CITOBOT: un enfoque de inteligencia artificial para la detección temprana del cáncer de cuello uterino

DAVID STEVEN RIVERO URBANO

Nota de Aceptación

Certificamos que el presente Trabajo de Grado Satisface, en alcances y calidad, todos los requisitos que demanda un Trabajo de Grado de Maestría.

HERNÁN DARÍO VARGAS CARDONA
Director

GERARDO SARRIA
Jurado Uno

SUSANA MEDINA
Jurado Dos

Aprobado en cumplimiento de los requisitos exigidos por la Pontificia Universidad Javeriana Cali, para optar el título de Magister en Ingeniería de Software.

HERNÁN CAMILO ROCHA NIÑO Ph. D.
Decano Facultad de Ingeniería y Ciencias

JUAN CARLOS MARTÍNEZ ARIAS
Director Posgrados de Ingeniería y Ciencias

Santiago de Cali, 08 de febrero del 2024

Santiago de Cali, 08 de Febrero de 2024

Ingeniero:
Juan Carlos Martínez Arias
Director Posgrados de Ingeniería
Facultad de Ingeniería y Ciencias
Pontificia Universidad Javeriana - Cali

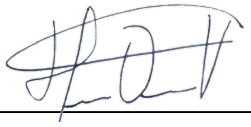
Con el fin de cumplir con los requisitos exigidos por la Universidad para llevar a cabo el Trabajo de Grado y posteriormente optar por el título de Magíster en Ingeniería de Software, nos permitimos presentar a su consideración el proyecto de Trabajo de Grado denominado ***"CITOBOT: un enfoque de inteligencia artificial para la detección temprana del cáncer de cuello uterino"***, el cual será realizado por el (la) estudiante David Steven Rivero Urbano con código 8935103 perteneciente al énfasis en Software, bajo la dirección del profesor Hernán Darío Vargas Cardona.

El suscrito director del Trabajo de Grado autoriza para que se proceda a hacer la evaluación de este Proyecto ante el Tribunal que para el efecto se designe, toda vez que ha revisado cuidadosamente el documento y avala que ya se encuentra listo para ser presentado oficialmente.

Atentamente,

Firma
David Steven Rivero Urbano

C.C. 1.107.531.622 de Cali



Firma
Hernán Darío Vargas Cardona

C.C. 1.097.721.437 de Montenegro

Maestría en Ingeniería de Software
Facultad de Ingeniería y Ciencias

FICHA RESUMEN
TRABAJO DE GRADO DE MAESTRÍA

TITULO: “CITOBOT: un enfoque de inteligencia artificial para la detección temprana del cáncer de cuello uterino”

1. ÉNFASIS: Ingeniería de Software
2. TIPO DE PROYECTO: Investigación
3. ÁREA DE TRABAJO: Inteligencia Artificial
4. ESTUDIANTE (S): David Steven Rivero Urbano
5. CORREO ELECTRÓNICO: drivero1999@javerianacali.edu.co
6. DIRECCIÓN Y TELÉFONO: Carrera 91A #48-181, +57 3118397047
7. DIRECTOR: Hernán Darío Vargas Cardona
8. VINCULACIÓN DEL DIRECTOR: Planta. Director del programa de Ingeniería Biomédica de la Facultad de Ingeniería.
9. CORREO ELECTRÓNICO DEL DIRECTOR: hernan.vargas@javerianacali.edu.co
10. CO-DIRECTOR(ES) (Si aplica): N/A
11. GRUPO O EMPRESA QUE LO AVALA (Si aplica): Oficina de Investigación y Desarrollo en Pontificia Universidad Javeriana Cali, grupos ECGSA y DESTINO.
12. OTROS GRUPOS O EMPRESAS: N/A
13. PALABRAS CLAVE: aprendizaje automático, aprendizaje profundo, cáncer de cuello uterino, redes convolucionales, redes transformers, MLOps, transfer learning.
14. ODS QUE APLICA EL PROYECTO (Agenda 2030): Salud y Bienestar
15. FECHA DE INICIO: 1/08/2022
16. RESUMEN: Hoy en día, el cáncer de cuello uterino es una enfermedad que sigue siendo preocupación en términos de salud pública a nivel mundial, por su alta incidencia y mortalidad, especialmente en países en desarrollo. En el 2022, en Colombia se reportaron 30.997 casos prevalentes, significando un incremento del 17\% en la proporción de casos nuevos reportados. A pesar de los avances y disponibilidad de pruebas de detección, estas cifras continúan generando inquietud principalmente en áreas rurales debido a la dificultad para obtener imágenes diagnósticas y la falta de expertos médicos capacitados para proporcionar una evaluación precisa en estos sitios.

    En el ámbito de la ingeniería, el uso de algoritmos de aprendizaje automático y profundo han demostrado ser efectivos en aplicaciones de imágenes médicas, permitiendo identificar patrones y extraer características de distintas enfermedades, obteniendo un diagnóstico preciso en segundos. Además, la metodología de MLOps, o DevOps para Machine Learning, se ha posicionado como una solución para llevar estos modelos a producción de manera efectiva, automatizando los flujos de trabajo y garantizando la escalabilidad y fiabilidad de los algoritmos. MLOps fusiona las prácticas de desarrollo de software (DevOps) con los procesos específicos de Machine Learning (ML), facilitando la implementación efectiva de

modelos en entornos de producción y asegurando la integridad y confiabilidad de los algoritmos a lo largo del ciclo de vida del modelo.

Bajo este contexto, el proyecto CITOBOT busca desarrollar un sistema portátil basado en inteligencia artificial para el tamizaje del cáncer de cuello uterino. Principalmente, implementando metodologías que permitan integrar adecuadamente un modelo predictivo de imágenes colposcópicas en una aplicación móvil que impulsa el dispositivo CITOBOT. El objetivo es mejorar la detección temprana del cáncer de cuello uterino y garantizar un diagnóstico preciso y confiable de la enfermedad. Con esta iniciativa, se busca aliviar la carga de los centros médicos al proporcionar una herramienta de apoyo para el diagnóstico del cáncer de cuello uterino. Además, se pretende abordar las limitaciones en el acceso a servicios especializados en áreas rurales, al ofrecer una solución que puede ser utilizada en dichas regiones.

CITOBOT: un enfoque de inteligencia artificial para la detección temprana del cáncer de cuello uterino

David Steven Rivero Urbano

Proyecto de grado presentada(o) como requisito parcial para optar al título de:
Magister en Ingeniería de Software

Director(a):
Ph.D. Hernán Darío Vargas Cardona

Pontificia Universidad Javeriana Cali
Facultad de Ingeniería
Departamento de Electrónica y Ciencias de la Computación
Cali, Colombia
8 de febrero de 2024

Índice

Índice de figuras

Índice de tablas

Resumen

Hoy en día, el cáncer de cuello uterino es una enfermedad que sigue siendo preocupación en términos de salud pública a nivel mundial, por su alta incidencia y mortalidad, especialmente en países en desarrollo. En el 2022, en Colombia se reportaron 30.997 casos prevalentes, significando un incremento del 17 % en la proporción de casos nuevos reportados. A pesar de los avances y disponibilidad de pruebas de detección, estas cifras continúan generando inquietud principalmente en áreas rurales debido a la dificultad para obtener imágenes diagnósticas y la falta de expertos médicos capacitados para proporcionar una evaluación precisa en estos sitios.

En el ámbito de la ingeniería, el uso de algoritmos de aprendizaje automático y profundo han demostrado ser efectivos en aplicaciones de imágenes médicas, permitiendo identificar patrones y extraer características de distintas enfermedades, obteniendo un diagnóstico preciso en segundos. Además, la metodología de MLOps, o DevOps para Machine Learning, se ha posicionado como una solución para llevar estos modelos a producción de manera efectiva, automatizando los flujos de trabajo y garantizando la escalabilidad y fiabilidad de los algoritmos. MLOps fusiona las prácticas de desarrollo de software (DevOps) con los procesos específicos de Machine Learning (ML), facilitando la implementación efectiva de modelos en entornos de producción y asegurando la integridad y confiabilidad de los algoritmos a lo largo del ciclo de vida del modelo.

Bajo este contexto, el proyecto CITOBOT busca desarrollar un sistema portátil basado en inteligencia artificial para el tamizaje del cáncer de cuello uterino. Principalmente, implementando metodologías que permitan integrar adecuadamente un modelo predictivo de imágenes colposcópicas en una aplicación móvil que impulsa el dispositivo CITOBOT. El objetivo es mejorar la detección temprana del cáncer de cuello uterino y garantizar un diagnóstico preciso y confiable de la enfermedad. Con esta iniciativa, se busca aliviar la carga de los centros médicos al proporcionar una herramienta de apoyo para el diagnóstico del cáncer de cuello uterino. Además,

se pretende abordar las limitaciones en el acceso a servicios especializados en áreas rurales, al ofrecer una solución que puede ser utilizada en dichas regiones.

Palabras Clave: aprendizaje automático, aprendizaje profundo, cáncer de cuello uterino, MLOps, imágenes colposcopia.

Abstract

Today, cervical cancer remains a concern in terms of public health worldwide due to its high incidence and mortality rates, particularly in developing countries.

In 2022, 30,997 prevalent cases were reported in Colombia, meaning an increase of 17 % in the proportion of new cases reported.

Despite advancements and availability of detection tests, these figures continue to raise concerns, particularly in rural areas, where obtaining diagnostic images and lacking trained medical experts for accurate evaluations pose challenges.

In the field of engineering, the utilization of machine learning and deep learning algorithms has proven effective in medical imaging applications, enabling the identification of patterns and extraction of disease-specific features, resulting in precise diagnoses within seconds. In addition, the MLOps methodology, or DevOps for Machine Learning, has emerged as a solution to effectively deploy these models into production, automating workflows, and ensuring the scalability and reliability of algorithms. MLOps combines software development practices (DevOps) with specific Machine Learning (ML) processes, facilitating the effective implementation of models in production environments and ensuring the integrity and reliability of algorithms throughout the model's lifecycle.

In this context, the CITOBOT project aims to develop a portable system based on artificial intelligence for cervical cancer screening. Primarily, it focuses on implementing methodologies that enable seamless integration of a predictive model using colposcopic images into a mobile application that powers the CITOBOT device. The objective is to improve early detection of cervical cancer and ensure accurate and reliable diagnosis of the disease. This initiative seeks to alleviate the burden on healthcare centers by providing a supportive tool for cervical cancer diagnosis. Furthermore, it aims to address limitations in accessing specialized services in rural areas by offering a solution that can be utilized in these regions.

Keywords: machine learning, deep learning, cervical cancer, MLOps, colposcopic images.

1. Introducción

En los últimos años, la conexión entre la ingeniería y la medicina ha ido estrechándose [3], brindando principalmente beneficios a esta última. Esto debido al rápido desarrollo de nuevas tecnologías, que en particular permiten, facilitan y mejoran la obtención de diagnósticos médicos respecto de los métodos tradicionales [4]. Evidentemente, uno de los campos que está tomando relevancia en los últimos años es la detección temprana del cáncer de cuello uterino.

Esta enfermedad representa un desafío significativo para la salud pública a nivel mundial, tanto en términos de incidencia como de mortalidad. A pesar de ser una enfermedad prevenible, continúa siendo una de las amenazas más graves para la vida de las mujeres [5]. En este contexto, Colombia enfrenta particularmente altas tasas de mortalidad, especialmente en áreas rurales donde el acceso a los servicios de salud es limitado [6].

Claramente, el diagnóstico temprano del cáncer de cuello uterino es fundamental para mejorar los resultados y las tasas de supervivencia. Sin embargo, existen desafíos significativos asociados con la disponibilidad de las pruebas de detección y la falta de expertos médicos capacitados para llevarlas a cabo [5]. Es en este escenario que la aplicación de tecnologías de vanguardia, como el aprendizaje automático y el aprendizaje profundo, se convierten en una solución prometedora.

De modo que, en este trabajo se presenta como objetivo abordar estos desafíos mediante el proyecto CITOBOT, que busca desarrollar un sistema portátil y basado en inteligencia artificial para el tamizaje del cáncer de cuello uterino. Para lograrlo, se aplicará la metodología de MLOps, que permitirá la integración efectiva de los modelos predictivos en una aplicación móvil, asegurando su funcionamiento óptimo y eficacia en la detección temprana de la enfermedad.

2. Definición del problema

2.1. Planteamiento del problema

El cáncer de cuello uterino sigue siendo un importante problema de salud pública a nivel mundial, tanto en términos de incidencia como de mortalidad. A pesar de ser totalmente prevenible, esta enfermedad representa una de las amenazas más graves para la vida de las mujeres. Según la Organización Mundial de la Salud (OMS) [5] [7], el cáncer de cuello uterino es el cuarto tipo de cáncer más común en mujeres en todo el mundo. De hecho, cada año, alrededor de 604,000 mujeres son diagnosticadas con cáncer de cuello uterino y la tasa de mortalidad asociada supera el 50 % [8]. La mayoría de estos casos ocurren en países en desarrollo [9]; esto se debe en gran medida a la falta de conocimiento y recursos sobre el cáncer de cuello uterino, además que usualmente son diagnosticados en etapas avanzadas.

En Colombia, el cáncer de cuello uterino es una preocupación importante en términos de salud pública. Según el Ministerio de Salud y Protección Social [10], ocupa el segundo lugar como causa de muerte por cáncer en mujeres en el país. La tasa cruda de incidencia de este tipo de cáncer es de 15.3 por cada 100,000 mujeres al año [11] [6], lo que demuestra la alta prevalencia de la enfermedad. Además, según los estudios epidemiológicos [11] la tasa de mortalidad asociada es de 7.1 por cada 100,000 mujeres al año.

El diagnóstico del cáncer de cuello uterino se realiza mediante la búsqueda de anomalías o lesiones en la región del cuello uterino; por lo general se utiliza una combinación de pruebas, como el Papanicolaou y el examen colposcópico. El primero, consiste en un examen de detección inicial y tradicional que busca identificar cambios anormales en las células del cuello uterino, mientras que la colposcopia es un procedimiento de seguimiento más detallado que permite una evaluación visual directa del cuello uterino y la toma de biopsias si es necesario [12] [13]. Siendo este último ampliamente reconocido como el método de referencia por los médicos exper-

tos para evaluar el cáncer de cuello uterino [14]. El resultado de este examen genera un conjunto de imágenes que la OMS [14] [1] clasifica en cinco categorías: imágenes normales, neoplasia intraepitelial cervical (CIN) 1 (leve), CIN2 (moderada), CIN3 (grave) e imágenes de cáncer.

A pesar de la disponibilidad de pruebas de detección, la tasa de mortalidad por cáncer de cuello uterino sigue siendo alta en un país como Colombia, especialmente en áreas rurales y remotas donde el acceso a los servicios de salud es limitado [15]. Una de las razones principales es la dificultad para obtener imágenes de colposcopia, que son consideradas como la práctica estándar para evaluar el cáncer de cuello uterino. Sin embargo, este proceso es generalmente lento y requiere la presencia de médicos especializados que puedan realizar el diagnóstico preciso [14]. De hecho, según Marcela Arrivillaga en [16] "en Cali los resultados de la citología pueden tardar entre 15 y 20 días hábiles en llegar a algunas instituciones que atienden mujeres del régimen subsidiado, lo que hace que muchas pacientes olviden reclamar los resultados y no continúen con la atención, en caso de que sea necesario". Además, lamentablemente, en muchas zonas rurales, la falta de estos expertos médicos capacitados dificulta el acceso a esta importante herramienta de detección y diagnóstico [15]. De modo que, se resalta la necesidad de encontrar una solución que permita obtener un diagnóstico sin la presencia de un experto en un menor tiempo y que pueda ser de fácil acceso para estas zonas.

Desde el campo de la ingeniería, el enfoque que se utiliza para abordar esta necesidad es la aplicación de algoritmos basados en aprendizaje automático (ML, machine learning) y aprendizaje profundo (DP, deep learning), debido a que los enfoques basados en estas técnicas han ganado impulso recientemente en una variedad de aplicaciones de imágenes médicas [17]. Estos conjuntos de métodos son capaces de detectar automáticamente patrones de datos [18], lo que permite realizar predicciones o tomar decisiones en entornos de incertidumbre. De hecho, este tipo de técnicas han generado un gran avance en todos los campos en los que han sido aplicadas [19], debido a que no solo aumentan el poder predictivo, sino que también mejoran la

eficiencia de procesos a través de la automatización.

Ahora bien, el gran poder predictivo de estos algoritmos debe ir acompañado de una metodología que permita llevar estos modelos a producción de manera efectiva y eficiente. En este sentido, la técnica de MLOps se ha posicionado como una solución fundamental debido a que proporciona un enfoque sistemático y estructurado [20] que abarca desde la construcción y entrenamiento de los modelos de Machine Learning hasta su implementación, monitorización y mantenimiento continuo en un entorno de producción. Según [21], esta técnica se centra en la automatización de los flujos de trabajo, la gestión de versiones, la colaboración entre equipos y la monitorización constante de los modelos; lo que garantiza la reproducibilidad, la escalabilidad y la fiabilidad de los algoritmos en producción [20] [21]. De modo que, es una técnica idónea para poner en producción los modelos predictivos ya que asegura la implementación exitosa y el funcionamiento óptimo de los modelos en un entorno real, maximizando así su valor y efectividad.

Bajo esta premisa surge el proyecto CITOBOT, el cual consiste en el desarrollo y validación clínica de un sistema portable para el tamizaje de cáncer de cuello uterino [22]. El objetivo es crear un dispositivo innovador que utilice tecnología de inteligencia artificial para ayudar en la detección temprana del cáncer de cuello uterino. Para lograr esto, se plantea una solución que aproveche modelos y técnicas de vanguardia en Machine y Deep Learning, y que funcione bajo la metodología MLOps. La implementación de MLOps permitirá asegurar la integración adecuada del modelo predictivo en la aplicación móvil que impulsa el funcionamiento del dispositivo CITOBOT. De esta manera, se busca garantizar un diagnóstico preciso y confiable del cáncer de cuello uterino, mejorando así la eficacia y la efectividad de la detección temprana de esta enfermedad.

2.2. Formulación del problema

¿Cómo generar un clasificador de imágenes de colposcopia a partir de algoritmos y técnicas basadas en aprendizaje de máquina profundo que pueda ser integrado en la aplicación móvil del CITOBOT?

2.3. Sistematización

- ¿Cómo garantizar un proceso de recolección y preparación de datos que favorezcan el funcionamiento del algoritmo?
- ¿Cómo determinar cuál es la técnica de aprendizaje profundo que nos permita clasificar imágenes de colposcopia?
- ¿Cómo evaluar el desempeño de los métodos implementados teniendo como referente las métricas definidas e imágenes cáncer de cuello uterino de validación?
- ¿Cómo integrar el modelo de inteligencia artificial en la aplicación móvil del CITOBOT para permitir obtener predicciones de forma constante y escalable?

3. Objetivos del proyecto

3.1. Objetivo General

Identificar las etapas de cáncer de cuello uterino para ofrecer una herramienta de soporte al diagnóstico mediante la aplicación de técnicas de Machine/Deep Learning en imágenes digitales de colposcopia.

3.2. Objetivos específicos

- Gestionar bases de datos de distintas fuentes necesarias para el entrenamiento de los algoritmos de aprendizaje profundo.
- Implementar modelos de aprendizaje profundo en imágenes de colposcopia para la identificación de etapas de cáncer de cuello uterino.
- Evaluar las técnicas de aprendizaje profundo mediante la medición de métricas de desempeño para imágenes de cáncer de cuello uterino etiquetadas por expertos.
- Aplicar la metodología que permita la implementación, gestión y mantenibilidad del modelo predictivo para su integración en un prototipo móvil.

3.3. Resultados esperados

- Un archivo Jupyter Notebook que consta del código documentado en Python del preprocesamiento de los datos, la construcción y evolución de los modelos y técnicas de aprendizaje profundo utilizados para la clasificación de imágenes de colposcopia, permitiendo evidenciar como fue el manejo de la base de datos previamente etiquetada por expertos.
- Documento con la descripción de las respectivas técnicas de aprendizaje profundo, proceso de análisis y construcción de algoritmos, además de los resultados obtenidos que permitan evidenciar el trabajo realizado.

- Prototipo móvil funcional que permite al usuario obtener un diagnóstico de cáncer de cuello uterino a partir de una imagen de colposcopia enviada al sistema.

4. Alcance

- Se emplearán imágenes de diversas fuentes y bases de datos, tanto privadas como públicas, seleccionando aquellas que presenten menos interferencias o ruido (datos innecesarios que puedan afectar el entrenamiento de los modelos).
- Se llevará a cabo un proceso de normalización para estandarizar las imágenes provenientes de distintas fuentes.
- Se realizará una transformación de las imágenes para asegurar que todas tengan una resolución de píxeles uniforme, lo cual facilitará la comparación efectiva de los diferentes modelos utilizados.
- El prototipo constará de una interfaz móvil que permitirá al usuario activar la cámara y mostrar una imagen de colposcopia, brindando como resultado una probabilidad de diagnóstico.

5. Justificación

El uso de procedimientos intervencionistas guiados por imágenes se ha vuelto indispensable en los hospitales de todo el mundo [3]. Estas técnicas de imagen médica han revolucionado el diagnóstico de numerosas afecciones y han facilitado la implementación de tratamientos para los pacientes. Gracias a estos avanzados procedimientos, se ha logrado una mayor precisión diagnóstica al permitir una detección temprana de enfermedades [3] [4]. La visualización directa de anomalías en tejidos y órganos internos ha llevado a mejores resultados clínicos [4], con tratamientos más oportunos y efectivos, lo que beneficia la curación y calidad de vida de los pacientes.

Sin duda, el cáncer de cuello uterino sigue siendo una preocupación significativa en términos de salud a nivel mundial [5] [23]. A pesar de ser una enfermedad completamente prevenible, las cifras de incidencia y mortalidad continúan siendo alarmantes. De modo que, para la detección temprana de esta enfermedad generalmente se emplea el flujo de trabajo llamado tamizaje [24] [25], el cual consiste en la realización diversas técnicas diagnósticas. Inicialmente, se lleva a cabo la citología, también conocida como prueba de Papanicolaou, en la cual se recolectan células del cuello uterino y se examinan bajo el microscopio para identificar posibles células anormales o lesiones precancerosas [26] [24]. En casos donde los resultados de la citología son anormales o existen factores de riesgo adicionales, se suelen realizar una colposcopia. Esta prueba consiste en la visualización ampliada y detallada del cuello uterino con un instrumento llamado colposcopio [23]. Permite identificar áreas sospechosas y tomar biopsias si es necesario para un análisis más detallado en el laboratorio. Generalmente, esta prueba retorna una imagen que se puede clasificar en Normal (sin células precancerosas), CIN1 (lesión leve), CIN2 o CIN3 (lesiones graves y presencia de células precancerosas) o Cáncer (imagen con presencia de células cancerosas) [5] [24]. Finalmente, en la prueba de VPH se utiliza en el tamizaje para detectar la presencia del virus del papiloma humano, que es la principal causa del cáncer de cuello uterino [23] [26]. Se realiza tomando una muestra de células del cuello uterino y analizando su material genético para detectar la presencia de cepas

de VPH de alto riesgo [24]. Estas técnicas se complementan entre sí y se utilizan de manera conjunta para obtener resultados más precisos.

A pesar de los avances en los tratamientos, los índices de nuevos casos de cáncer de cuello uterino continúan siendo preocupantes en Colombia [6]. Esta problemática se atribuye en gran medida a diversos factores que afectan el diagnóstico y la atención oportuna de las mujeres afectadas [27].

En primer lugar, los prolongados tiempos de espera para obtener los resultados de las pruebas de citología representan un desafío importante. De hecho, en el sistema de salud subsidiado, los informes de estas pruebas pueden tardar entre 15 y 20 días en entregarse [16]. Estas demoras pueden tener consecuencias graves, ya que el diagnóstico temprano y el inicio rápido del tratamiento son cruciales para mejorar los resultados en los casos de cáncer de cuello uterino. Además, la realización de una colposcopia, siendo este el procedimiento más preciso, debe ser realizado por especialistas en ginecología, lo cual puede requerir esperar hasta 2 meses solo para obtener una cita [27] [16]. Sin contar que, en muchos casos, también resulta complicado establecer un diagnóstico preciso, lo que lleva al especialista a realizar pruebas adicionales que prolongan aún más los tiempos de espera.

Un problema adicional se presenta en las áreas rurales, donde la falta de acceso a la colposcopia es más evidente [27]. En estas zonas desconectadas del país, resulta desafiante para el personal de salud llegar a los sitios remotos. Por lo que, esto conlleva que las mujeres se vean obligadas a desplazarse a otras ciudades en busca de atención médica adecuada. En algunos casos, estas barreras geográficas y de transporte pueden resultar insuperables, lo que lleva a la falta de realización de los procedimientos necesarios y permite que la enfermedad avance sin control.

Ahora bien, esta propuesta va enfocada en aprovechar las imágenes obtenidas mediante colposcopia y utilizarlas como datos de entrada para desarrollar un clasificador de inteligencia artificial. El objetivo es crear un sistema capaz de analizar y evaluar las imágenes colposcópicas, identificando patrones y características asocia-

das al cáncer de cuello uterino. De modo que, al utilizar algoritmos y técnicas de Machine/Deep Learning, se busca entrenar al clasificador para que pueda distinguir entre imágenes normales y aquellas que muestran indicios de células precancerosas o cancerosas. Este enfoque podría proporcionar una herramienta adicional y eficiente para el diagnóstico temprano de esta enfermedad, mejorando así los tiempos de respuesta y la atención oportuna a las mujeres afectadas. Además, al automatizar el proceso de clasificación, se busca reducir los tiempos de espera y la dependencia de la disponibilidad de especialistas en ginecología, sobre todo en áreas rurales del país donde el acceso a la colposcopia es limitado.

Como se ha mencionado anteriormente, la solución propuesta se basa en el uso de herramientas de Machine Learning y Deep Learning para construir un clasificador de imágenes colposcópicas. Estas herramientas imitan el aprendizaje humano al extraer características e identificar patrones en las imágenes [18], permitiendo diferenciar entre imágenes sin cáncer y aquellas con lesiones o células precancerosas y cancerosas. Esto se logra mediante el uso de redes neuronales y redes convolucionales [28], que funcionan de manera similar a las neuronas humanas al identificar características y transmitir conocimiento entre ellas para distinguir los diferentes tipos de imágenes y la presencia de células cancerosas. Desde el estado del arte, existen varios modelos que han sido utilizados en distintos enfoques médicos para identificar enfermedades [28], como lo son las redes UNET, Resnet, googlenet, VGGNET, entre otras. Además, es posible adaptar redes neuronales ya entrenadas en otras tareas, mediante transfer learning [29], para clasificar imágenes colposcópicas. Por otro lado, en los últimos años, las redes Transformers han sido muy relevantes, ya que han demostrado ser muy poderosas en el procesamiento de lenguaje natural [30]. Resulta interesante explorar su desempeño en tareas de diagnóstico de imágenes médicas, como las colposcópicas, aunque originalmente están diseñadas para el procesamiento de texto.

Generalmente, estos modelos son altamente efectivos cuando la cantidad de muestras disponibles es grande durante la etapa de entrenamiento [18]. Sin embargo, en el contexto de aplicaciones médicas, como el cáncer de cuello uterino, es común dispo-

ner de un número limitado de imágenes debido a la sensibilidad de los datos, lo que plantea un desafío construir modelos sin sobreajuste. Para abordar este problema, se puede recurrir a la generación de muestras artificiales para ampliar el conjunto de datos [31] [19]. Además, se requiere una etapa de preprocesamiento adecuada para las imágenes, ya que estas suelen presentar mucho ruido. Por lo tanto, es fundamental garantizar un buen proceso de segmentación de las imágenes de colposcopia, eliminando el ruido y resaltando las características relevantes.

Para lograr una implementación efectiva del modelo más eficiente en la aplicación de CITOBOT, es crucial seguir la metodología de MLOps, que proporciona prácticas para garantizar un despliegue y mantenimiento adecuados de los modelos [20] [21]. Esto implica asegurarse de que los modelos continúen aprendiendo con nuevas muestras a medida que se disponga de ellas. Una de las herramientas clave para lograr esto es TensorFlow Lite, una biblioteca que permite integrar modelos de inteligencia artificial en aplicaciones móviles y permite su funcionamiento sin necesidad de conexión a internet [32]. Al utilizar TensorFlow Lite, CITOBOT puede ser accesible y utilizado en dispositivos móviles, lo que brinda una mayor comodidad y accesibilidad a los usuarios. Además, la capacidad de actualizar y mejorar los modelos en tiempo real garantiza un rendimiento óptimo a medida que se obtienen más datos y se realiza un aprendizaje continuo.

Realizar este trabajo, generaría un impacto económico y social significativo tanto para los pacientes como para los centros de salud. Debido a que, este sistema busca simplificar el canal de trabajo del tamizaje al combinar la prueba de colposcopia y VPH en un mismo procedimiento, lo que reduce el número de pasos y consultas requeridas. Esto no solo ahorra tiempo para los hospitales y los pacientes, sino que también tiene un impacto económico positivo al disminuir los costos asociados a los múltiples exámenes. Por otro lado, el uso de CITOBOT puede beneficiar a los especialistas médicos al proporcionarles un segundo diagnóstico para casos en los que tengan dudas. Esto puede ayudar a verificar sus propias evaluaciones y evitar la necesidad de pruebas adicionales, como biopsias. Al eliminar o reducir la necesidad

de estas pruebas adicionales, se optimizan los recursos y se minimizan los riesgos y las molestias para los pacientes. Particularmente, para las personas que viven en zonas rurales, la implementación de CITOBOT es especialmente beneficiosa. Ahora, los centros médicos en áreas rurales pueden ofrecer un diagnóstico confiable en una etapa inicial, lo que evita que los pacientes tengan que desplazarse repetidamente a otras ciudades para realizar múltiples pruebas. Además, dado que CITOBOT no requiere conexión a internet, puede llegar a estas áreas remotas sin problemas, brindando un acceso más equitativo a servicios médicos especializados.

Finalmente, hay un impacto tecnológico puesto que se aporta al estudio y desarrollo de diversos algoritmos de aprendizaje automático para la detección temprana del cáncer de cuello uterino. Se analizan detenidamente las ventajas y desventajas de cada uno de estos algoritmos, con el objetivo de determinar cuál de ellos ofrece un modelo de clasificación más eficiente y preciso. Este análisis es fundamental para seleccionar el algoritmo adecuado que pueda ser integrado de manera efectiva en la aplicación del proyecto CITOBOT. Al impulsar la investigación y aplicación de algoritmos de aprendizaje automático en esta área, se contribuye al avance tecnológico y a la mejora continua de los métodos de detección y diagnóstico del cáncer de cuello uterino.

6. Marco teórico de referencia y antecedentes

Este proyecto se centra en dos áreas de conocimiento fundamentales. En primer lugar, se enfoca en el campo de la inteligencia artificial, explorando diversas técnicas que nos permitan desarrollar modelos predictivos basados en imágenes médicas. El objetivo es utilizar estas técnicas para obtener resultados precisos y confiables para dar soporte al diagnóstico médico.

En segundo lugar, se enfoca en MLOps, que se refiere a las prácticas y metodologías utilizadas para garantizar un despliegue y mantenimiento eficiente del producto de inteligencia artificial en una aplicación móvil. Esto implica establecer un proceso sólido que permita implementar, monitorear y actualizar de manera efectiva los modelos de IA en la aplicación móvil, garantizando su funcionamiento óptimo y brindando soporte continuo.

6.1. Bases Teóricas

- **Tamizaje:** Es un proceso de detección temprana utilizado en medicina para identificar enfermedades en etapas iniciales [25]. En el caso del cáncer de cuello uterino, el tamizaje se realiza mediante pruebas como la citología (prueba de Papanicolaou), la colposcopia y la prueba de VPH (virus del papiloma humano) [24]. Estas pruebas ayudan a detectar células anormales, lesiones precancerosas y la presencia de cepas de alto riesgo del VPH, permitiendo un diagnóstico temprano y un tratamiento oportuno.

- **Citología:** También conocida como prueba de Papanicolaou, es un procedimiento inicial en el que se recolectan células del cuello uterino para su posterior análisis bajo un microscopio [24]. Esta prueba se utiliza en el tamizaje del cáncer de cuello uterino y tiene como objetivo detectar células anormales o lesiones precancerosas en el tejido cervical [24].

- **Prueba de VPH:** La prueba de detección del virus del papiloma humano es un análisis médico que se utiliza para identificar la presencia de cepas de VPH de alto riesgo en el cuello uterino de las mujeres [24]. Esta prueba consiste en

la toma de una muestra de células cervicales que posteriormente se analizan en busca de material genético del virus. La detección temprana del VPH es crucial [24], ya que este virus es la principal causa del cáncer de cuello uterino.

- **Colposcopia:** Es un procedimiento médico que permite examinar de manera ampliada y detallada el cuello uterino, la vagina y la vulva [25] [24]. Se utiliza un instrumento llamado colposcopio, que es similar a un microscopio con una luz intensa, para visualizar con mayor precisión las estructuras y tejidos en estas áreas. La colposcopia se realiza generalmente cuando hay anormalidades en la citología o en otros estudios [24], y ayuda a identificar áreas sospechosas o anormales que podrían requerir biopsias adicionales para un análisis más detallado en el laboratorio.

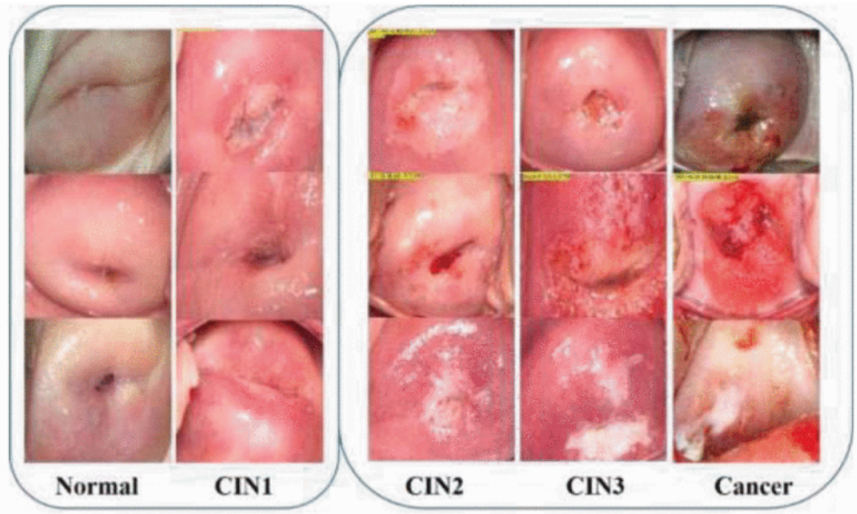


Figura 1: Cinco tipos de imágenes colposcópicas (Normal, CIN1, CIN2, CIN3, Cáncer) [1].

Como se puede ver en la Figura 1, las imágenes de colposcopia pueden clasificarse en 5 tipos según su diagnóstico. Las Imágenes tipo Normal muestran tejidos cervicales y vaginales sin anomalías aparentes. En este caso, no se re-

quieren medidas adicionales, pero se recomienda realizar seguimiento regular para detectar cualquier cambio en el futuro [33].

En las imágenes CIN1 (Neoplasia Intraepitelial Cervical de bajo grado) se observan células anormales en el cuello uterino que indican un grado leve de *displasia*[1]. En este caso, pueden tomarse medidas como realizar una biopsia o realizar pruebas adicionales, como la prueba del virus del papiloma humano (VPH) [33] [24] y se recomienda un seguimiento cercano para monitorear la progresión o regresión de la condición.

En el caso de CIN2 (Neoplasia Intraepitelial Cervical de grado intermedio), se observan células anormales en el cuello uterino que indican un grado moderado de displasia. Las medidas tomadas pueden incluir la realización de una biopsia dirigida para confirmar el diagnóstico y determinar el tratamiento adecuado [33], que puede ser una extirpación de las células anormales o una vigilancia más estrecha.

Cuando se presentan el tipo CIN3 (Neoplasia Intraepitelial Cervical de alto grado) se observan células anormales en el cuello uterino que indican un grado severo de displasia. En este caso, se recomienda una biopsia dirigida para confirmar el diagnóstico y se considera un tratamiento más agresivo [33], como la extirpación de las células anormales mediante escisión o ablación, para prevenir la progresión hacia el cáncer cervical.

Finalmente, cuando se cataloga como Cáncer, se observan células cancerosas en el cuello uterino [1]. En este caso, se requiere una evaluación más exhaustiva, que puede incluir biopsias adicionales, estudios de estadificación y la derivación a un especialista en oncología para el tratamiento adecuado [33], que puede involucrar cirugía, radioterapia, quimioterapia u otros enfoques terapéuticos.

[1]Condición médica en la que las células de un tejido u órgano muestran anormalidades en su forma, tamaño o estructura, lo que puede indicar un mayor riesgo de desarrollo de cáncer en algunos casos. La displasia se clasifica en diferentes grados según la severidad de las anormalidades celulares.

- **Inteligencia Artificial:** La inteligencia artificial (IA) es una disciplina de la ciencia de la computación que busca desarrollar sistemas y programas capaces de imitar y realizar tareas que normalmente requerirían de la inteligencia humana [34]. Utilizando algoritmos y modelos matemáticos, la IA permite a las máquinas aprender, razonar, tomar decisiones y resolver problemas de manera autónoma [35]. En su esencia, la inteligencia artificial se centra en la creación de algoritmos y modelos que permiten a las máquinas procesar información, comprender su entorno, adaptarse a cambios, resolver problemas y tomar acciones basadas en ese conocimiento [34] [2]. Esto puede incluir la capacidad de reconocer patrones en datos, comprender el lenguaje natural, interactuar con los seres humanos o realizar tareas específicas de manera autónoma. En la Figura 2podemos ver las diferentes ramas que la componen.

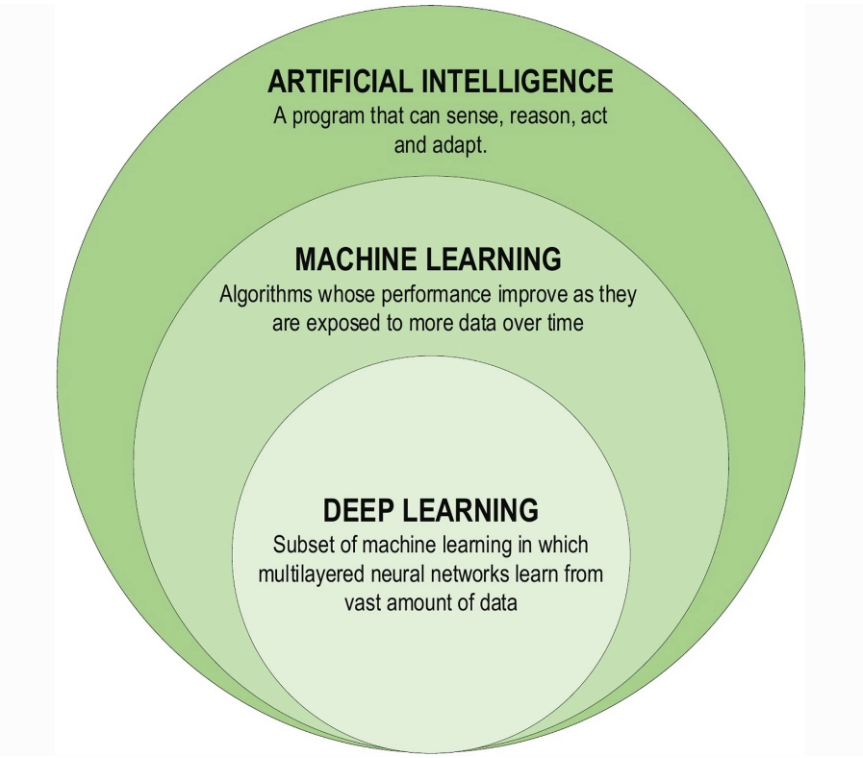


Figura 2: Sistema jerárquico de la inteligencia artificial y sus ramas [2].

- **Machine Learning:** También llamado aprendizaje automático en español, es un campo de la inteligencia artificial que se enfoca en el desarrollo de algoritmos y modelos que permiten a las máquinas aprender y mejorar automáticamente a partir de los datos sin ser programadas explícitamente [2]. En lugar de seguir instrucciones específicas, los algoritmos de machine learning detectan patrones y correlaciones en los datos para realizar predicciones y tomar decisiones basadas en esa información. El proceso de aplicación de machine learning implica tres etapas principales: adquisición y preparación de datos, entrenamiento de modelos y evaluación del rendimiento [34].

  En primer lugar, se busca y se recopila la información necesaria para resolver un problema específico. Luego, se realiza un preprocesamiento de los datos para limpiarlos y prepararlos adecuadamente. Después, se procede al entrenamiento de los modelos de machine learning. Esto implica utilizar los datos preparados para que el modelo aprenda y encuentre patrones relevantes en ellos. Una vez que el modelo ha sido entrenado, se evalúa su rendimiento utilizando conjuntos de datos de prueba. Esto permite determinar qué tan bien el modelo puede realizar predicciones precisas y generalizar a nuevos datos. Este proceso iterativo se repite para mejorar continuamente el rendimiento del modelo y ajustarlo según sea necesario.

- **Deep Learning:** El aprendizaje profundo o Deep Learning es una rama de la inteligencia artificial que se centra en la construcción y entrenamiento de redes neuronales artificiales profundas. Estas redes están compuestas por múltiples capas de neuronas interconectadas, lo que les permite aprender y extraer representaciones complejas de datos [34].

  A diferencia del Machine Learning, el cual se basa en algoritmos y características diseñadas por humanos, el Deep Learning busca aprender automáticamente características y patrones más abstractos y de alto nivel directamente de los datos [36]. Esto se logra mediante el uso de algoritmos de aprendizaje profundo que aprovechan la capacidad de las redes neuronales para aprender y ajustar los pesos de las conexiones entre las neuronas.

- **Clasificación:** Es un proceso en el que se agrupan objetos o datos en diferentes categorías o clases según sus características o propiedades. En el contexto de la clasificación de imágenes, se refiere a la tarea de asignar una etiqueta o categoría a una imagen en función de su contenido visual [37]. Este proceso utiliza algoritmos de aprendizaje automático para identificar patrones y características específicas en las imágenes y luego asociarlas con las clases predefinidas [2]. El objetivo es entrenar un modelo que pueda reconocer y clasificar automáticamente nuevas imágenes en las categorías adecuadas.

- **Transfer Learning:** Es un enfoque de Machine Learning en el cual se utiliza el conocimiento adquirido por un modelo entrenado previamente en una tarea específica para mejorar el rendimiento en otra tarea relacionada [38]. En lugar de entrenar un modelo desde cero, se aprovechan los pesos y las características aprendidas por el modelo preentrenado, que ha sido entrenado en un conjunto de datos más grande y diverso. Al utilizar Transfer Learning, se pueden lograr mejoras significativas en la eficiencia de entrenamiento y en la precisión de la predicción en nuevos conjuntos de datos [38] [29], especialmente cuando hay disponibilidad limitada de datos para la nueva tarea.

- **Procesamiento de Imágenes:** Es un proceso algorítmico en el que se toma una imagen de entrada y retorna una imagen de salida. Está compuesto por un sinnúmero de áreas de investigación tales como: adquisición, compresión, segmentación, registro, restauración, seguimiento, etiquetado, reconocimiento de patrones, clasificación, regresión, y otras [39]. Básicamente, consiste en tener un dato de entrada y salida, que son imágenes, y en el medio un algoritmo que realiza una serie de transformaciones que ayudan a mejorar el aprendizaje, dependiendo de cuál sea la naturaleza del problema que se desea resolver.

- **Data-Augmentation:** Es una técnica que consiste en generar nuevas muestras de datos a partir de las existentes mediante transformaciones y manipulaciones [40]. Estas transformaciones amplían y diversifican el conjunto de datos original, mejorando la capacidad de generalización y adaptación del modelo.

Es especialmente útil cuando se tiene un conjunto de datos limitado [31] [40], ya que permite aumentar su cantidad y variedad sin necesidad de recopilar más datos reales. Además, ayuda a abordar desequilibrios en los datos y mejorar la capacidad del modelo para reconocer patrones menos comunes [19].

- **Segmentación de Imágenes:** La segmentación en el campo de la visión artificial es el proceso de dividir una imagen digital en varias partes (grupos de píxeles) u objetos [41] [42]. El objetivo de la segmentación es simplificar y/o cambiar la representación de una imagen en otra más significativa y más fácil de analizar. La segmentación se usa tanto para localizar objetos como para encontrar los límites de estos dentro de una imagen. Más precisamente, la segmentación de la imagen es el proceso de asignación de una etiqueta a cada píxel de la imagen de forma que los píxeles que compartan la misma etiqueta también tendrán ciertas características visuales similares [42].

- **Red neuronal convolucional:** Una red neuronal convolucional (CNN, por sus siglas en inglés) es un tipo de red neuronal artificial utilizada en el reconocimiento y procesamiento de imágenes que está diseñada específicamente para procesar datos de píxeles. Las CNN son una potente herramiento para el procesamiento de imágenes [2], debido a que utilizan el aprendizaje profundo para realizar tareas tanto generativas como descriptivas, a menudo utilizando una visión artificial que incluye reconocimiento de imágenes y videos, junto con sistemas de recomendación y procesamiento de lenguaje natural. La principal ventaja de CNN en comparación con sus predecesores es que detecta automáticamente las características importantes sin supervisión humana, lo que lo convirtió en la arquitectura más utilizada para realizar tareas de clasificación en imágenes digitales. [43] [44].

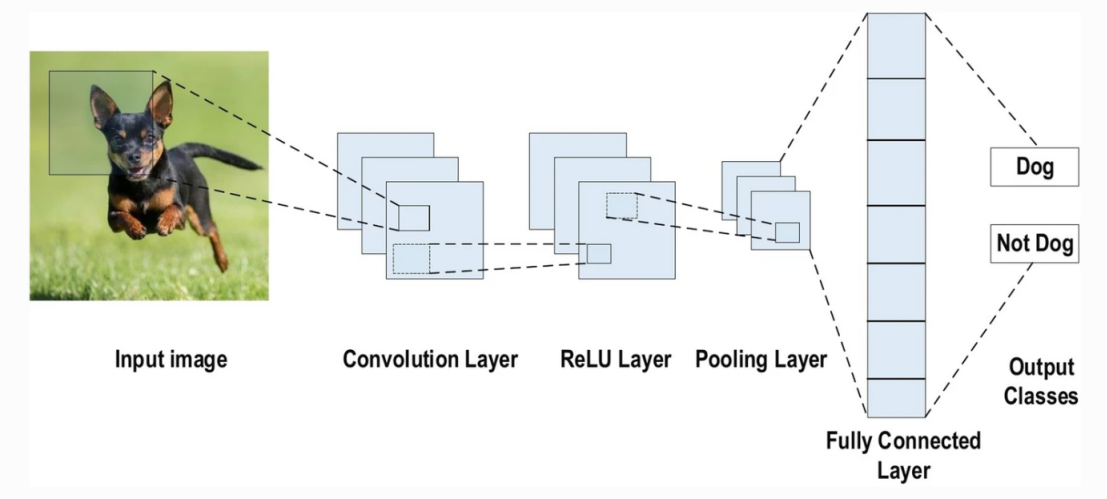


Figura 3: Ejemplo de arquiectura CNN para clasificación [2].

Como vemos en la Figura 3 un modelo basado en CNN recibe una imagen como entrada, a la cual internamente evalúa por medio de unas capas de convolución, que se encargan de encontrar el mayor número de características de la imagen para generar una salida. La entrada de cada capa en un modelo está organizada en tres dimensiones: alto, ancho y profundidad [2]. Donde la altura es igual al ancho, y la profundidad se conoce como número de canal. Por ejemplo, en una imagen RGB, la profundidad es igual a tres.

- **Redes Transformers:** Las redes Transformers son un tipo de arquitectura de redes neuronales que ha revolucionado el campo del procesamiento de lenguaje natural [30] y ha obtenido resultados sobresalientes en diversas tareas relacionadas con el lenguaje. A diferencia de las redes neuronales convencionales, que procesan secuencias de manera secuencial, las redes Transformers utilizan mecanismos de atención para capturar las relaciones entre las palabras [30] o elementos en una secuencia de manera simultánea.

  La atención es un mecanismo que permite a la red enfocarse en partes relevantes de la secuencia durante el procesamiento [30], lo que facilita la comprensión de relaciones a largo plazo y la captura de dependencias no lineales. Esta ca-

pacidad de atención ha demostrado ser especialmente efectiva para modelar el contexto y la coherencia en el lenguaje natural. Además, las redes Transformers se benefician de la capacidad de aprendizaje en paralelo y la posibilidad de capturar información contextual en diferentes niveles de abstracción [30].

- **Overfitting:** Es un fenómeno que hace que un algoritmo predictivo presente un bajo porcentaje de acierto en sus resultados, ofreciendo previsiones con una alta varianza [45]. El sobreajuste en español, indica que un modelo se ajusta demasiado a los datos de entrenamiento y no generaliza bien a nuevos datos [46]. Se puede identificar por un rendimiento excelente en los datos de entrenamiento pero un rendimiento deficiente en los datos de prueba, así como discrepancias entre las métricas de rendimiento en los dos conjuntos de datos. Para evitar el sobreajuste, se pueden aplicar técnicas como aumentar los datos de entrenamiento, usar técnicas de regularización y ajustar los hiperparámetros del modelo de manera óptima [47]. El objetivo es asegurar que el modelo aprenda patrones generalizables en lugar de memorizar los datos específicos de entrenamiento.

- **Normalización:** Proceso de ajustar y estandarizar los datos para que tengan una escala común [48], lo que facilita el entrenamiento de modelos y mejora su rendimiento. En el caso de imágenes, la normalización suele implicar la escala de píxeles para que estén en un rango específico, como [0, 1] o [-1, 1]. La fórmula matemática para normalizar un valor x del rango original $[x_{\min}, x_{\max}]$ al nuevo rango $[a, b]$ es:

$$x_{\text{norm}} = a + \frac{(x - x_{\min}) \cdot (b - a)}{x_{\max} - x_{\min}}$$

  En términos de la fórmula anterior, si consideras $x_{\min} = 0$, $x_{\max} = 255$, $a = 0$, y $b = 1$, la fórmula se reduce a la siguiente expresión: $x_{\text{norm}} = \frac{x}{255}$

- **Matriz de Confusión:** Tabla que describe el rendimiento de un modelo de clasificación al mostrar el número de verdaderos positivos (TP), verdaderos negativos (TN), falsos positivos (FP) y falsos negativos (FN) [49]. Es útil para

evaluar la precisión y el comportamiento del modelo en términos de errores y aciertos en la clasificación.

- **Specificity:** La especificidad es una medida que evalúa la capacidad de un modelo de clasificación para identificar correctamente las instancias negativas [49]. Especifica la proporción de instancias negativas reales que el modelo clasifica correctamente. Su fórmula está dada por:

$$\text{Specificity} = \frac{TN}{TN + FP}$$

- **Sensitivity:** La sensibilidad, también conocida como "recall" o "verdadero positivo rate", es una métrica de evaluación en modelos de clasificación que mide la capacidad del modelo para identificar correctamente todas las instancias positivas. En otras palabras, la sensibilidad indica la proporción de instancias positivas reales que el modelo logra clasificar correctamente [49]. Se calcula utilizando la siguiente fórmula:

$$\text{Sensitivity } = \frac{TP}{TP + FN}$$

- **Accuracy:** La exactitud es una métrica de evaluación en modelos de clasificación que mide la proporción de instancias correctamente clasificadas sobre el total de instancias [49]. Es una medida general del rendimiento del modelo. La fórmula de exactitud es:

$$\text{Accuracy} = \frac{\text{TP + TN}}{\text{Total de Instancias}}$$

- **Precision:** La precisión es una métrica de evaluación en modelos de clasificación que mide la proporción de instancias positivas identificadas correctamente entre todas las instancias clasificadas como positivas [49]. En otras palabras, la precisión se enfoca en la calidad de las predicciones positivas. La fórmula de precisión está dada por:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

- **F1-Score:** El F1-Score es una métrica que combina precisión y sensibilidad en un solo valor, proporcionando una medida equilibrada del rendimiento de un modelo de clasificación [49]. Es especialmente útil cuando hay un desequilibrio entre las clases. La fórmula del F1-Score es:

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Sensitivity}}{\text{Precision} + \text{Sensitivity}}$$

- **Loss:** La pérdida en el contexto de modelos de aprendizaje automático es una medida que evalúa la discrepancia entre las predicciones del modelo y los valores reales. Cuanto menor es el valor de la pérdida, mejor es el rendimiento del modelo en la tarea específica [50]. El objetivo del entrenamiento del modelo es minimizar esta pérdida, ajustando los parámetros del modelo durante el proceso de optimización.

- **Hiperparámetros:** Son parámetros externos que no se aprenden directamente durante el proceso de entrenamiento de un modelo de aprendizaje automático. Son configuraciones ajustables que influyen en el rendimiento y comportamiento del modelo, pero deben ser establecidos antes de iniciar el entrenamiento [51]. Ejemplos comunes de hiperparámetros incluyen la tasa de aprendizaje, el número de capas ocultas en una red neuronal, y la profundidad de un árbol de decisión. La elección adecuada de hiperparámetros es crucial para lograr un buen rendimiento del modelo [51].

- **Learning Rate:** La tasa de aprendizaje es un hiperparámetro en algoritmos de aprendizaje automático y redes neuronales que controla el tamaño de los pasos que se toman durante el proceso de optimización. Es una medida de cuánto deben ajustarse los pesos del modelo en cada iteración para minimizar la función de pérdida [52]. Una tasa de aprendizaje adecuada es crucial, ya que una tasa demasiado baja puede hacer que el modelo tarde mucho en converger, mientras

que una tasa demasiado alta puede causar oscilaciones o incluso que el modelo no converja [52]. Este hiperparámetro influye en la velocidad y estabilidad del proceso de aprendizaje del modelo.

- **Batch Size:** El tamaño del lote se refiere al número de ejemplos de entrenamiento utilizados en una iteración para actualizar los pesos del modelo [53]. Un lote es una porción del conjunto de datos total, y el tamaño del lote determina cuántas instancias se procesan simultáneamente antes de realizar una actualización de los parámetros del modelo durante el entrenamiento. Un tamaño de lote pequeño puede acelerar el entrenamiento, pero puede introducir más variabilidad, mientras que un tamaño de lote grande puede mejorar la estabilidad a expensas de la eficiencia computacional [53].

- **Epochs:** Las épocas representan el número de veces que un modelo atraviesa todo el conjunto de entrenamiento durante el proceso de entrenamiento [53]. Es un hiperparámetro que indica cuántas iteraciones completas se realizarán para ajustar los parámetros del modelo [53]. La elección de un número adecuado de épocas es crucial para lograr un equilibrio entre un modelo bien entrenado y evitar el sobreajuste o el subajuste.

6.2. Estado del Arte

El artículo **Cervical Cancer Diagnostics Healthcare System Using Hybrid Object Detection Adversarial Networks** se centra en el diagnóstico del cáncer de cuello uterino utilizando imágenes de colposcopia de una base de datos de Kaggle. El estudio aborda el desafío de clasificar estas imágenes mediante el uso de diversos algoritmos de Machine y Deep Learning, incluido el uso de técnicas como Transfer Learning. Además, se presenta una arquitectura específica denominada FSOD-GAN [14] como parte de la propuesta del sistema de salud para la detección y diagnóstico de esta enfermedad.

En el artículo **Impact of Variation in Number of Channels in CNN Clas-**

sification model for Cervical Cancer Detection se enfocan en el análisis de imágenes de células de cáncer de cuello uterino y presenta tres modelos de redes neuronales convolucionales (CNN) que consisten en dos capas de convolución con diferentes números de canales (4,8), (8,16) y (32,64) [28], así como dos capas de agrupación. Los resultados muestran que el modelo CNN con el mayor número de canales en las capas de convolución logra el mejor rendimiento, con una precisión del 93,38 % y una sensibilidad del 93,75 % [28]. Estos resultados demuestran la eficacia de la arquitectura propuesta para la clasificación de células cancerígenas.

En los últimos años, se ha prestado una atención significativa al estudio de la clasificación precisa de las células del cuello uterino [54], con el objetivo de mejorar la detección temprana de esta enfermedad. En este contexto, el artículo **Cervical Cell Image Classification Based on Multiple Attention Fusion** propone un nuevo enfoque para la clasificación de imágenes de células cervicales cancerígenas. El método propuesto se basa en la fusión de atención múltiple [54], una técnica que permite capturar y combinar de manera efectiva la información relevante de diferentes regiones de interés en las imágenes.

Este estudio de **Automated Pap Smear Cervical Cancer Screening Using Deep Learning** tiene como objetivo aplicar la red neuronal convolucional regional de máscara (Mask R-CNN) a la detección del cáncer de cuello uterino utilizando frotis histológicos de Papanicolaou [55]. Este enfoque, que según nuestra revisión de la literatura es el primero en su tipo [55], utiliza Mask R-CNN para detectar y analizar el núcleo de las células cervicales, identificando tanto características nucleares normales como anormales. Los datos utilizados en este estudio consistieron en portaobjetos histológicos de base líquida obtenidos del Hospital de la Universidad de Thammasat (TU). Los resultados mostraron que el algoritmo propuesto logró una precisión media promedio del 57,8 %, una exactitud del 91,7 %, una sensibilidad del 91,7 % y una especificidad del 91,7 % [55].

En el artículo **A deep learning based framework for accurate segmenta-**

tion of cervical cytoplasm and nuclei también se emplean imágenes obtenidas a través de la citología, y se propone un método de segmentación utilizando redes neuronales convolucionales y superpíxeles (CNN) para la segmentación de células del cáncer de cuello uterino [56]. Dado que el contraste entre el fondo y el citoplasma no es muy evidente, se realiza primero la segmentación del citoplasma. Además, se explora el aprendizaje profundo basado en CNN para la detección de regiones de interés [56]. Los resultados obtenidos muestran una precisión del 94 %.

Cervical Lesion Detection Net propone un modelo llamado CLDNet, que se enfoca en la detección de lesiones cervicales en imágenes de colposcopia utilizando una red neuronal convolucional profunda. En este enfoque, se utiliza la red neuronal convolucional Squeeze-Excitation (SE-CNN) para extraer características detalladas de toda la imagen [1]. Además, se emplea la red de propuesta de región (RPN) para generar una región de interés (ROI) que contiene posibles lesiones. Los resultados muestran que el modelo logra una precisión promedio del 91,87 % en la extracción de regiones de lesiones [1].

El estudio **MDFI: Multi-CNN Decision Feature Integration for Diagnosis of Cervical Precancerous Lesions** propone un método para la clasificación de imágenes de colposcopia que consta de varias etapas. En el preprocesamiento de datos, se utiliza el algoritmo k-means para agrupar los datos de entrenamiento en clases específicas [57]. Luego, se realiza un entrenamiento en validación cruzada para mejorar la capacidad de generalización del modelo. En la etapa de aprendizaje, se ajustan dos tipos de redes neuronales convolucionales (CNN) mediante el uso de aprendizaje de transferencia [57]. Finalmente, se utiliza el algoritmo XGBoost para integrar los diferentes resultados de decisión obtenidos de las CNN y optimizar la predicción final.

Del mismo modo, en el estudio **CVM-Cervix: A hybrid cervical Pap-smear image classification framework using CNN, visual transformer and multilayer perceptron** presenta el marco CVM-Cervix, que utiliza técnicas de aprendizaje profundo para la clasificación de células cervicales en diapositivas de Papanico-

laou [58]. El marco propuesto consta de varios módulos. En primer lugar, se emplea un módulo de red neuronal convolucional para extraer características locales de las células. Luego, se utiliza un módulo de transformador visual para capturar características globales en la diapositiva. Por último, se incorpora un módulo multicapa Perceptron para fusionar las características locales y globales y realizar la clasificación final de las células cervicales. El marco CVM-Cervix se destaca por su capacidad para analizar de manera rápida y precisa las diapositivas de Papanicolaou [58], contribuyendo así a una detección y clasificación más eficiente del cáncer cervical.

Finalmente, en el artículo **Android Device-Based Cervical Cancer Screening for Resource-Poor Settings** de Vidya Kudva, Keerthana Prasad y Shyamala Guruvare, se propone un algoritmo para el análisis de imágenes cervicales adquiridas mediante un dispositivo Android [59]. La motivación radica en la necesidad de una solución asequible y eficiente para la detección de cáncer cervical en entornos con recursos limitados. Se aborda la limitación de la experiencia necesaria en la evaluación de imágenes cervicales y se propone un sistema basado en Android que permite la adquisición de imágenes y la generación de resultados instantáneos durante la prueba de inspección visual con ácido acético (VIA) [59]. El algoritmo desarrollado demostró una precisión del 97.94 %, una sensibilidad del 99.05 % y una especificidad del 97.16 %, lo que sugiere su potencial utilidad como un sistema de apoyo a decisiones para la detección de cáncer cervical en entornos con recursos limitados.

7. Metodología de la investigación

7.1. Tipo de Estudio

Este trabajo es de tipo desarrollo aplicado, ya que tiene como objetivo probar diversas técnicas de Machine y Deep Learning para mejorar la clasificación de imágenes colposcópicas. Se seguirá una metodología completa que abarca la recolección, procesamiento y generación de datos, junto con el uso de algoritmos y técnicas adecuadas. Además, se considerará la implementación de librerías específicas para garantizar un despliegue adecuado del producto final. El propósito principal es proporcionar una metodología que contribuya de manera positiva al procesamiento digital de imágenes diagnósticas en el campo médico. Del mismo modo, este estudio hace parte del proyecto CITOBOT, el cual fue parcialmente financiado por el Ministerio de Ciencia, Tecnología e Innovación de Colombia [Grant Id 125189783229, 897/2021] y la Pontificia Universidad Javeriana Cali, Colombia [Grant Id 130100131].

7.2. Actividades

Para la solución de este proyecto se proponen las siguientes actividades basadas en cada uno de los objetivos específicos propuestos.

1. **Objetivo Específico 1:** *Gestionar bases de datos de distintas fuentes necesarias para el entrenamiento de los algoritmos de aprendizaje profundo.* Para este objetivo se pretende realizar la gestión para obtener las imágenes con las cuales se trabajará, para posteriormente realizar el proceso de preparación de los datos. Donde finalmente, con los datos que se tienen poder ir planeando las métricas con las cuales se evaluará el producto final.

   1.1 Obtener imágenes de colposcopia junto a sus etiquetas.

   1.2 Visualizar los datos (tipo de archivo y resolución de los datos/imágenes).

   1.3 Analizar y limpiar los datos.

   1.4 Transformar las imágenes para unificar las distintas fuentes de datos.

1.5 Plantear métricas para calificar el resultado final.

2. **Objetivo Específico 2:***Implementar modelos de aprendizaje profundo en imágenes de colposcopia para la identificación de etapas de cáncer de cuello uterino.* En esta parte se realizará el proceso de investigación y desarrollo de las técnicas. Donde se busque cuáles son los algoritmos de Deep Learning que nos permiten llegar a una solución, y cuál de estos proporciona más beneficios para el área seleccionada (cuello uterino). Con el fin de aplicar la técnica seleccionada y llevar a cabo el entrenamiento.

   2.1 Investigar técnicas y arquitecturas de DL para clasificación de imágenes.

   2.2 Análisis de las ventajas y desventajas para el proyecto de cada arquitectura.

   2.3 Selección de los modelos.

   2.4 Implementación de los modelos.

   2.5 Entrenamiento de los modelos seleccionados.

3. **Objetivo Específico 3:***Evaluar las técnicas de aprendizaje profundo mediante la medición de métricas de desempeño para imágenes de cáncer de cuello uterino etiquetadas por expertos.* En esta etapa se busca analizar los resultados obtenidos, haciendo entrega de los mismos en un documento, donde se logre evidenciar la precisión de la solución.

   3.1 Evaluación de resultados.

   3.2 Análisis de las métricas de clasificación (especificidad y sensibilidad).

   3.3 Clasificar modelos según el rendimiento en las métricas definidas.

   3.4 Comparación de los diferentes métodos utilizados.

   3.5 Documentación y reporte de resultados.

4. **Objetivo Específico 4:***Aplicar la metodología que permita la implementación, gestión y mantenibilidad del modelo predictivo para su integración en un prototipo móvil.* Finalmente, en esta etapa se pretende tomar el modelo con mejores

métricas e integrarlo en la aplicación móvil del proyecto CITOBOT, siguiendo la metodología MLOps, con la finalidad de garantizar buenas prácticas de despliegue y mantenibilidad del producto.

4.1 Seleccionar el modelo con mejores métricas para ser integrado.

4.2 Investigar distintas librerías que permitan integrar modelos de Machine/Deep Learning en aplicativos móviles.

4.3 Seleccionar la herramienta más adecuada en función de los requisitos y restricciones del sistema CITOBOT.

4.4 Adaptar y configurar el modelo seleccionado para que sea compatible con la aplicación móvil.

4.5 Diseño e implementación del modelo en un prototipo funcional móvil.

4.6 Realizar pruebas y evaluaciones del rendimiento del modelo en el prototipo móvil.

4.7 Documentación del prototipo, que incluya especificaciones para garantizar su mantenibilidad y la posibilidad de reentrenamiento continuo del modelo a lo largo del tiempo.

8. Desarrollo Metodológico

8.1. Metodología

El desarrollo de este proyecto se basó en la metodología CRISP-DM (Cross-Industry Standard Process for Data Mining), ampliamente reconocida en proyectos de minería de datos y aprendizaje automático debido a su capacidad para proporcionar una estructura sólida que guía a los equipos a lo largo de las etapas esenciales en la ejecución de proyectos de análisis de datos y modelado predictivo [60]. La metodología CRISP-DM se compone de seis fases fundamentales:

1. Comprensión del negocio: En esta etapa inicial, el propósito es adquirir un entendimiento profundo de los objetivos y requisitos del negocio. Se busca definir de manera precisa el problema a abordar y cómo la utilización del análisis de datos puede contribuir a lograr dichos objetivos.

2. Comprensión de los datos: Aquí se procede a la recopilación y exploración de los datos disponibles. Esto incluye obtener una visión general de la calidad de los datos, identificar posibles problemas en los mismos y seleccionar las variables relevantes para el análisis.

3. Preparación de los datos: En esta fase, se realiza la limpieza, transformación y preparación de los datos para su utilización en el proceso de modelado. Esto puede implicar la eliminación de valores atípicos, la imputación de valores faltantes y la ingeniería de características.

4. Modelado: En este punto, se seleccionan y aplican técnicas de modelado, como algoritmos de aprendizaje automático, para construir modelos predictivos. Los modelos son entrenados con los datos preparados y ajustados para lograr un rendimiento óptimo.

5. Evaluación: Luego de la construcción de los modelos, se procede a su evaluación mediante el uso de métricas de desempeño apropiadas. Esto permite determinar

cuán bien se ajustan los modelos a los datos y si cumplen con los objetivos del negocio. En caso necesario, se realizan ajustes en esta fase.

6. Despliegue: En la última etapa, los modelos que han superado la evaluación se implementan en un entorno de producción. Esto puede implicar la integración de los modelos en sistemas empresariales existentes o su publicación en una plataforma accesible para su uso continuo.

8.2. Comprensión del Negocio

Para la compresión del negocio del proyecto CITOBOT es crucial entender que el propósito central de la iniciativa radica en el desarrollo de un sistema capaz de identificar el cáncer de cuello uterino en sus etapas iniciales, lo cual conlleva una contribución significativa a la disminución de la mortalidad relacionada con esta enfermedad. En este sentido, se exige que el sistema ofrezca diagnósticos de alta precisión y fiabilidad, en concordancia con los estándares médicos, asegurando así que las pacientes reciban la atención y tratamiento adecuados de manera oportuna.

Asimismo, se plantea la necesidad de que el sistema sea accesible y esté disponible en áreas rurales y remotas, donde el acceso a servicios médicos y la presencia de expertos en colposcopia son limitados. Por lo que, resulta necesario que la herramienta no dependa del acceso a internet al proporcionar diagnósticos. Dada la frecuencia de inconvenientes en la conectividad en estas áreas, contar con una herramienta que funcione de manera autónoma garantiza que, incluso en situaciones adversas, se pueda obtener un diagnóstico. Esto se vuelve especialmente vital en contextos rurales donde la disponibilidad de internet puede ser intermitente o limitada, asegurando así la continuidad en la prestación de servicios de diagnóstico.

Así, de esta forma, el proyecto asume el desafío de proporcionar una solución que permita efectuar diagnósticos de forma ágil y eficiente, reduciendo la dependencia de la presencia de profesionales médicos y, de este modo, agilizando el proceso de diagnóstico y tratamiento. La utilización de algoritmos basados en Machine Lear-

ning y Deep Learning adquiere una relevancia crítica en la obtención de diagnósticos precisos, lo que implica la incorporación de tecnología de vanguardia para potenciar la capacidad de detección. Para garantizar la exitosa integración de los modelos predictivos en una aplicación móvil y asegurar su óptimo funcionamiento, se hace imprescindible la aplicación de la metodología de MLOps. Esta metodología automatiza los flujos de trabajo y garantiza la escalabilidad y fiabilidad de los algoritmos en un entorno de producción, lo que resulta fundamental para el éxito del proyecto.

8.3. Comprensión de los Datos

El diagnóstico del cáncer de cuello uterino implica la detección de posibles anomalías o lesiones en la región cervical, y este proceso, conocido como tamizaje, implica una serie de pasos y exámenes médicos. La Colposcopia destaca como el método más utilizado y preciso, ya que ofrece una visualización directa y detallada del cuello uterino y la vagina. La OMS clasifica las imágenes resultantes en cinco categorías: normales, NIC1 (leve), NIC2 (moderada), NIC3 (grave) e imágenes de cáncer.

Hoy en día, la adquisición de estas imágenes se realiza comúnmente a través de cámaras o colposcopios digitales, y el proceso de análisis por personal capacitado se conoce como cervicografía. La ventaja de esta técnica radica en la posibilidad de almacenar y enviar las imágenes a través de Internet, permitiendo así el diagnóstico remoto. Aunque la colposcopia digital facilita la captura de imágenes con gran detalle, su alto costo limita su viabilidad en entornos con recursos limitados. En este contexto, la opción de un dispositivo Android con una aplicación incorporada para la adquisición de imágenes y la generación de resultados instantáneos, como lo plantea CITOBOT, se presenta como una alternativa evidente en entornos con limitaciones de recursos, ofreciendo una solución eficiente y accesible para la detección temprana del cáncer de cuello uterino.

8.3.1. Recolección de Datos

La complejidad inherente a este desafío se manifiesta principalmente en la obtención de imágenes, dado que se trata de datos clínicos delicados. En consecuencia, se llevaron a cabo gestiones exhaustivas para recopilar imágenes de pacientes en distintos estados de la enfermedad, provenientes de diversos repositorios, tanto públicos como privados. Este proceso, si bien vital, presentó un desafío significativo debido a la diversidad de fuentes, lo que en ocasiones afectó la eficacia de nuestros modelos. Se logró obtener la autorización para utilizar 1.333 imágenes de la OMS, y gracias a la colaboración con el proyecto CITOBOT, se accedió a una impresionante colección de 76.658 imágenes del Banco de Cáncer de los Institutos Nacionales de Salud (NIH) de Estados Unidos.

Sin embargo, es esencial destacar que un porcentaje considerable de estas imágenes presentaba un nivel significativo de ruido, con la presencia de artefactos médicos y elementos indeseados que complicaban la visualización precisa del cuello uterino. Además, gran parte de estos datos carecían de etiquetas o tenían un diagnóstico desconocido, lo que convirtió el desafío relacionado con la calidad de la imagen en un aspecto central que debíamos abordar. En la actualidad, hemos logrado seleccionar 723 imágenes útiles de esta base de datos.

Por otra parte, se llevaron a cabo pruebas con un conjunto de datos público proveniente de un desafío en la plataforma Kaggle, que constaba de 1.469 imágenes. Sin embargo, estos datos no lograron mantener una uniformidad adecuada con las bases de datos anteriores, lo que resultó en una disminución de la efectividad de nuestros modelos y, finalmente, su exclusión en algunos casos.

En aras de mejorar y ampliar la base de datos del proyecto CITOBOT, se gestionó la obtención de imágenes del Hospital de Siloé en la ciudad de Cali. Este hospital realiza jornadas semanales de colposcopia, y las imágenes correspondientes son etiquetadas por especialistas algunas semanas después. En perspectiva, espera-

mos integrar estas imágenes a nuestros modelos para seguir fortaleciendo la robustez de nuestro proyecto.

8.3.2. Exploración de Datos

Durante la exploración de las imágenes, se notó que la gran mayoría presentaba artefactos médicos, así como fechas y horas en algunos casos, que impedían una visualización completa del cuello uterino, como se ilustra en la Figura 20. Desde el inicio, se reconoció la necesidad de segmentar estas imágenes para garantizar un entrenamiento óptimo.

Adicionalmente, se identificaron casos en los cuales el cuello uterino no era visible debido a la opacidad o desenfoque de la imagen. En respuesta a esta variabilidad, se establecieron métricas que permitieran parametrizar la idoneidad de una imagen para su utilización en el modelo de Machine/Deep Learning. Estas métricas también desempeñan un papel fundamental en la evaluación del sistema colposcópico de CITOBOT durante las pruebas, facilitando la identificación de imágenes óptimas para la construcción de la base de datos del proyecto. Las métricas definidas incluyen aspectos como la visualización del cuello, resolución, tamaño, luminosidad, profundidad y contraste.

Aunque algunas de estas métricas pueden ser subjetivas y dependientes del criterio del experto, el proyecto CITOBOT desarrolló un programa específico para procesar estas imágenes y definir sus niveles de contraste, resolución, luminosidad y profundidad, comparándolos con medidas estándar. Si una imagen cumple con los umbrales de aceptación establecidos, se considera apta para su inclusión.

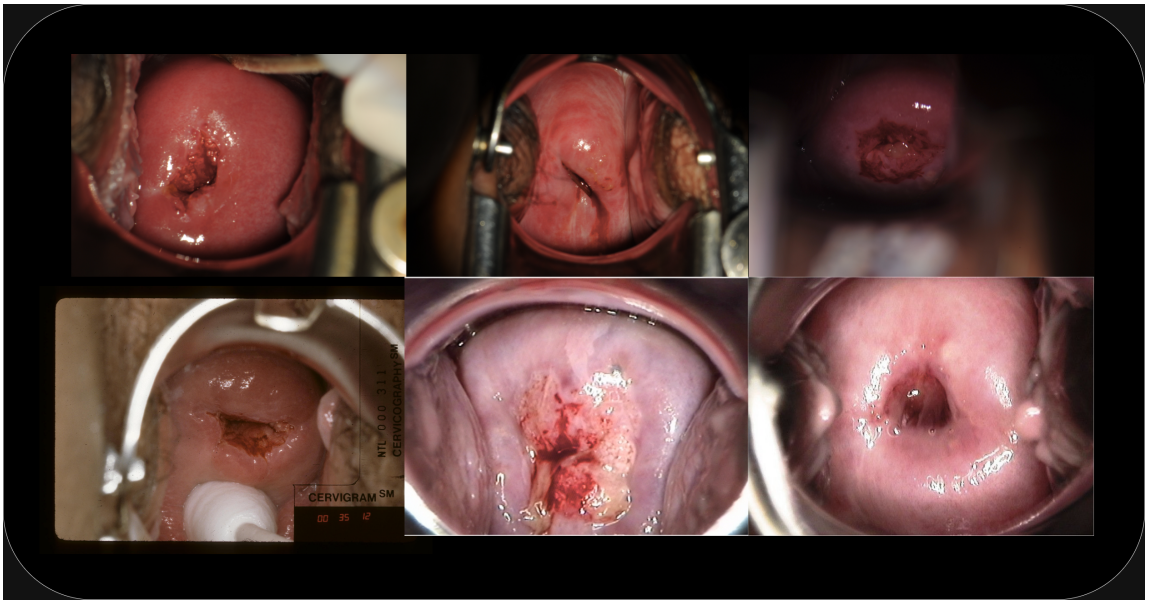


Figura 4: Exploración y visualización de las imágenes de colposcopia.

En la Tabla 1 se presenta una visión detallada de las tres bases de datos de las cuales se obtuvieron datos, así como su distribución según los niveles de displasia, brindando una perspectiva integral del conjunto de imágenes disponibles.

	OMS	**NIH**	**Kaggle**
No Diagnóstico	612	40,186	0
Normal	124	531	244
CIN1	133	192	778
CIN2	308	0	0
CIN3	156	0	0
Cancer	0	0	447
Total	1,333	40,909	1,469
Dimensiones	960 x 720	2,893 x 1,972	2,448 x 3,264

Tabla 1: Distribución por clases de las bases de datos

Es importante destacar que la base de datos del NIH alberga un total de 76,658 imágenes, como se mencionó anteriormente, distribuidas en cuatro carpetas. La repre-

sentación presentada en la Tabla 1 se refiere exclusivamente a una de estas carpetas, pues la calidad de las otras tres carpetas no es muy buena. Por otro lado, se evidencia la razón por la cual la mayoría de las imágenes de esta base de datos fueron descartadas, debido a que, a pesar de esta carpeta contar con 40.909 imágenes, prácticamente su mayoría son casos a los cuales los médicos especialistas no realizaron la respectiva biopsia, con lo cual no es posible asignarles un diagnóstico y en consecuencia imposibilita el poder utilizarlas.

8.3.3. Distribución de Clases

Después de adquirir las imágenes, se llevó a cabo un proceso de categorización basado en su grado de displasia y los criterios especificados por los médicos especialistas de CITOBOT, dando lugar a la formulación de dos problemas de clasificación.

8.3.3.1. Clasificación Dos Clases

El primer desafío consistió en llevar a cabo una clasificación binaria, dividiendo las imágenes en dos clases: "No-Risk" y "High-Risk". La clase "No-Risk" engloba imágenes normales, es decir, aquellas que no exhiben niveles de displasia. En contraste, la clase "High-Risk" incluye todas las categorías de displasia, desde leves hasta graves, así como imágenes con presencia de cáncer, como se muestra en la Figura 5. Esta agrupación se realizó con la consideración de que el agente inteligente debe tener la capacidad de generalizar no solo imágenes completamente saludables, sino también aquellas que presentan diversos grados de enfermedad, para cumplir con las expectativas de los expertos en el proyecto CITOBOT.

Este enfoque de clasificación binaria facilita una comprensión inicial y distintiva de las imágenes, permitiendo al sistema distinguir entre casos sin riesgo aparente y aquellos que sugieren la presencia de posibles anomalías. Además, posibilita la identificación temprana de imágenes que requieren atención y análisis más detallado durante el proceso de evaluación y diagnóstico.

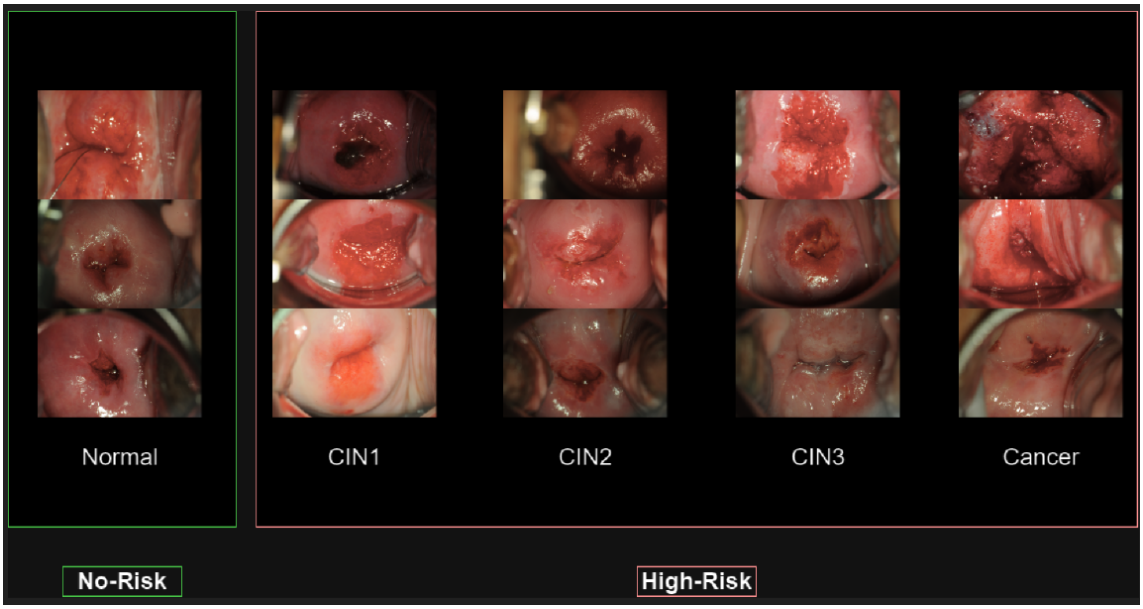


Figura 5: Distribución clasificación de dos clases.

8.3.3.2. Clasificación Tres Clases

Es preciso subrayar que cada tipo de displasia conlleva exámenes y procedimientos específicos, especialmente en los casos de displasia leve y grave. En situaciones de displasia grave, se requiere una mayor rigurosidad en ciertos exámenes, comúnmente incluyendo pruebas de VPH. Por lo tanto, para los expertos surge la necesidad de lograr diferenciar no solo entre imágenes saludables y no saludables, sino también de distinguir entre los tipos de displasia dentro de este último grupo.

Este nuevo desafío de clasificación involucra tres categorías: "No-Risk", "Low-Risk" y "High-Risk". La primera clase sigue representando exclusivamente las imágenes consideradas normales. Ahora, las imágenes de displasia leve CIN1 se clasifican como "Low-Risk", y las imágenes de displasia grave, que abarcan CIN2, CIN3, así como las imágenes de cáncer, se agrupan en la clase "High-Risk". Este enfoque trinario brinda una perspectiva más detallada y diferenciada, permitiendo una evaluación más precisa de las imágenes en términos de riesgo de enfermedad. La Figura 6 pro-

porciona una representación visual de esta clasificación.

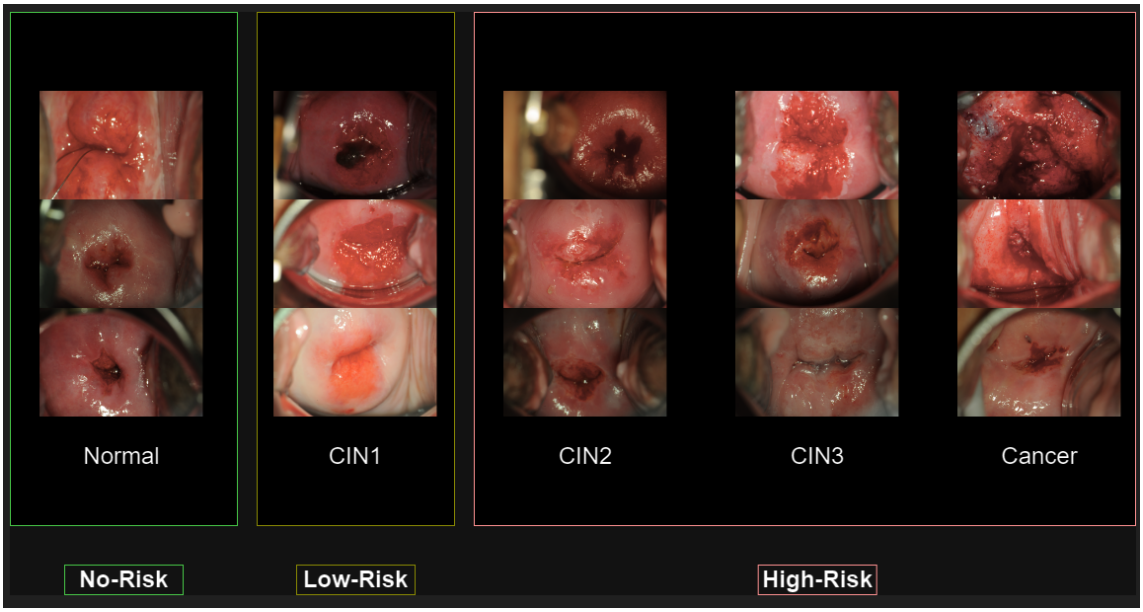


Figura 6: Distribución clasificación de tres clases.

8.3.4. Entorno de Ejecución

El entorno de desarrollo integrado (IDE) utilizado para el proceso fue Google Colaboratory o Google Colab, este es un entorno de notebook Jupyter gratuito que no requiere configuración y se ejecuta íntegramente en la nube. Con esta herramienta, es posible escribir y ejecutar código, guardar y compartir nuestros análisis, y acceder a potentes recursos informáticos realizando la ejecución desde el navegador.

En su versión gratuita, la herramienta dispone de 12.7GB de RAM, 107GB de disco y entorno de GPU limitado. Por lo que, para agilizar los tiempos de ejecución de los modelos se utilizó la versión PRO, que dispone de 51.0GB de RAM del sistema, 166.8GB de disco y 15GB de RAM de GPU.

Dentro de las ventajas que tiene Google Colab, se destaca su compatibilidad con el lenguaje de programación Python (versión más actual), en el cual se realizaron

los análisis y procedimientos de las distintas arquitecturas dentro del campo de Machine/Deep Learning. Para este proceso, se utilizaron librerías propias del lenguaje como: (i) numpy para el manejo de arreglos, matrices y las operaciones entre estas. (ii) opencv para leer y visualizar las imágenes. (iii) matplotlib para graficar los resultados. (iv)sklearn para utilizar las métricas con las cuales se evalúan los resultados. Y, finalmente (v)keras y tensforflow para hacer uso de las diferentes arquitecturas junto a sus parámetros.

Además, como se mencionó anteriormente, Colab cuenta con la posibilidad de agregar un entorno de ejecución mediante GPU, el cual permite reducir tiempos de cómputo y beneficiar la calidad de los resultados.

8.4. Preparación de los Datos

En la crucial fase de preparación de los datos, se llevó a cabo un proceso minucioso para asegurar la calidad y coherencia de la información que empleada en la construcción y entrenamiento de los modelos predictivos. Esto con la finalidad de optimizar la efectividad y fiabilidad de los algoritmos. Durante esta etapa, se realizaron tareas como la limpieza y transformación de los datos para mejorar la representación de la información. Además, se buscó garantizar la cohesión y homogeneidad de las entre las imágenes de diferentes bases de datos, permitiendo así que los modelos se ajusten de manera precisa a los datos preparados. La calidad de los resultados obtenidos en las fases posteriores del proyecto depende en gran medida de la calidad y preparación rigurosa de los datos en esta etapa fundamental.

8.4.1. Base de Datos

A continuación, se detalla la distribución del número de imágenes para cada clase, considerando los dos problemas de clasificación. Es importante recordar que contamos con 1.333 imágenes provenientes de la OMS, 723 imágenes del NIH y 1.469 imágenes obtenidas de Kaggle.

8.4.1.1. Distribución para Dos Clases

Para la clasificación de dos clases, se optó por utilizar exclusivamente las imágenes provenientes de la OMS y del NIH. La integración de las imágenes de Kaggle afectaba negativamente la calidad de los resultados, impidiendo una uniforme incorporación a la base de datos existente. En consecuencia, la distribución de las imágenes se ajustó según lo reflejado en la Tabla 2.

		OMS	**NCI**	**Total**
No-Risk	**Normal**	124	531	655
High-Risk	**CIN1**	133	192	789
	CIN2	308	0	
	CIN3	156	0	
	Cancer	0	0	
	Dimensiones	960 x 720	2,893 x 1,972	

Tabla 2: Distribución de las bases de datos para la clasificación de dos clases.

Como se evidencia, la clase "No-Risk" cuenta con un total de 655 imágenes, mientras que la clase "High-Risk" presenta 789 imágenes, resultando en un desbalanceo que podría impactar negativamente el entrenamiento de los modelos y, en algunos casos, propiciar el sobreajuste (overfitting). Por ende, se tomó la decisión de equilibrar las clases, alcanzando la siguiente configuración:

Dos Clases	
No-Risk	655 imágenes
High-Risk	656 imágenes
Total	1310 imágenes

Tabla 3: Balanceo de clases para la clasificación de dos clases.

8.4.1.2. Distribución para Tres Clases

Para abordar la clasificación de tres clases, se adoptaron estrategias distintas, ya que, al observar la Tabla 2, la distribución inicial de las imágenes provenientes exclu-

sivamente de esas bases de datos resultaría en la clase “No-Risk” con 655 imágenes, la clase “Low-Risk” con 325 imágenes y la clase “High-Risk” con 464 imágenes. Este desequilibrio generaría un total de 325 imágenes por clase al buscar la equidad, lo cual representa una cantidad limitada, especialmente considerando la necesidad de subdividir posteriormente en conjuntos de entrenamiento, validación y prueba.

Con el objetivo de aumentar el número de imágenes, se optó por incorporar las imágenes de la base de datos de Kaggle, a pesar de que los resultados al unificar estos datos con el dataset de dos clases no fueron tan satisfactorios. En este caso, la prioridad era incrementar la cantidad de datos, dado que, en problemas de clasificación con múltiples clases, la escasez de datos puede complicar significativamente la obtención de resultados óptimos. La nueva distribución de los datos quedó configurada de la siguiente manera:

		OMS	**NCI**	**Kaggle**	**Total**
No-Risk	Normal	124	531	244	899
Low-Risk	CIN1	133	192	778	1,103
High-Risk	CIN2	308	0	0	911
	CIN3	156	0	0	
	Cancer	0	0	0	
	Dimensiones	960 x 720	2,893 x 1,972	2,448 x 3,264	

Tabla 4: Distribución de las bases de datos para la clasificación de tres clases.

Al lograr el balanceo de las clases, los datos se distribuyen de la siguiente forma:

Tres Clases	
No-Risk	899 imágenes
Low-Risk	899 imágenes
High-Risk	899 imágenes
Total	2697 imágenes

Tabla 5: Balanceo de clases para la clasificación de tres clases.

8.4.2. Segmentación de Imágenes

En la mayoría de las imágenes, independientemente de su base de datos de origen, se identificó un factor común: la presencia de ruido. Este fenómeno se manifestaba a través de artefactos médicos, fechas u otros datos superfluos que podían distorsionar la visualización del cuello uterino. Durante el proceso de entrenamiento, estos elementos podrían inducir a interpretaciones incorrectas por parte de los algoritmos, ya que podrían ser interpretados como información relevante.

Para mitigar este problema, se llevó a cabo un crucial proceso de segmentación de las imágenes. Esta técnica permitió eliminar el ruido y focalizar la atención únicamente en el área del cuello uterino, como se ilustra detalladamente en la Figura 7. La segmentación no solo contribuye a mejorar la calidad de los datos, sino que también facilita una interpretación más precisa por parte de los algoritmos de Machine/Deep Learning, potenciando así la efectividad de los modelos en el proceso de entrenamiento y predicción. La importancia de este paso radica en la necesidad de proporcionar a los modelos datos limpios y relevantes para lograr resultados más precisos y confiables en la detección temprana del cáncer de cuello uterino.

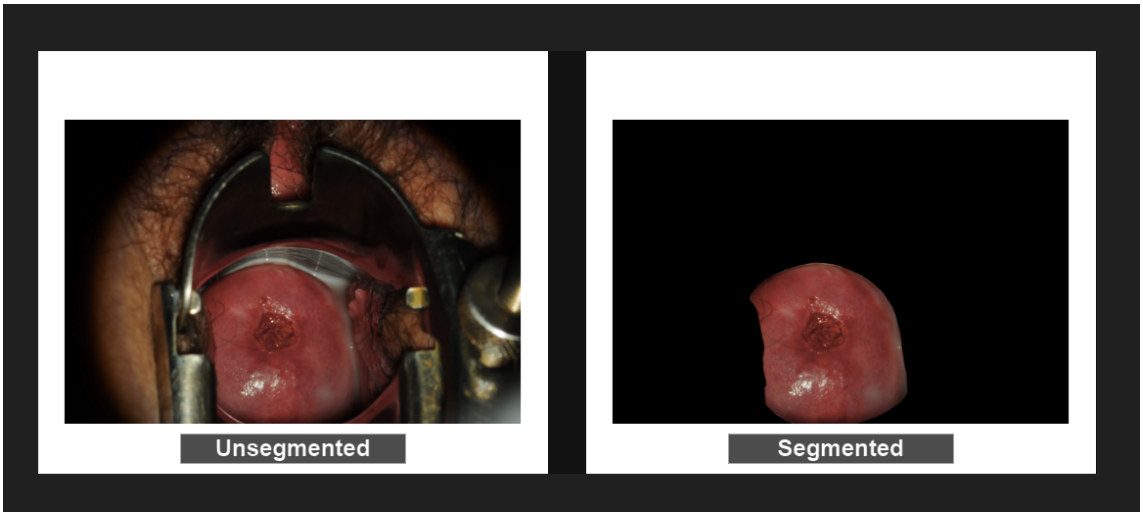


Figura 7: Segmentación de las imágenes.

8.4.3. Procesar y Cargar Imágenes

Ahora bien, como se observa en la Tabla 1, las imágenes de cada base de datos tienen un tamaño diferente, lo que motiva la necesidad de estandarizarlos a una medida uniforme con el objetivo de homogeneizar las imágenes. La selección de este tamaño estándar es de gran importancia ya que influye en el desempeño y la eficiencia de los modelos. Dada la naturaleza de las arquitecturas de modelos a utilizar, caracterizadas por su peso computacional, es esencial evitar tamaños excesivamente grandes que generen modelos pesados y poco prácticos para su integración y exportación en una aplicación móvil. Sin embargo, reducir el tamaño de manera significativa podría resultar en la pérdida de información valiosa al eliminar píxeles importantes. En busca de un equilibrio, se optó finalmente por un tamaño estándar ampliamente utilizado en el estado del arte: 224x224 píxeles. Este tamaño proporciona una solución equilibrada, facilitando la eficacia del modelo y su posterior implementación en una aplicación móvil.

De esta forma, se realiza una función que lleva a cabo una serie de transformaciones fundamentales en las imágenes con el propósito de prepararlas para su posterior procesamiento en modelos de aprendizaje automático. En primer lugar, la función utiliza TensorFlow para leer el archivo de la imagen y decodificar el formato JPEG. Posteriormente, la imagen se redimensiona a una dimensión de 224x224 píxeles como se definió. Seguidamente, se convierte la imagen a escala de grises mediante el uso de tf.image.rgb_to_grayscale, reduciendo la complejidad de los datos al prescindir de los canales de color RGB. Para asegurar la compatibilidad con los modelos, la imagen se convierte al tipo de datos float32. Finalmente, se normalizan los píxeles al rango [0, 1] mediante la división por 255.0, asegurando que los valores de los píxeles estén dentro de un rango adecuado para su procesamiento en modelos de aprendizaje profundo. Estas transformaciones garantizan que las imágenes estén listas y adaptadas para su entrada en modelos de clasificación o detección.

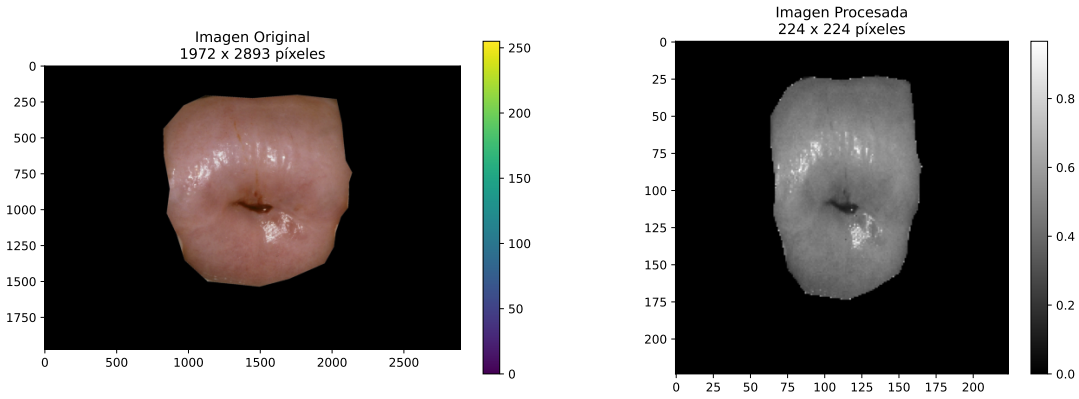


Figura 8: Imágenes procesadas.

8.4.4. Aumento de Datos

Generalmente, para que los modelos enfocados en la clasificación de imágenes puedan desempeñarse de manera eficiente en sus tareas, requieren de un conjunto extenso de imágenes para el entrenamiento. Sin embargo, esta necesidad se convierte en una limitación en nuestro caso, dada la naturaleza de los datos, que son escasos y difíciles de obtener debido a las restricciones de permisos. Para abordar esta limitación, se implementó la técnica de Data-Augmentation (aumento de datos), que consiste en generar variantes de las imágenes existentes mediante transformaciones y manipulaciones controladas. Esta estrategia ayuda a mejorar la generalización y la capacidad de los modelos para reconocer patrones en nuevas instancias. En el contexto del aprendizaje profundo, donde se requieren grandes cantidades de datos para entrenar modelos complejos, el aumento de datos es una herramienta valiosa para aprovechar al máximo conjuntos de datos limitados.

En primera instancia se logra una diversificación del conjunto de datos, pues al aplicar transformaciones como rotaciones, zoom, desplazamientos y reflejos, se generan múltiples versiones de una imagen original, introduciendo variabilidad y diversidad en el conjunto de datos. Del mismo modo, se ayuda a prevenir el sobreajuste al exponer el modelo a una mayor variabilidad durante el entrenamiento. Esto per-

mite que el modelo aprenda patrones más robustos y aplicables a nuevas muestras. Además, al generar instancias adicionales de datos, se reduce la probabilidad de que el modelo memorice el conjunto de entrenamiento, lo que es particularmente útil en conjuntos de datos pequeños como el nuestro.

Para lograr esto, se definió un generador de datos de imagen utilizando la biblioteca Keras. Los parámetros especificados controlan las transformaciones que se aplicarán a las imágenes durante el entrenamiento:

- **rotation_range:** Rango de ángulos de rotación en grados.
- **zoom_range:** Rango para aplicar zoom aleatorio a las imágenes.
- **width_shift_range y height_shift_range:** Rangos de desplazamiento horizontal y vertical aleatorio.
- **shear_range:** Rango de deformación.
- **horizontal_flip y vertical_flip:** Indican si se aplicará volteo horizontal o vertical de manera aleatoria.

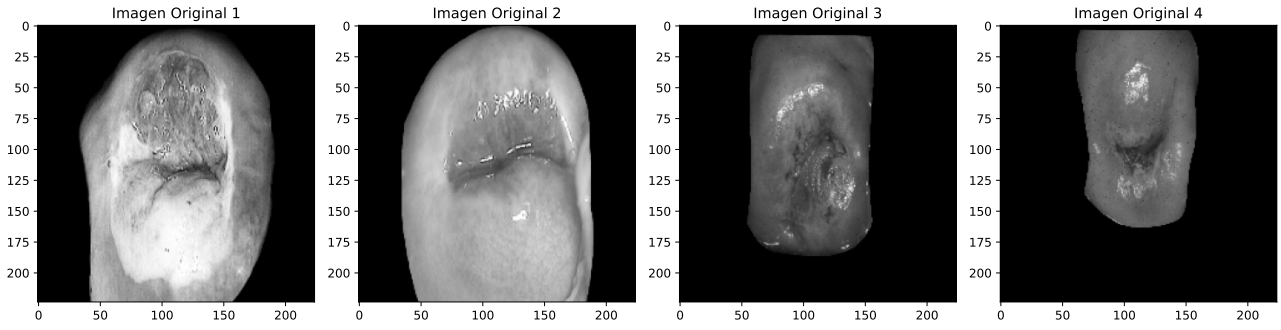


Figura 9: Imágenes originales.

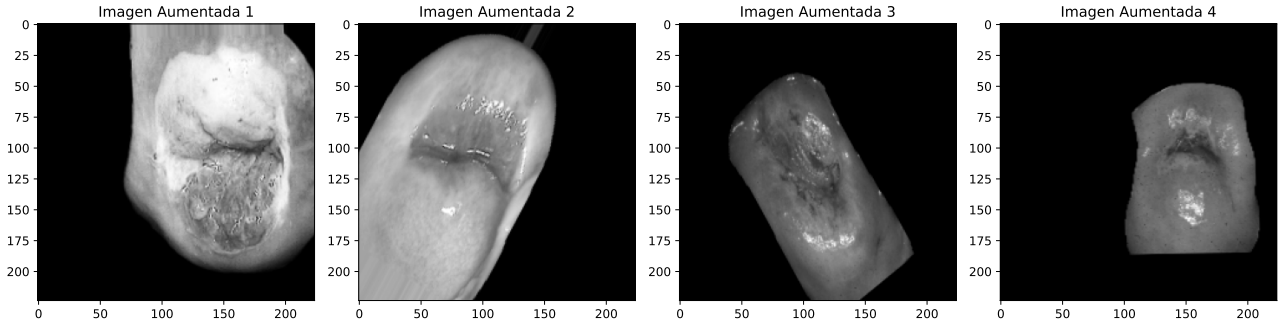


Figura 10: Imágenes generadas por medio de aumento de datos.

8.4.5. Datos de Entrenamiento, Validación y Pruebas

En cuanto a la partición del conjunto de datos, se asignó una división del 80 % para entrenamiento (Train) y el 20 % restante se reservó para pruebas (Test). Además, se creó un conjunto adicional llamado validación (Validation) a partir del conjunto de datos de entrenamiento. Esta división permite evaluar el desempeño del modelo en un entorno de validación independiente, contribuyendo a la selección de hiperparámetros óptimos y mejorando la confiabilidad de las métricas de desempeño del modelo.

Cabe resaltar que, para realizar esta tarea se usa la función train_test_split, la cual es propia de Python, y permite la división de los datos se realice de manera aleatoria, preservando la proporción de las clases existentes en el conjunto de datos original. Esta práctica es esencial para evaluar la capacidad de generalización del modelo, ya que proporciona un conjunto independiente y no visto previamente durante la fase de entrenamiento. Al asignar parte de los datos a un conjunto de prueba, podemos medir con mayor precisión la capacidad del modelo para generalizar a nuevos datos, lo cual es crucial para evaluar su rendimiento en situaciones del mundo real. En consecuencia, la estructura del conjunto de datos fue la siguiente:

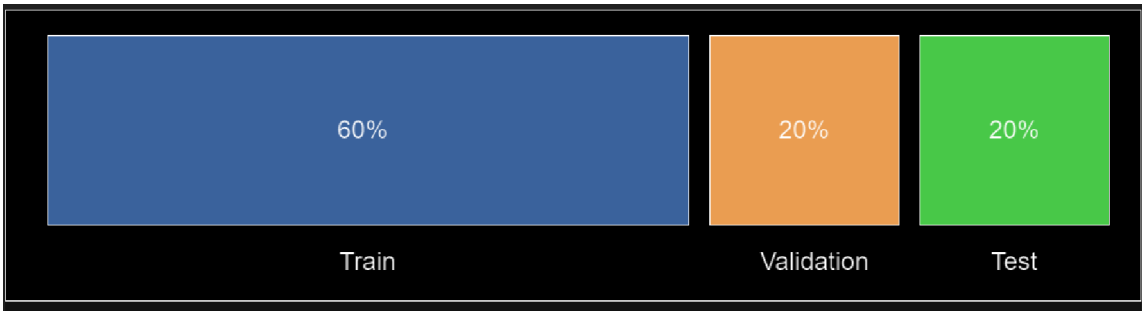


Figura 11: Partición del dataset en Train, Validation y Test.

De modo que, en la Tabla 6, observamos como quedo esta distribución para cada clase:

	Train	**Validation**	**Test**	**Total**
Dos Clases	838	210	262	1,310
Tres Clases	1,725	432	540	2,697

Tabla 6: Partición del dataset en Train, Validation y Test para cada tipo de clasificación.

8.4.6. Técnica One-Hot-Enconding

Antes de alimentar los datos al modelo, se realiza una última transformación en las etiquetas de las imágenes. Con el propósito de mejorar el rendimiento, se emplea comúnmente la técnica conocida como one-hot-encoding. Esta técnica consiste en convertir las etiquetas categóricas en un formato binario, donde cada categoría única se representa como un vector de bits. En otras palabras, se asigna un valor binario a cada categoría posible, y solo un bit en cada vector es "encendido" para indicar la clase correspondiente.

Esta representación numérica única de las etiquetas categóricas facilita la interpretación para los modelos de aprendizaje automático. Algunos de los beneficios de utilizar one-hot-encoding incluyen la eliminación de la ambigüedad inherente a las etiquetas categóricas y la mejora en la capacidad del modelo para comprender la

relación entre las clases. Además, esta técnica permite una manipulación matemática más eficiente de las etiquetas, contribuyendo así a un mejor rendimiento general durante el entrenamiento y la evaluación del modelo. De modo que, las etiquetas quedan de la siguiente forma:

Dos Clases		
Clase	**Etiqueta**	**Etiqueta One-Hot-Encoding**
No-Risk	0	[1,0]
High-Risk	1	[0,1]

Tabla 7: Etiquetas one-hot-encoding para las clases de la clasificación binaria.

Tres Clases		
Clase	**Etiqueta**	**Etiqueta One-Hot-Encoding**
No-Risk	0	[1,0,0]
Low-Risk	1	[0,1,0]
High-Risk	2	[0,0,1]

Tabla 8: Etiquetas one-hot-encoding para las clases de la clasificación ternaria.

8.5. Modelado

Luego de completar la fase de preprocesamiento de datos, avanzamos hacia la selección e implementación de modelos que nos posibilitaran generar predicciones óptimas. Para abordar eficazmente la tarea de clasificación de imágenes, optamos por emplear redes convolucionales, reconocidas por su eficacia en este ámbito. Específicamente, nos centramos en arquitecturas ampliamente reconocidas en el estado del arte por su capacidad para manejar imágenes médicas. Paralelamente, implementamos técnicas de optimización de modelos, ajustando hiperparámetros y empleando métodos de validación para asegurar un rendimiento óptimo.

8.5.1. Arquitecturas Utilizadas

A continuación, se describen las arquitecturas y técnicas utilizadas para la clasificación de imágenes de cáncer de cuello uterino. Es importante destacar que todos los

modelos comparten la misma entrada (independientemente de cuál sea el tipo de clasificación), es decir, imágenes de 224x224 píxeles. La distinción principal entre ellos reside en la capa de salida. Dado que, como se empleó la técnica one-hot-encoding, entonces el número de clases en la capa de salida varía entre 2 o 3, dependiendo si se trata de una clasificación binaria o ternaria, respectivamente.

8.5.1.1. AlexNet

AlexNet es una arquitectura pionera en redes neuronales convolucionales (CNN) diseñada para abordar desafíos complejos de clasificación de imágenes. Su estructura, propuesta por Alex Krizhevsky, incorpora capas convolucionales profundas, seguidas de capas de max-pooling para reducir la dimensionalidad y extraer características distintivas. La red consta de cinco capas convolucionales, algunas de las cuales están seguidas por capas de max-pooling para aprender jerarquías complejas de características visuales. AlexNet también introduce el uso de funciones de activación ReLU para mejorar la convergencia y superar el problema de desvanecimiento del gradiente. Con capas totalmente conectadas al final, la red realiza la clasificación final. Además, el modelo utiliza técnicas como la regularización mediante la eliminación de nodos (Dropout) para evitar el sobreajuste.

Concretamente se implementó esta red como se muestra en la Figura 12 y en el Anexo C, la red comienza con una capa convolucional de 96 filtros (kernels) con un tamaño de 11x11 y una función de activación ReLU. Las capas de max-pooling (capas de reducción de dimensionalidad) con tamaños de ventana 3x3 y pasos (strides) de 2x2 siguen a las capas convolucionales. Luego, se incorporan capas convolucionales adicionales con tamaños de filtro decrecientes (256, 384, 384, 256) para aprender representaciones jerárquicas más complejas. El modelo utiliza la función de activación ReLU en estas capas para introducir no linealidades. Después de las capas convolucionales, se agrega una capa Flatten para convertir la salida tridimensional en un vector unidimensional, seguida por dos capas densas de 4096 nodos con activación ReLU y capas de Dropout para evitar el sobreajuste. Finalmente, la capa de salida

utiliza la función de activación softmax para permitir la clasificación entre dos/tres clases y utiliza la función de pérdida de entropía cruzada categórica para la optimización. Este modelo se compila con el optimizador Adam y se presenta un resumen detallado de la arquitectura y parámetros del modelo.

```python
#Arquitectura AlexNet
import tensorflow as tf
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Dense,Dropout,Activation,Conv2D, MaxPooling2D,Flatten,BatchNormalization

def modelCompile():
  model = Sequential()
  model.add(Conv2D(96, (11, 11), strides=(4, 4), input_shape=(224,224,1), padding='valid', activation='relu',
                   kernel_initializer='uniform'))
  model.add(MaxPooling2D(pool_size=(3, 3), strides=(2, 2)))
  model.add(Conv2D(256, (5, 5), strides=(1, 1), padding='same', activation='relu', kernel_initializer='uniform'))
  model.add(MaxPooling2D(pool_size=(3, 3), strides=(2, 2)))
  model.add(Conv2D(384, (3, 3), strides=(1, 1), padding='same', activation='relu', kernel_initializer='uniform'))
  model.add(Conv2D(384, (3, 3), strides=(1, 1), padding='same', activation='relu', kernel_initializer='uniform'))
  model.add(Conv2D(256, (3, 3), strides=(1, 1), padding='same', activation='relu', kernel_initializer='uniform'))
  model.add(MaxPooling2D(pool_size=(3, 3), strides=(2, 2)))
  model.add(Flatten())
  model.add(Dense(4096, activation='relu'))
  model.add(Dropout(0.5))
  model.add(Dense(4096, activation='relu'))
  model.add(Dropout(0.5))
  model.add(Dense(2, activation='softmax'))
  model.compile(loss='categorical_crossentropy', optimizer=tf.keras.optimizers.Adam(learning_rate=1e-4), metrics=['accuracy'])
  return model

model = modelCompile()
model.summary()
```

Figura 12: Código de la red AlexNet implementada en Python.

8.5.1.2. GoogleNet

También conocida como Inception, es una red neuronal convolucional desarrollada por Google. Su innovador diseño se centra en la eficiencia y la capacidad de aprendizaje profundo. Una característica distintiva de GoogleNet es la introducción del módulo "Inception", que utiliza múltiples filtros de diferentes tamaños en paralelo para capturar patrones a diferentes escalas en la misma capa. Esto permite que la red aprenda representaciones más ricas y complejas de las imágenes.

La arquitectura Inception tiene múltiples capas convolucionales, seguidas por módulos Inception, y utiliza capas de reducción de dimensionalidad para disminuir la carga computacional. Además, incorpora conexiones residuales para abordar el

problema del desvanecimiento del gradiente, permitiendo un entrenamiento más efectivo de redes profundas. La capacidad de GoogleNet para aprender representaciones jerárquicas y su eficiencia computacional la ha convertido en una opción popular para tareas de clasificación de imágenes en conjunto con otras aplicaciones. Su diseño modular y la utilización de bloques Inception la hacen especialmente adecuada para aprender patrones complejos en datos visuales.

De modo que, nuestra implementación sigue el diseño modular de la arquitectura Inception, utilizando funciones para definir capas convolucionales con normalización por lotes (Batch Normalization) y el bloque Inception en sí mismo. La función Conv2d_BN define una capa convolucional seguida de Batch Normalization, aplicando activación ReLU. Luego, la función Inception construye un bloque Inception combinando diferentes tamaños de filtros en paralelo y concatenándolos. Este bloque incluye convoluciones 1x1, 3x3, 5x5 y una capa de agrupación máxima (max pooling), cada una seguida de Batch Normalization.

El modelo general comienza con una capa de entrada de tamaño (224, 224, 1) y sigue con capas convolucionales y bloques Inception. La última parte del modelo consiste en capas de agrupación y bloques Inception adicionales. Finalmente, se aplica una capa de agrupación promedio (average pooling), se aplica Dropout para la regularización, se aplana la salida y se conecta a dos capas densas para la clasificación final, y en el caso de clasificación ternaria se conecta a tres capas densas, nótese la arquitectura en el Anexo D.

```python
#Arquitectura googlenet
import tensorflow as tf
from tensorflow import keras
from keras.models import Sequential, Model
from keras.layers import Dense,Dropout,Activation,Conv2D, MaxPooling2D,Flatten,BatchNormalization, concatenate, Input, AveragePooling2D


def Conv2d_BN(x, nb_filter, kernel_size, padding='same', strides=(1, 1), name=None):
    if name is not None:
        bn_name = name + '_bn'
        conv_name = name + '_conv'
    else:
        bn_name = None
        conv_name = None

    x = Conv2D(nb_filter, kernel_size, padding=padding, strides=strides, activation='relu', name=conv_name)(x)
    x = BatchNormalization(axis=3, name=bn_name)(x)
    return x


def Inception(x, nb_filter):
    branch1x1 = Conv2d_BN(x, nb_filter, (1, 1), padding='same', strides=(1, 1), name=None)

    branch3x3 = Conv2d_BN(x, nb_filter, (1, 1), padding='same', strides=(1, 1), name=None)
    branch3x3 = Conv2d_BN(branch3x3, nb_filter, (3, 3), padding='same', strides=(1, 1), name=None)

    branch5x5 = Conv2d_BN(x, nb_filter, (1, 1), padding='same', strides=(1, 1), name=None)
    branch5x5 = Conv2d_BN(branch5x5, nb_filter, (1, 1), padding='same', strides=(1, 1), name=None)

    branchpool = MaxPooling2D(pool_size=(3, 3), strides=(1, 1), padding='same')(x)
    branchpool = Conv2d_BN(branchpool, nb_filter, (1, 1), padding='same', strides=(1, 1), name=None)

    x = concatenate([branch1x1, branch3x3, branch5x5, branchpool], axis=3)

    return x

def modelCompile():
  inpt = Input(shape=(224,224, 1))
  x = Conv2d_BN(inpt, 64, (7, 7), strides=(2, 2), padding='same')
  x = MaxPooling2D(pool_size=(3, 3), strides=(2, 2), padding='same')(x)
  x = Conv2d_BN(x, 192, (3, 3), strides=(1, 1), padding='same')
  x = MaxPooling2D(pool_size=(3, 3), strides=(2, 2), padding='same')(x)
  x = Inception(x, 64)  # 256
  x = Inception(x, 120)  # 480
  x = MaxPooling2D(pool_size=(3, 3), strides=(2, 2), padding='same')(x)
  x = Inception(x, 128)  # 512
  x = Inception(x, 128)
  x = Inception(x, 128)
  x = Inception(x, 132)  # 528
  x = Inception(x, 208)  # 832
  x = MaxPooling2D(pool_size=(3, 3), strides=(2, 2), padding='same')(x)
  x = Inception(x, 208)
  x = Inception(x, 256)  # 1024
  x = AveragePooling2D(pool_size=(7, 7), strides=(7, 7), padding='same')(x)
  x = Dropout(0.7)(x)
  x = Flatten()(x)
  x = Dense(1000, activation='relu')(x)
  x = Dense(2, activation='softmax')(x)

  model = Model(inpt, x, name='inception')
  #model = multi_gpu_model(model, gpus=2)

  # model = multi_gpu_model(model,4)
  #model.compile(loss='categorical_crossentropy', optimizer='sgd', metrics=['accuracy'])
  model.compile(loss='categorical_crossentropy', optimizer=tf.keras.optimizers.Adam(learning_rate=1e-4), metrics=['accuracy'])
  return model

model = modelCompile()
model.summary()
```

Figura 13: Código de la red GoogleNet implementada en Python.

8.5.1.3. ResNet

La arquitectura ResNet, que significa "Redes Residuales", introdujo una innovadora estructura de bloques residuales para abordar el desafío del entrenamiento profundo. A diferencia de las arquitecturas convencionales que siguen un enfoque de apilamiento de capas, ResNet utiliza bloques residuales que incorporan conexiones de atajos, permitiendo que la información fluya directamente a través de las capas sin procesamiento adicional. Cada bloque consta de una ruta principal que realiza transformaciones en la entrada y una ruta de atajo que proporciona una conexión directa entre las capas de entrada y salida.

Como se observa en la Figura 14 y en el Axeno E, la imagen de entrada pasa por una capa de convolución y max pooling para extraer características iniciales. Luego, se emplean bloques residuales que contienen capas convolucionales con conexiones de atajos. Estas conexiones facilitan el flujo del gradiente y mitigan los problemas de desvanecimiento del gradiente en redes profundas. La red se compone de varias repeticiones de estos bloques, incrementando gradualmente el número de filtros en cada bloque para aprender representaciones más complejas de las imágenes. Finalmente, después de múltiples capas, se realiza un promedio de pooling seguido de una capa totalmente conectada con una función softmax para la clasificación binaria. La red se entrena con la función de pérdida de entropía cruzada categórica y utiliza el optimizador Adam para el aprendizaje, logrando un equilibrio entre la profundidad del modelo y la eficiencia del entrenamiento.

```python
#Arquitectura Resnet
import tensorflow as tf
from tensorflow import keras
from keras.models import Sequential,Model
from keras.layers import Dense,Dropout,Activation,Conv2D, MaxPooling2D,Flatten,Input,BatchNormalization,AveragePooling2D,concatenate,ZeroPadding2D,add

def Conv2d_BN(x, nb_filter, kernel_size, strides=(1, 1), padding='same', name=None):
    if name is not None:
        bn_name = name + '_bn'
        conv_name = name + '_conv'
    else:
        bn_name = None
        conv_name = None

    x = Conv2D(nb_filter, kernel_size, padding=padding, strides=strides, activation='relu', name=conv_name)(x)
    x = BatchNormalization(axis=3, name=bn_name)(x)
    return x


def Conv_Block(inpt, nb_filter, kernel_size, strides=(1, 1), with_conv_shortcut=False):
    x = Conv2d_BN(inpt, nb_filter=nb_filter[0], kernel_size=(1, 1), strides=strides, padding='same')
    x = Conv2d_BN(x, nb_filter=nb_filter[1], kernel_size=(3, 3), padding='same')
    x = Conv2d_BN(x, nb_filter=nb_filter[2], kernel_size=(1, 1), padding='same')
    if with_conv_shortcut:
        shortcut = Conv2d_BN(inpt, nb_filter=nb_filter[2], strides=strides, kernel_size=kernel_size)
        x = add([x, shortcut])
        return x
    else:
        x = add([x, inpt])
        return x

def modelCompile():
  inpt = Input(shape=(224,224,1))
  x = ZeroPadding2D((3, 3))(inpt)
  x = Conv2d_BN(x, nb_filter=64, kernel_size=(7, 7), strides=(2, 2), padding='valid')
  x = MaxPooling2D(pool_size=(3, 3), strides=(2, 2), padding='same')(x)

  x = Conv_Block(x, nb_filter=[64, 64, 256], kernel_size=(3, 3), strides=(1, 1), with_conv_shortcut=True)
  x = Conv_Block(x, nb_filter=[64, 64, 256], kernel_size=(3, 3))
  x = Conv_Block(x, nb_filter=[64, 64, 256], kernel_size=(3, 3))

  x = Conv_Block(x, nb_filter=[128, 128, 512], kernel_size=(3, 3), strides=(2, 2), with_conv_shortcut=True)
  x = Conv_Block(x, nb_filter=[128, 128, 512], kernel_size=(3, 3))
  x = Conv_Block(x, nb_filter=[128, 128, 512], kernel_size=(3, 3))
  x = Conv_Block(x, nb_filter=[128, 128, 512], kernel_size=(3, 3))

  x = Conv_Block(x, nb_filter=[256, 256, 1024], kernel_size=(3, 3), strides=(2, 2), with_conv_shortcut=True)
  x = Conv_Block(x, nb_filter=[256, 256, 1024], kernel_size=(3, 3))
  x = Conv_Block(x, nb_filter=[256, 256, 1024], kernel_size=(3, 3))
  x = Conv_Block(x, nb_filter=[256, 256, 1024], kernel_size=(3, 3))
  x = Conv_Block(x, nb_filter=[256, 256, 1024], kernel_size=(3, 3))
  x = Conv_Block(x, nb_filter=[256, 256, 1024], kernel_size=(3, 3))

  x = Conv_Block(x, nb_filter=[512, 512, 2048], kernel_size=(3, 3), strides=(2, 2), with_conv_shortcut=True)
  x = Conv_Block(x, nb_filter=[512, 512, 2048], kernel_size=(3, 3))
  x = Conv_Block(x, nb_filter=[512, 512, 2048], kernel_size=(3, 3))
  x = AveragePooling2D(pool_size=(2, 2))(x)
  x = Flatten()(x)
  x = Dense(2, activation='softmax')(x)

  model = Model(inputs=inpt, outputs=x)

  #sgd = SGD(decay=0.0001, momentum=0.9)
  #model = multi_gpu_model(model, gpus=1)

  model.compile(loss='categorical_crossentropy', optimizer=tf.keras.optimizers.Adam(learning_rate=1e-4), metrics=['accuracy'])
  return model

model = modelCompile()
model.summary()
```

Figura 14: Código de la red ResNet implementada en Python.

8.5.1.4. U-Net

segmentación semántica de imágenes, especialmente en tareas de segmentación de imágenes médicas. Su estructura única se asemeja a la letra "U" en su diseño, con una parte descendente (encoder) y una parte ascendente (decoder). En la fase de codificación, las capas convolucionales y de pooling capturan características de

alto nivel en la imagen, reduciendo su resolución espacial. La fase de decodificación reconstruye la imagen segmentada a partir de las características de bajo y alto nivel mediante capas de upsampling y convoluciones transpuestas. La innovación radica en la conexión directa de las capas del encoder a las del decoder mediante conexiones skip (también conocidas como conexiones residuales o shortcuts), que permiten la recuperación de detalles finos.

En la Figura 15 y en el Anexo F, se comienza definiendo la entrada del modelo con una imagen de tamaño (224, 224, 1). Luego, sigue el camino de contracción, donde se aplican capas convolucionales seguidas de normalización por lotes, activación ReLU y regularización Dropout. Después de cada par de capas convolucionales, se realiza un muestreo máximo para reducir las dimensiones espaciales. En el camino expansivo, se utilizan capas de transposición convolucional para realizar upsampling y se concatenan las salidas del camino de contracción correspondientes. Se aplican nuevamente capas convolucionales, normalización por lotes y dropout. La capa de salida consiste en una capa densa con activación softmax para clasificar píxeles en dos clases. El modelo se compila con la función de pérdida 'categorical_crossentropy' y el optimizador Adam.

```python
#Modelo UNET
import tensorflow as tf
from tensorflow import keras
from keras.models import Sequential, Model
from keras.layers import Dense, Flatten, Activation, Dropout, BatchNormalization, concatenate, Lambda, Input, Conv2D, MaxPooling2D, Conv2DTranspose

def modelCompile():
  # Definimos la entrada al modelo
  Image_input = Input((224, 224, 1))
  Image_in = Image_input

  #contracting path
  conv1 = Conv2D(64, (3, 3), padding='same')(Image_in)
  conv1 = BatchNormalization()(conv1)
  conv1 = Activation('relu')(conv1)
  conv1 = Dropout(0.1)(conv1)

  conv1 = Conv2D(64, (3, 3), padding='same')(conv1)
  conv1 = BatchNormalization()(conv1)
  conv1 = Activation('relu')(conv1)
  maxp1 = MaxPooling2D((2, 2))(conv1)

  conv2 = Conv2D(128, (3, 3), padding='same')(maxp1)
  conv2 = BatchNormalization()(conv2)
  conv2 = Activation('relu')(conv2)
  conv2 = Dropout(0.1)(conv2)

  conv2 = Conv2D(128, (3, 3), padding='same')(conv2)
  conv2 = BatchNormalization()(conv2)
  conv2 = Activation('relu')(conv2)
  maxp2 = MaxPooling2D((2, 2))(conv2)

  conv3 = Conv2D(256, (3, 3), padding='same')(maxp2)
  conv3 = BatchNormalization()(conv3)
  conv3 = Activation('relu')(conv3)
  conv3 = Dropout(0.2)(conv3)

  conv3 = Conv2D(256, (3, 3), padding='same')(conv3)
  conv3 = BatchNormalization()(conv3)
  conv3 = Activation('relu')(conv3)
  maxp3 = MaxPooling2D((2, 2))(conv3)

  conv4 = Conv2D(512, (3, 3), padding='same')(maxp3)
  conv4 = BatchNormalization()(conv4)
  conv4 = Activation('relu')(conv4)
  conv4 = Dropout(0.2)(conv4)

  conv4 = Conv2D(512, (3, 3), padding='same')(conv4)
  conv4 = BatchNormalization()(conv4)
  conv4 = Activation('relu')(conv4)
  maxp4 = MaxPooling2D(pool_size=(2, 2))(conv4)

  conv5 = Conv2D(1024, (3, 3), padding='same')(maxp4)
  conv5 = BatchNormalization()(conv5)
  conv5 = Activation('relu')(conv5)
  conv5 = Dropout(0.3)(conv5)

  conv5 = Conv2D(1024, (3, 3), padding='same')(conv5)
  conv5 = BatchNormalization()(conv5)
  conv5 = Activation('relu')(conv5)

  #expansive path
  up6 = Conv2DTranspose(512, (2, 2), strides=(2, 2), padding='same')(conv5)
  up6 = concatenate([up6, conv4])
  conv6 = Conv2D(512, (3, 3), activation='relu', padding='same')(up6)
  conv6 = BatchNormalization()(conv6)
  conv6 = Dropout(0.2)(conv6)
  conv6 = Conv2D(512, (3, 3), activation='relu', padding='same')(conv6)
  conv6 = BatchNormalization()(conv6)

  up7 = Conv2DTranspose(256, (2, 2), strides=(2, 2), padding='same')(conv6)
  up7 = concatenate([up7, conv3])
  conv7 = Conv2D(256, (3, 3), activation='relu', padding='same')(up7)
  conv7 = BatchNormalization()(conv7)
  conv7 = Dropout(0.2)(conv7)
  conv7 = Conv2D(256, (3, 3), activation='relu', padding='same')(conv7)
  conv7 = BatchNormalization()(conv7)

  up8 = Conv2DTranspose(128, (2, 2), strides=(2, 2), padding='same')(conv7)
  up8 = concatenate([up8, conv2])
  conv8 = Conv2D(128, (3, 3), activation='relu', padding='same')(up8)
  conv8 = BatchNormalization()(conv8)
  conv8 = Dropout(0.1)(conv8)
  conv8 = Conv2D(128, (3, 3), activation='relu', padding='same')(conv8)
  conv8 = BatchNormalization()(conv8)

  up9 = Conv2DTranspose(64, (2, 2), strides=(2, 2), padding='same')(conv8)
  up9 = concatenate([up9, conv1])
  conv9 = Conv2D(64, (3, 3), activation='relu', padding='same')(up9)
  conv9 = BatchNormalization()(conv9)
  conv9 = Dropout(0.1)(conv9)
  conv9 = Conv2D(64, (3, 3), activation='relu', padding='same')(conv9)
  conv9 = BatchNormalization()(conv9)

  conv10 = Flatten()(conv9)
  outputs = Dense(2, activation='softmax')(conv10)

  model = Model(inputs=[Image_input], outputs=[outputs])
  model.compile(loss='categorical_crossentropy', optimizer=tf.keras.optimizers.Adam(learning_rate=1e-4), metrics=['accuracy'])
  return model

model = modelCompile()
model.summary()
```


Figura 15: Código de la red U-Net implementada en Python.

8.5.1.5. VGGNET

También conocida como Visual Geometry Group Network, es una arquitectura de red neuronal convolucional desarrollada por el Grupo de Geometría Visual de la Universidad de Oxford. Fue propuesta por los investigadores Karen Simonyan y Andrew Zisserman en 2014. La VGGNet se destacó por su simplicidad y profundidad, estableciendo nuevos estándares en la construcción de redes neuronales convolucionales más profundas.

La característica distintiva de la VGGNet es su enfoque en tener capas convolucionales con filtros de tamaño muy pequeño (3x3), pero con una profundidad significativa. Su arquitectura se caracteriza por la repetición de bloques de capas convolucionales seguidos por capas de pooling, culminando en capas completamente conectadas. La variante más conocida es la VGG16, que tiene 16 capas (13 convolucionales y 3 completamente conectadas), y la VGG19, que tiene 19 capas.

La arquitectura implementada del Anexo G comienza con un par de capas convolucionales con 64 filtros de 3x3, seguidas de una capa de pooling max de 2x2 para reducir las dimensiones espaciales. Luego, se repite un patrón similar dos veces: dos capas convolucionales con 128 filtros y 3x3 seguidas de una capa de pooling max. Posteriormente, se repite el mismo patrón con tres capas convolucionales, cada una con 256 filtros de 3x3. Después, se tiene una repetición de tres capas convolucionales con 512 filtros de 3x3. La última sección de capas convolucionales consta de tres capas con 512 filtros de 3x3. Luego, la red se aplana y se conecta a dos capas densas de 2048 unidades con funciones de activación ReLU, seguidas de capas de Dropout para prevenir el sobreajuste. Finalmente, se tiene una capa densa de salida con 2 unidades y activación softmax para la clasificación binaria.

```python
#Arquitectura VGGNET
import tensorflow as tf
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Dense,Dropout,Activation,Conv2D, MaxPooling2D,Flatten,BatchNormalization
#from keras.utils import np_utils,multi_gpu_model

def modelCompile():
  model = Sequential()
  model.add(Conv2D(64, (3, 3), strides=(1, 1), input_shape=(224,224,1), padding='same', activation='relu',
                   kernel_initializer='uniform'))
  model.add(Conv2D(64, (3, 3), strides=(1, 1), padding='same', activation='relu', kernel_initializer='uniform'))
  model.add(MaxPooling2D(pool_size=(2, 2)))
  model.add(Conv2D(128, (3, 2), strides=(1, 1), padding='same', activation='relu', kernel_initializer='uniform'))
  model.add(Conv2D(128, (3, 3), strides=(1, 1), padding='same', activation='relu', kernel_initializer='uniform'))
  model.add(MaxPooling2D(pool_size=(2, 2)))
  model.add(Conv2D(256, (3, 3), strides=(1, 1), padding='same', activation='relu', kernel_initializer='uniform'))
  model.add(Conv2D(256, (3, 3), strides=(1, 1), padding='same', activation='relu', kernel_initializer='uniform'))
  model.add(Conv2D(256, (3, 3), strides=(1, 1), padding='same', activation='relu', kernel_initializer='uniform'))
  model.add(MaxPooling2D(pool_size=(2, 2)))
  model.add(Conv2D(512, (3, 3), strides=(1, 1), padding='same', activation='relu', kernel_initializer='uniform'))
  model.add(Conv2D(512, (3, 3), strides=(1, 1), padding='same', activation='relu', kernel_initializer='uniform'))
  model.add(Conv2D(512, (3, 3), strides=(1, 1), padding='same', activation='relu', kernel_initializer='uniform'))
  model.add(MaxPooling2D(pool_size=(2, 2)))
  model.add(Conv2D(512, (3, 3), strides=(1, 1), padding='same', activation='relu', kernel_initializer='uniform'))
  model.add(Conv2D(512, (3, 3), strides=(1, 1), padding='same', activation='relu', kernel_initializer='uniform'))
  model.add(Conv2D(512, (3, 3), strides=(1, 1), padding='same', activation='relu', kernel_initializer='uniform'))
  model.add(MaxPooling2D(pool_size=(2, 2)))
  model.add(Flatten())
  model.add(Dense(2048, activation='relu'))
  model.add(Dropout(0.5))
  model.add(Dense(2048, activation='relu'))
  model.add(Dropout(0.5))
  model.add(Dense(2, activation='softmax'))
  #model = multi_gpu_model(model, gpus=2)
  #model.compile(loss='categorical_crossentropy', optimizer='sgd', metrics=['accuracy'])
  model.compile(loss='categorical_crossentropy', optimizer=tf.keras.optimizers.Adam(learning_rate=1e-4), metrics=['accuracy'])
  return model

model = modelCompile()
model.summary()
```

Figura 16: Código de la red VGGNET implementada en Python.

8.5.1.6. Transfer Learning

Es evidente, que a pesar de que contamos con buenas técnicas de preprocesamiento y arquitecturas bastante robustas, una limitante sigue siendo el número de imágenes con el que contamos. Por lo que, una de las estrategias más efectivas para lograr modelos con predicciones precisas incluso con conjuntos de datos limitados es el Transfer Learning. Esta técnica se basa en la idea de aprovechar el conocimiento previamente adquirido por un modelo entrenado en un conjunto de datos grande y general, y aplicarlo a una tarea específica con un conjunto de datos más pequeño. En lugar de comenzar el entrenamiento de un modelo desde cero, se utilizan pesos preentrenados de un modelo existente, como una red neuronal convolucional entrenada en un conjunto de datos masivo, como "ImageNet".

Al transferir este conocimiento a un dominio específico o tarea de interés, el modelo puede beneficiarse de patrones aprendidos en datos más generales, mejorando así su capacidad de generalización y rendimiento en tareas específicas con datos limitados. Esto se traduce en la capacidad de obtener modelos más robustos y precisos, incluso cuando la disponibilidad de datos de entrenamiento es escasa. En nuestro caso, se usaron dos redes previamente entrenadas, que nos sirvieron para extraer características.

8.5.1.6.1. VGG19

En primera instancia se utilizó la arquitectura preentrenada VGG19, cargada desde Keras con los pesos correspondientes al conjunto de datos "Imagenet". Se configura el modelo para que no incluya la capa superior (clasificador) y se establece que las capas cargadas no sean entrenables. Esto significa que los pesos y las conexiones ya aprendidas durante el entrenamiento en "Imagenet" se mantendrán fijos, y el modelo se utilizará como un extractor de características. Al establecer las capas como no entrenables, se evita que los pesos se actualicen durante el entrenamiento subsiguiente. Al cargarlo se verifica que no haya parámetros entrenables, ya que la intención es conservar los conocimientos previos en lugar de ajustarlos a nuestro conjunto de datos. Con el sistema de obtienen 25.088 características.

Al obtener un número tan grande de características, se optó por implementar técnicas de reducción de dimensionalidad, entre ellas PCA, Kernel PCA y T-SNE. Notablemente, PCA demostró ser la más eficaz, logrando conservar el 97.947 % de la información al reducir el conjunto de datos a 200 características. Esta elección no solo permitió una significativa compresión de la información original, sino que también facilitó la representación y comprensión efectiva de las relaciones clave dentro del conjunto de datos, contribuyendo así a una mayor eficiencia computacional y una interpretación más simplificada de los patrones presentes.

8.5.1.6.2. Resnet50

Se realizo un proceso similar con una red neuronal convolucional profunda ampliamente reconocida, que es la Resnet50; esta se carga desde la librería Keras. Se establecen las capas como no entrenables, asegurando que los pesos preentrenados en el conjunto de datos "ImageNet" se conserven durante cualquier tarea subsiguiente. Se verifica nuevamente que no haya parámetros entrenables en el modelo, ya que se ha optado por mantener los pesos aprendidos previamente. De modo que, con este enfoque de Transfer learning permite aprovechar el conocimiento adquirido por ResNet50 en tareas anteriores, mejorando así la capacidad del modelo para extraer características relevantes en nuestras imágenes colposcópicas sin la necesidad de entrenar desde cero.

Finalmente, se obtienen 100.352 características, y en este caso resalta más la importancia de usar técnicas de reducción de la dimensionalidad. Por lo que, nuevamente usamos PCA para obtener 1.000 características que representan el 99.86 % de la información.

8.5.1.6.3. Clasificadores Machine Learning

Una vez extraídas las características de la red convolucional, se procede a aplanarlas mediante la función "Flatten". Esta operación es esencial para convertir la salida de las capas convolucionales en una forma compatible con las capas totalmente conectadas, como las capas de salida "Dense". En lugar de dirigir estas características directamente a una capa de salida, optamos por utilizar potentes algoritmos de Machine Learning para la clasificación, incluyendo Random Forest, Logistic Regression, XGBoost y Support Vector Machine (SVM). Además, se exploró la transformación de una red convolucional tipo U-Net a un formato unidimensional (1D), permitiendo así que también procesara y clasificara estos datos, ampliando las opciones de modelado y análisis. Este enfoque versátil y combinado ofrece una mayor flexibilidad en la tarea de clasificación de características extraídas.

8.5.1.7. Redes Transformers

Las redes transformers han emergido como una arquitectura revolucionaria en el campo del aprendizaje profundo en los últimos años. Su diseño se basa en mecanismos de atención, permitiendo a la red procesar secuencias de entrada completa simultáneamente, en lugar de depender de un procesamiento secuencial. Esta capacidad inherente de capturar relaciones de largo alcance entre elementos de entrada ha llevado a su éxito en diversas tareas, como procesamiento de lenguaje natural y visión por computadora. Su popularidad en el estado del arte se debe a su capacidad para modelar complejas dependencias, su flexibilidad y su rendimiento sobresaliente en grandes conjuntos de datos. En el contexto del proyecto Citobot, que busca aplicar arquitecturas de vanguardia, las redes transformers fueron adaptadas para abordar el desafío de la clasificación de imágenes de cáncer de cuello uterino. No solo se utilizaron como extractores de características eficientes, sino que también se implementaron directamente para llevar a cabo la clasificación. Esta estrategia no solo refleja la versatilidad de las redes transformers, sino también su capacidad para mejorar significativamente el rendimiento en tareas de clasificación médica.

8.5.1.7.1. Extractor de Características

Se implementa un modelo Transformer utilizando la arquitectura Vision Transformer (ViT). Inicialmente se definió el tamaño de entrada de las imágenes y se crea un modelo de entrada. Luego, se utiliza la función vit.vit_b16 para instanciar un modelo ViT con ciertas configuraciones, como un tamaño de imagen de 256x256, función de activación softmax, y se carga con pesos preentrenados. Posteriormente, se construye un nuevo modelo que incorpora la capa de entrada y el modelo ViT. Para garantizar que no se realice un ajuste fino durante el entrenamiento, todas las capas de este nuevo modelo se establecen como no entrenables mediante un bucle for. Finalmente, se muestra un resumen del modelo, destacando que no hay parámetros entrenables, lo que significa que el modelo se utiliza como un extractor de características fijas sin ajustar sus pesos durante el entrenamiento. A partir de aquí, se obtienen 768

características, por lo que no fue necesario usar reducción de la dimensionalidad, y dichos datos fueron enviados nuevamente a los clasificadores de Machine Learning.

8.5.1.7.2. Clasificador

Si se quiere usar el modelo como clasificador directamente, se omite la parte superior preentrenada. Luego, se agrega una capa de aplanado (Flatten) para convertir la salida del modelo ViT en un formato unidimensional. Posteriormente, se conecta una capa densa (Dense) con dos o tres unidades (depende el tipo de clasificación) y activación softmax, que sirve como capa de salida para realizar la clasificación. Finalmente, se compila el modelo con la función de pérdida de entropía cruzada categórica y el optimizador Adam. Este enfoque permite adaptar el modelo ViT preentrenado a la tarea específica de clasificación binaria o trinaría mediante el ajuste de los parámetros de la capa densa agregada. Evidentemente, es posible realizar ciertas transformaciones adicionales antes de enviar los datos a la capa Dense de salida, pero esta arquitectura de por si ya es bastante pesada, por lo que agregar transformaciones o convoluciones adicionales generaría que no pueda ser integrado en una aplicación móvil.

8.5.2. Hiperparámetros

Una vez que el modelo está definido, se torna crucial ajustar los hiperparámetros para optimizar el proceso de entrenamiento y, por ende, mejorar las predicciones del modelo. Los hiperparámetros son configuraciones externas al modelo y desempeñan un papel vital en la eficacia del aprendizaje profundo. En particular, las redes neuronales emplean el algoritmo de optimización de descenso de gradiente estocástico para ajustar sus pesos durante el entrenamiento. Entre los hiperparámetros esenciales se encuentra la tasa de aprendizaje (learning rate), que determina la magnitud de los ajustes realizados en los pesos de la red en cada iteración. Otros hiperparámetros, como la tasa de dropout para prevenir el sobreajuste y el tamaño del lote (batch) que especifica cuántas muestras se utilizan en cada actualización, también influyen significativamente en el rendimiento general del modelo. Por lo que, se hizo un proce-

so para ajustar estos hiperparámetros de manera cuidadosa y adaptativa, pues esto es fundamental para maximizar la capacidad de generalización y eficiencia de los modelos.

8.5.2.1. Learning Rate

En nuestro escenario, se estableció un rango de valores para la tasa de aprendizaje, abarcando desde 0.1 hasta 0.0000001. Se llevaron a cabo múltiples iteraciones de los modelos, evaluando el rendimiento a través de la métrica de pérdida (loss). La elección del mejor valor de la tasa de aprendizaje, finalmente fijada en 0.0001, se basó en la comparación de los resultados obtenidos durante estas iteraciones. Este proceso de búsqueda y ajuste cuidadoso del hiperparámetro de la tasa de aprendizaje es esencial para lograr un equilibrio óptimo entre la convergencia eficiente y la prevención de divergencias durante el entrenamiento de modelos de aprendizaje profundo.

8.5.2.2. Batch Size

El "batch" se refiere a un conjunto de datos utilizado en una iteración del algoritmo de entrenamiento. Durante el entrenamiento de un modelo, los datos se dividen en lotes más pequeños para procesarlos de manera eficiente. El tamaño del lote, se conoce como "batch size", el cual es un hiperparámetro que determina cuántos ejemplos de entrenamiento se utilizan en cada paso de actualización de los pesos del modelo. En lugar de actualizar los pesos después de cada ejemplo individual (lo que se conoce como descenso de gradiente estocástico), se actualizan después de procesar un conjunto de ejemplos (el tamaño del lote).

Elegir el tamaño del lote es un equilibrio entre eficiencia computacional y capacidad de generalización del modelo. Tamaños de lote más grandes pueden acelerar el proceso de entrenamiento al aprovechar la paralelización de hardware, pero a veces a expensas de la capacidad de generalización. Tamaños de lote más pequeños pueden proporcionar una generalización mejor, pero pueden ser computacionalmente más

costosos.

Generalmente, cuando se dispone de un conjunto de datos pequeño como en nuestro, se suele preferir un tamaño de lote menor. Pues, un tamaño de lote más pequeño permite al modelo actualizar sus pesos con mayor frecuencia, lo que puede ser beneficioso cuando hay limitada cantidad de datos disponibles. Además, puede ayudar a prevenir el sobreajuste, ya que el modelo no verá la misma cantidad de ejemplos en cada iteración. De modo que, se dejo fijado un batch size de 32 para los modelos.

8.5.2.3. Epochs

Una época se refiere a una iteración completa a través de todo el conjunto de datos de entrenamiento durante el proceso de entrenamiento de un modelo. Durante cada época, el modelo ve todos los ejemplos de entrenamiento una vez. El proceso de entrenamiento implica dividir el conjunto de datos en lotes, como se explicó anteriormente, y alimentar estos lotes al modelo para actualizar sus pesos. Después de cada época, el modelo ha tenido la oportunidad de ver y aprender de todos los ejemplos de entrenamiento.

El número total de épocas es un hiperparámetro que se ajusta durante el entrenamiento del modelo. Ajustar el número de épocas es importante para lograr un equilibrio entre el subajuste (modelo insuficientemente entrenado) y el sobreajuste (modelo memoriza los datos de entrenamiento pero no generaliza bien a nuevos datos).

Por lo que, para solucionar este problema se utilizó la función EarlyStopping que es propia de la librería. Esta es una técnica de regularización que se utiliza como callback durante el entrenamiento de un modelo de aprendizaje profundo. Su propósito es evitar el sobreajuste deteniendo el entrenamiento si la mejora en la métrica especificada (en nuestro caso, la pérdida en el conjunto de validación) no es

significativa después de un número específico de épocas (patience=20). El parámetro 'min_delta' establece el cambio mínimo necesario para considerar una mejora. Si la pérdida no mejora más allá de este umbral después de la paciencia establecida, el entrenamiento se detiene. 'Restore_best_weights=True' asegura que el modelo se restaure a los pesos que dieron la mejor métrica en el conjunto de validación antes de detenerse, lo que ayuda a evitar la degradación del rendimiento. En resumen, si después de 20 épocas el "loss" no disminuye se procede a detenerse el entrenamiento, y el modelo se queda con los mejores pesos obtenidos durante el proceso.

8.5.2.4. GridSearch

Ahora bien, si recordamos, cuando se usó Transfer Learning para extraer características, se enviaron estos datos a clasificadores de Machine Learning, los cuales tienen sus propios hiperparámetros. Con lo cual, se utiliza GridSearch, que es una técnica utilizada en Machine Learning para la búsqueda sistemática de los mejores hiperparámetros de un modelo en un conjunto predeterminado de valores. En lugar de ajustar manualmente los hiperparámetros, que son configuraciones externas al modelo que afectan su rendimiento, GridSearch automatiza este proceso al probar todas las combinaciones posibles de valores en un conjunto dado. Este método realiza una búsqueda exhaustiva a través de una "rejilla" de hiperparámetros, evaluando el rendimiento del modelo para cada combinación. Añadiendo que, en nuestro caso se utilizó junto con la validación cruzada para obtener una estimación más robusta del rendimiento del modelo en datos no vistos.

8.6. Evaluación

A continuación, describimos las técnicas empleadas para validar y evaluar el rendimiento de nuestros modelos, junto con las métricas utilizadas para calificarlos. Estas prácticas nos permitieron identificar y seleccionar el modelo más adecuado para abordar eficazmente nuestro problema de clasificación de imágenes colposcópicas.

8.6.1. Validación Holdout

En nuestra estrategia de validación, optamos por implementar múltiples iteraciones de la validación holdout para abordar los desafíos asociados con un conjunto de datos limitado. De hecho, para ser preciso utilizamos 5 iteraciones. En cada iteración, dividimos aleatoriamente el conjunto de datos en los porcentajes específicos ya definidos previamente para entrenamiento, validación y prueba, estableciendo un equilibrio práctico que nos permite obtener evaluaciones más estables y confiables del rendimiento del modelo. La elección de utilizar holdout en lugar de k-folds (validación cruzada) se fundamenta en la consideración de que, con nuestro conjunto de datos reducido, la división en k subconjuntos resultaría en grupos demasiado pequeños para obtener evaluaciones significativas. Esta estrategia iterativa nos ha permitido aprovechar al máximo nuestros datos limitados y obtener una comprensión más robusta del rendimiento del modelo en diversas configuraciones de entrenamiento y evaluación.

8.6.2. Métricas de Evaluación

En la evaluación de nuestros modelos, nos enfocamos principalmente en métricas clave como especificidad y sensibilidad. La sensibilidad (Sensitivity), también conocida como recall, representa la proporción de casos positivos reales correctamente identificados por el modelo, es decir, la capacidad para capturar verdaderos positivos entre todos los casos positivos reales. Por otro lado, la especificidad (Specificity) mide la habilidad del modelo para identificar correctamente los casos negativos, destacando su capacidad para evitar falsos positivos. En otras palabras, la sensibilidad es la probabilidad de que el modelo identifique como enfermo a aquél que efectivamente lo está, mientras que la especificidad es la probabilidad de que la prueba identifique como no enfermo a aquél que efectivamente no lo está.

Además, consideramos métricas complementarias para tener una comprensión más completa del rendimiento. La precisión (Precision) representa la calidad de las predicciones positivas en relación con todas las predicciones positivas realizadas por

el modelo. El F1 Score combina precisión y sensibilidad, proporcionando una métrica equilibrada que es especialmente útil cuando se busca un equilibrio entre falsos positivos y falsos negativos. La exactitud (Accuracy) general ofrece una visión global del rendimiento del modelo.

Durante las 5 iteraciones de holdout, calculamos estas métricas en cada conjunto de prueba y, al final, obtuvimos la media y la desviación estándar de cada métrica. Este enfoque nos permitió no solo evaluar el rendimiento promedio del modelo, sino también entender la variabilidad en su desempeño en diferentes configuraciones de entrenamiento y validación. Además, identificamos la iteración específica que mostró el mejor rendimiento para guardar y exportar ese modelo.

8.7. Despliegue

Una vez seleccionado el modelo para la aplicación móvil en el proyecto CITOBOT, se enfrentan requisitos técnicos y consideraciones específicas del proyecto. La tecnología utilizada por el proyecto para la aplicación es Android Studio, de modo que, la interfaz debía implementarse en este entorno. Inicialmente, la opción de cargar el modelo mediante una API se consideró, pero esto presenta limitaciones en entornos rurales con conexión a internet inestable. Por lo tanto, MLOps propone una estrategia para el despliegue e integración de la agente inteligente en la interfaz móvil, lo que permitiría un diagnóstico en tiempo real sin necesidad de conexión a internet.

1. **Conversión del Modelo:** Se utilizo la herramienta TensorFlow Lite Converter para convertir el modelo seleccionado a un formato compatible con dispositivos móviles, el cual se conoce como ".tflite".

2. **Optimización del Modelo:** Como se mencionó, la interfaz debe ser implementada en Android Studio, y realizando un proceso de investigación se encontró que esta herramienta solo permitía cargar modelos que tuvieran un peso inferior a 200MB, con lo cual se plantean formas de optimizar nuestro mo-

delo sin dañar el rendimiento. Por dicha razón, desde un principio de definió que el tamaño de las imágenes debía ser de 224x224 pixeles, generando que las arquitecturas sean más livianas. En algunos casos, esto no fue suficiente, por lo que se planteó incluir la técnica de la cuantización del modelo, que consiste en reducir la precisión numérica de los parámetros, pues en lugar de utilizar números de punto flotante de 32 bits, la cuantización reduce la representación de los números a enteros de menor tamaño, como enteros de 8 bits o incluso menos.

3. **Integración con Android Studio:** Se creó una interfaz en Android Studio, el cual consta de una pantalla donde se habilitan las opciones de "Tomar una foto" o "Seleccionar una foto desde la galería". La idea es que, desde la cámara del celular o una cámara conectada al celular, se pueda enfocar el cuello uterino y tomar una foto las veces que sea necesario hasta tener una buena imagen, habilitando las opciones de poder repetir la foto. En caso de estar bien, se vuelve a la pantalla principal y se muestra la foto junto al diagnóstico final, además de mostrar la probabilidad de cada clase de salida. Del mismo modo, es posible cargar una foto que este guardada en la galería de dispositivo, una vez seleccionado aparece en la pantalla principal junto al diagnóstico de igual forma.

4. **Pruebas:** Se realizó la prueba con un dispositivo Android, validando que las funcionalidades de la interfaz funcionaran correctamente, y que efectivamente se pudiera tomar y cargar una imagen, y posteriormente esta de inmediatamente el diagnóstico.

9. Resultados y Discusión

A continuación, se presentan los resultados de las ejecuciones de los modelos para los datos de prueba previamente definidos.

9.1. Resultados

9.1.1. Dos Clases

En la Tabla 9 y en la Tabla 10 se observa la media y desviación estándar respectivamente, obtenida por cada métrica luego realizar las 5 iteraciones de Holdout.

Dos Clases-Media					
	Specificity	**Sensitivity**	**Accuracy**	**Precision**	**F1-Score**
AlexNet	0,7590	0,9514	0,8580	0,8720	0,8554
GoogleNet	0,7955	0,8733	0,8328	0,8396	0,8322
ResNet	0,6353	0,8580	0,7458	0,6968	0,7128
U-Net	0,7395	0,8184	0,8183	0,8268	0,8166
VGGNET	0,7533	0,9324	0,8412	0,8532	0,8399
VGG19 + U-Net 1D	0,8355	0,7912	0,8122	0,8150	0,8120
VGG19 + RandomForest	0,7833	0,8896	0,8382	0,8409	0,8366
VGG19 + LogisticRegression	0,8198	0,7738	0,7954	0,7965	0,7950
VGG19 + XGBoost	0,7693	0,8498	0,8084	0,8108	0,8082
VGG19 + SVC	0,7884	0,8966	0,8412	0,8456	0,8410
VGG19 + PCA + U-Net 1D	0,7359	0,8483	0,7931	0,7998	0,7913
VGG19 + PCA + RandomForest	0,7392	0,8828	0,8069	0,8144	0,8066
VGG19 + PCA + LogisticRegression	0,8407	0,8130	0,8267	0,8270	0,8264
VGG19 + PCA + XGBoost	0,7745	0,8585	0,8160	0,8176	0,8154
VGG19 + PCA + SVC	0,8012	0,8679	0,8351	0,8367	0,8345
ResNet50 + U-Net 1D	0,8270	0,7951	0,8069	0,8116	0,8062
ResNet50 + RandomForest	0,7809	0,8814	0,8305	0,8311	0,8299
ResNet50 + LogisticRegression	0,8311	0,8128	0,8214	0,8212	0,8212
ResNet50 + XGBoost	0,8002	0,8471	0,8229	0,8232	0,8232
ResNet50 + SVC	0,8847	0,6095	0,7466	0,7682	0,7412
ResNet50 + PCA + U-Net 1D	0,8129	0,7556	0,7824	0,7858	0,7821
ResNet50 + PCA + RandomForest	0,7442	0,8298	0,7863	0,7888	0,7870
ResNet50 + PCA + LogisticRegression	0,8345	0,8072	0,8206	0,8208	0,8205
ResNet50 + PCA + XGBoost	0,7514	0,8641	0,8076	0,8121	0,8064
ResNet50 + PCA + SVC	0,7790	0,8823	0,8282	0,8320	0,8280
Transformer + RandomForest	0,7375	0,9199	0,8282	0,8387	0,8264
Transformer + LogisticRegression	0,8028	0,6354	0,7214	0,7274	0,7182
Transformer + XGBoost	0,7439	0,9087	0,8252	0,8347	0,8242
Transformer	0,9015	0,7385	0,8206	0,7778	0,8351

Tabla 9: Media de los resultados de los modelos dos clases después de 5 iteraciones de Holdout.

Dos Clases-Desviacion Estandar					
	Specificity	**Sensitivity**	**Accuracy**	**Precision**	**F1-Score**
AlexNet	0,0364	0,0182	0,0159	0,0697	0,0256
GoogleNet	0,0683	0,0542	0,0173	0,0594	0,0205
ResNet	0,3185	0,0801	0,1273	0,2521	0,2428
U-Net	0,0984	0,0874	0,0253	0,0621	0,0321
VGGNET	0,0381	0,0175	0,0217	0,0743	0,0252
VGG19 + U-Net 1D	0,0314	0,0527	0,0166	0,0399	0,0178
VGG19 + RandomForest	0,0257	0,0119	0,0168	0,0369	0,0239
VGG19 + LogisticRegression	0,0231	0,0328	0,0089	0,0387	0,0126
VGG19 + XGBoost	0,0280	0,0160	0,0166	0,0396	0,0176
VGG19 + SVC	0,0329	0,0228	0,0265	0,0525	0,0270
VGG19 + PCA + U-Net 1D	0,0652	0,0512	0,0235	0,0460	0,0313
VGG19 + PCA + RandomForest	0,0259	0,0296	0,0131	0,0673	0,0151
VGG19 + PCA + LogisticRegression	0,0193	0,0246	0,0142	0,0254	0,0163
VGG19 + PCA + XGBoost	0,0262	0,0188	0,0203	0,0394	0,0233
VGG19 + PCA + SVC	0,0218	0,0311	0,0204	0,0315	0,0223
ResNet50 + U-Net 1D	0,0537	0,0800	0,0333	0,0692	0,0350
ResNet50 + RandomForest	0,0210	0,0220	0,0177	0,0434	0,0210
ResNet50 + LogisticRegression	0,0253	0,0207	0,0203	0,0306	0,0212
ResNet50 + XGBoost	0,0286	0,0150	0,0154	0,0346	0,0183
ResNet50 + SVC	0,0450	0,0444	0,0282	0,0869	0,0474
ResNet50 + PCA + U-Net 1D	0,0341	0,0679	0,0319	0,0507	0,0332
ResNet50 + PCA + RandomForest	0,0434	0,0149	0,0202	0,0399	0,0240
ResNet50 + PCA + LogisticRegression	0,0164	0,0158	0,0121	0,0202	0,0127
ResNet50 + PCA + XGBoost	0,0473	0,0406	0,0315	0,0546	0,0353
ResNet50 + PCA + SVC	0,0297	0,0265	0,0198	0,0540	0,0208
Transformer + RandomForest	0,0202	0,0202	0,0158	0,0682	0,0251
Transformer + LogisticRegression	0,0485	0,0104	0,0255	0,0456	0,0375
Transformer + XGBoost	0,0258	0,0113	0,0101	0,0602	0,0172
Transformer	0,0358	0,0102	0,0189	0,0359	0,0133

Tabla 10: Desviación estándar de los resultados de los modelos dos clases después de 5 iteraciones de Holdout.

Observando los resultados, podemos destacar varios modelos por sus sólidos desempeños en las métricas definidas. AlexNet, con una especificidad promedio de 0.7590 y una sensibilidad promedio de 0.9514, demostró un equilibrio notable entre precisión y capacidad para identificar verdaderos positivos. VGGNet, con una especificidad promedio de 0.7533 y una sensibilidad promedio de 0.9324, también exhibió un rendimiento destacado.

ResNet50 combinado con una máquina de soporte vectorial (SVC) mostró una alta especificidad promedio de 0.8847, destacándose en la capacidad para identificar

verdaderos negativos. Por otro lado, los modelos basados en Transformers, especialmente Transformer con RandomForest y Transformer con XGBoost, presentaron una sensibilidad promedio significativa de 0.9199 y 0.9087, respectivamente.

De modo que, tras analizar estos resultados y teniendo en cuenta los requisitos específicos, se tomó la decisión de seleccionar AlexNet como el modelo a exportar e integrar en la interfaz móvil. Este modelo logró un rendimiento sólido en múltiples métricas y cumplió con los requisitos de tamaño, con un peso de 178.27 MB.

La elección de AlexNet como el modelo preferido se fundamenta en su rendimiento equilibrado y robusto en las métricas específicas de nuestro problema. AlexNet ha demostrado consistentemente su eficacia en tareas de clasificación de imágenes, y en este contexto específico de detección de cáncer de cuello uterino, su combinación de especificidad y sensibilidad sobresalientes lo destacó entre las opciones disponibles. Además, la gestión eficiente de recursos computacionales y su capacidad para adaptarse a conjuntos de datos más pequeños, como el nuestro, lo convirtieron en una elección idónea para nuestro proyecto. La estructura profunda de AlexNet y sus capas convolucionales han demostrado ser efectivas en la extracción de características relevantes de las imágenes médicas, contribuyendo así a su selección como el modelo principal para la implementación en la interfaz móvil del proyecto CITOBOT.

9.1.1.1. Mejor Modelo

A continuación, se muestra el rendimiento específico del modelo AlexNet, graficando los resultados obtenidos en su mejor iteración.

Iteración 4					
	Specificity	**Sensitivity**	**Accuracy**	**Precision**	**F1-Score**
No-Risk	0,8143	0,9590	0,8817	0,9580	0,8803
High-Risk				0,8182	0,8830

Tabla 11: Resultados de las métricas del modelo AlexNet en su cuarta iteración Holdout.

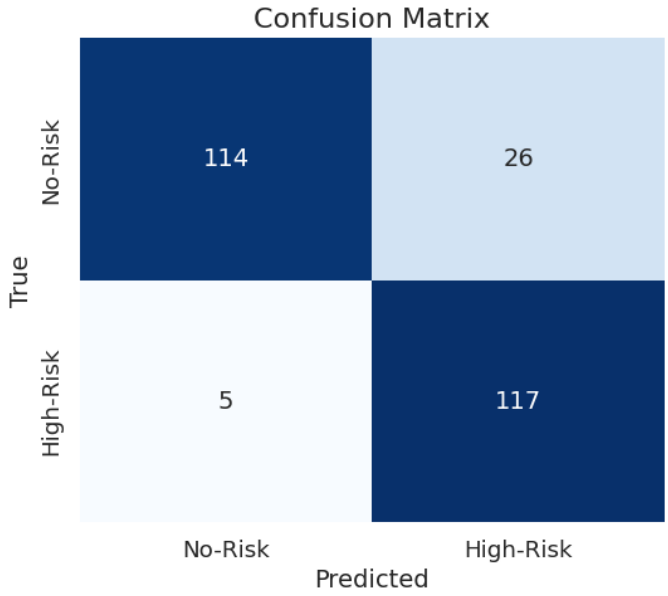


Figura 17: Matriz de confusión del modelo AlexNet en la cuarta iteración.

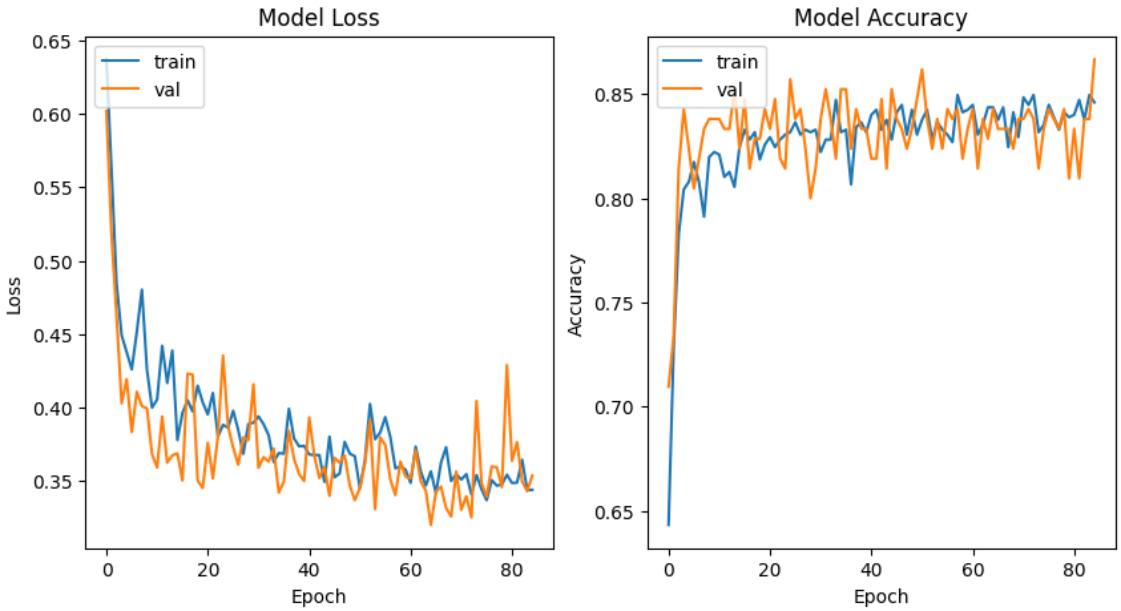


Figura 18: Gráficas de rendimiento del modelo AlexNet durante el entrenamiento.

9.1.2. Tres Clases

En la Tabla 12 y en la Tabla 13 se observa la media y desviación estándar respectivamente, obtenida por cada métrica luego realizar las 5 iteraciones de Holdout.

Tres Clases-Media						
		Specificity	**Sensitivity**	**Accuracy**	**Precision**	**F1-Score**
AlexNet	**No-Risk**	0,9131	0,6186	0,6826	0,6947	0,6816
	Low-Risk	0,7971	0,6698			
	High-Risk	0,8119	0,7587			
GoogleNet	**No-Risk**	0,8499	0,6420	0,6448	0,6518	0,6430
	Low-Risk	0,8036	0,6155			
	High-Risk	0,8129	0,6717			
VGGNET	**No-Risk**	0,9240	0,5942	0,6830	0,6971	0,6815
	Low-Risk	0,8294	0,6577			
	High-Risk	0,7707	0,7990			
VGG19 + U-Net 1D	**No-Risk**	0,6534	0,6845	0,5122	0,5155	0,5020
	Low-Risk	0,8246	0,3771			
	High-Risk	0,7876	0,4667			
VGG19 + RandomForest	**No-Risk**	0,7312	0,6135	0,5374	0,5391	0,5344
	Low-Risk	0,8115	0,4433			
	High-Risk	0,7632	0,5541			
VGG19 + LogisticRegression	**No-Risk**	0,6352	0,7072	0,5263	0,5362	0,5168
	Low-Risk	0,8586	0,3780			
	High-Risk	0,7937	0,4881			
VGG19 + XGBoost	**No-Risk**	0,8001	0,5921	0,5674	0,5677	0,5667
	Low-Risk	0,7767	0,5207			
	High-Risk	0,7745	0,5886			
ResNet50 + U-Net 1D	**No-Risk**	0,6884	0,7023	0,5404	0,5590	0,5284
	Low-Risk	0,8280	0,4230			
	High-Risk	0,7931	0,4953			
ResNet50 + RandomForest	**No-Risk**	0,7683	0,5976	0,5448	0,5555	0,5383
	Low-Risk	0,8685	0,3781			
	High-Risk	0,6835	0,6679			
ResNet50 + LogisticRegression	**No-Risk**	0,7786	0,6442	0,5593	0,5600	0,5581
	Low-Risk	0,8104	0,4755			
	High-Risk	0,7504	0,5637			
ResNet50 + XGBoost	**No-Risk**	0,8143	0,6026	0,5730	0,5755	0,5724
	Low-Risk	0,8038	0,5147			
	High-Risk	0,7420	0,6026			
ResNet50 + SVC	**No-Risk**	0,4933	0,8041	0,4967	0,5455	0,4770
	Low-Risk	0,9197	0,2892			
	High-Risk	0,8353	0,4058			

Tabla 12: Media de los resultados de los modelos tres clases después de 5 iteraciones de Holdout.

Tres Clases-Desviacion Estandar						
		Specificity	**Sensitivity**	**Accuracy**	**Precision**	**F1-Score**
AlexNet	**No-Risk**	0,0302	0,0656	0,0106	0,0700	0,0329
	Low-Risk	0,0411	0,0677			
	High-Risk	0,0248	0,0316			
GoogleNet	**No-Risk**	0,0624	0,0725	0,0223	0,0623	0,0493
	Low-Risk	0,0558	0,0950			
	High-Risk	0,0406	0,1064			
VGGNET	**No-Risk**	0,0212	0,0478	0,0204	0,0794	0,0340
	Low-Risk	0,0091	0,0401			
	High-Risk	0,0276	0,0234			
VGG19 + U-Net 1D	**No-Risk**	0,0582	0,0448	0,0123	0,0334	0,0681
	Low-Risk	0,0370	0,0630			
	High-Risk	0,0349	0,0365			
VGG19 + RandomForest	**No-Risk**	0,0175	0,0233	0,0096	0,0383	0,0330
	Low-Risk	0,0153	0,0146			
	High-Risk	0,0288	0,0209			
VGG19 + LogisticRegression	**No-Risk**	0,0551	0,0652	0,0179	0,0613	0,0734
	Low-Risk	0,0709	0,1147			
	High-Risk	0,1151	0,1282			
VGG19 + XGBoost	**No-Risk**	0,0175	0,0233	0,0096	0,0383	0,0330
	Low-Risk	0,0153	0,0146			
	High-Risk	0,0288	0,0209			
ResNet50 + U-Net 1D	**No-Risk**	0,0551	0,0652	0,0179	0,0613	0,0734
	Low-Risk	0,0709	0,1147			
	High-Risk	0,1151	0,1282			
ResNet50 + RandomForest	**No-Risk**	0,0101	0,0303	0,0271	0,0571	0,0629
	Low-Risk	0,0221	0,0425			
	High-Risk	0,0318	0,0482			
ResNet50 + LogisticRegression	**No-Risk**	0,0197	0,0225	0,0176	0,0382	0,0465
	Low-Risk	0,0235	0,0268			
	High-Risk	0,0104	0,0374			
ResNet50 + XGBoost	**No-Risk**	0,0241	0,0328	0,0209	0,0502	0,0392
	Low-Risk	0,0271	0,0295			
	High-Risk	0,0181	0,0314			
ResNet50 + SVC	**No-Risk**	0,0313	0,0173	0,0189	0,0903	0,0759
	Low-Risk	0,0079	0,0409			
	High-Risk	0,0268	0,0426			

Tabla 13: Desviación estándar de los resultados de los modelos tres clases después de 5 iteraciones de Holdout.

Observando detenidamente los resultados obtenidos en la clasificación de tres clases ("No-Risk", "Low-Risk", "High-Risk"), se destacan diversos modelos que han demostrado un desempeño sólido en las métricas consideradas.

AlexNet, aunque presenta una sensibilidad promedio de 0.6186 para la clase "No-Risk", muestra una especificidad promedio notable de 0.9131, lo que indica su ca-

pacidad para identificar correctamente la ausencia de riesgo. La baja sensibilidad podría deberse a una mayor focalización en la precisión de las predicciones negativas, lo cual puede ser crucial en escenarios de detección de ausencia de riesgo. En cuanto a VGGNet, destaca por su elevada especificidad promedio de 0.9240 para la clase "No-Risk", resaltando su habilidad para prever situaciones sin riesgo con alta precisión.

Los modelos basados en ResNet50, especialmente cuando se combina con RandomForest, LogisticRegression y XGBoost, también exhiben resultados prometedores. ResNet50 + RandomForest, en particular, presenta una especificidad promedio considerable de 0.7683 para la clase "No-Risk" y una sensibilidad promedio razonable de 0.5976, indicando una capacidad equilibrada de identificar tanto la presencia como la ausencia de riesgo.

Finalmente, se tomó la decisión de seleccionar el modelo VGGNet como la opción preferida para la exportación e integración futura en la interfaz móvil del proyecto. VGGNet demostró un rendimiento sólido, destacando especialmente por su elevada especificidad de 0.9240 para la clase "No-Risk", lo que subraya su capacidad para prever situaciones sin riesgo con una precisión notable. Si bien, lo esperado es que al igual que en los modelos de dos clases, nos interesa tener mayor efectividad a la hora de identificar los casos con alto riesgo, a estos modelos en general les cuesta mucho realizar esta asociación, debido a que les es difícil poder encontrar diferencias entre las clases "Low-Risk" y "High-Risk", de modo que por ahora tenemos un modelo que logra predecir de gran manera los casos que efectivamente son sin riesgo.

9.1.2.1. Mejor Modelo

A continuación, se muestra el rendimiento específico del modelo VGGNET, graficando los resultados obtenidos en su mejor iteración.

Iteración 5					
	Specificity	**Sensitivity**	**Accuracy**	**Precision**	**F1-Score**
No-Risk	0,9574	0,6277	0,7111	0,8872	0,7352
Low-Risk	0,8338	0,7318		0,6859	0,7081
High-Risk	0,7793	0,7803		0,6250	0,6941

Tabla 14: Resultados de las métricas del modelo VGGNET en su quinta iteración Holdout.

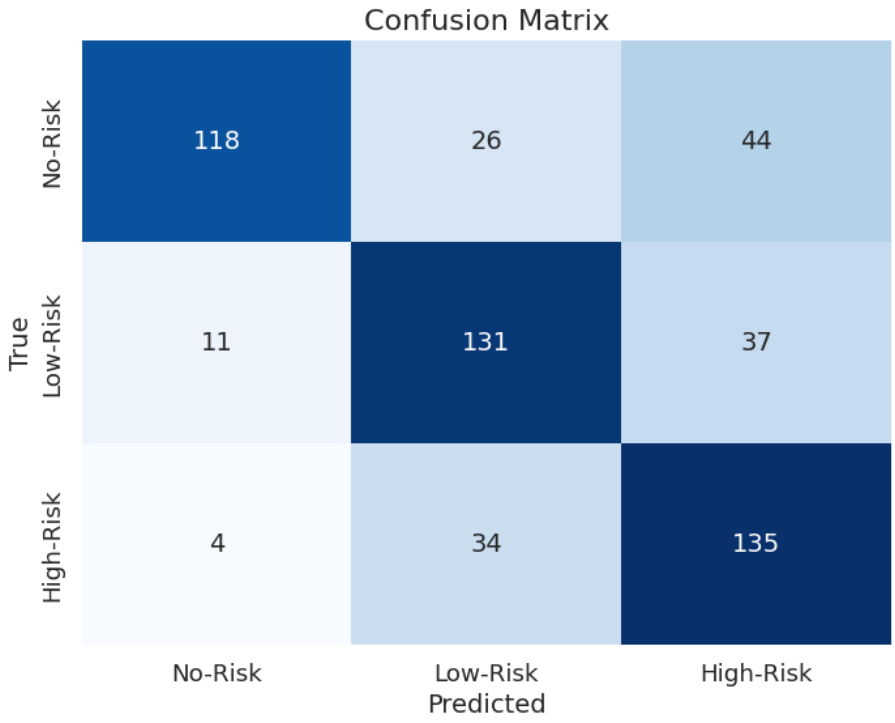


Figura 19: Matriz de confusión del modelo VGGNET en la cuarta iteración.

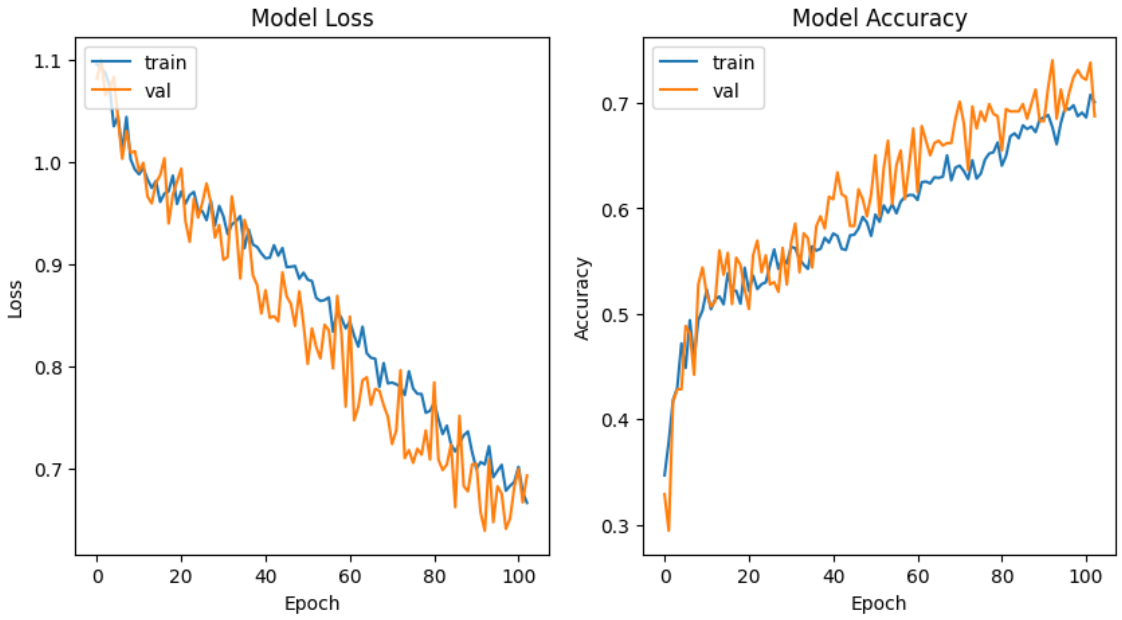


Figura 20: Gráficas de rendimiento del modelo VGGNET durante el entrenamiento.

9.2. Comparación de los Modelos

En base a los resultados obtenidos, se destaca la notable eficacia de las métricas en ambas modalidades de clasificación, un logro que se alinea con el objetivo inicial del proyecto CITOBOT de alcanzar métricas de especificidad y sensibilidad superiores al 60 %. Esta meta se ha alcanzado con éxito en ambos casos. Al examinar las tablas de desviación estándar y los gráficos de rendimiento, se evidencia la selección acertada de los hiperparámetros, pues la desviación es muy baja y el rendimiento tanto en "loss" como en accuracy de los datos de entrenamiento y validación son muy similares. Garantizando de esta forma, que los modelos no sufran de una alta tasa de overfitting y, por ende, asegurando un aprendizaje efectivo.

En la comparativa entre las dos modalidades de clasificación, surge la constatación de que enfrentar un problema de clasificación con un mayor número de clases presenta desafíos adicionales. Para la clasificación de dos clases, las métricas y resultados

permanecen consistentes tanto para la clase “No-Risk” como para la “High-Risk”, siendo incluso más destacados para esta última. Esta situación es beneficiosa, ya que, dada la naturaleza del problema, es preferible que el sistema diagnostique erróneamente a una persona sin cáncer como si lo tuviera, en lugar de la situación inversa, que podría tener consecuencias más graves.

En el caso de la clasificación de tres clases, la complejidad aumenta, ya que el modelo muestra buenos resultados al identificar la clase “No-Risk”, pero enfrenta dificultades al diferenciar entre las clases “Low-Risk” y “High-Risk”. Este escenario resalta la necesidad de ajustar las arquitecturas y mejorar la base de datos para perfeccionar el diagnóstico de estas dos clases.

Además, la utilización de modelos preentrenados como extractores de características ha demostrado ser una estrategia eficaz para abordar la limitación de datos, permitiendo a los clasificadores de Machine Learning identificar y diagnosticar las clases de salida. La relevancia de estas características se evidencia al constatar que, al realizar la reducción de dimensionalidad con técnicas como PCA, Kernel PCA y TSNE, no se logró superar las métricas de los modelos que prescindieron de dicha reducción.

Finalmente, se destaca la idoneidad de las redes Transformers como una alternativa prometedora. Aunque originalmente diseñadas para otros tipos de problemas, estas redes han demostrado ser adaptables al enfoque de clasificación de imágenes en el proyecto CITOBOT, funcionando tanto como extractores de características como clasificadores. Es importante señalar que, a pesar de su eficacia, la exportación de estas redes implica desafíos, especialmente en términos de peso, incluso para una red relativamente no densa. Por ende, la implementación inicial de estas redes requerirá un proceso de cuantización.

9.3. Interfaz Móvil

La interfaz móvil se presenta como una herramienta avanzada y accesible que combina la potencia de la inteligencia artificial con la conveniencia de dispositivos Android. Su funcionalidad principal reside en la capacidad de analizar y categorizar imágenes en tiempo real, ofreciendo a los usuarios la opción de tomar instantáneas con la cámara del dispositivo o seleccionar imágenes almacenadas en la galería. Este proceso de clasificación se realiza mediante un modelo de aprendizaje automático implementado con TensorFlow Lite, proporcionando diagnósticos precisos y las probabilidades asociadas a diferentes categorías, incluso en condiciones offline.

La aplicación, desarrollada en Android Studio, se estructura en módulos fundamentales, incluyendo la interfaz de usuario, la lógica de la aplicación y el componente responsable de la clasificación de imágenes basado en el modelo de aprendizaje automático. La interfaz de usuario, definida en el archivo 'activity_main.xml', se crea mediante el uso de Java y el Android SDK, mientras que la lógica de la aplicación, implementada en 'MainActivity.java', gestiona la interacción del usuario y la presentación de resultados. Por otro lado, el componente relacionado con el modelo de aprendizaje automático, contenido en 'model.tflite', se encarga de la inferencia y clasificación de imágenes, funcionando de manera análoga a un backend local. Esta estructura modular facilita la interacción fluida entre el frontend y el backend, permitiendo a los usuarios explorar la capacidad de la inteligencia artificial para analizar imágenes de manera eficiente.

9.3.1. Backend

En este contexto, la noción de backend adquiere una perspectiva particular, ya que gran parte de la complejidad computacional recae en la ejecución local de un modelo de aprendizaje automático. En lugar de depender de un servidor remoto para procesamiento y toma de decisiones (API), la aplicación confía en un backend local representado por el archivo 'model.tflite'. Este componente se encarga de la inferencia y clasificación de imágenes, llevando a cabo tareas esenciales para la aplicación,

como la interpretación de patrones visuales y la asignación de categorías. Esta aproximación descentralizada aporta eficiencia y rapidez, ya que las operaciones críticas se realizan directamente en el dispositivo móvil, permitiendo una experiencia de usuario ágil y autónoma. Aunque el término "backend" tradicionalmente evoca la idea de un servidor remoto, en este contexto específico, se refiere a la lógica de procesamiento local que impulsa la funcionalidad principal de la aplicación.

9.3.2. Frontend

En el frente del desarrollo móvil, la interfaz presenta un frontend robusto y amigable que facilita la interacción del usuario con las funciones avanzadas de clasificación. La interfaz de usuario, definida en el archivo 'activity_main.xml' y respaldada por la lógica implementada en 'MainActivity.java', constituye el núcleo del frontend. Este conjunto de archivos define la disposición visual de la aplicación, incluyendo botones para tomar o cargar imágenes, así como elementos de presentación como TextViews e ImageViews para mostrar diagnósticos y resultados de clasificación. En la Figura 21, se observa la estructura y diseño de la pantalla:

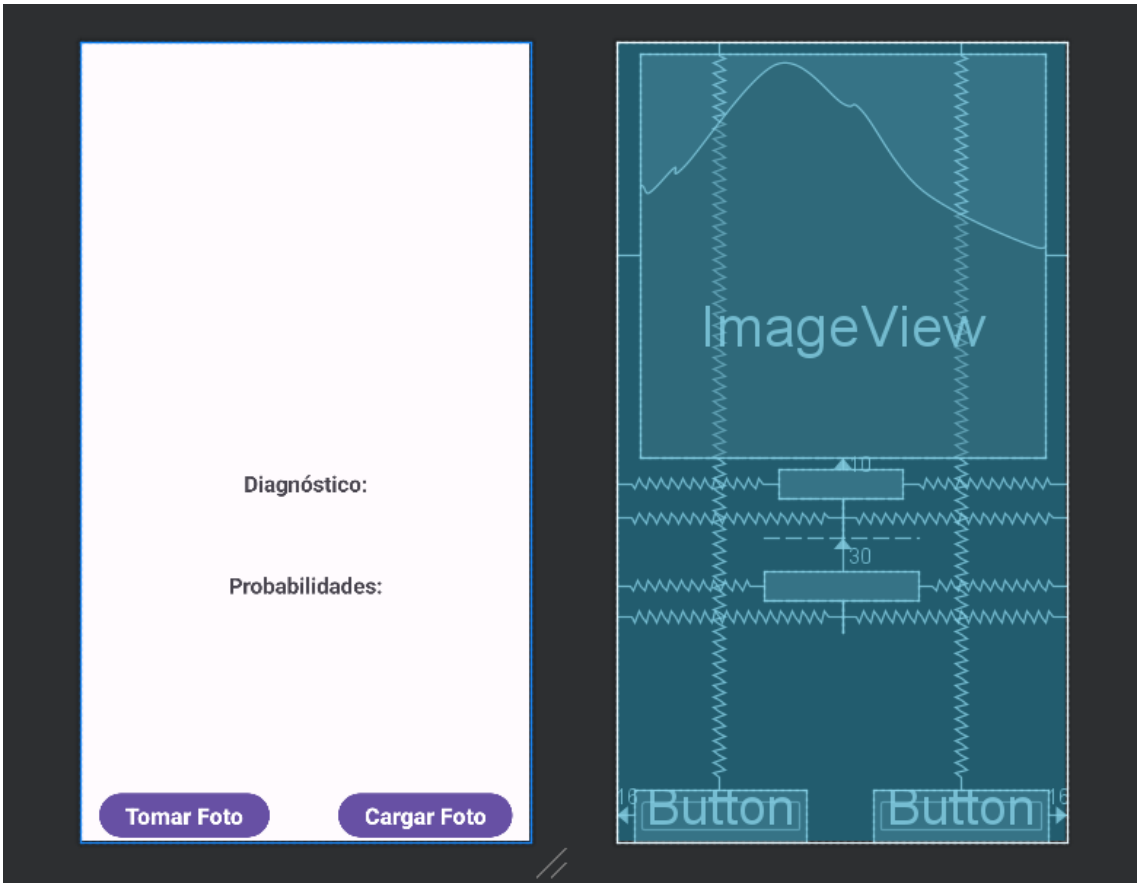


Figura 21: Estructura de la pantalla principal de la interfaz móvil..

El lenguaje principal utilizado en el frontend es Java, complementado por el framework Android SDK, que proporciona las herramientas esenciales para el desarrollo de aplicaciones Android. La estructuración en XML, como se observa en 'activity_main.xml', facilita la descripción de la interfaz de usuario, permitiendo una visualización clara y una manipulación eficiente de los elementos visuales.

El frontend de la aplicación va más allá de simplemente gestionar la representación visual, ya que integra la lógica esencial para interactuar con el usuario. En el núcleo de estas interacciones se encuentra el código contenido en 'MainActivity.java', el cual responde a eventos como la captura de imágenes, la selección desde la galería

y la actualización de la interfaz con los resultados de la clasificación. Además de estas funciones, este componente realiza transformaciones críticas en la imagen, preparándola para ser procesada por el modelo de inteligencia artificial.

Durante el proceso de preparación de la imagen para la clasificación por el modelo de aprendizaje automático, 'MainActivity.java' ejecuta transformaciones fundamentales. En una primera etapa, la imagen se redimensiona a un formato estándar de 224x224 píxeles, una dimensión clave para el modelo. Posteriormente, se realiza la normalización de los valores de los píxeles RGB. Cada valor se divide por 255.0, escalándolos a un rango entre 0 y 1, una práctica común en modelos de aprendizaje automático. Estas transformaciones no solo ajustan la imagen a las expectativas del modelo, sino que también contribuyen a una presentación estandarizada y optimizada para la inferencia.

Esta integración de lógica de aplicación proporciona una experiencia de usuario fluida y educativa al hacer accesibles las avanzadas capacidades de clasificación de imágenes impulsadas por el modelo de aprendizaje automático. En este contexto, el componente frontend no solo sirve como interfaz visual, sino que se convierte en un facilitador clave para que los usuarios aprovechen al máximo las capacidades de la IA en la clasificación precisa y consistente de imágenes.

9.3.3. Arquitectura

La arquitectura de esta aplicación móvil sigue un patrón de diseño modelo-vista-controlador (MVC), que organiza la lógica y la funcionalidad en tres componentes principales. En el modelo, se encuentra el modelo de aprendizaje automático contenido en el archivo 'model.tflite', el cual realiza la inferencia y clasificación de imágenes, actuando como una suerte de backend local. Por otro lado, el controlador está representado por la clase 'MainActivity.java', que se encarga de gestionar la interacción del usuario y orquestar la lógica de la aplicación. Este controlador responde a eventos como la captura de imágenes y la selección desde la galería, implementando además

las transformaciones necesarias en la imagen para su procesamiento por el modelo de inteligencia artificial. La vista, por su parte, está definida en el archivo 'activity_main.xml', que describe la disposición y la apariencia visual de la interfaz de usuario, incluyendo botones, textos y una imagen para mostrar los resultados de la clasificación. Esta arquitectura modular facilita la extensión y el mantenimiento del código, permitiendo una clara separación de responsabilidades entre el frontend y el modelo de IA, lo que contribuye a un desarrollo eficiente y escalable de la aplicación.

9.4. Visualización de la Interfaz Móvil

Como se mencionó previamente, la interfaz fue desarrollada utilizando Android Studio para integrar el modelo AlexNet en la aplicación. En la Figura 22, se presenta la pantalla principal de la interfaz, compuesta por un área de carga de imágenes, la cual mostrará el resultado de la predicción en "Diagnóstico", y la probabilidad de cada clase en "Probabilidades". Además, en la parte inferior se encuentran dos botones: uno para tomar fotografías y otro para cargarlas desde la galería del dispositivo.

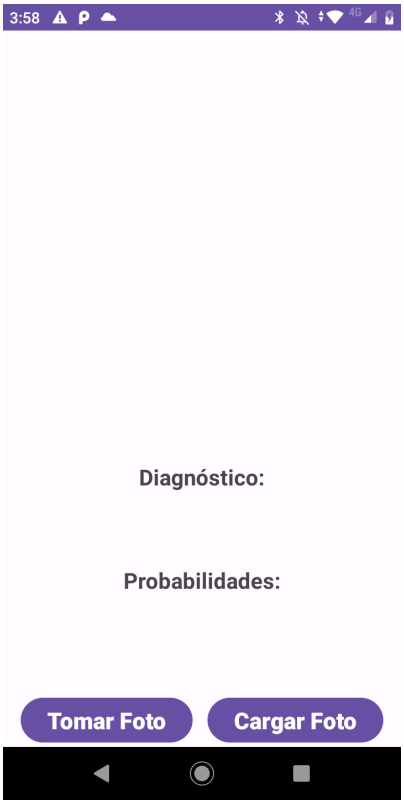


Figura 22: Pantalla principal de la interfaz móvil.

La primera funcionalidad de la interfaz permite al usuario tomar fotografías de manera directa y realizar predicciones en tiempo real. Como se ilustra en la Figura 23, la interfaz posibilita la captura de imágenes mediante la cámara del dispositivo, brindando al usuario la opción de repetir la toma si no cumple con los estándares de calidad deseados. En este caso, se simuló un cuello uterino enfocando una fotografía desde una pantalla de computadora.

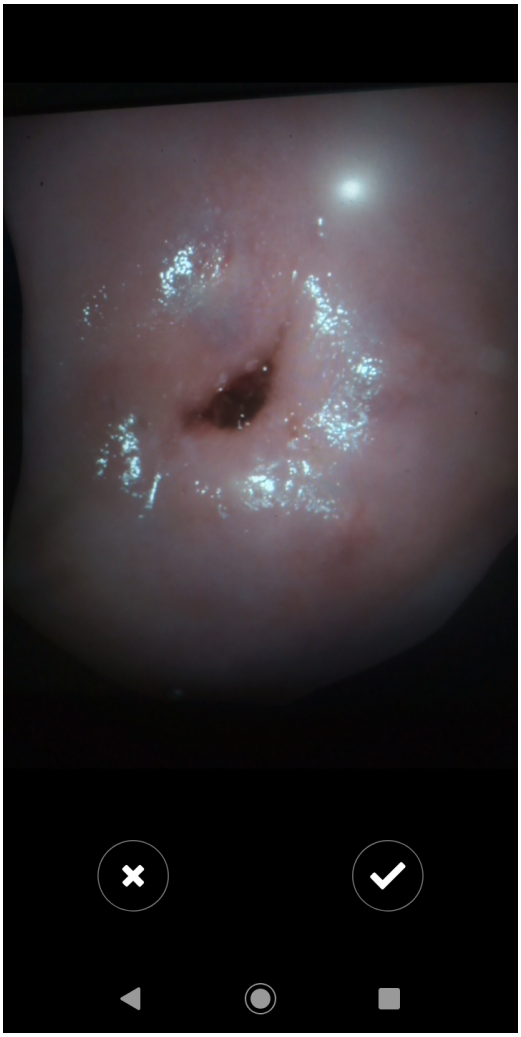


Figura 23: Funcionalidad de tomar foto.

Figura 24: Predicción después de tomar la foto.

Una vez seleccionada la foto, el usuario regresa a la pantalla principal, donde se visualiza la predicción del diagnóstico, como se muestra en la Figura 24.

Por otro lado, la siguiente funcionalidad permite cargar fotografías previamente almacenadas en la galería del dispositivo, notese en la Figura 25.

Figura 25: Funcionalidad de cargar foto desde la galería.

Figura 26: Predicción después de cargar la foto.

Cuando se selecciona la foto, esta es enviada al sistema inteligente que devuelve el diagnóstico correspondiente, como se muestra en la Figura 26.

10. Conclusiones y Trabajos Futuros

El presente trabajo ha proporcionado un marco de referencia sólido para la identificación y diferenciación de imágenes relacionadas con el cáncer de cuello uterino, clasificándolas según su grado de riesgo o displasia. En el transcurso de este proyecto, se logró un profundo entendimiento de las razones que motivan el abordaje de este problema en el estado del arte, especialmente desde la perspectiva del Deep Learning.

Una contribución significativa de este estudio fue la capacidad de abordar el conjunto de datos relacionado con el cáncer de cuello uterino, incluso sin experiencia médica previa por parte del estudiante. Este logro permitió comprender el dataset y, lo que es aún más crucial, adaptarlo y gestionarlo eficazmente en el contexto de modelos de aprendizaje profundo. La verdadera potencia de esta herramienta radica en sus modelos, capaces de establecer conexiones entre sí, transmitir conocimientos y destacar factores y características claves para la generalización de los detalles en cada tipo de imagen.

Además, se destacó la diversidad de alternativas disponibles para abordar los desafíos encontrados en el proceso. Desde la elección de redes y arquitecturas disponibles hasta la implementación de técnicas para contrarrestar el sobreajuste y mitigar el impacto de bases de datos pequeñas, se exploraron múltiples enfoques para garantizar un rendimiento óptimo.

El rendimiento destacado de los modelos también se atribuyó a una etapa rigurosa de preprocesamiento. Desde la carga inicial de imágenes hasta las transformaciones y estandarizaciones aplicadas para lograr uniformidad, incluso cuando las imágenes provenían de diversas fuentes, se implementaron medidas para garantizar la coherencia del conjunto de datos. El uso de aumentos de datos y la segmentación de imágenes se revelaron como pasos cruciales, especialmente para contrarrestar el ruido presente en imágenes del NCI que podría afectar el entrenamiento y sesgar los modelos.

Se subrayó la importancia del ajuste de hiperparámetros y del proceso de Tuning para encontrar el punto óptimo en el rendimiento de cada modelo. La técnica seleccionada, que demostró un rendimiento destacado, fue la red AlexNet. Su integración exitosa en la interfaz diseñada refuerza su preferencia para la clasificación de dos clases, una decisión tomada considerando los futuros pasos del proyecto CITOBOT. Del mismo modo, fue interesante ver los resultados de las redes Transformers, las cuales, aunque no fueron diseñadas para este tipo de problemas siguen demostrando porque son de las mas usadas en el estado del arte actualmente, logrando tener notables resultados.

Es evidente que la complejidad aumenta al trabajar con sistemas inteligentes que deben clasificar en múltiples clases. Se reconoce la necesidad crucial de mejorar el conjunto de datos de imágenes para continuar elevando las métricas en este tipo de problemas. Es decir que, el modelo se afinará y mejorará en la medida que se logren obtener más imágenes etiquetadas correctamente, las cuales puedan ser incluidas en nuestro conjunto de datos de entrenamiento y validación. La experiencia ganada revela que el aumento de clases de salida complica la generalización y la identificación de características distintivas.

En última instancia, los objetivos iniciales del proyecto, que buscaban alcanzar un 60 % en las métricas de especificidad y sensibilidad, se cumplieron. Se proporcionan modelos implementados en Python capaces de realizar un diagnóstico preciso para la clasificación de imágenes de cuello uterino. La red AlexNet se destaca como la preferida para la clasificación de dos clases, mientras que la red VGGNET muestra eficacia para la clasificación de tres clases.

Este trabajo también destaca la importancia de todas las etapas en la construcción de modelos de Machine/Deep Learning, desde la comprensión de los datos hasta el procesamiento y la utilización de la información. Se proyecta la implementación e integración futura de estas redes en la aplicación oficial de CITOBOT, actualmente en construcción. Las métricas proporcionadas al proyecto facilitarán la selección de

imágenes válidas para la base de datos futura, permitiendo que los modelos mejoren continuamente su rendimiento.

Además, gracias a los resultados obtenidos, se llevará a cabo un artículo como trabajo futuro. Este artículo se publicará con el propósito de compartir los resultados y métricas obtenidas durante el desarrollo del proyecto. Este documento servirá como contribución al conocimiento en el campo de Machine/Deep Learning aplicado a la detección y clasificación de imágenes colposcópicas de cáncer cervical.

Por otro lado, una vez integrada la red inteligente en la aplicación oficial de CITOBOT, se tiene la intención de seguir entrenándola y actualizándola de manera continua. El objetivo es permitir que la red mejore sus predicciones a medida que el conjunto de datos aumente, como se mencionó anteriormente, asegurando así un rendimiento óptimo en la clasificación de imágenes. Adicionalmente, se planea la incorporación de un módulo de segmentación dentro de la aplicación. Este módulo se encargará de segmentar previamente las imágenes que lleguen a la aplicación, proporcionando datos más refinados para el clasificador. Esta mejora en la calidad de los datos contribuirá a la precisión general del sistema.

Finalmente, se contempla también la implementación de un algoritmo de inteligencia artificial adicional. Este algoritmo realizará un análisis previo sobre las imágenes, verificando que efectivamente se trate únicamente de una imagen de cuello uterino y que cumpla con las métricas de calidad establecidas. Esta adición ayudará a fortalecer la robustez y precisión del sistema, asegurando que solo se utilicen datos relevantes y de alta calidad en el proceso de clasificación.

10.1. Anexos, Códigos, y Participaciones

El proyecto de grado “CITOBOT: un enfoque de inteligencia artificial para la detección temprana del cáncer de cuello uterino” fue seleccionado para participar en el XXV Simposio Nacional e Internacional de Investigaciones en Salud organizado por

la Universidad del Valle, según se detalla en los Anexos A y B. Durante este evento, se presentaron los resultados obtenidos y se describió detalladamente el proceso de construcción y preparación de datos para los modelos de Machine/Deep Learning.

Adicionalmente, al final del documento, se encuentran los anexos relacionados con la implementación de las arquitecturas de cada modelo. Por otro lado, se proporciona el Anexo H, que incluye un enlace al repositorio de Github. Este enlace le llevará a la ubicación donde puede revisar el código de los modelos implementados, así como el código correspondiente a la interfaz móvil. Asimismo, se han cargado los archivos .pdf de las arquitecturas en caso de que desee descargarlos para una visualización más detallada.

11. Referencias Bibliográficas

Referencias

[1] B. Bai, Y. Du, P. Li, and Y. Lv, "Cervical lesion detection net," in *2019 IEEE 13th International Conference on Anti-counterfeiting, Security, and Identification (ASID)*, 2019, pp. 168–172.

[2] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan, "Review of deep learning: Concepts, cnn architectures, challenges, applications, future directions," *Journal of big Data*, vol. 8, pp. 1–74, 2021.

[3] X. Han, "Mr-based synthetic ct generation using a deep convolutional neural network method," *Medical physics*, vol. 44, no. 4, pp. 1408–1419, 2017.

[4] H. N. Wagner Jr and P. S. Conti, "Advances in medical imaging for cancer diagnosis and treatment," *Cancer*, vol. 67, no. S4, pp. 1121–1128, 1991.

[5] A. Jemal, F. Bray, M. M. Center, J. Ferlay, E. Ward, and D. Forman, "Global cancer statistics," *CA: A Cancer Journal for Clinicians*, vol. 61, no. 2, pp. 69–90, 2011. [Online]. Available: https://acsjournals.onlinelibrary.wiley.com/doi/abs/10.3322/caac.20107

[6] E. De Vries, I. Arroyave, and C. Pardo, "Re-emergence of educational inequalities in cervical cancer mortality, colombia 1998–2015," *Journal of cancer policy*, vol. 15, pp. 37–44, 2018.

[7] WHO, "Global strategy to accelerate the elimination of cervical cancer as a public health problem," 2020.

[8] Organización Mundial de la Salud, "Cáncer de cuello uterino," https://www.who.int/es/news-room/fact-sheets/detail/cervical-cancer, 2021.

[9] M. Schiffman, J. Doorbar, N. Wentzensen, S. De Sanjosé, C. Fakhry, B. J. Monk, M. A. Stanley, and S. Franceschi, "Carcinogenic human papillomavirus infection," *Nature reviews Disease primers*, vol. 2, no. 1, pp. 1–20, 2016.

[10] Ministerio de Salud y Protección Social, "Cáncer de cuello uterino," https://www.minsalud.gov.co/salud/publica/ssr/Paginas/Cancer-de-cuello-uterino.aspx, 2020.

[11] N. Muñoz and L. E. Bravo, "Epidemiology of cervical cancer in colombia," *Colombia Médica*, vol. 43, no. 4, pp. 298–304, 2012.

[12] N. Santesso, R. A. Mustafa, H. J. Schünemann, M. Arbyn, P. D. Blumenthal, J. Cain, M. Chirenje, L. Denny, H. De Vuyst, L. O. Eckert *et al.*, "World health organization guidelines for treatment of cervical intraepithelial neoplasia 2–3 and screen-and-treat strategies to prevent cervical cancer," *International Journal of Gynecology & Obstetrics*, vol. 132, no. 3, pp. 252–258, 2016.

[13] S. Gordon, G. Zimmerman, and H. Greenspan, "Image segmentation of uterine cervix images for indexing in pacs," in *Proceedings. 17th IEEE Symposium on Computer-Based Medical Systems*, 2004, pp. 298–.

[14] R. Elakkiya, V. Subramaniyaswamy, V. Vijayakumar, and A. Mahanti, "Cervical cancer diagnostics healthcare system using hybrid object detection adversarial networks," *IEEE Journal of Biomedical and Health Informatics*, vol. 26, no. 4, pp. 1464–1471, 2022.

[15] Instituto Nacional de Salud, "Boletín epidemiológico semana 23," https://www.ins.gov.co/buscador-eventos/BoletinEpidemiologico/2018%20Bolet%C3%ADn%20epidemiol%C3%B3gico%20semana%2023.pdf, 2018.

[16] Pontificia Universidad Javeriana, "Citobot, inteligencia artificial contra el cáncer de cuello uterino," https://www.javeriana.edu.co/documents/12789/10853545/P%C3%A1g.+6.+Citobot%2C+inteligencia+artificial+contra+el+c%C3%A1ncer+de+cuello+uterino+HJ+mayo+2019+WEB.pdf, 2019.

[17] H. Arabi, G. Zeng, G. Zheng, and H. Zaidi, "Novel deep learning-based ct synthesis algorithm for mri-guided pet attenuation correction in brain pet/mr imaging," in *2018 IEEE Nuclear Science Symposium and Medical Imaging Conference Proceedings (NSS/MIC)*. IEEE, 2018, pp. 1–3.

[18] Management Solutions, "Machine learning: Una pieza clave en la transformación de los modelos de negocio," https://www.managementsolutions.com/es/publicaciones-y-eventos/informes-sectoriales/white-papers/machine-learning-una-pieza-clave-en-la-transformacion-de-los-modelos-de-negocio, 2018.

[19] D. Shen, G. Wu, and H.-I. Suk, "Deep learning in medical image analysis," *Annual review of biomedical engineering*, vol. 19, pp. 221–248, 2017.

[20] D. Kreuzberger, N. Kühl, and S. Hirschl, "Machine learning operations (mlops): Overview, definition, and architecture," *IEEE Access*, vol. 11, pp. 31 866–31 879, 2023.

[21] S. Garg, P. Pundir, G. Rathee, P. Gupta, S. Garg, and S. Ahlawat, "On continuous integration / continuous delivery for automated deployment of machine learning models using mlops," in *2021 IEEE Fourth International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, 2021, pp. 25–28.

[22] M. Arrivillaga, P. C. Bermúdez, J. P. García-Cifuentes, M. Rodríguez-López, D. Neira, and H. D. Vargas-Cardona, "Women's critical experiences with the pap smear for the development of cervical cancer screening devices," *Heliyon*, 2023.

[23] S. E. Waggoner, "Cervical cancer," *The lancet*, vol. 361, no. 9376, pp. 2217–2225, 2003.

[24] T. J. Eun and R. B. Perkins, "Screening for cervical cancer," *Medical Clinics*, vol. 104, no. 6, pp. 1063–1078, 2020.

[25] S. A. Pimple and G. A. Mishra, "Global strategies for cervical cancer prevention and screening." *Minerva ginecologica*, vol. 71, no. 4, pp. 313–320, 2019.

[26] J. Potter, S. M. Peitzmeier, I. Bernstein, S. L. Reisner, N. M. Alizaga, M. Agénor, and D. J. Pardee, "Cervical cancer screening for patients on the female-to-male spectrum: a narrative review and guide for clinicians," *Journal of general internal medicine*, vol. 30, pp. 1857–1864, 2015.

[27] I. C. Garcés, D. C. Rubio, and I. C. Scarinci, "Factores asociados con el tamizaje de cáncer de cuello uterino en mujeres de nivel socioeconómico medio y bajo en bogotá, colombia," *Revista Facultad Nacional de Salud Pública*, vol. 30, no. 1, pp. 7–16, 2012.

[28] N. K. Chauhan and K. Singh, "Impact of variation in number of channels in cnn classification model for cervical cancer detection," in *2021 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, 2021, pp. 1–6.

[29] Q. Yang, Y. Zhang, W. Dai, and S. J. Pan, *Transfer learning*. Cambridge University Press, 2020.

[30] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[31] M. Chen, A. Carass, A. Jog, J. Lee, S. Roy, and J. L. Prince, "Cross contrast multi-channel image registration using image synthesis for mr brain images," *Medical image analysis*, vol. 36, pp. 2–14, 2017.

[32] R. David, J. Duke, A. Jain, V. Janapa Reddi, N. Jeffries, J. Li, N. Kreeger, I. Nappier, M. Natraj, T. Wang *et al.*, "Tensorflow lite micro: Embedded machi-

ne learning for tinyml systems," *Proceedings of Machine Learning and Systems*, vol. 3, pp. 800–811, 2021.

[33] V. Pavlov, S. Fyodorov, S. Zavjalov, T. Pervunina, I. Govorov, E. Komlichenko, V. Deynega, and V. Artemenko, "Simplified convolutional neural network application for cervix type classification via colposcopic images," *Bioengineering*, vol. 9, no. 6, p. 240, 2022.

[34] J. M. Helm, A. M. Swiergosz, H. S. Haeberle, J. M. Karnuta, J. L. Schaffer, V. E. Krebs, A. I. Spitzer, and P. N. Ramkumar, "Machine learning and artificial intelligence: definitions, applications, and future directions," *Current reviews in musculoskeletal medicine*, vol. 13, pp. 69–76, 2020.

[35] P. Wang, "On defining artificial intelligence," *Journal of Artificial General Intelligence*, vol. 10, no. 2, pp. 1–37, 2019.

[36] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[37] M. Goyal, T. Knackstedt, S. Yan, and S. Hassanpour, "Artificial intelligence-based image classification methods for diagnosis of skin cancer: Challenges and opportunities," *Computers in Biology and Medicine*, vol. 127, p. 104065, 2020.

[38] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43–76, 2020.

[39] J. C. Russ, J. R. Matey, A. J. Mallinckrodt, and S. McKay, "The image processing handbook," *Computers in Physics*, vol. 8, no. 2, pp. 177–178, 1994.

[40] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of big data*, vol. 6, no. 1, pp. 1–48, 2019.

[41] S. Minaee, Y. Boykov, F. Porikli, A. Plaza, N. Kehtarnavaz, and D. Terzopoulos, "Image segmentation using deep learning: A survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 7, pp. 3523–3542, 2021.

[42] H.-H. Chang and C.-C. Hsieh, "Brain segmentation in mr images using a texture-based classifier associated with mathematical morphology," in *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE, 2017, pp. 3421–3424.

[43] A. Dhillon and G. K. Verma, "Convolutional neural network: a review of models, methodologies and applications to object detection," *Progress in Artificial Intelligence*, vol. 9, no. 2, pp. 85–112, 2020.

[44] G. Yao, T. Lei, and J. Zhong, "A review of convolutional-neural-network-based action recognition," *Pattern Recognition Letters*, vol. 118, pp. 14–22, 2019.

[45] D. M. Hawkins, "The problem of overfitting," *Journal of chemical information and computer sciences*, vol. 44, no. 1, pp. 1–12, 2004.

[46] I. Bilbao and J. Bilbao, "Overfitting problem and the over-training in the era of data: Particularly for artificial neural networks," in *2017 eighth international conference on intelligent computing and information systems (ICICIS)*. IEEE, 2017, pp. 173–177.

[47] X. Ying, "An overview of overfitting and its solutions," in *Journal of physics: Conference series*, vol. 1168. IOP Publishing, 2019, p. 022022.

[48] J. I. Blanco. (28/04/2023) Por qué la normalización es clave e importante en machine learning y ciencia de datos. [Online]. Available: https://jorgeiblanco.medium.com/por-qu%C3%A9-la-normalizaci%C3%B3n-es-clave-e-importante-en-machine-learning-y-ciencia-de-datos-4595f15d5be0

[49] J. Barrios. (26/07/2019) La matriz de confusión y sus métricas. [Online]. Available: https://www.juanbarrios.com/la-matriz-de-confusion-y-sus-metricas/

[50] J. Brownlee. (23/10/2019) Loss and loss functions for training deep learning neural networks. [Online]. Available: https://machinelearningmastery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks/

[51] D. Rodríguez. (16 de diciembre de 2019) ¿cuál es la diferencia entre parámetro e hiperparámetro? [Online]. Available: https://www.analyticslane.com/2019/12/16/cual-es-la-diferencia-entre-parametro-e-hiperparametro/#google_vignette

[52] J. Brownlee. (12 de septiembre de 2020) Understand the impact of learning rate on neural network performance. [Online]. Available: https://machinelearningmastery.com/understand-the-impact-of-learning-rate-on-neural-networks/

[53] ——. (15 de agosto de 2022) Difference between a batch and an epoch in a neural network. [Online]. Available: https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/

[54] X. Su, J. Shi, Y. Peng, and L. Zheng, "Cervical cell image classification based on multiple attention fusion," in *2021 14th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, 2021, pp. 1–5.

[55] N. Sompawong, J. Mopan, P. Pooprasert, W. Himakhun, K. Suwannarurk, J. Ngamvirojcharoen, T. Vachiramon, and C. Tantibundhit, "Automated pap smear cervical cancer screening using deep learning," in *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2019, pp. 7044–7048.

[56] Y. Song, L. Zhang, S. Chen, D. Ni, B. Li, Y. Zhou, B. Lei, and T. Wang, "A deep learning based framework for accurate segmentation of cervical cytoplasm and nuclei," in *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 2014, pp. 2903–2906.

[57] Y.-M. Luo, T. Zhang, P. Li, P.-Z. Liu, P. Sun, B. Dong, and G. Ruan, "Md-fi: Multi-cnn decision feature integration for diagnosis of cervical precancerous lesions," *IEEE Access*, vol. 8, pp. 29 616–29 626, 2020.

[58] W. Liu, C. Li, N. Xu, T. Jiang, M. M. Rahaman, H. Sun, X. Wu, W. Hu, H. Chen, C. Sun, Y. Yao, and M. Grzegorzek, "Cvm-cervix:

A hybrid cervical pap-smear image classification framework using cnn, visual transformer and multilayer perceptron," *Pattern Recognition*, vol. 130, p. 108829, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320322003107

[59] V. Kudva, K. Prasad, and S. Guruvare, "Andriod device-based cervical cancer screening for resource-poor settings," *Journal of digital imaging*, vol. 31, pp. 646–654, 2018.

[60] C. Schröer, F. Kruse, and J. M. Gómez, "A systematic literature review on applying crisp-dm process model," *Procedia Computer Science*, vol. 181, pp. 526–534, 2021.

A. Certificado de Ponente

Cali y el Valle Investigan en Salud

25 SIMPOSIO de Investigaciones En Salud INTERNACIONAL

Universidad del Valle — Facultad de Salud

Cali, 7 de Julio de 2023

UNIVERSIDAD DEL VALLE - FACULTAD DE SALUD

Certifica:

Que el investigador(a) David Steven Rivero Urbano identificado(a) con el número de documento 1107531622 participó como ponente del proyecto CITOBOT: un enfoque de inteligencia artificial para la detección temprana de cáncer de cuello uterino. bajo la modalidad de Artículo Original e identificado con el código interno V131139-2507 en el marco del XXV Simposio Nacional e Internacional de Investigaciones en Salud. Según resolución No. 308 del Consejo de Facultad del 04 de Octubre de 2022 Realizado del 5 al 8 de Junio de 2023. Intensidad Horaria: 70 horas

Carlos H. Valencia Ll.
Coordinador Académico Simposio
Universidad del Valle

Jhonathan Stick Guerrero
Director de Extension y proyección social
Facultad de Salud, Universidad del Valle

B. Certificado de Asistente

Cali y el Valle Investigan en Salud

Cali, 7 de Julio de 2023

UNIVERSIDAD DEL VALLE - FACULTAD DE SALUD

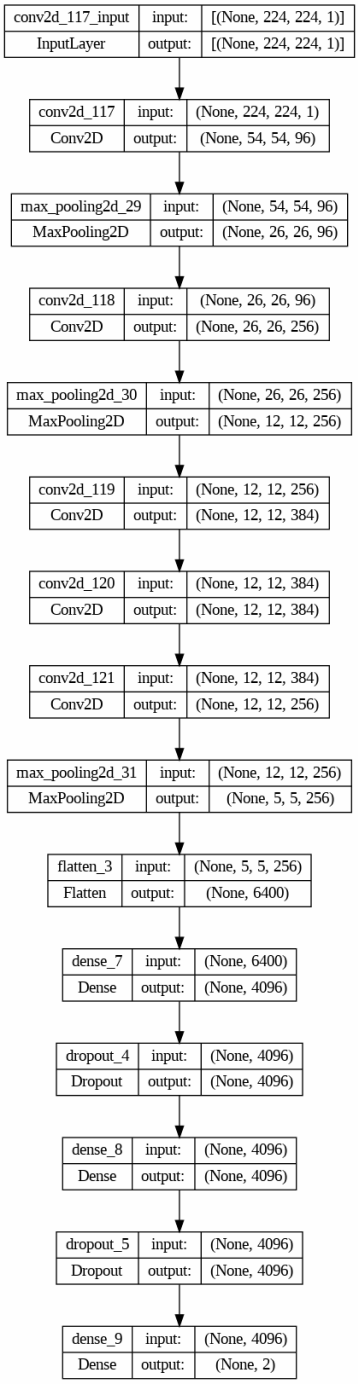
Certifica:

Que David Steven Rivero Urbano identificado(a) con el número de documento 1107531622 participó como asistente en el marco del XXV Simposio Nacional e Internacional de Investigaciones en Salud. Según resolución No. 308 del Consejo de Facultad del 04 de Octubre de 2022 Realizado del 5 al 8 de Junio de 2023. Intensidad Horaria: 70 horas

Carlos H. Valencia Ll.
Coordinador Académico Simposio
Universidad del Valle

Jhonathan Stick Guerrero
Director de Extension y proyección social
Facultad de Salud, Universidad del Valle

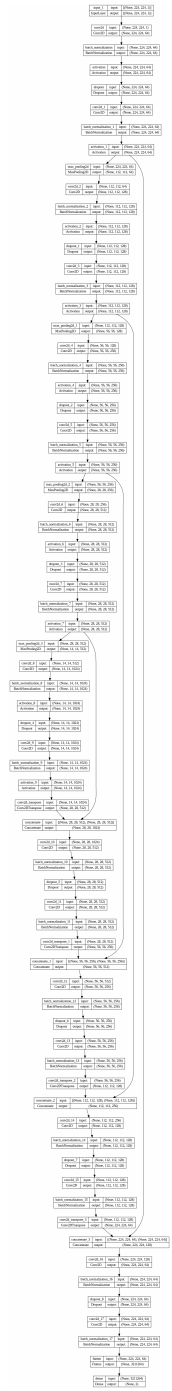
C. Arquitectura AlexNet implementada



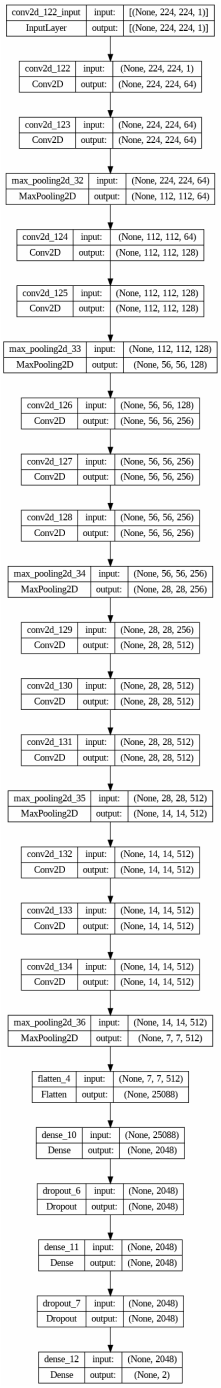
D. Arquitectura GoogleNet implementada

E. Arquitectura Resnet implementada

F. Arquitectura U-Net implementada



G. Arquitectura VGGNET implementada



H. Códigos

A continuación se adjunta el enlace del repositorio donde encontrara la implementación de los modelos en Python, y el código de la interfaz móvil implementada en Android Studio.

https://github.com/drivero1999/Cervical-Cancer-Detection-From-Deep-Learning-Based-Colposcop