [23] PyTorch documentation on LeakyReLU activation function: <https://pytorch.org/docs/stable/generated/torch.nn.LeakyReLU.html>
- [24] D. A. Clevert, T. Unterthiner, Hochreiter, S. Fast and accurate deep network learning by exponential linear units (ELUs), 2015.
- [25] D. Hendrycks, K. Gimpel. Gaussian Error Linear Units (GELUs). *Conferencia Internacional sobre Aprendizaje Representacional en Sistemas de Inteligencia Artificial (ICLR)*, 2018.
- [26] G. Klaumbauer, T. Unterthiner, A. Mayr, S. Hochreiter. Self-Normalizing Neural Networks. *Conferencia Internacional sobre Aprendizaje Representacional en Sistemas de Inteligencia Artificial (ICLR)*, 2017.
- [27] P. Ramachandran, B. Zoph, Q. V. Le. Searching for Activation Functions. *Conferencia Internacional sobre Aprendizaje Representacional en Sistemas de Inteligencia Artificial (ICLR)*, 2017.
- [28] D. E. Rumelhart, G. E. Hinton, R. J. Williams. Learning representations by back-propagating errors. *Nature*, 1986.
- [29] B. E. Bejnordi, M. Veta, P. J. van Diest, B. van Ginneken, N. Karssemeijer, LG. itjens, M. Hermsen. Diagnostic assessment of deep learning algorithms for detection of lymph node metastases in women with breast cancer, 2017.
- [30] H. Chen, X. Qui, L. Yu, P. A. Heng, J. Quin. DCAN: Deep contour-aware networks for object instance segmentation from histology images. *Medical image analysis*, 36, 135-146, 2016.
- [31] B. Ehteshami Bejnordi, M. Veta, P. Johannes van Diest, B. van Ginneken, N. Karssemeijer, G. Litjens, van der Laak JAWM, and the CAMELYON16 Consortium. Diagnostic Assessment of Deep Learning Algorithms for Detection of Lymph Node Metastases in Women With Breast Cancer. *JAMA*, 2017.
- [32] K. Sirinukunwattana, J. P. Pluim, H. Chen, X. Qi, P. AHeng, Y. Guo. Gland segmentation in colon histology images: The glas challenge contest. *Medical image analysis*, 35, 489-502, 2016.
- [33] TensorFlow. Crea modelos de aprendizaje automático de nivel de producción con TensorFlow, <https://www.tensorflow.org>, visita mayo 2023.

- [34] B. Wieder, TensorFlow. TensorFlow, <https://github.com/tensorflow/tensorflow>, visita mayo 2023.
- [35] Raspberry Pi Foundation. (s. f.). Downloads. Recuperado de <https://www.raspberrypi.org/downloads/>
- [36] J. D. Smith, A. B. Johnson. Convolutional neural networks for image classification. *Journal of Artificial Intelligence*, 15(2), 123-145, 2019.
- [37] C. D. Brown. *Deep Learning: Fundamentals and Applications*. New York, NY: Springer, 2017.
- [38] Y. LeCun, Y. Bengio, G. Hinton. Deep learning. *Nature* 17-22, 2015.
- [39] A. Krizhevsky, I. Sutskever, G. E. Hinton. ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25, 1097-1105, 2012.
- [40] D. Zhang, Z. H. Zhou. *Understanding deep learning networks: A practical guide*. CRC Press, 2018.
- [41] I. Goodfellow, Y. Bengio, A. Courville. *Deep Learning*. MIT Press, 2016.
- [42] C. Bishop. *Pattern Recognition And Machine Learning*. Probabilistics Network, 2006.

Apéndice A

Instalación del sistema operativo de Raspberry Pi Os

Observando la Figura 2.11, al pulsar 'CHOOSE OS' se abrirá la segunda ventana para elección del sistema operativo, una referencia de apoyo es el Cuadro 2.2, para este caso se seleccionó Raspberry Pi OS (32-bit)"(Figura A.1) y posteriormente seleccionar el puerto USB que esté conectado en ese momento y debe tener agregada la tarjeta microSD (Figura A.2).

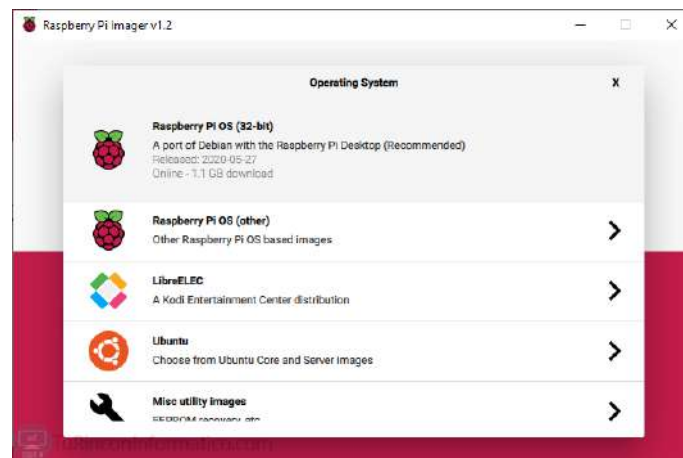


Figura A.1: Selección de sistema operativo.

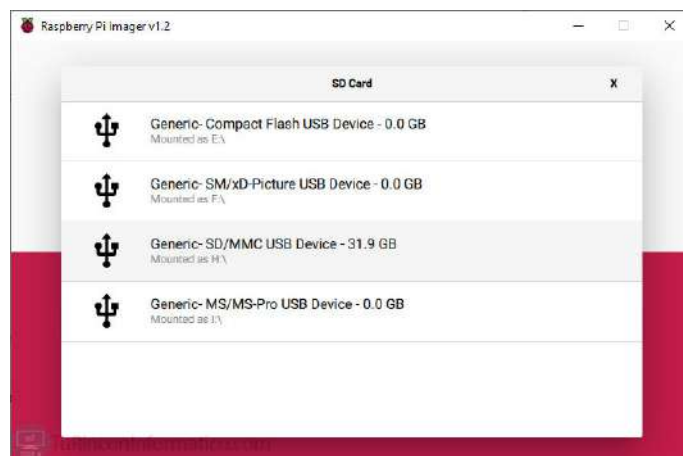


Figura A.2: Selección de puerto USB.

También se puede seleccionar otros sistemas operativos como Ubuntu, Manjaro y Apertis, en la Figura A.3 se muestra un ejemplo de los diferentes modelos del sistema operativo Ubuntu.

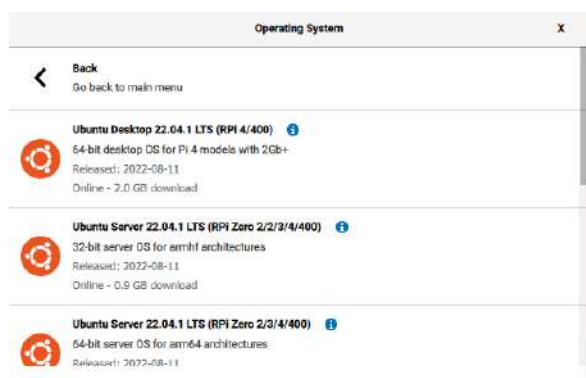


Figura A.3: Selección de algún otro sistema operativo como Ubuntu en diferentes versiones.

Por último, pulsar sobre el botón WRITE y el programa empezará la descarga de Raspberry Pi OS y la instalará en la microSD, este proceso puede tardar varios minutos.



Figura A.4: Paso final de la instalación.

Cuando termine mostrará un mensaje marcando que ha finalizado. Ahora solo queda extraer la tarjeta e introducirla en la Raspberry Pi.

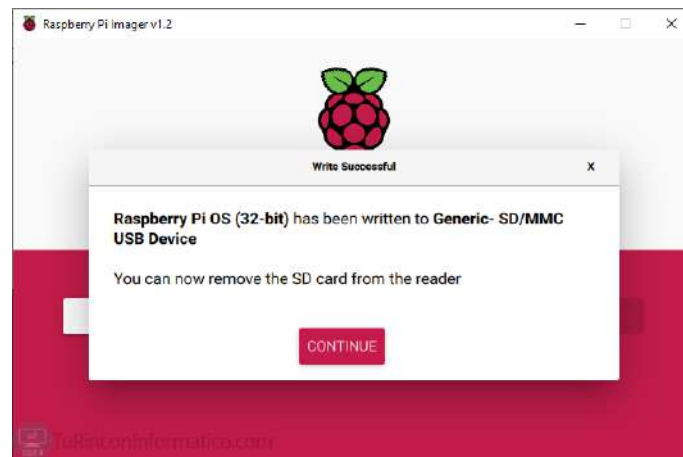


Figura A.5: Pasar la microSD con el sistema operativo ya instalado a la Raspberry Pi 4.

Configuración y primeros pasos en Raspberry Pi OS

Cuando se arranca la Raspberry Pi OS se mostrará un mensaje de bienvenida, en él se pueden configurar las opciones básicas del sistema.

- Región y Zona Horaria
- Lenguaje
- Ajuste del borde negro de pantalla
- Contraseña
- Conexión a Internet
- Actualización del sistema

Lo primero que pedirá es que se seleccione la región, lenguaje y zona horaria. Al terminar de seleccionarlo se debe pulsar Next.

El siguiente paso es asignar una contraseña a Raspberry Pi OS. Posteriormente, pulsar Next.



Figura A.6: Interfaz de cambio de contraseña.

Si la pantalla se ve con un borde negro, activar la casilla donde aparece 'This screen shows a black border around the desktop', si no es así, no se debe activar. Seguidamente, pulsar Next (Ver Figura A.7).



Figura A.7: Pantalla para configuración de pantalla.

En la Figura A.8, se debe seleccionar la red wifi. Una vez que se haya seleccionado la red wifi, pulsar Next, en el caso de no querer conectar la Raspberry Pi por red wifi, conectarle un cable de red y pulsar Skip.



Figura A.8: Selección de redes Wifi.

Cuando se seleccione la red wifi introducir la contraseña. Al pulsar Next, la Raspberry Pi se conectará a la red seleccionada y el icono de la red wifi de la parte superior derecha aparecerá en azul.



Figura A.9: Configurar contraseña para WiFi.

La página indica que se van a realizar actualizaciones del sistema, es recomendable actualizar el sistema para un mejor uso y seguridad. Este proceso puede tardar un tiempo en realizarse.



Figura A.10: Confirmación de carga del sistema operativo.

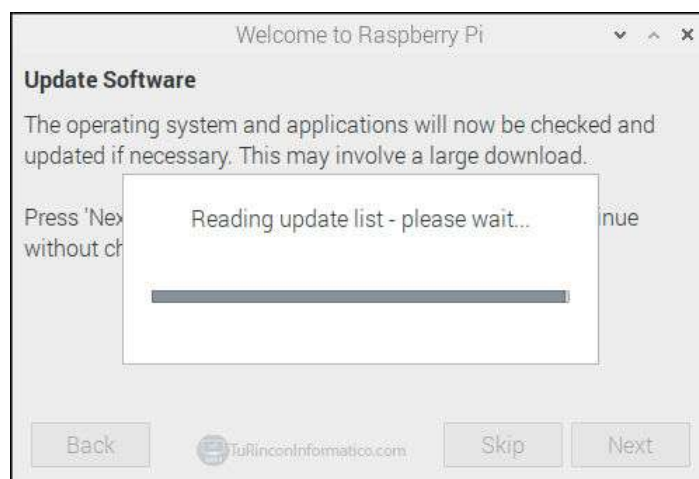


Figura A.11: Lectura para verificación de carga de lista del sistema operativo.

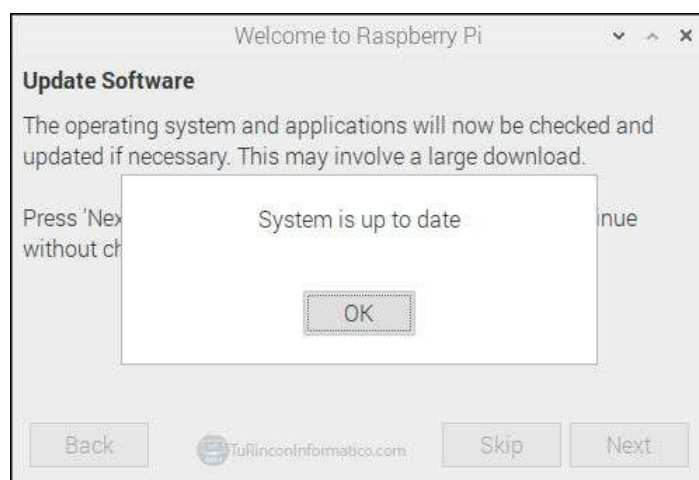


Figura A.12: Selección de país y región horaria.

Una vez el proceso ha terminado, se debe reiniciar la Raspberry Pi pulsando en

Restart.



Figura A.13: Confirmación de configuración completa.

Cuando la Raspberry Pi se reinicie se podrá observar que el menú de bienvenida ya no aparece, la red wifi está conectada, el sistema actualizado, la pantalla se ha ajustado sin los bordes negros, la fecha y hora es la correcta y el idioma que se ha elegido [11].

Apéndice B

Lista de acrónimos

- CDI: Carcinoma Ductal Invasivo.
- RN: Redes Neuronales.
- RNC: Redes Neuronales Convolucionales.
- IA: Inteligencia Artificial.
- ML: Machine Learning.
- DL: Deep Learning.