Pontificia Universidad
JAVERIANA
Cali

Facultad de Ingeniería
y Ciencias
Ingeniería Electrónica

Monografía de Trabajo de Grado

SEGMENTACIÓN DEL CUELLO UTERINO EN IMÁGENES DE COLPOSCOPIA MEDIANTE TÉCNICAS DE APRENDIZAJE DE MAQUINA

Ana Maria Bolaños Semanate
Santiago Hurtado Bustos

Director
Dr. Hernan Dario Vargas Cardona

14 de febrero de 2024

Pontificia Universidad JAVERIANA Cali

Facultad de Ingeniería y Ciencias

Acta de Correcciones al Proyecto de Grado
Ingeniería Electrónica

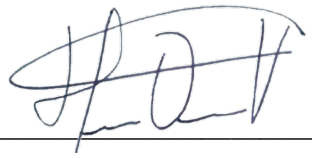
Fecha: 14/febrero/2024

Autores:
Ana Maria Bolaños Semanate
Santiago Hurtado Bustos

Nombre del Proyecto de Grado:
SEGMENTACIÓN DEL CUELLO UTERINO EN IMÁGENES DE COLPOSCOPIA MEDIANTE TÉCNICAS DE APRENDIZAJE DE MAQUINA

Director:

Como indica el artículo 2.27 de las Directrices de Trabajo de Grado, he verificado que los estudiantes indicados arriba han implementado todas las correcciones que los Jurados del Proyecto de Grado definieron que se efectuaran, como consta en el Acta de Calificación correspondiente.

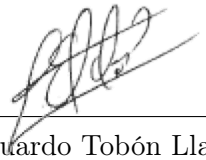


Firma de Director(a) del Proyecto de Grado

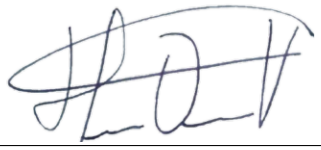
Nota de Aceptación

Aprobado por el Comité de Trabajo de Grado en cumplimiento de los requisitos exigidos por la Pontificia Universidad Javeriana para optar al título de Ingeniero Electrónico.

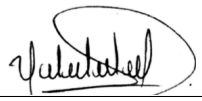
Dr. Hernán Camilo Rocha Niño
Decano Facultad de Ingeniería y Ciencias



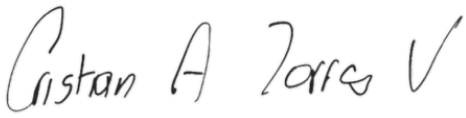
Dr. Luis Eduardo Tobón Llano
Director Carrera Ingeniería Electrónica



Dr. Hernan Dario Vargas Cardona
Director Trabajo de Grado



Mag. Valentina Corchuelo Guzmán
Jurado 1



PhD. Cristian Alejandro Torres Valencia
Jurado 2

Santiago de Cali, 14 de febrero de 2024

Señores
Pontificia Universidad Javeriana – Cali
Dr. Hernán Camilo Rocha Niño
Decano
Facultad de Ingeniería y Ciencias
Ciudad

Cordial Saludo.

Por medio de la presente nos permitimos presentarle el Trabajo de Grado titulado "SEGMENTACIÓN DEL CUELLO UTERINO EN IMÁGENES DE COLPOSCOPIA MEDIANTE TÉCNICAS DE APRENDIZAJE DE MAQUINA".

Esperamos que este trabajo reúna todos los requisitos académicos, cumpla el propósito para el cual fue creado y sirva de apoyo para futuros proyectos relacionados con la materia.

Atentamente,

Ana Maria Bolaños Semanate

Ana Maria Bolaños Semanate

Santiago Hurtado Bustos

Santiago Hurtado Bustos

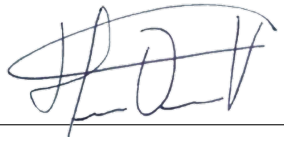
Santiago de Cali, 14 de febrero de 2024

Señores
Pontificia Universidad Javeriana – Cali
Dr. Hernán Camilo Rocha Niño
Decano
Facultad de Ingeniería y Ciencias
Ciudad

Cordial Saludo.

Certifico que el presente Trabajo de Grado titulado "SEGMENTACIÓN DEL CUELLO UTERINO EN IMÁGENES DE COLPOSCOPIA MEDIANTE TÉCNICAS DE APRENDIZAJE DE MAQUINA", realizado por Ana Maria Bolaños Semanate y Santiago Hurtado Bustos, estudiantes de Ingeniería Electrónica, se encuentra terminado y puede ser presentado para su sustentación.

Atentamente,



Dr. Hernan Dario Vargas Cardona
Director Trabajo de Grado

Agradecimientos

La culminación de esta tesis ha sido un viaje desafiante pero gratificante, y nos gustaría aprovechar esta oportunidad para expresar nuestro agradecimiento a todas las personas que han desempeñado un papel fundamental en este proceso.

En primer lugar agradecemos a nuestras familias por su amor, apoyo incondicional y aliento constante han sido una fuente de esperanza e inspiración durante los momentos más retadores.

También agradecemos a la universidad Javeriana por brindar todas las herramientas para realizar el desarrollo de esta tesis. Seguidamente, deseamos agradecer a nuestro tutor, Hernan Vargas , por su guía experta y apoyo constante a lo largo de esta investigación, su sabiduría, paciencia y dedicación fueron esenciales para nuestro crecimiento académico.

De igual manera extendemos agradecimientos al profesor Hernan Benitez, por su tiempo, retroalimentación valiosa que mejoraron la calidad de esta tesis. Queremos expresar nuestro agradecimiento a todos las personas que han hecho parte de este proyecto,que con sus conocimientos, experiencias, discusiones y colaboraciones han enriquecido este trabajo.

Agradecemos a Minciencias, ya que financiaron esta investigación y su apoyo financiero hizo posible este proyecto, a la OID (Oficina de investigación y desarrollo) por su gestión y apoyo.También agradecemos al equipo interdisciplinario del proyecto Citobot por su dedicación, compromiso y colaboración incansable.

A todas las personas que mencioné anteriormente y a aquellas que de alguna manera contribuyeron a esta tesis, les agradecemos sinceramente. Su apoyo ha sido fundamental en la realización de este logro.

¡Gracias a todos!

Glosario

Acrónimos y Abreviaturas

VPH	Virus del papiloma humano
CU	Cáncer de cuello uterino
OMS	Organización mundial de la salud
IoU	Intersecction over union ó Intersección sobre Unión
MG	Modelo de mezclas gausianas
EM	Expectation-maximization ó Maximización de expectativas

Resumen

El virus del papiloma humano (VPH) es una enfermedad de transmisión sexual que puede desencadenar cáncer de cuello uterino, siendo esta la cuarta neoplasia más frecuente en las mujeres a nivel mundial y la segunda a nivel nacional, convirtiéndose en uno de los principales problemas de salud pública. Por esta situación, actualmente se realizan diferentes campañas de salud que promueven los exámenes de tamización del cáncer de cuello uterino como: citología, colposcopia y prueba del virus del papiloma humano, los cuales permiten la detección de esta enfermedad. Sin embargo, los tiempos de espera para obtener los resultados son altos debido a la infraestructura deficiente de los laboratorios, incluso las pruebas recolectadas en los exámenes están sujetas a una manipulación o almacenamiento inadecuado afectando el diagnóstico, ocasionando que la prueba se deba tomar de nuevo, postergando el diagnóstico.

Ante esta situación donde la salud de millones de mujeres se ve afectada cada año, se opta por aprovechar las imágenes de colposcopia para analizar el estado del cuello uterino mediante técnicas de aprendizaje de máquina como una herramienta para soportar el diagnóstico de las pacientes, empleando métodos supervisados y no supervisados, para mejorar el tiempo de diagnóstico de esta enfermedad, debido a que esto permite evaluar un gran número de datos en menor tiempo.
Con el fin de ayudar a dar solución a este problema de salud pública, se desarrolló un aplicativo de escritorio que cuenta con modelos de aprendizaje supervisado y no supervisado. Para llegar a este producto fue necesario acondicionar las bases de datos para que las imágenes sean uniformes, revisar documentación de trabajos previos, así como también modelos, para poder implementarlos, finalmente se evaluaron todos los modelos para definir cuales cumplen con las métricas establecidas que garantizan un buen rendimiento para el proyecto de CITOBOT, el cual es financiado por Minciencias y está siendo desarrollado por un equipo multidisciplinario de las facultades de ingeniería y ciencias de la salud, en colaboración con la ESE Ladera de la ciudad de Cali.

Palabras Clave: segmentación, aprendizaje de máquina, cuello uterino, imágenes de colposcopia, VPH.

Abstract

The human papillomavirus (HPV) is a sexually transmitted disease that can trigger cervical cancer, being the fourth most frequent neoplasm in women worldwide and the second at national level, becoming one of the main public health problems. Due to this situation, different health campaigns are currently being carried out to promote cervical cancer screening tests such as: cytology, colposcopy and human papillomavirus test, which allow the detection of this disease. However, the waiting times to obtain the results are high due to the deficient infrastructure of the laboratories, even the tests collected in the exams are subject to inadequate handling or storage affecting the diagnosis, causing the test to be taken again, delaying the diagnosis.

Faced with this situation where the health of millions of women is affected every year, it is decided to take advantage of colposcopy images to analyze the state of the cervix through machine learning techniques as a tool to support the diagnosis of patients, using supervised and unsupervised methods, to improve the time of diagnosis of this disease, because this allows to evaluate a large number of data in less time.

To build the application capable of segmenting any colposcopy image using machine learning, it will be necessary to condition the database so that the images are uniform. Likewise, the documentation process of previous works and methods will be important at the moment of implementing machine learning algorithms, from the results obtained with the different methods it is determined which results comply with the metrics and serve correctly to be used in the CITOBOT project for the identification of cancer stages.

Translated with www.DeepL.com/Translator (free version)

Keywords: segmentation, machine learning, cervix, colposcopy imaging, HPV.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

Este trabajo de grado abordará la temática del Virus del Papiloma Humano (VPH) debido a su alta incidencia en las mujeres de Latinoamérica, donde el cáncer de cuello uterino es la principal causa de muerte en los países en vía de desarrollo. Cada año, se producen alrededor de 466000 nuevos casos a nivel mundial y el 80 % se dan en la zona mencionada anteriormente [6]. La principal razón de esta situación se debe a la falta de programas enfocados a diagnosticar lesiones precancerosas y tratarlas antes de que evolucionen a un cáncer invasor, este aparece en los tejidos que se encuentran en la parte baja del útero y se expande hacia el epitelio exocervical. Si las lesiones en estos tejidos son detectados tempranamente, la probabilidad de cura total es muy alta.

Lo anterior plantea un problema a nivel social, dado que por cada mujer que fallece quedan entre 5 a 7 niños huérfanos anualmente, lo que equivale aproximadamente a 17000 niños. Los procedimientos actuales presentan altos porcentajes de error, 30 % son falsos positivos y el 20 % son falsos negativos. En el mejor de los casos se realizan biopsias innecesarias, soportadas en falsos positivos, generando altos costos para el sector salud. Otro factor importante, se relaciona con las molestias causadas debido a los procedimientos clínicos, dado que en ocasiones, se debe retirar tejido de la zona 'afectada' más de una vez [6].

Actualmente se lleva a cabo el examen de Papanicolau o citología, el cual solo tarda unos minutos. Previo al examen pueden solicitar a la paciente retirarse las prendas solo de la cintura para abajo, posteriormente, se recostara en una camilla de examen con las rodillas flexionadas y apoyadas sobre unos soportes llamados estribos. Luego, el médico introducirá un instrumento un espéculo con el objetivo de mantener las paredes vaginales separadas y visualizar fácilmente el cuello uterino. El procedimiento descrito anteriormente puede generar presión en la zona pélvica. Finalmente, el médico tomara las muestras de las células cervicales empleando un cepillo suave y otro implemento denominado espátula. Después de tomadas las muestras el médico las depositará en un recipiente con un líquido especial para preservar la muestra (prueba de Papanicolau en base liquida) o portaobjetos de vidrio (prueba de papanicolaou convencional). Cuando la toma de las muestras finaliza, se trasladan a un laboratorio donde son analizadas con microscopio para detectar características de las células o un trastorno precanceroso. Este tipo de exámenes es recomendable iniciarlos a la edad de los 21 años y luego repetir este proceso cada 3 años, sin embargo, es preferible que las mujeres de 30 años en adelante se realicen el examen cada 5 años. Es importante destacar que la prueba deberá llevarse a cabo con mayor frecuencia, si la paciente tiene una infección por VIH, antecedentes de tabaquismo, debilitación del sistema inmunitario debido al trasplante de órganos, quimioterapias o uso crónico de corticoesteroides, así como exposición al diestilestilbestrol. Los profesionales en la salud aconsejan dejar de tomar la prueba a partir de los 65 años y si los exámenes anteriores resultaron negativos [10].

Lo descrito anteriormente demuestra que esta patología es una problemática real la cual puede

abordarse desde el campo de la electrónica, más específicamente en el área de análisis digital de imágenes. Ya que se emplea el colposcopio para realizar inspecciones visuales sobre el útero. En la actualidad se comercializan los colposcopios digitales con los cuales se puede tomar imágenes y vídeos con un mayor contraste y definición [11], posibilitando el uso de programas informáticos. Por ende, se emplearán diversas técnicas de segmentación de imágenes a través de machine learning, incluyendo aprendizaje supervisado y no supervisado, con algoritmos como: K-means, K-medians, Watershed, Otsu, histogramas, combinación de métodos, entre otros. El objetivo de la segmentación es extraer información del cuello uterino a partir de las imágenes médicas, cumpliendo con las diferentes métricas propias de los modelos y generando imágenes adecuadas para análisis posteriores en este trabajo de grado. Las imágenes con las cuales se entrenarán los modelos pertenecen a una base de datos de la organización mundial de la salud y el instituto de cáncer de Estados Unidos adquiridos para este trabajo, es importante recalcar que antes de usar la base de datos, se realizó una preselección y segmentación manual de los datos que se usarán para que la información sea uniforme.

Capítulo 2

Planteamiento del Problema

El Virus del Papiloma Humano (VPH) es una infección de transmisión sexual, pero otros factores como sistemas inmunitarios debilitados, contacto con personas o superficies infectadas, así como el uso de duchas comunes o piscinas, pueden aumentar el riesgo de contraer esta enfermedad. La Sociedad Americana del Cáncer indica que existen alrededor de 150 variaciones del VPH, algunas de las cuales pueden causar verrugas que son tumores benignos. Sin embargo, ciertos tipos de VPH, como los tipos 6, 11, 16 y 18, pueden generar lesiones que conducen al cáncer [12]. Debido a su atracción por las células epiteliales escamosas, que recubren tanto la piel como las superficies mucosas del cuerpo, el Virus del Papiloma Humano (VPH) se clasifica en dos grupos principales: el VPH cutáneo, que produce verrugas en la piel, y el VPH mucoso, que puede sobrevivir y propagarse en áreas mucosas como los genitales, el revestimiento de la boca y la garganta. El VPH mucoso tiene dos clasificaciones: bajo y alto riesgo. Ambos tipos pueden causar afecciones en el cuerpo humano, pero varían en su gravedad.

En el caso de las verrugas causadas por el VPH de "bajo riesgo", estas pueden aparecer en brazos, manos, pies o el pecho, pero generalmente no causan problemas de salud y suelen desaparecer por sí solas o con un tratamiento menor realizado por un médico. Por otro lado, el VPH de .ªlto riesgo"puede causar síntomas que pueden desaparecer en algunos años, pero también puede provocar alteraciones en la estructura de las células que pueden desencadenar algún tipo de cáncer [10] [13].

Como se mencionó anteriormente, la clasificación de alto riesgo puede causar diferentes tipos de cáncer entre ellos está el cáncer de cuello uterino, cáncer de ano, cáncer de vulva, cáncer de garganta, cáncer de boca o cáncer de vagina. Siendo el cáncer de cuello uterino, uno de los más frecuentes en las mujeres, después del cáncer de mama, y el quinto más frecuente de todos los cánceres [14]. Una de las afecciones más representativas del VPH es el cáncer de cuello uterino, ya que es la segunda neoplasia más común en mujeres de América latina, con aproximadamente 68,818 casos al año, donde el 83 % de la población afectada se encuentra en países que están en vía de desarrollo, como: Colombia, Perú, Venezuela, México, Brasil y Argentina [14]. Si bien, es cierto que los programas de tamización de cáncer de cuello uterino han reducido la tasa de mortalidad en un 80 % respecto al año 1997, actualmente esta cifra es de 75 % correspondiente a 28.565 defunciones anuales en latino América [15].

El método utilizado para detectar esta enfermedad es el Papanicolaou, también conocido como citología. Este examen ayuda a detectar células precancerosas y a prevenir el desarrollo del cáncer mediante chequeos periódicos, que varían según la edad de la paciente, por ejemplo: cada 3 años para mujeres entre los 21-29 años. El objetivo de este examen es detectar anomalías en las células cervicales, que se encuentran en el revestimiento del cuello uterino, y determinar si han cambiado de apariencia, lo que se conoce como displasia cervical. Para complementar el estudio realizado con el Papanicolaou, se emplea la colposcopia, un examen que ayuda a detectar lesiones, verrugas,

mediante el uso de un colposcopio, el cual está dotado de lentes que permite la visualización con aumento de las zonas de interés [11]. En torno al impacto que tiene la variación del VPH asociado a la neoplasia de cérvix en Latinoamérica, se ha identificado que el proceso de diagnóstico del cáncer de cuello uterino puede presentar falsos positivos, así como también, lapsos de espera superiores a los 5 días en la obtención de los resultados. Los diagnósticos equivocados se generan por diferentes factores, una de estas razones se debe a la recolección inadecuada de las muestras, generando que la cantidad de células recolectadas sea insuficiente para ser analizadas, además, la sangre o células inflamadas bloquearían las células afectadas o anómalas [16]. La segunda causa se relaciona con los tiempos de respuesta [17][18] en los laboratorios para evaluar las muestras, a razón de que en algunos hospitales hay carencia de equipos especializados para analizar los datos, debido a los altos costos, por lo que deben solicitarse instalaciones a modo de préstamo para este proceso. Los tiempos de espera en la obtención de los resultados del Papanicolaou son, uno de los muchos casos en donde se puede evidenciar tiempos de espera altos para acceder al servicio médico. Referente a esta situación, se han observado casos en donde los tiempos de espera para una cita de ginecología puede tardar hasta dos meses, haciendo que los procesos de diagnóstico de patologías como el cáncer sean tardías, ocasionando que el éxito de los tratamientos se reduzca [19].

Por otro lado, a nivel de ingeniería uno de los problemas con el procesamiento de imágenes está relacionado con el ruido, entendiendo por ruido, un píxel que tiene un valor que no corresponde al real. En la mayoría de las ocasiones, este ruido proviene del equipo electrónico con el cual se han tomado la medición, por ejemplo, ruido de cuantización de la imagen y efecto de 'Blurring'. La fuente principal de ruido de una imagen médica aparece durante la adquisición de la misma y/o transmisión. Esto se debe a que los sensores empleados por los equipos biomédicos pueden ser afectados por condiciones de climatización durante la obtención de los datos y la calidad del dispositivo empleado. Adicional a ello, durante la transmisión de la información, las imágenes son degradadas por las interferencias presentes en el medio de transmisión.

Respecto a la densidad de los datos, se tiene que los algoritmos basados en densidad probabilística, según los estudios producen mejores resultados, dado que tienen en consideración grupos de forma arbitrarias y adicionalmente tienen la capacidad de adaptarse a la dinámica del flujo y a la presencia de ruido. Entre las principales PDF (Funciones de distribución probabilística) se encuentra la distribución Gaussiana, puesto que los tipos más comunes de ruido son generados por la contribución de varias señales diferentes, lo cual es una consecuencia del teorema del límite central. Este teorema establece que la suma de varias variables aleatorias con diversas PDFs tiene como resultado una señal con una PDF Gaussiana. Este modelo es ampliamente usado gracias a la facilidad del tratamiento matemático. Es importante mencionar que este ruido aparece debido a factores como ruidos del circuito electrónico y sensores que aparecen por factores como iluminación no apta para los procesos clínicos y/o temperaturas altas [20].

Estos factores conllevan a analizar el concepto de calidad en las imágenes médicas, es la capacidad que una imagen tiene de representar el objeto original, es decir, la exactitud o parecido entre ambos. Dentro de los modelos de medida de calidad, en lugar de tener una imagen y un objeto, la definición se extiende a dos imágenes de las cuales una es considerada la de referencia u original y la otra que ha sido procesada. Para lograr una calidad óptima se debe saber los objetivos de la imagen médica, por el ejemplo extraer información de organismos vivos (señales) y proveer trazado espacial discreto

(información anatómica). Es importante recalcar que la calidad no está definida por un único factor, sino al menos por una combinación de cinco, los cuales son; contraste, manchas, ruido, artefactos, y distorsión. Por ende, el propósito de la imagenología médica es la representación de la forma más clara posible de la morfología y fisiología humana.[21].

Otra dificultad existente en la colposcopia está relacionada con la baja especificidad, es decir que, si el entorno en donde se realizará la prueba no cuenta con las condiciones de iluminación apropiados, las lesiones aceto-blancas, es decir las secciones del cuello uterino que se tornan blancas después de aplicar ácido acético indicando la presencia de lesiones, generar confusiones durante el análisis, ya que, estas lesiones no se deben únicamente al VHP. [22].

Capítulo 3

Justificación

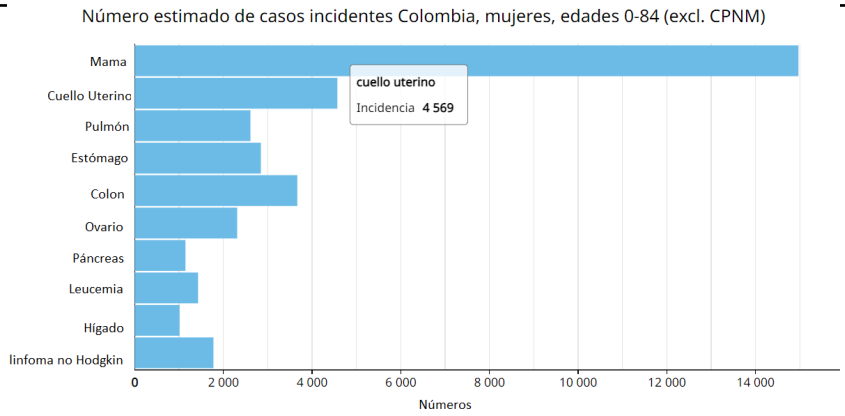
El cáncer de cuello uterino es una de las principales causas de muerte en mujeres a nivel mundial, lo que lo convierte en uno de los problemas de salud pública, con mayor incidencia en Latino América. Muestra de ello es que en Colombia para el año 2020, cerca de 4569 mujeres padecían de esta enfermedad y 2309 fallecieron a causa de esta, así lo indica la agencia internacional para la investigación de cáncer de la organización mundial de la salud (OMS). A partir de la información concentrada en la base de datos de la institución también se puede mencionar que este es el segundo cáncer con mayor índice de mortalidad y frecuencia a nivel nacional como se muestra en la figura 1.1 [1]

La neoplasia de cuello uterino es un problema de salud pública siendo este el cuarto tipo de cáncer más frecuente en las mujeres en todo el mundo, afectando en mayor medida a las población femenina de regiones de ingresos bajos y medios, pues el 90 % de los casos y muertes se presentan en estas regiones, así lo afirma la organización mundial de la salud [23]. Sin embargo, esta enfermedad se puede prevenir con tratamiento cuando se detectan cambios precancerosos en el cérvix de manera temprana, pero esto solo ocurre si se hacen chequeos periódicos.

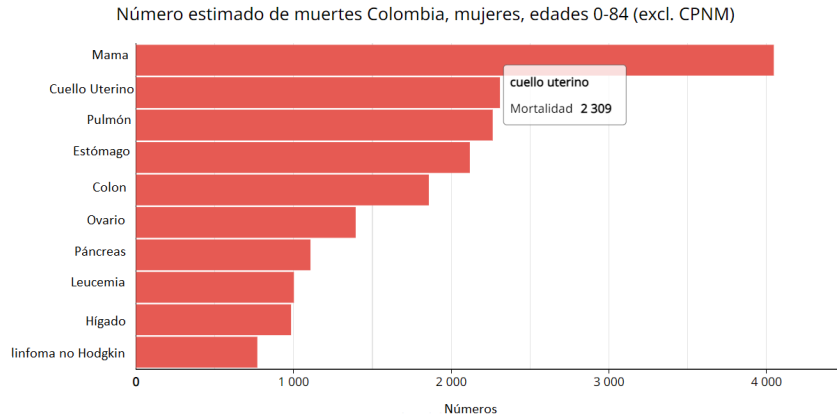
Por esta razón resulta importante realizar pruebas que permitan evaluar el estado del cuello uterino, ya que, mediante su análisis es posible determinar si el paciente tiene cambios celulares en el cuello uterino y si están relacionados con el virus de papiloma humano (VPH), para esto existen diferentes maneras para detectar el cáncer de cuello uterino, a continuación, se mencionan y explican algunos de los métodos:

- Pruebas de los virus del papiloma humano (VPH): consiste en un análisis celular para identificar si hay o no presencia del virus del VPH de alto riesgo. [24]
- Prueba de Papanicolaou (Pap) también conocida como citología: en este examen es posible identificar cambios celulares, así como también otro tipo de afecciones. [24]
- Colposcopia: el examen consiste en la aplicación de una solución de ácido acético del 3 - 5 % con ayuda de unas pinzas y algodón en el cuello uterino, después de unos minutos en las zonas donde hay presencia de células escamosas anormales toman un color blanco, para visualizar esto se emplea una luz y un microscopio de baja densidad. En caso de identificarse un aspecto anormal se toma una biopsia para hacer un análisis a nivel celular[25].

Los métodos de diagnóstico como el VPH o el Pap son más demorados, el tiempo de espera para tener los resultados oscila entre 8 y 15 días, dado que los laboratorios tienen una infraestructura limitada que no permite la evaluación de una alta demanda de muestras de manera simultánea. Adicional a ello problemas en la recolección y almacenamiento de la muestra afectan los resultados,



(a) Indecencia del cáncer en población femenina en Colombia



(b) Cifras de mortalidad en población femenina en Colombia

Figura 3.0: Cifras de incidencia y mortalidad a causa del Cáncer en mujeres Colombianas entre 0-84 años.(fuente: [1])

generando que se requiera tomar el examen de nuevo postergando el momento del diagnóstico[26], finalmente estos estudios tienen el riesgo de generar falsos negativos [27]

Por lo anterior, la implementación de técnicas de aprendizaje de máquina para la segmentación de imágenes de Colposcopia aplicado en la identificación de etapas de cáncer, es una herramienta que busca soportar el diagnóstico médico, reduce el tiempo para el diagnóstico, permite el análisis de grandes flujos de datos condensados en una imagen, también se puede eliminar el ruido proveniente del equipo médico utilizado para la captura de imágenes [28]. Si bien es cierto que el método de detección tradicional ha permitido el diagnóstico de cáncer de cuello uterino en miles de mujeres, se ha podido identificar que estos enfoques pueden presentar falsos positivos debido a un mal proceso en la recolección o almacenamiento de las muestras, incurriendo en diagnósticos erróneos. Ante esta situación, resulta apropiado la implementación de un método de respaldo, que basado en el análisis de imágenes mediante técnicas de aprendizaje de máquina, segmente una imagen de Colposcopia.

La realización de este proyecto es pertinente por el impacto que puede generar no solo en la medicina sino también, en la ingeniería. Además, el desarrollo de este proyecto es viable dado que se

tienen acceso a bases de datos validadas por el instituto nacional de cáncer y la OMS, que contiene la cantidad necesaria de imágenes de colposcopia, también se cuenta con equipo de cómputo para el procesamiento de las mismas. Además, el conocimiento técnico necesario ya ha sido desarrollado y está disponible para que las personas puedan aprenderlo fácilmente; este conocimiento puede ser implementado de diversas maneras en numerosas aplicaciones, como en la navegación autónoma de vehículos, donde se utiliza para identificar objetos, carriles y el espacio por donde se puede conducir [29] [30], también se emplea la segmentación de imágenes diagnósticas, como en las tomografías de personas que han tenido un accidente cerebro vascular (ACV) para así poder identificar las zonas que han sido más afectadas [31], otro campo de aplicación está relacionado con la robótica, ejemplo de ello es un robot que asiste una cirugía laparoscópia y se mueve a partir del seguimiento de los instrumentos médicos los cuales los reconoce por su color [32]. A partir de la información presentada anteriormente es posible afirmar que los diferentes métodos de segmentación pueden ser aplicados casi a cualquier tipo de imagen, y también que uno de los campos donde más se usa es en el área de procesamiento de imágenes médicas, de manera que la segmentación de cuello uterino es pertinente y viable. Como se presentó anteriormente, los métodos de aprendizaje de máquina pueden ser empleados en diferentes áreas y generar mejoras en los procesos, haciéndolos más rápidos, versátiles y precisos. Por tanto, aplicarlo en el proceso de segmentación de imágenes de colposcopia también tendrá un impacto positivo en la salud pública de la mujer, pues este proceso funciona como una herramienta que soporta el proceso de diagnóstico. Se pueden procesar varias imágenes en poco tiempo, haciendo que el tiempo de espera para obtener un diagnóstico sea menor que el de los procesos actuales, además, reduce el número de diagnósticos errados generados por falsos positivos o falsos negativos, los cuales en algunas ocasiones conllevan a practicar biopsias innecesarias.

Con el fin de aportar a este problema de salud pública, el proyecto CITOBOT tiene como objetivo ser implementado en zonas de difícil acceso, ya que, implementar este tipo de tecnología el cual soporta el diagnostico de enfermedades garantiza resultados con alto nivel de precisión y requiere menos recursos en comparación con los métodos tradicionales, proporcionando alternativas para las poblaciones más vulnerables.

Capítulo 4

Objetivos

4.1. Objetivo General

Implementar técnicas de aprendizaje de máquina para la segmentación automática en imágenes de cuello uterino de colposcopia para la identificación de las etapas de cáncer en la tercera fase del proyecto CITOBOT.

4.2. Objetivos Específicos

- Organizar una base de datos de imágenes de colposcopia a partir de ilustraciones almacenadas en otro banco de datos las cuales tengan el cuello uterino identificado, para segmentar manualmente las imágenes y establecer un estándar de referencia (gold standard).
- Implementar algoritmos de aprendizaje de máquina que permitan la segmentación automática del cuello uterino.
- Evaluar el rendimiento de los métodos de aprendizaje de máquina utilizando métricas estándar del estado del arte, como: el coeficiente Dice, IoU y Accuracy, para la selección de los mejores modelos.

Capítulo 5

Marco de Referencia

5.1. Áreas Temáticas

Lista de las áreas temáticas del proyecto:

- Inteligencia computacional y artificial - Sistemas de aprendizaje - Aprendizaje autónomo.
- Inteligencia computacional y artificial - Aprendizaje de máquina - Aprendizaje supervisado.
- Inteligencia computacional y artificial - Aprendizaje de máquina - Aprendizaje no supervisado.
- Inteligencia computacional y artificial - Modelo predictivo.
- Ingeniería en medicina y biología - Computación Biomédica - Procesamiento de imagen Biomédica.
- Ingeniería en medicina y biología - Examen médico - Detección de cáncer.
- Ingeniería en medicina y biología - Examen médico - Ginecología.
- Imágenes - Imágenes Biomédicas.

5.2. Marco Teórico

- **Cuello uterino / Cérvix**

Es la porción más baja del útero y está localizado en la parte superior de la vagina. El canal entre el útero y la vagina tiene una longitud aproximada de 2.5 a 3.5 centímetros de largo [33].

- **VPH**

El virus de papiloma humano es la infección de transmisión sexual más común y es responsable de diversos trastornos, que afectan a hombres y mujeres, con lesiones precancerosas que pueden evolucionar hasta ser cáncer, o simplemente se presentarse protuberancias en las zonas genitales. En las mujeres, la infección persistente por el VPH-16 y el VPH-18, pueden llegar a generar lesiones precancerosas que, de no ser tratadas a tiempo, pueden progresar hasta convertirse en cáncer de cuello uterino, actualmente estas variaciones son el responsable de que el 70 % de los casos de cáncer de cérvix en el mundo [34].

- **Papanicolau**

El desarrollo de la citología exfoliativa, también conocida como la prueba de Papanicolaou, es utilizada para el diagnóstico pertinente de lesiones precursoras, significó un avance en la prevención de cáncer de cuello uterino en mujeres con una vida sexual activa. La prueba consiste en un frotis cervical que permite recoger muestras de células de un área específica en el cuello uterino. Las células se extienden en una lámina portaobjeto, que luego se tiñe y se fija con laca para evitar la alteración de las células. Para determinar la existencia de posibles alteraciones celulares, un patólogo o citólogo entrenado, debe analizar en el microscopio, la apariencia y características de las células del frotis cervical. Esta prueba es considerada, como una prueba muy específica para detección de lesiones de alto grado, o cáncer, pero con un grado moderado de sensibilidad [35].

- **Exocervix**

Es la parte externa del cuello uterino que un médico puede observar durante un examen con espéculo (prueba del Papanicolau), y que está cubierto de células escamosas [36].

- **Endocervix**

Es la abertura del cuello uterino que lleva hasta el útero, y que está cubierto de células glandulares [36].

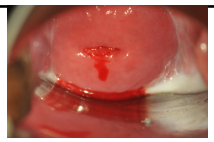
- **Anomalías precancerosas de cuello uterino**

El aspecto del cuello uterino es un indicador importante para determinar si hay o no presencia de cáncer, así como también identificar el nivel de cáncer a partir de la clasificación de las lesiones. Algunas de estas anomalías epiteliales escamosas son:

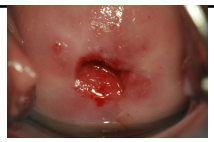
- Epitelio Blancos Atípicos: estas anomalías se presentan como placas blancas que están adheridas a la mucosa y son consideradas precancerosas.
- Puntillado: corresponde a un patrón de puntos. Si los puntos son finos está relacionado con una metaplasia, por el contrario si las lesiones son más grandes y toscas, estas corresponden a una lesión de mayor grado.
- Mosaico: Apariencia de bloques geométricos, donde las lesiones más finas corresponden a bajo grado y lesiones profundas e irregulares, son de mayor grado [37].

- **Imagen colposcopia**

Es un método de estudio del cuello uterino con una herramienta de aumento similar a una lupa, incluso en algunos casos se realiza por medio de cámaras. Esta exploración es de gran importancia, si los resultados obtenidos en la citología son anormales, mediante la colposcopia es posible identificar el tejido conjuntivo, es decir, es posible identificar la estructura, la vascularización, así como también, una posible inflamación en el tejido. Es importante aclarar que con este estudio no se pretende hacer un diagnóstico, pero se emplea para poder apreciar la gravedad de las lesiones. La sensibilidad de la colposcopia para distinguir entre un cuello uterino anormal y normal es buena, pero cuando la zona de unión visible es posible identificar la mayoría de las lesiones, el 73 – 100 % [38].

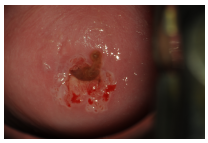


(a) Colposcopia normal

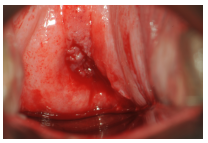


(b) Neoplasia cervical intraepitelial nivel 1

(c) Neoplasia cervical intraepitelial nivel 2



(d) Neoplasia cervical intraepitelial nivel 3



(e) Cáncer

Figura 5.1: Imágenes de colposcopia de diferentes casos de la base de datos del instituto de cáncer de Estados Unidos.

- **Aprendizaje de maquina**

El aprendizaje de máquina es una herramienta que convierte un conjunto de datos en un modelo matemático, este campo surgió a partir de la estadística y la inteligencia artificial, por esta razón, estos algoritmos son capaces de recopilar, así como también, procesar un gran volumen de datos, predecir valores futuros, detectar comportamientos extraños por un fenómeno bajo observación. Dadas sus características, esta es una disciplina que puede ser aplicada a diferentes campos de estudio [39].

- **Aprendizaje supervisado**

El aprendizaje supervisado es una forma de realizar el aprendizaje de máquina y se denomina supervisado porque el proceso se realiza a partir de información que se le ha asignado previamente una etiqueta. Es decir, para el entrenamiento del conjunto de datos, el sistema recibe un conjunto de datos que se refieren a las características cuantitativas y cualitativas, se genera como salida una etiqueta, para ello son empleados algoritmos que incluyen regresión multilínea, logística, k-vecino más cercano, entre otros [40].

- **Aprendizaje no supervisado**

El aprendizaje no supervisado emplea datos sin etiquetas, sin clasificación para el entrenamiento, por esta razón no se puede aplicar directamente una regresión o problema de clasificación, ya que no el sistema no conoce cuales podrían ser los valores de salida, es por esto, que el algoritmo mas empelado en el aprendizaje no supervisado es el de agrupación por clúster, ya que partir de ellos es posible analizar la información [38].

- **Deep Learning**

El aprendizaje profundo permite que los modelos computacionales compuestos por múltiples capas de procesamiento aprendan representaciones de datos con múltiples niveles de abstracción. Estos métodos han mejorado drásticamente el estado del arte en reconocimiento de voz, reconocimiento visual de objetos, detección de objetos y muchos otros dominios, como el descubrimiento de fármacos y la genómica. El aprendizaje profundo descubre una estructura compleja en grandes conjuntos de datos mediante el uso del algoritmo de retropropagación para indicar cómo una máquina debe cambiar sus parámetros internos que se utilizan para calcular la representación en cada capa a partir de la representación en la capa anterior. Las redes convolucionales profundas han producido avances en el procesamiento de imágenes, vídeo, voz y audio, mientras que las redes recurrentes han arrojado luz sobre datos secuenciales como texto y voz [41].

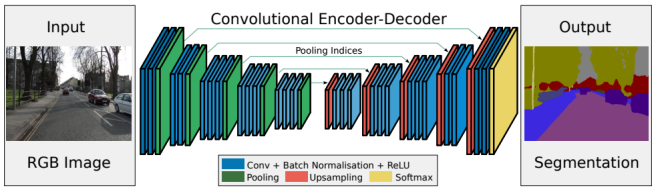


Figura 5.2: Arquitectura segnet

- **Clustering**

El aprendizaje no supervisado emplea datos sin etiquetas, sin clasificación para el entrenamiento, por esta razón no se puede aplicar directamente una regresión o problema de clasificación, ya que no el sistema no conoce cuales podrían ser los valores de salida, es por esto, que el algoritmo mas empelado en el aprendizaje no supervisado es el de agrupación por clúster, ya que partir de ellos es posible analizar la información [40].

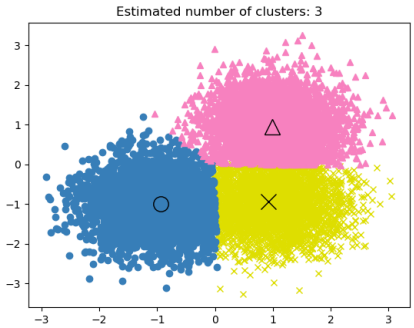


Figura 5.3: Ejemplo gráfico de clustering (Fuente:[2])

- **Algoritmo K-means**

Es un método que tiene como objetivo generar una partición de un conjunto de n observaciones en k grupos. Cada grupo está representado por el promedio de los puntos que lo componen. El representante de cada grupo se denomina centroide. La cantidad de grupos se debe definir previo a la ejecución del algoritmo. Cuando se ha elegido el valor de k, el cual hace referencia a los centroides que se ubican de manera aleatoria, los centroides son reubicados de acuerdo con el valor promedio de los datos cercanos al mismo, dicho proceso se hace reiteradamente, hasta que los resultados de los centroides y sus agrupamientos comiencen a repetirse [42]. A continuación, se presenta la descripción de este algoritmo: "1, seleccionar centroides, especificando el número de clusters deseados. 2, asignar cada punto al centroide más cercano y cada colección de puntos asignados a un centroide es un cluster (Región de Voronoi). 3, actualizar los centroides de cada cluster, basados en los puntos asignados al cluster. 4, repetir el proceso de asignación y actualización hasta que ningún punto cambie de cluster, o lo que es lo mismo, hasta que los centroides permanezcan iguales. 5. Fin."

- **Algoritmo Fuzzy K-means**

Es una extensión del K-means. Mientras K-means encuentra particiones para las que un punto pertenece a un solo cluster, Fuzzy K-means es un método estadísticamente formalizado que encuentra K clusters donde un punto puede pertenecer a más de un cluster con cierto valor de pertenencia (Jain, Murty, & Flynn, 1999). Basa su funcionamiento en la teoría de conjuntos imprecisos presentada por Zadeh [42].

- **Función de distribución de tensiones**

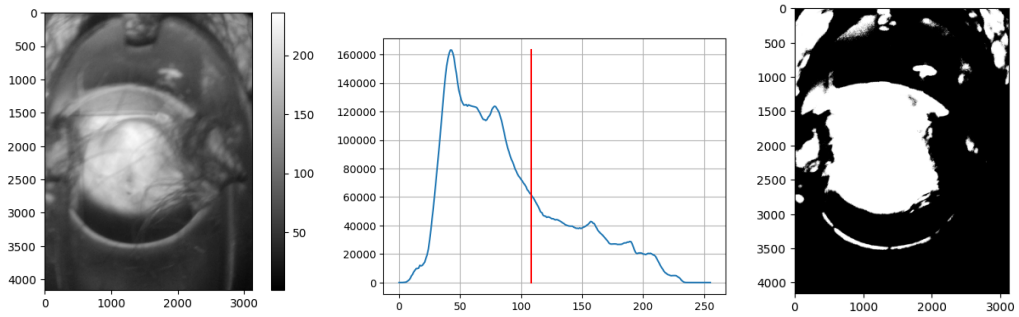
Con base al estudio "Caracterización del color del citoplasma en imágenes de citología cérvico-vaginal" realizado por Santiago Romaní, Pilar Sobrevilla, Eduard Montseny, Enrique Lerma, estudiantes de la Universidad Rovira, Universidad Politécnica de Cataluña y Universidad Autónoma de Barcelona, donde se plantea la implementación de un sistema automático de detección y análisis de células en imágenes de citología cérvico vaginal, interactuando con el cito-patólogo. Se plantea que la detección de las características de los colores relevantes está basada en la detección de los clusters correspondientes a los colores más relevantes del citoplasma. Para detectar dichos clusters se calcula lo que llamamos Función de Distribución de Tensiones o Función de Carpa. El comportamiento de la función de distribución de tensiones o FC se basa en la siguiente idea: "Si se dejara caer una cuerda sobre el histograma, cada bin ejercería una fuerza (o tensión) hacia arriba proporcional a su número de ocurrencias, y la cuerda tendería a caer hacia abajo por la fuerza de la gravedad. Además, cada punto de la cuerda recibiría y ejercería tensiones sobre sus dos puntos vecinos" [43].

- **VGG16**

Una herramienta que favorece el trabajo con redes neuronales profundas es emplear redes preentrenadas con gran número de datos para resolver un problema general, y posteriormente estos datos son adaptados al caso de interés. La razón por la cual se emplea este método está relacionada con, la facilidad de entender e implementar esta arquitectura, también es un método que tiene excelentes resultados, adicional a ello se este modelo tiene relativamente pocas definiciones (13 capas convolucionales y 3 densas), finalmente, existen redes en Keras con modelos y pesos entrenados [44].

- **Otsu**

Este es un modelo estadístico que busca separar los objetos de una imagen basados en la intensidad de los píxeles que componen la imagen. El modelo de umbralización extrae objetos de su fondo basado en la selección de umbral óptimo global que maximiza la varianza entre clases de una imagen en escala de grises, de manera que el píxel cuyo nivel de gris sea menor que el umbral se asignará al fondo, en caso contrario, al segmento de objeto.[45]

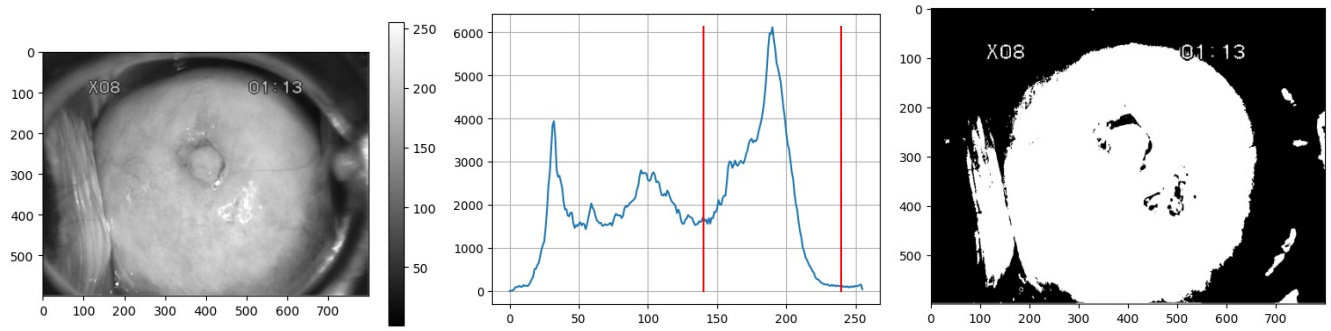


(a) Imagen colposcopia escala de grises (b) Histograma Otsu (c) Mascara Otsu

Figura 5.4: Ejemplo modelo Otsu

- **Rango umbral**

El modelo de segmentación por rango umbral es un método estadístico que, al igual que el modelo Otsu, busca diferenciar el cuello uterino del fondo de una imagen. Para ello, se calcula el tono de gris que tiene mayor frecuencia en la imagen, este tono de gris se utiliza como punto de partida para definir un rango inferior y superior. El rango inferior se obtiene restando cincuenta al tono de gris de mayor frecuencia, y el rango superior se obtiene sumando cincuenta.[46]



(a) Imagen colposcopia escala de grises (b) Histograma rango umbral (c) Mascara rango umbral

Figura 5.4: Ejemplo modelo rango umbral

- **K-means**

El algoritmo k-means es un método de agrupamiento que trabaja con conjuntos de datos que son descritos por valores numéricos y es uno de los algoritmos más populares en minería de datos. Su objetivo es encontrar una partición de ese conjunto, la cual consiste en una serie de grupos que son representados por un centro. Para determinar el número de grupos a crear por el algoritmo, el usuario tiene que definir previamente el valor del parámetro "k", indicando cuantos grupos se formarán en la partición resultante del conjunto de datos. La idea del algoritmo es ir afinando la posición de los centros representativos en el espacio de objetos, es decir, encontrar a los grupos que integren a los individuos más entre ellos.

Para llevar a cabo el proceso del algoritmo k-means se define a priori el número de grupos a formar. Considerando ese valor se inicializan los centros que representarán a cada grupo. Inicialmente, los centroides deben asegurar que al menos un elemento del total de los datos sea asignado a un grupo, para seguir el siguiente proceso iterativo:[47]

1. Calcular la distancia entre los objetos y los centros de cada grupo.

2. Identificar el centro más cercano a cada objeto.

3. Asignar cada objeto al grupo más cercano.

4. Recalcular los centros de los grupos considerando los objetos asignados a cada grupo. Para calcular el nuevo centro del grupo j, se obtiene la media de los elementos que lo conforman.

5. Si los centros calculados en la iteración "n"son iguales a los centros calculados en la iteración n-1, entonces la ejecución del algoritmo termina.

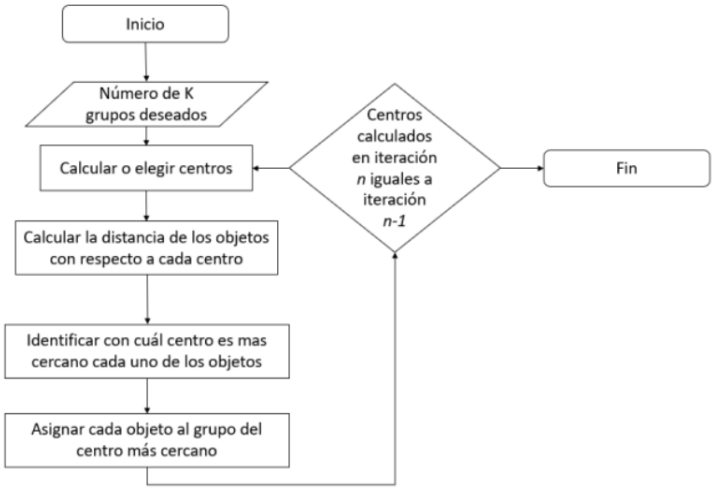


Figura 5.5: Descripción algoritmo k-means

Después de explicar cómo funciona el algoritmo, se presentan como ejemplo una máscara binaria generada con el modelo K-means, esta máscaras es la base para la segmentación de las imágenes de cuello uterino.

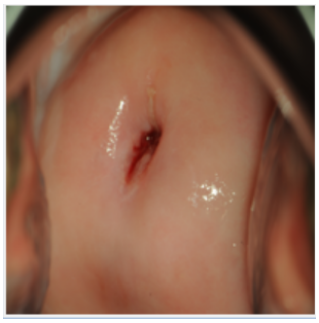


Figura 5.6: Imagen original

Figura 5.7: Mascara generada por k-means

- **Mezclas Gaussianas**

El modelo de mezclas gaussianas (GMM) es un modelo probabilístico que retorna centroides y limites de grupos que describen distribuciones gaussianas. Para la implementación del modelo se empleó la librería sklearn.mixture que hace uso del algoritmo expectation-maximization(EM), el cual cosiste en un proceso iterativo para estimar los parámetros de un modelo de mezclas gaussianas para ajustar un modelo mixto a un conjunto de datos [48]. Si bien las mascaras binarias permiten diferenciar el fondo del elemento, la manera en la que el modelo asigna la región del fondo y del elemento es aleatoria, por este motivo, en algunos casos la predicción del modelo podría ser buena pero la mascara binaria estaba al revés, además este fenómeno no tiene un patrón establecido para tanto no fue posible hacer una corrección. El algoritmo de mezclas gaussianas sigue los siguientes pasos:

- Convertir la imagen en escala de grises
- Definir los parámetros de la distribución gaussiana, es decir el numero de objetos que se quieren diferenciar, tipo de covarianza, así como el método para inicializar los pesos e iterar.
- Se repite el paso anterior hasta que los parámetros de las distribuciones gaussianas se estabilicen.

- **Grabcut**

Grabcut es una técnica de segmentación de imágenes donde se requiere poca intervención del usuario. Inicialmente el usuario debe seleccionar un recuadro alrededor de la zona de interés y luego la segmentación se realiza de manera automática. Posteriormente el usuario puede seleccionar manualmente ciertas áreas de la imagen para mejorar el resultado obtenido. La figura 5.8 ilustra el proceso.[49]

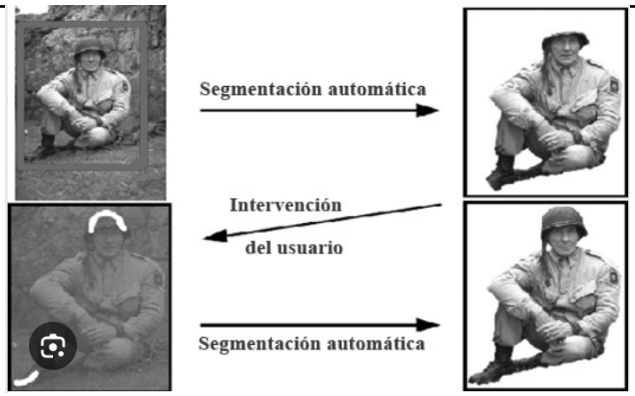


Figura 5.8: Pasos segmentación grabcut

La técnica consiste en crear un grafo de flujo de redes a partir de la imagen a segmentar, donde por cada píxel se genera un nodo que lo representa en el grafo. Luego, cada nodo se conecta con sus 8 vecinos próximos a través de arcos no dirigidos los cuales se denominan N-Link. Adicionalmente, se requieren 2 nodos especiales en el grafo de flujo: fuente y destino. El nodo fuente representa el objeto a segmentar en la imagen (foreground), y el destino representa el fondo de la imagen (background). Cada uno de los nodos del grafo se conecta a través de un arco con la fuente y con el destino, estos arcos se denominan T-Link. El peso de los arcos se calcula empleando una función de energía potencial basado en los modelos mixtos gaussianos (Gaussian Mixture Models - GMM), uno para el foreground y otro para el background. Cada GMM está formado por 5 componentes gaussianos. Con base a lo anterior, se muestran los resultado obtenidos al usar el método grabcut.

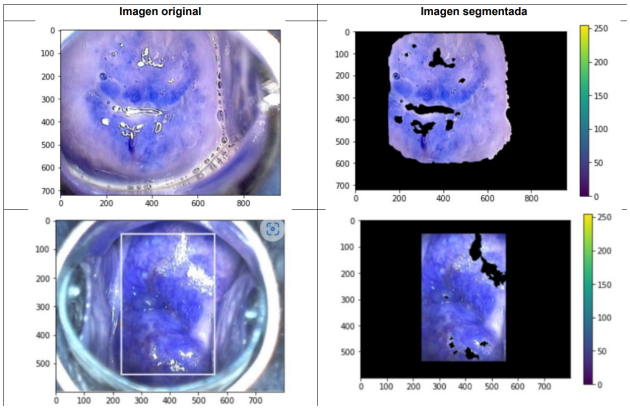


Figura 5.9: Resultados grabcut con y sin recuadro para mejorar resultados

- **Modelo RGB**

El modelo RGB segmenta una imagen de color asignando un segmento a cada píxel. Para ello, el modelo selecciona una región central de la imagen y separa los canales RGB de la sección. Luego,

calcula el valor mínimo y máximo para cada canal, estos valores definen un rango. Finalmente, el modelo verifica que píxeles tienen un tono que está en el rango especificado, a aquellos que si lo están se les asigna un valor de uno y aquellos que no un cero, de esta manera se construye la mascara binaria del método. Mascara que después se multiplica por la imagen original para finalmente obtener la imagen segmentada. [50] [51]

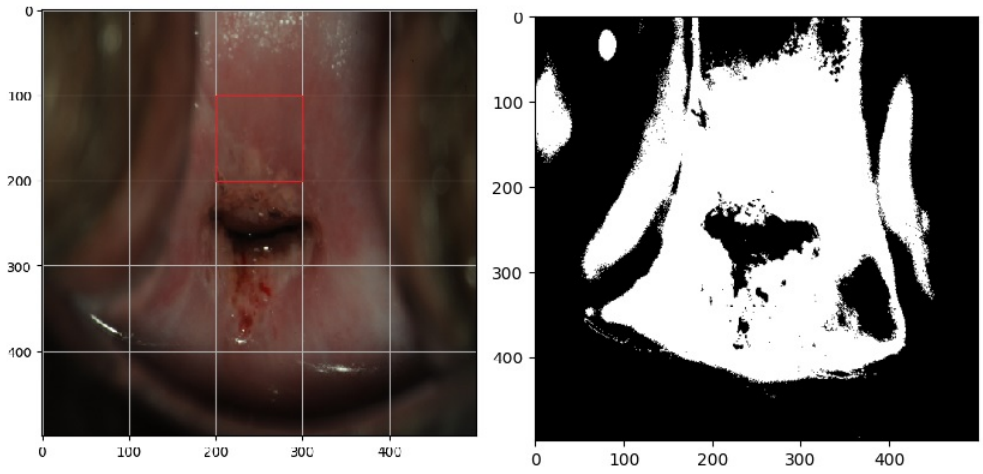


Figura 5.10: Ejemplo modelo RGB

- **Modelo Watershed**

El algoritmo de watershed es una técnica de segmentación basada en morfología matemática, que permite extraer las fronteras de las regiones que hay en una imagen. A la vez, se considera una técnica de segmentación basada en regiones, debido a que clasifica los píxeles debido a su proximidad espacial, el gradiente de sus niveles de gris y la homogeneidad de sus texturas. Por ello se toma como método de detección de contornos y crecimiento de regiones al mismo tiempo.

El algoritmo puede considerar una imagen en escala de grises como la imagen topográfica de un relieve terrestre, en donde a cada píxel se le asocia como valor de "altura" su nivel de gris correspondiente. En este sentido se puede pensar que las intensidades de gris de mayor amplitud corresponden con montañas, mientras que las intensidades de menor valor son valles o ríos. La técnica además incorpora un proceso de inundación de los valles desde los niveles más bajos de altura (mínimos locales) hasta los más altos.

Las zonas de baja intensidad de gris también se conocen como "basins" por donde fluirá el agua e inundara toda la topografía de la imagen. Es decir, el agua circulará por todas las "basins"

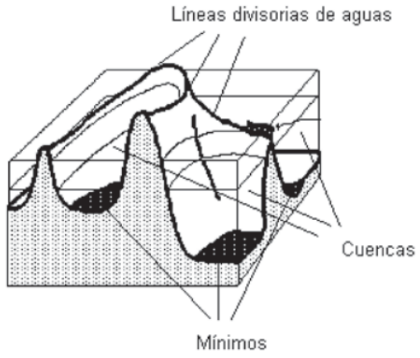


Figura 5.11: Representación didáctica del algoritmo watershed

identificadas. Este proceso continua hasta que las aguas de cuencas contiguas se unan, formando lineas de unión que representan las fronteras de regiones homogéneas y constituyen el resultado de la segmentación. Lo explicado hasta el momento se puede apreciar en la figura 7.9.[52]

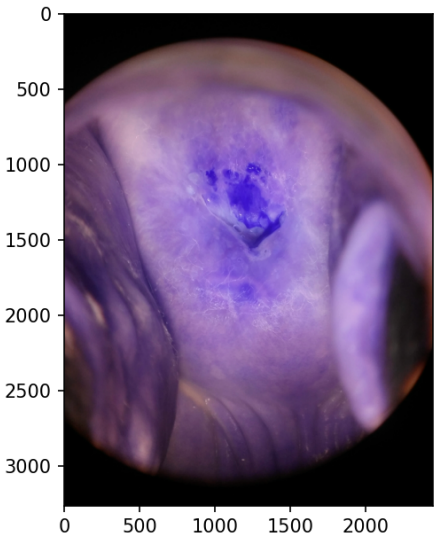


Figura 5.12: Imagen original

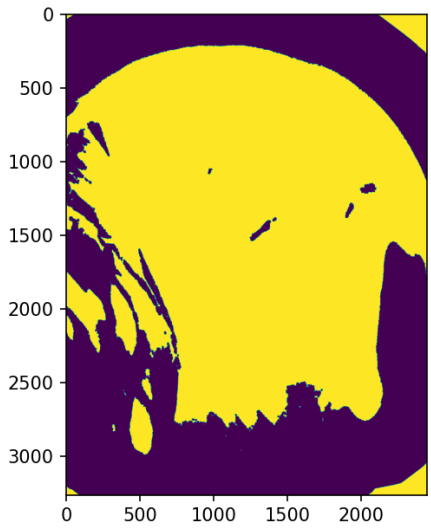


Figura 5.13: Mascara generada por watershed

- **Segnet**

Segnet es una arquitectura para realizar la segmentación de imágenes, esta conformada por una red de codificación y una red de decodificación, seguida por una capa final de clasificación de píxeles. La red de codificación consta de 13 capas convolucionales que corresponden a las 13 primeras capas convolucionales en la red VGG16, diseñada para la clasificación de objetos. Por lo tanto, podemos inicializar el proceso de entrenamiento a partir de pesos previamente entrenados para la clasificación de grandes volúmenes de datos.También se puede descartar las capas totalmente conectadas en

favor de retener mapas de características de alta resolución en las salidas más profundas de la red de codificación. Esto también reduce el número de parámetros en la red de codificación de Segnet. Cada capa de codificación tiene una capa de decodificación correspondiente, por lo tanto, la red de decodificación consta de 13 capas. La salida final del decodificador se alimenta a un clasificador softmax multiclase para producir probabilidades de clase para cada píxel de manera independiente.

Cada codificador en la red de codificación realiza una convolución con un banco de filtros para producir un conjunto de mapas de características. Luego se normalizan por lotes. A continuación, se aplica una no linealidad ReLU (Rectified Linear Unit) por elementos (max(0, x)). Después de eso, se realiza un max-pooling con una ventana de 2×2 y un paso de 2 (ventana no superpuesta), y la salida resultante se submuestrea en un factor de 2.

El decodificador apropiado en la red de decodificación aumenta la resolución de su mapa de características de entrada utilizando los índices memorizados de max-pooling de los mapas de características del codificador correspondientes. Este paso produce mapas de características dispersos. Luego, estos mapas de características se convolucionan con un banco de filtros de decodificador entrenable para producir mapas de características densos. Luego, se aplica un paso de normalización por lotes a cada uno de estos mapas.

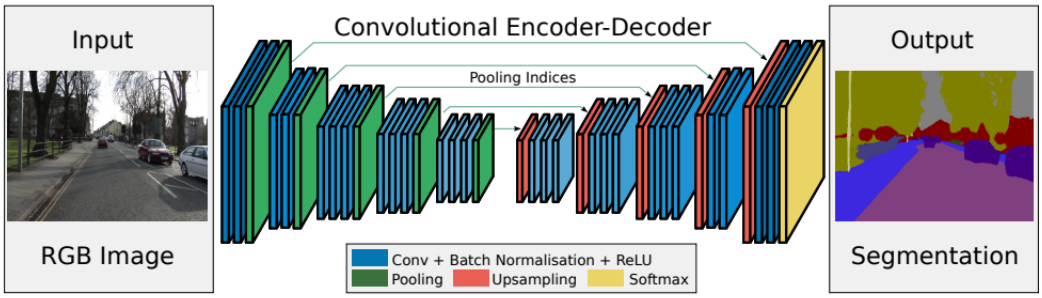
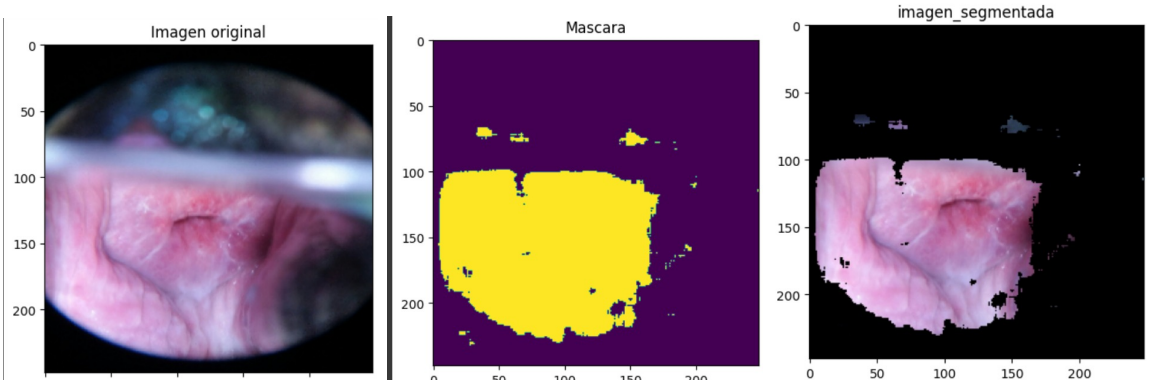


Figura 5.14: Una ilustración de la arquitectura SegNet

No hay capas totalmente conectadas y, por lo tanto, es solo convolucional. Un decodificador aumenta la resolución de su entrada utilizando los índices de pooling transferidos de su codificador para producir un mapa de características dispersas. Luego, realiza una convolución con un banco de filtros entrenables para densificar el mapa de características. Los mapas de características finales de salida del decodificador se alimentan a un clasificador de soft-max para la clasificación píxel a píxel. Una vez explicada la arquitectura Segnet, se mostrarán los resultados obtenidos al ser entrenada con imágenes de cuello uterino.



(a) Imagen original (b) Mascara SegNet (c) Imagen segmentada

Figura 5.15: Resultado arquitectura SegNet

- **U-net**

La arquitectura U-net es un modelo de red neuronal usado en tareas de visión artificial, más específicamente en problemas de segmentación semántica de imágenes biomédicas, de manera que se le asigna una etiqueta una clase a cada píxel, permitiendo un análisis detallado de la composición de la imagen. Las redes neuronales se componen de codificador y un decodificador, para el caso de U-net, el codificador son una serie de capas convolucionales o Down Convolutions, que reducen el tamaño de las imágenes, pero se incrementan el número de canales, de manera que se extraen el mayor número de características, identificando la información más relevante, en cuanto al decodificador se hacen una serie de capas de deconvolución o Up Convolutions, siendo esta una estructura simétrica al codificador, el decodificador ayuda a producir una salida de las mismas dimensiones de la entrada, pero en la medida que se ejecutan las capas de Up Covolution se fusiona la imagen con las características extraídas en el codificador. Algo que distingue a este método es que se hace un proceso de Copy and Drop, en el cual se concatenan las salidas de cada capa de la Down Convolution con cada capa del decodificador, de esta manera se puede conservar detalles finos [53] [54]. A continuación, se muestra la estructura de la arquitectura U-net, en donde se plasma cada uno de los procesos del codificador y decodificador explicados anteriormente.

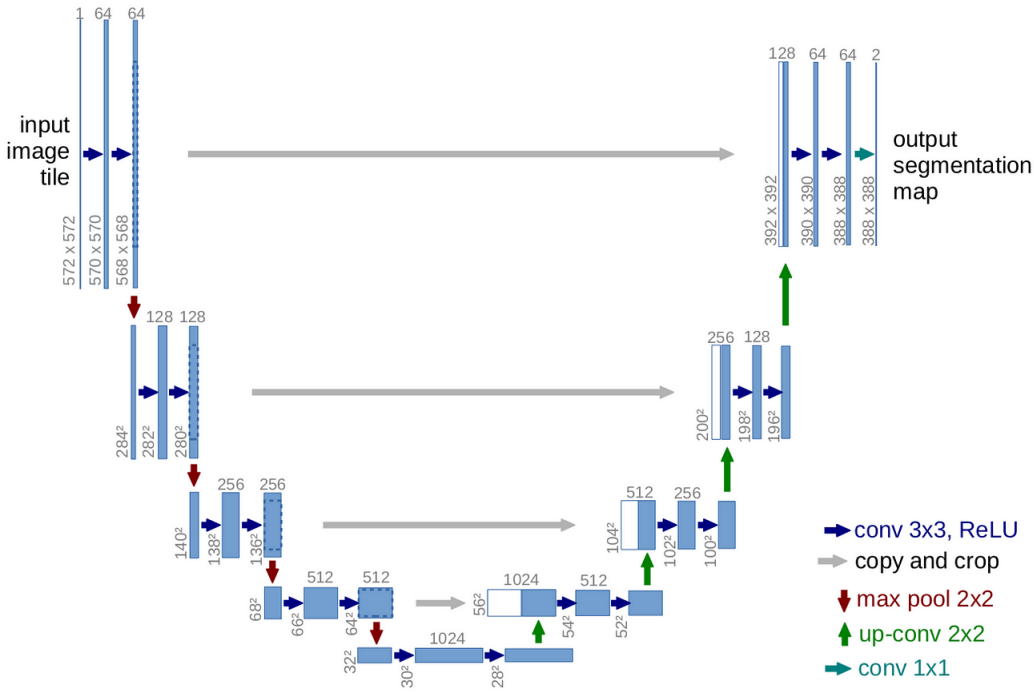


Figura 5.16: Arquitectura Unet (fuente: [3])

- **FPN**

El modelo FPN o Feature Pyramid Network es una arquitectura empleada en aplicaciones de visión por computadora, como la detección de objetos o segmentación de imágenes, esto es posible mediante la representación de características multidimensionales para cualquier tamaño de imagen. Esta extracción de características se hace mediante una red neuronal convolucional en forma de pirámide a partir de una imagen. La estructura FPN está basada en una red pre-entrenada y en la medida que los datos de la imagen pasa por la red se construye una pirámide de características mediante fusiones ascendentes, de esta manera se aumenta la resolución de las características mediante la interpolación y la suma de mapas característicos de resoluciones inferiores, y descendentes, donde se concatenan las mapas de resoluciones superiores con mapas de resolución inferior, de esta manera es posible mejorar la precisión y robustez del modelo de segmentación semántica.[4]

La construcción del mapa de características empieza con la extracción de características con la red convolucional "resnet101", siendo estas la base de la pirámide dado que tiene la resolución más alta y contiene características de nivel inferior para detalles finos de las imágenes, después de obtener la base se realiza una fusión descendente para crear mapas característicos de mayo resolución, en cada nivel se realiza un upsampling o convolución transpuesta para aumentar el tamaño hasta que coincida con la resolución del mapa característico superior, luego se concatenan los mapas superiores con la capa base. También se hacen fusiones ascendentes para obtener mapas característicos con mayor

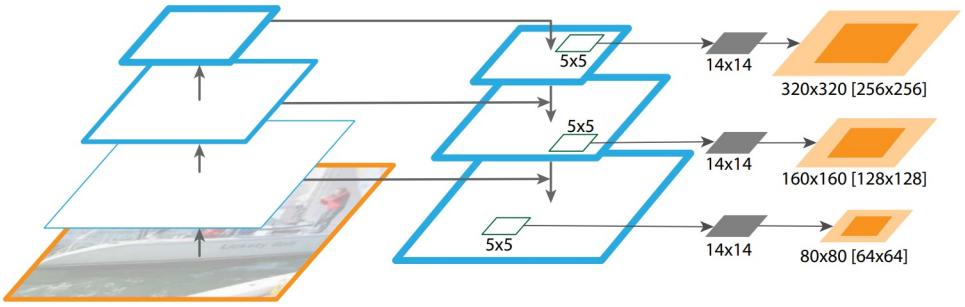


Figura 5.17: Arquitectura FPN (fuente: [4])

resolución, en donde en cada nivel se toma el mapa característico de resolución superior y se aplica una operación de pooling para reducir su tamaño, luego se agrega este mapa característico. Después de los procesos de fusiones se obtienen mapas característicos que permiten obtener información detallada, se realiza una operación de convolución para unificar el mapa de características general [55]. Después de implementar y entrenar el algoritmo los resultados obtenidos fueron los siguientes.

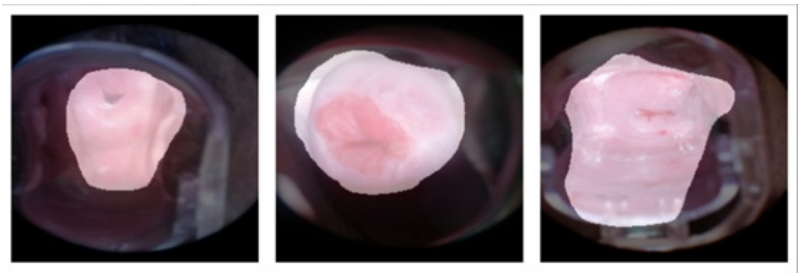


Figura 5.18: Ejemplo resultados parciales modelo FPN

- **IoU o Intersection over Union**

Es una métrica de evaluación utilizada par medir la precisión de un detector de objetos en un conjunto de datos en particular. Cualquier algoritmo que proporcione cuadros delimitadores (Bounding box) o mascaras puede emplear este método. Para usarlo, se requieren dos aspectos:

1. Bouding box o mascaras generados por el sistema.
2. Bouding box o mascaras etiquetados manualmente.

Con esta información se debe realizar la razón entre las áreas solapadas (Area of overlap) y área de unión (Areao of Union) obteniendo resultados entre cero y uno, se espera que los resultados superen los valores de 0.8. Es extremadamente improbable que las coordenadas (x,y) de nuestro

bounding box previsto coincida exactamente con el bounding box generado manualmente, debido a los diferentes parámetros de nuestro modelo (escala de pirámide de imagen, tamaño de ventana deslizante, método de extracción de características, etc.), una coincidencia completa y total entre los cuadros delimitadores predichos y reales es simplemente poco realista [56]. A continuación se presenta como es el cálculo de IoU en la ecuación 5.1.

$$IoU = \frac{Prediccin \cap GoldStandard}{Prediccin \cup GoldStandard} \tag{5.1}$$

- **Coeficiente DICE**

El coeficiente DICE se utiliza para evaluar la precisión de los algoritmos de segmentación en imágenes mediante la comparación de la segmentación generada por el algoritmo con la segmentación de referencia (etiquetada manualmente). Este coeficiente mide la similitud o superposición entre dos conjuntos, especialmente en el contexto de la teoría de conjuntos y análisis de datos.

El resultado del coeficiente DICE proporciona una medida de la superposición entre la segmentación generada por el algoritmo y la segmentación de referencia. Un valor cercano a 1 indica una alta similitud y precisión en la segmentación del algoritmo, mientras que un valor más cercano a 0 indica una menor precisión [57]. La ecuación 5.2 describe cómo se calcula este parámetro.

$$Dice = \frac{2 * (Prediccin \cap GoldStandard)}{Prediccin + GoldStandard} \tag{5.2}$$

- **Accuracy**

Es una medida verdaderamente intuitiva, puesto que se limita a calcular la relación entre las predicciones correctas de un modelo, versus el total de predicciones que llevó a cabo. Para realizar el calculo de esta métrica se debe seguir la siguiente formula:

$$accuracy = \frac{\sum_{i=1}^{N} (if\ prediction_i\ =\ actual_i\ then\ 1\ else\ 0)}{N}$$

Figura 5.19: Formula accuracy

No es más que la sumatoria de 1, cuando la predicción es igual a la etiqueta real, o 0 cuando no coinciden, dividida por N, donde N es el total de elementos en nuestro conjunto de datos.

No obstante, el Accuracy sólo destaca los aciertos, pero no repara en las fallas.[58]

- **Loss**

Mide el desempeño de un modelo de clasificación en el que la entrada de la predicción es un valor de probabilidad entre 0 y 1. La pérdida logarítmica aumenta a medida que la probabilidad predicha se aleja de la etiqueta real. El objetivo de cualquier modelo de aprendizaje automático es minimizar este valor. Por lo tanto, una pérdida logarítmica menor es mejor, con un modelo perfecto teniendo una pérdida logarítmica de 0.[59]

5.3. Trabajos Relacionados

- **Segmentación de la región acetoblanca en colposcopías del cérvix / Aceto - white region segmentation in cervix colposcopies**

En el presente articulo se proponen estrategias para procesar las imágenes colposcopicas, detectar de forma automática el área cervical y segmentar la región acetoblanca combinando algoritmos de agrupamiento y morfología matemática en el espacio de colores Lab. Siendo está, una zona del cérvix que adquiere o acentúa una coloración blanca tras aplicarse una solución de ácido acético. Inicialmente se encuentran los algoritmos de agrupamiento empleados, tal como k-means, Mini-batch k-means, modelos de mezclas Gaussianas y modelos de mezclas Gaussianas por Inferencia Bayesiana. Es importante aclara que el Mini-batch K-means es una variante del algoritmo original ya que emplea subconjuntos aleatorios del conjunto de datos de entrada para calcular los centroides. Dichos conjuntos se denominan mini-batch y tienen la propiedad de poseer un tamaño fijo que se introduce como parámetro. Aunque su convergencia es mucho más rápido que k-means, la calidad de los resultados, medida por la suma de las distancias de cada punto a su centroide más cercano, no es tan efectivo.Posteriormente se realiza el preprocesamiento, dado que este tipo de imágenes poseen rasgos peculiares, tal como regiones de brillo o especulares dado las características húmedas propias de los tejidos del cuello uterino y las diferentes soluciones aplicadas. Además de poseer ruido, y mala distribución de la luz, objetos o tejidos adicionales en la fotografía, por ejemplo, otras partes de la vagina. A continuación, se elimina la zona brillante producto del reflejo de la luz de la cámara del colposcopio, es decir, región especular. Es necesario eliminarlo dado que afecta los algoritmos de segmentación. Así mismo, se hace la corrección de iluminación, la cual consiste en aclarar los píxeles más oscuros y dejar los más claros con su propia iluminación. Para finalizar el preprocesamiento de la imagen se elimina el ruido.Un paso importante para la segmentación de la región acetoblanca, consiste en detectar el ROI, ya que las fotos pueden contener información irrelevante, por ejemplo el espéculo. En el artículo, se entiende por ROI, la región cervical, dado que solo en esta zona se puede encontrar algún epitelio acetoblanco. Finalmente se emplean los algoritmos de agrupación para realizar la segmentación del área de interés [5].

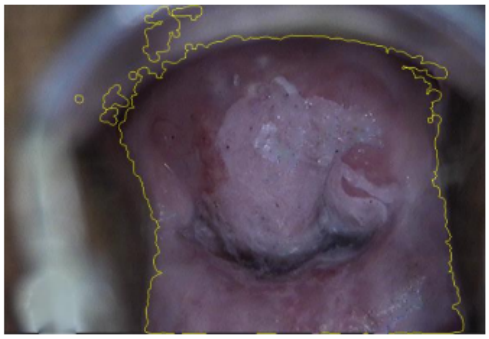


Figura 5.20: Detección del ROI.(Fuente: [5])

- **Segmentación de imágenes médicas digitales mediante técnicas de clustering**

En este trabajo se emplea técnicas de Clustering en la segmentación de imágenes médicas digitales para ser utilizadas en la reconstrucción de modelos anatómicos 3D a partir del estándar Digital Images and Comunication in Medicine (DICOM) con el fin de mejorar los resultados reportados en las fuentes bibliográficas. Por tal motivo se emplearon dos métodos existentes de clustering, el primero de ellos es el k-means, el cual es el método más empleado debido a su simplicidad, además que el parámetro k relacionado con los centroides se debe especificar. El segundo de ellos es el Fuzzy K-means "El análisis minucioso de los algoritmos de Clustering más significativos y la experimentación fueron los precedentes para la construcción de un módulo de segmentación de imágenes médicas digitales que dio cumplimiento a los objetivos planteados, para ello se determinaron los siguientes aspectos: 1. Las técnicas de Clustering presentan ventajas con respecto a las otras técnicas de segmentación de imágenes. 2. La diferencia de color es el indicador de disimilitud más adecuado para la ejecución de los algoritmos tratados. 3. La heurística de análisis de frecuencia de color para la inicialización de los centroides de los algoritmos vistos es la que proporciona los resultados más acertados. 4.Para el perfeccionamiento ulterior de las técnicas de segmentación de imágenes médicas mediante técnicas de clustering se ha detectado las siguientes vías: Búsqueda de procedimientos más avanzados de inicialización de los centroides. Utilización de algoritmos que permitan alcanzar el óptimo global o al menos aproximarse al mismo, tanto para el método K-means como para el Fuzzy K-means. 5. El módulo de segmentación de imágenes médicas digitales presenta un diseño extensible y refinado que no lo mantiene atado a alguna biblioteca, fue desarrollado sobre estándares y brinda la posibilidad de ser adaptado fácilmente a diferentes sistemas. Sobre los sistemas operativos Ubuntu 9.10 y Windows XP se realizaron las pruebas que alegan alta fiabilidad y robustez." [60].

- **Detección de cáncer cérvico-uterino mediante red neuronal función de base radial**

En este articulo no se busca sustituir al cito patólogo, persona encargada en dar a conocer los diagnósticos de las muestras citológicas, sino en brindar una herramienta que de un segundo diagnóstico para corroborar los resultados y evitar errores. Empleando procesamiento digital de imágenes es posible realizar mejoras a las fotografías, dado que se pueden resaltar mejor los detalles a través del aumento o disminución de la luminosidad y/o cromaticidad, posteriormente se emplearan redes neuronales artificiales la cuales imitan el funcionamiento neuronal biológico en la resolución de problemas durante lapsos de tiempo cortos, dichos avances tecnológicos se emplean en diversas áreas, tal como estadística, industria, seguridad, entre otras. El primer paso consistió en la adquisición de las imágenes, las cuales fueron obtenidas a través del departamento de anatomía patológica del Hospital Regional Metropolitano 1° Octubre mediante un microscopio Leica DME con cámara digital integrada Leica. Posteriormente se realizó la segmentación por medio de la separación de patrones de interés del resto de la imagen. Para ello, los píxeles de un patrón determinado son asignados a otra imagen, en la cual ocupan la misma posición relativa, resultando tantas imágenes como patrones se desee separar. Esta segmentación se llevo a cabo usando el mouse para seleccionar la región de interés de la imagen previamente guardada, esta región se asigna a una nueva variable la cual contiene únicamente el objeto de utilidad. Seguidamente se lleva a cabo la binarización de la imagen, proceso en el cual se reduce información de la misma. El uso de imágenes binarias

permite un proceso rápido para la obtención de datos geométricos en los que no intervienen el color, además de que permite un gasto menor en cuanto a memoria del PC. El siguiente paso consiste en la morfología matemática, ya que la imagen esta descrita por un conjunto finito de puntos, algunos de ellos serán alteradas para rellenar huecos, eliminar picos o redondear bordes. Las técnicas empleadas son dilatación y erosión. La primera se enfoca en establecer el control del proceso de crecimiento de una conjunto A respecto al conjunto B, el cual tiene forma y estructura simple. Este proceso se describe a través de la suma Minkowski de A y B. La segunda consiste en el decrecimiento controlado de un conjunto A con elemento estructurante B. El tamaño y forma final del conjunto erosionado dependerá del tamaño y forma del elemento estructurante. Posteriormente se emplea la escala de grises a las imágenes para disminuir el volumen de información, ya que se reduce el análisis tres capas (RGB). Luego se emplea el histograma ya que muestra el brillo de una imagen digital a través de una distribución gráfica de los niveles de gris de los píxeles. Del histograma se observa si una imagen es clara u oscura y si tiene alto o bajo contraste. Finalmente, el funcionamiento de la red es dar a conocer el resultado acerca del grado de displacía presente en una imagen digital a través de un programa computacional [61].

- **Red neuronal artificial para detectar lesiones precancerosas en el cuello uterino**

Cada año se producen alrededor de 466000 nuevos casos de cáncer de cuello uterino en el mundo y cerca del 80 % de los casos corresponden a los países en desarrollo, trayendo como consecuencia un alto costo social, ya que se estima que por cada mujer que fallece quedan entre 5 a 7 niños huérfanos, lo cual representa un aproximado de 17000 niños huérfanos por año. Basado en el contexto anterior, llena de preocupación la necesidad de herramientas informáticas que permitan la gestión de los datos almacenados (fotos digitales del cuello uterino) que conllevan a la toma de decisiones para realizar los diagnósticos o estrategias de despistajes. Cabe mencionar, que el método utilizando en la actualidad para la detección de anormalidades en el cuello uterino es la citología, la cual tiene un alto porcentaje de error ya que arroja resultados entre los cuales el 30 % son falsos positivos y el 20 % son falsos negativos . Llevando a realizar biopsias innecesarias. Una alternativa de solución a esta problemática, es la implementación de redes neuronales artificiales (RNA), dado que han sido usadas para aplicaciones médicas y han generado resultados con porcentajes de acierto alto, tal es el caso del desarrollo de una RNA capaz de realizar el reconocimiento automático de algunos tipos de células, con un porcentaje de acierto en un 87,4 %;también se han utilizado en citologías de orina, obteniendo una exactitud del 90.63 %. Lo anterior llevo al desarrollo de una aplicación basada en una red neuronal artificial que permita la realización del diagnostico de lesiones precancerosas en el cuello del útero. Para probar los modelos, se emplearon imágenes digitales con sus respectivos diagnósticos y toda la información necesaria para llevar a cabo la investigación. Los datos mencionados anteriormente fueron suministrados por el centro piloto Corposalud Aragua. Dos modulo a destacar del sistema implementado en este articulo, se relaciona con la adquisición de la imagen y la segmentación. En el primero se decidió limitar la información a fotografías de cuellos uterinos sin exéresis, es decir, cuellos uterinos que no hallan sufrido intervenciones quirúrgicas previas. El segundo, habla acerca del proceso de segmentación, el cual consiste en eliminar la mayor cantidad de información irrelevante posible, producto de la captura de la imagen a través del videocolposcopio, identificando el área de interés (Píxeles donde halla cuello uterino). Lo anterior conllevo a

crear un protocolo llamado "Protocolo de captación"para adquirir los datos de manera uniforme, posteriormente se implemento la segmentación estableciendo una relación de supervivencia, donde todos los píxeles que se encuentren dentro de una circunferencia concéntrica (El centro es el cuello uterino con base al protocolo de captación) permanecerán, por el contrario, todos aquellos píxeles que se encuentren fuera del mismo serán eliminados [6].

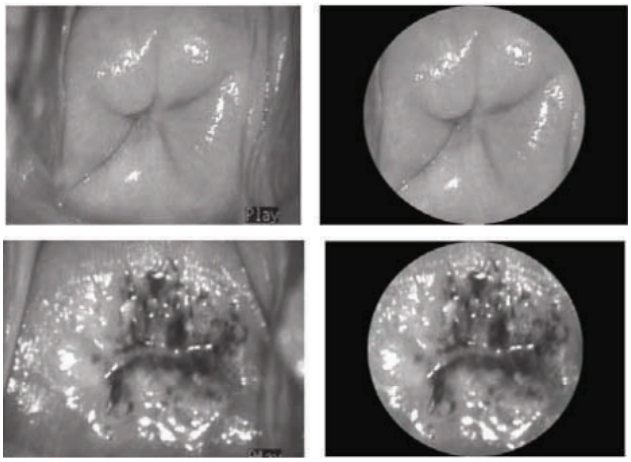


Figura 5.21: Segmentación de cuello uterino empleando la relación de supervivencia y el protocolo de captación (Fuente: [6])

- **Clasificación de cáncer cervical usando redes neuronales convolucionales, transferencia de aprendizaje y aumento de datos**

En el presente trabajo se busca demostrar que las redes neuronales convolucionales (CNNs) y la transferencia de aprendizaje son una buena estrategia para clasificar el grado de las lesiones presentes en el cuello uterino. Después de estudiar trabajos relacionado se encontró que la precisión para detectar el tipo de lesión no es buena, por tanto, se optó por separar la base de datos en cáncer cervical leve y agresivo. El algoritmo empleado cuenta con CNNs previamente entrenadas con ImageNet lo que permite tener un conjunto amplio de características lo que hace que sea mas sensible ante la detección de patrones, es importante destacar que la base de datos fue pre-procesada utilizando segmentación, este proceso se empleó para seleccionar el grupo de datos óptimo y que en la imagen se enfoque la lesión, además se aumentaron los datos ya que no se contaba con la cantidad de imágenes sin ruido y de calidad, se seleccionaron las mejores 1000 de cada tipo de cáncer cervical para que hubiera un equilibrio en los datos y etiquetado con el tipo de cáncer. Para la transferencia de aprendizaje se emplearon capas convolucionales para reentrenar la red y se adapte a la nueva clasificación (leve o agresivo). Con esto la precisión mejoró hasta 97,35 % sobre los datos de validación. [7]

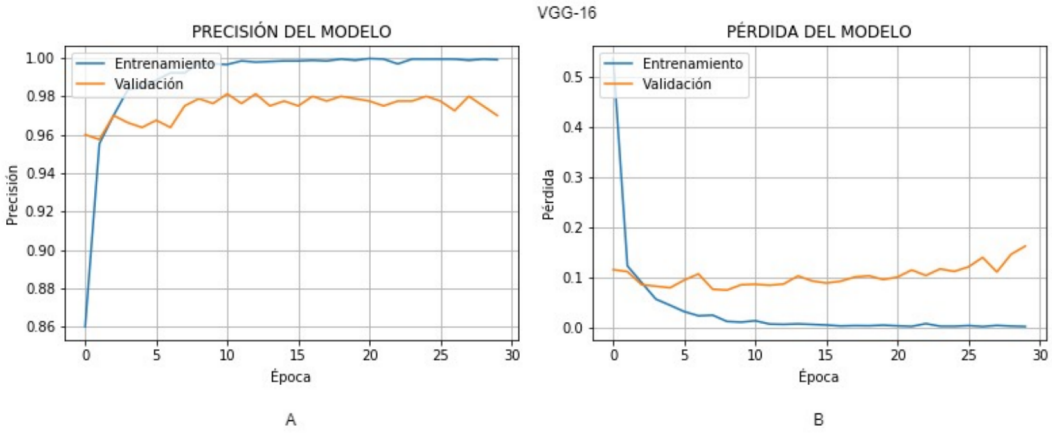


Figura 5.22: Precisión alcanzada y perdida del modelo sobre datos de entrenamiento y validación usando VGG16 (Fuente: [7])

- **A Deep Clustering Method For Analyzing Uterine Cervix Images Across Imaging Devices**

El estudio realizado por la doctora Zhinyun, consiste en que las técnicas de evaluación visual automatizada, para poder definir si una persona tiene precáncer basados en imágenes de cuello uterino acetoblanqueado, este proceso se refiere a la aplicación de ácido acético de una concentración entre el 3 - 5 %, siendo esta una estrategia que garantiza mejoras en el diagnostico ya que mejora el desempeño humano en cuanto a análisis de imágenes se refiere, adicional a ello es un proceso de bajo costo. Sin embargo, existen gran número de consideraciones a tener en cuenta si se desea hacer una implementación de una técnica automatizada de evaluación, una de estas consideraciones se encuentra relacionada con las características de imágenes capturadas con diferentes dispositivos, es por esto que los investigadores proponen un nuevo enfoque de agrupamiento basado en aprendizaje profundo para investigar si las imágenes tomadas por tres dispositivos diferentes, un teléfono inteligente, un dispositivo portátil basado en un teléfono inteligente personalizado para imágenes cervicales y un colposcopio. Durante el desarrollo de la investigación fue posible identificar una fuerte diferencia en la apariencia de las imágenes lo cual podría ser un factor de confusión significativo en el entrenamiento y la generalización del rendimiento de dicho proceso. Frente a esta situación, se plantea un método que consta de cuatro componentes: detección de la región del cuello uterino, extracción de características, codificación de las características y agrupación. Finalmente, el método propuesto logra una precisión de agrupamiento del 97 %, superando así significativamente a varios métodos representativos de agrupamiento profundo [8].

- **Implementación de métodos para reconocimiento de imágenes para el diagnóstico de cáncer cervicouterino**

Este articulo presenta cifras relacionadas a la cantidad de mujeres que adquieren esta enfermedad

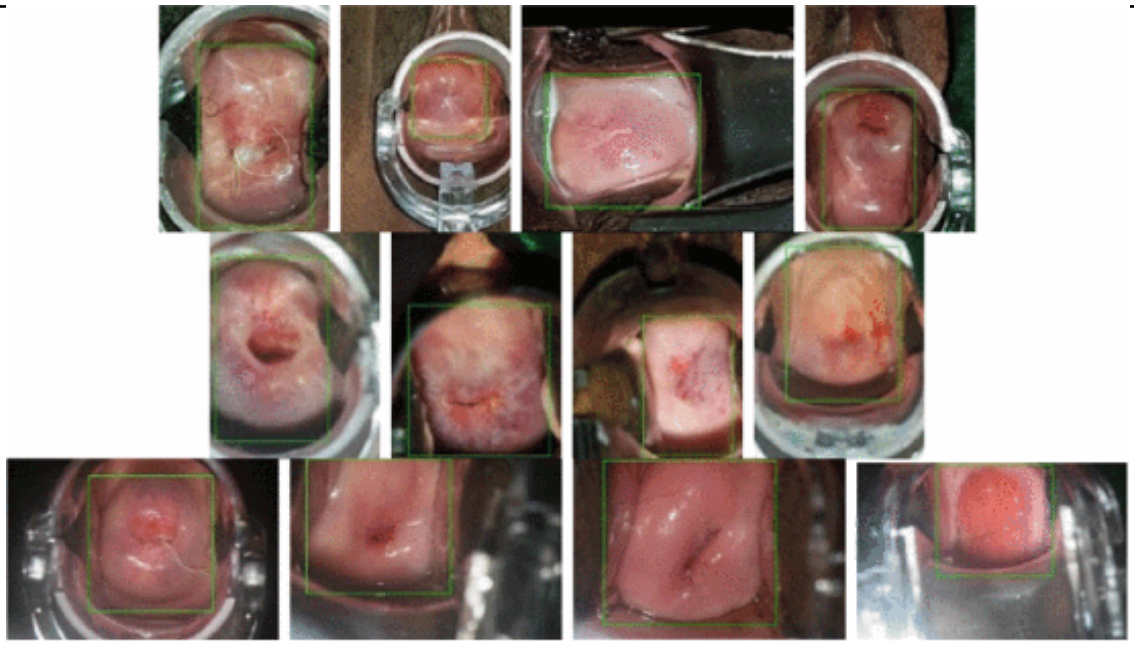


Figura 5.23: Detecc ión de la región del cérvix con tres diferentes dispositivos (1 fila: celular; 2 fila: EVA; 3 fila: Colposcopio. (Fuente: [8])

en el mundo e indica que los países en vía de desarrollo son los principales afectados con 85 % de los casos totales.

Por ende, una de las alternativas más prometedoras para generar diagnósticos a tiempo es emplear el método visual, el cual consiste en limpiar el cérvix con una solución de vinagre o yodo y someterlo a un simple examen visual sin instrumentos para detectar el tejido anormal. Dicha inspección presenta ciertas ventajas en contextos de bajos, por ejemplo, necesita poca infraestructura, puede realizarla personal no médico (Deben recibir capacitación previa y supervisión durante el procedimiento) por ende, los resultados son inmediatos y en teoría se podría ofrecer tratamiento y remitir al paciente en la misma visita al consultorio. Otra alternativa es detectar la presencia del VPH en el cérvix, sin embargo esta opción no se ha implementado a gran escala. a que implica elevados costos [9].

La investigación se centra la implementación de un sistema web desarrollado bajo el lenguaje de programación PHP, mismo que usa el SGB (Sistema Gestor de Base de Datos) MySQL, mediante el uso de software MATLAB se hace el reconocimiento de imágenes, utilizando técnicas como la segmentación de color, algoritmos de máximos y mínimos y distancia euclidiana, realizando el etiquetado de objetos en áreas de interés, todo lo anterior modelado en un sistema web administrado de forma local. Finalmente, el algoritmo le proporcional al paciente un prediagnóstico, el cual puede ser interpretado por el usuario, sin necesidad de consultar un especialista. En conclusión, el prediagnóstico busca hacer una contribución a al detección oportuna de esta enfermedad, agilizando el

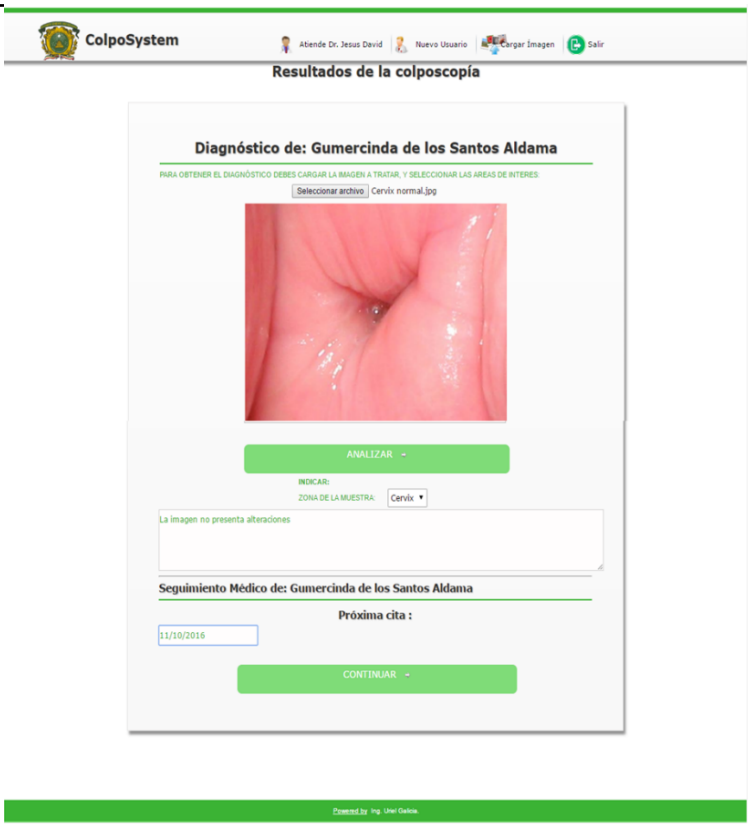


Figura 5.24: Interfaz donde hay que capturar la zona de análisis, un diagnostico hecho por el médico y la cita próxima de la paciente(Fuente: [9])

proceso para entregar los resultados ya que se apoya al personal médico en la toma de decisiones para el diagnóstico final [9].

Figura 5.25: Interfaz final donde se guarda toda la información y genera reporte PDF(Fuente: [9])

Capítulo 6

Materiales y Métodos

6.1. Tipo de Estudio

El presente trabajo de grado es de carácter exploratorio, puesto que se desea demostrar que la segmentación de imágenes de colposcopia mejora los índices de identificación de cáncer cuello uterino. Para el cumplir con ello, es necesario realizar una búsqueda de la información para identificar la manera como otras personas que han desarrollado proyectos relacionados han actuado, posterior a ello se buscan métodos de aprendizaje de máquina para entenderlos y para su posterior implementación. Después, se organiza la base de datos que contienen las imágenes de colposcopia y se construyen mascaras binarias para construir un marco de referencia de la segmentación.

Seguidamente, se implementaron 7 métodos de aprendizaje no supervisado, como son: Otsu, Rango Umbral, RGB, Grabcut, K-means, Wathersed y Mezclas Gaussianas, y 3 modelos de aprendizaje supervisado como son Unet, FPN y SegNet. Una vez implementados los modelos, se evalúan los resultados de los modelos en relación al gold standar para determinar el rendimiento de los métodos, para ello se hizo uso del coeficiente Dice, IoU y Acurracy, siendo este ultimo especifico para los modelos supervisados. Finalmente son seleccionados para el aplicativo final.

6.2. Actividades

De acuerdo a los objetivos planteados, se proponen las siguientes actividades y materiales, para el cumplimiento del objetivo general:

OE1: Organizar una base de datos con imágenes de colposcopia que tengan el cuello uterino identificado.

- **Actividades:**
  T.1.1 Gestionar base de datos validadas.
  T.1.2 Seleccionar las imágenes que no tengan artefactos que obstruyan el cuello uterino.
  T.1.3 El grupo de trabajo debe etiquetar manualmente las imágenes.

- **Materiales:**
  - Bases de datos
  - Computador portátil con procesador AMD Ryzen 5 2500U, memoria RAM 8,00 GB.
  - Programa de edición de imágenes.

OE2: Implementar algoritmos de aprendizaje de máquina que permitan la segmentación automática del cuello uterino.

- **Actividades:**
  T.2.1 Investigar modelos de segmentación de aprendizaje no supervisado.
  T.2.2 Investigar modelos de segmentación de aprendizaje supervisado.
  T.2.3 Implementar modelos de aprendizaje de máquina.

- **Materiales:**
  - Bases de datos
  - Computador portátil con procesador AMD Ryzen 5 2500U, memoria RAM 8,00 GB.
  - Python
  - Google Colab
  - Jupyter

OE3: Evaluar el rendimiento de los métodos de aprendizaje empleando métricas estándar del estado del arte como el coeficiente Dice, IoU, y el Accuracy.

- **Actividades:**
  T.3.1 Evaluar los resultados obtenidos con los métodos de aprendizaje supervisado.
  T.3.2 Evaluar los resultados obtenidos con los métodos de aprendizaje no supervisado.
  T.3.3 Determinar que métodos permiten alcanzar un mejor Dice, IoU, y el Accuracy.

- **Materiales:**
  - Bases de datos
  - Motor de búsqueda académico
  - Computador portátil con procesador AMD Ryzen 5 2500U, memoria RAM 8,00 GB.
  - Python
  - Google Colab
  - Excel

6.3. Recursos

6.3.1. Humanos

Director de tesis: Hernán Darío Vargas Cardona
Recibió su título de pregrado en Ingeniería Electrónica en la universidad del Quindío, y su título de Doctor en Ingeniería de la Universidad tecnológica de Pereira, en el año 2009 y 2018 respectivamente. Desde el año 2009 hasta el 2018 ha trabajado en la universidad tecnológica de Pereira. Desde febrero de 2019, ha estado trabajando en el departamento de Electrónica y Computación de la Pontifica Universidad Javeriana Sede Cali, desempeñado como profesor y director de carrera de Ingeniería Biomédica. Durante su carrera, ha realizado diferentes investigaciones que han aportado en el campo de investigación de la ingeniería biomédica, mediante el procesamiento digital de señales, machine learning, llevándolo a obtener un reconocimiento por el tener el mejor trabajo en el simposio de computación visual en el año 2018.

6.3.2. Técnicos

- **Bases de datos:**
  - Imágenes de colposcopia validadas por el instituto nacional de cáncer y la organización mundial de la salud, 1050 imágenes, de tamaño 500x500 con formato jpg.

- **Material bibliográfico:**
  - Libros, libros digitales, artículos, repositorios y revistas.

- **Material visual:**
  - Cursos virtuales, seminarios web.

- **Programas:**
  - Python, Google Colab.

- **Computador 1:**
  - AMD Ryzen 5 2500U
  - Tarjeta gráfica: Radeon Vega Mobile Gfx 2.00 GHz
  - Memoria RAM: 8,00 GB
  - GPU: AMD Radeon (TM) Vega 8 Graphics
  - Sistema operativo: Windows 10 Pro, sistema operativo de 64 bits

- **Computador 2:**
  - AMD Ryzen 5 5600U
  - Tarjeta gráfica: Radeon Graphics 2.30 GHz
  - Memoria RAM: 8,00 GB
  - GPU: AMD Radeon (TM) Graphics
  - Sistema operativo: Windows 11 Home Single Language, Sistema operativo de 64 bits

Capítulo 7

Resultados y Discusión

7.1. Resultados experimentales

En esta sección se presentan los hallazgos y el proceso que responde a los objetivos específicos 2 y 3 del estudio. El objetivo 2 consistió en implementar algoritmos de aprendizaje de máquina para permitir la segmentación automática del cuello uterino, para ello, se utilizaron diez modelos de aprendizaje de máquina diferentes, cada uno con sus propias características y enfoques.

Por otro lado, el objetivo 3 se enfocó en evaluar el rendimiento de estos métodos mediante métricas estándar del estado del arte, como el coeficiente Dice, IoU y Accuracy, con el fin de seleccionar los modelos más efectivos y precisos para la segmentación de imágenes de colposcopia.

7.1.1. Resultados modelos no supervisados

Se implementaron seis modelos no supervisados para la segmentación de imágenes de cuello uterino: Otsu, Rango umbral, K-means, Mezclas Gausianas, Grabcut, RGB y Watershed. Para evaluar la precisión de los modelos, se construyó un dataset de pruebas de 1050 imágenes, estas imagenes fueron extraidas de la base de datos de la OMS, NHS y kaggle, este dataset incluye imágenes de diferentes calidades, lo que permite evaluar el rendimiento de los modelos en una variedad de condiciones.

Los modelos se evaluaron segmentando las imágenes del dataset de pruebas, generando máscaras binarias para cada imagen y estas se compararon con las máscaras binarias generadas manualmente, para determinar la precisión se utilizaron los parámetros IoU y Dice. Los resultados para cada modelo se presentan a continuación.

7.1.1.1. Modelo Otsu

Se segmentaron 1050 imágenes con el modelo Otsu, el parámetro IoU se calculó para cada imagen resultante. Los resultados se presentan a continuación en un histograma.

Los resultados del parámetro Dice para el modelo Otsu tienen una distribución con un pico central alto. Dado el valor central, es posible afirmar que los resultados varían mucho. Adicional a ello el rango con mayor frecuencia es (0,463 - 0,540], lo que indica que la segmentación que el modelo tiene una baja precisión en la predicción para la aplicación de segmentación de imágenes de colposcopia. Por otra parte, los resultados del parámetro IoU aunque tienen una distribución de datos diferente la interpretación de los datos es la misma, el modelo en la mayoría de los casos no puede diferenciar por completo el cuello uterino del fondo.

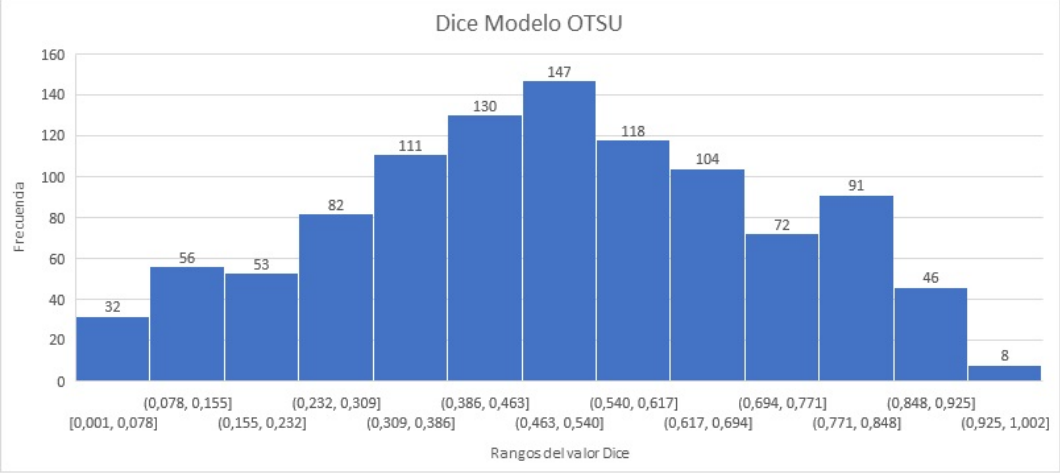


Figura 7.1: Resultado parámetro Dice para modelo Otsu (Fuente: Elaboración propia)

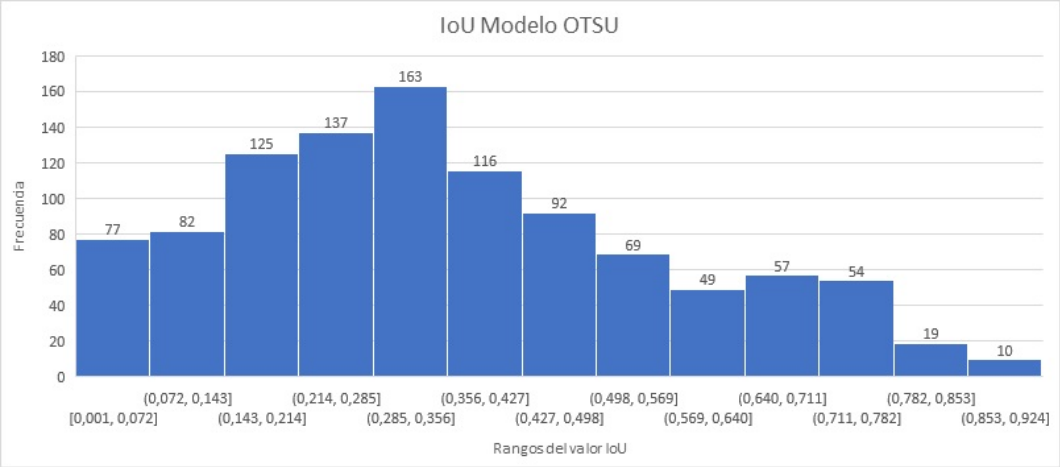


Figura 7.2: Resultado parámetro IoU para modelo Otsu (Fuente: Elaboración propia)

7.1.1.2. Rango umbral

Los resultados obtenidos para el modelo rango umbral de la prueba de segmentación para calcular los parámetros Dice e IoU se presentan a continuación.

El desempeño del método rango umbral es deficiente, en los histogramas se observa que en los dos casos de prueba más del 50 % de las imágenes evaluadas tienen una precisión entre 0 y 0.1, indicando así que el modelo no es recomendable para la aplicación dado el desempeño.

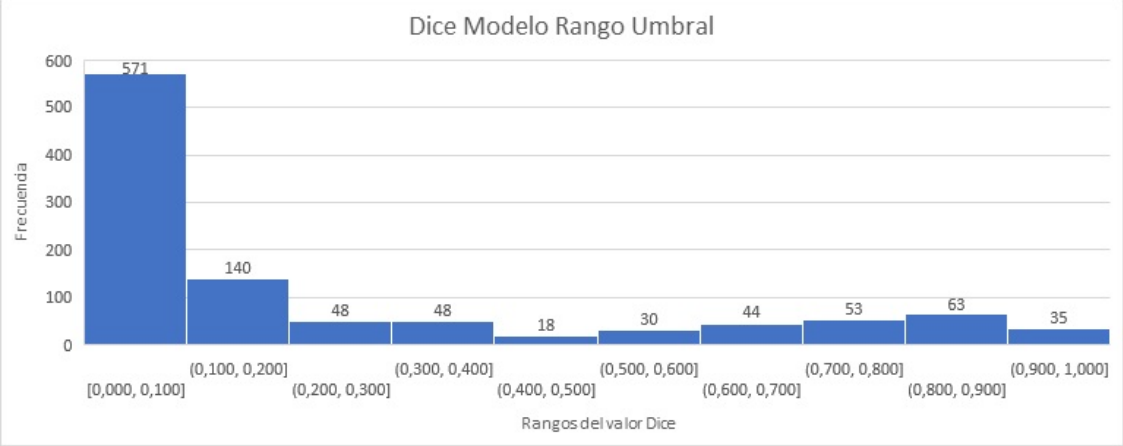


Figura 7.3: Resultado parámetro Dice para el modelo rango umbral (Fuente: Elaboración propia)

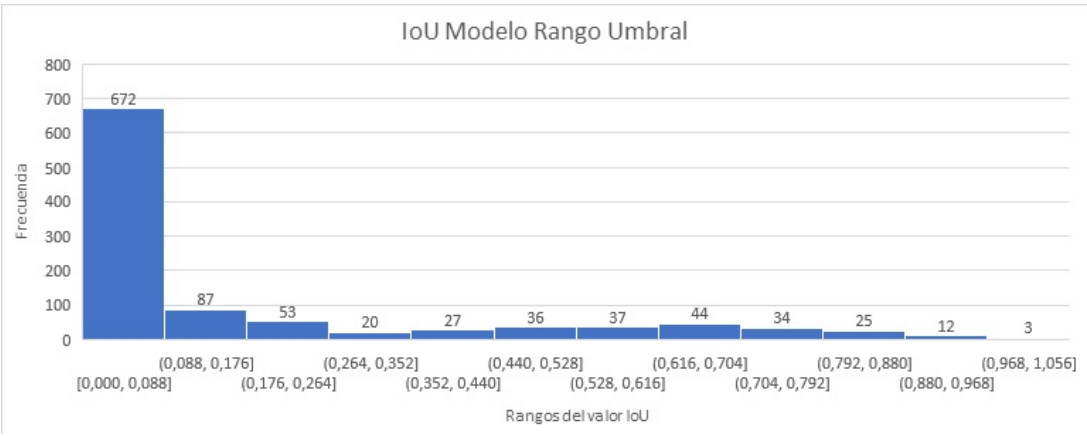


Figura 7.4: Resultado parámetro IoU para el modelo Rango umbral (Fuente: Elaboración propia)

7.1.1.3. K-means

A continuación, se presentan los resultados de los parámetros IoU y Dice después de efectuar la prueba de segmentación.

Los resultados consignados en el histograma muestra que el 81.51 % de las imágenes segmentadas tienen una precisión inferior a 0.5, además los valores de precisión con mayor frecuencia están en el rango de 0.115 a 0.215.

7.1.1.4. Mezclas Gaussianas

Los resultados obtenidos de la prueba para este modelo son los siguientes.

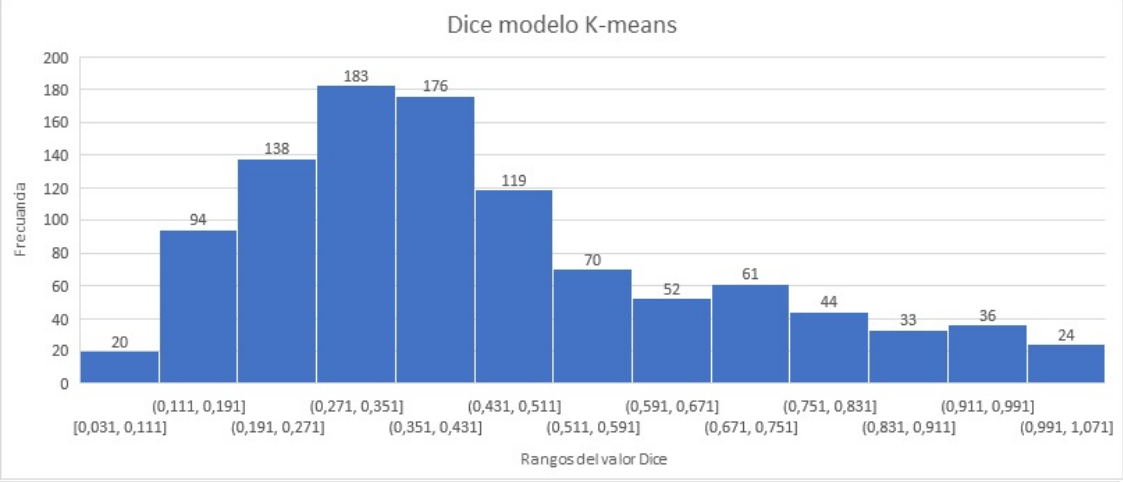


Figura 7.5: Resultado parámetro Dice para el modelo K-means (Fuente: Elaboración propia)

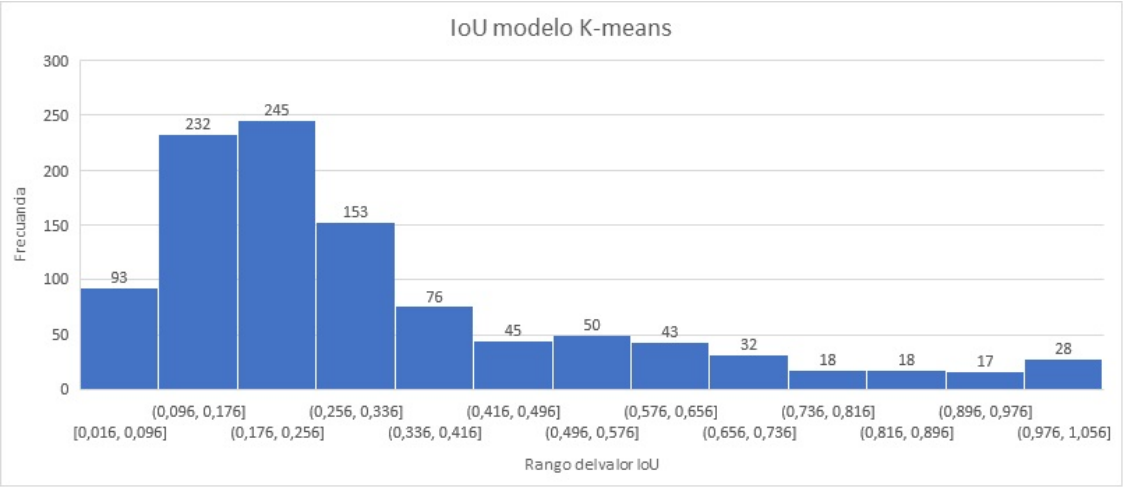


Figura 7.6: Resultado parámetro IoU para el modelo K-means (Fuente: Elaboración propia)

7.1.1.5. GrabCut

A continuación se presentan los resultados de la prueba de precisión de los modelos.

Como se observa en los histogramas la distribución de los datos es diferente, para el parámetro Dice se observa una tendencia se sesgo negativo, pues la mayoría de los datos se encuentra en el

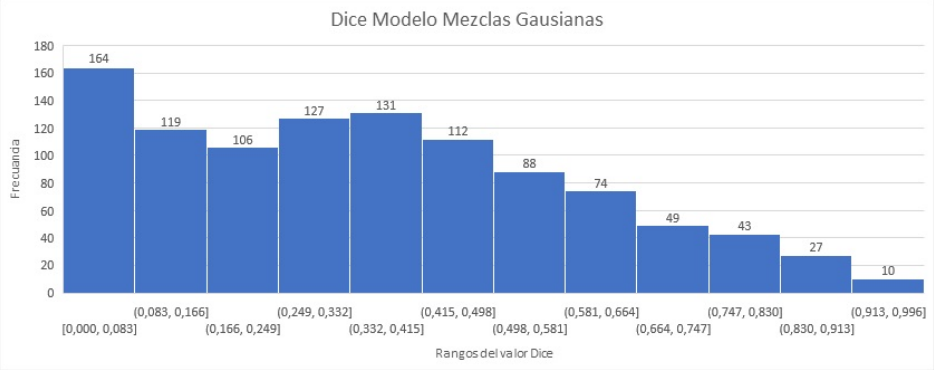


Figura 7.7: Resultado parámetro Dice para el modelo de mezclas gaussianas (Fuente: Elaboración propia)

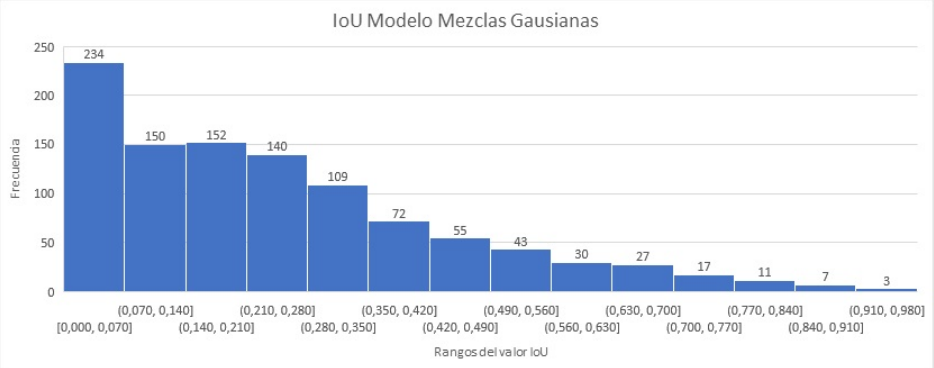


Figura 7.8: Resultado parámetro IoU para el modelo Mezclas Gaussianas (Fuente: Elaboración propia

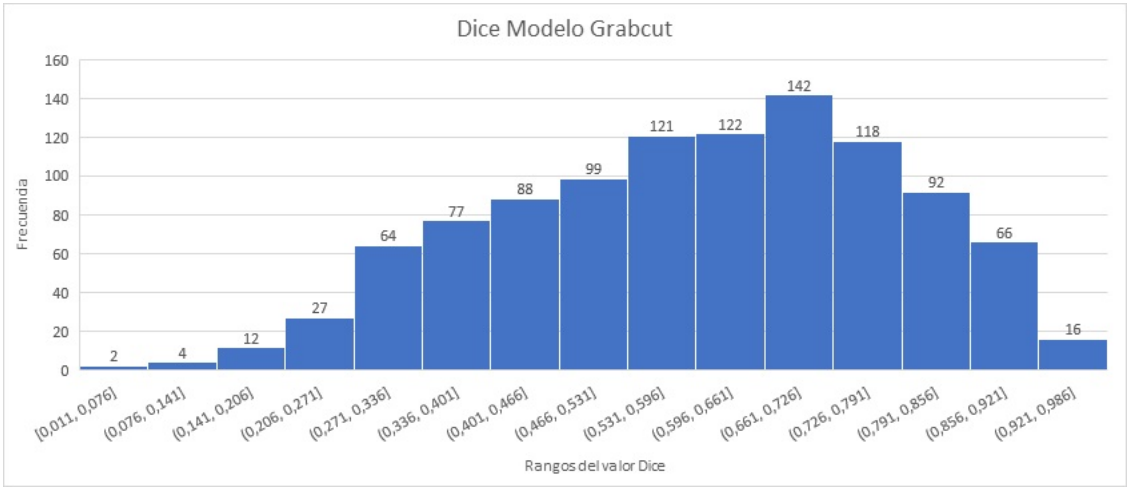


Figura 7.9: Resultado parámetro Dice para el modelo Grabcut (Fuente: Elaboración propia)

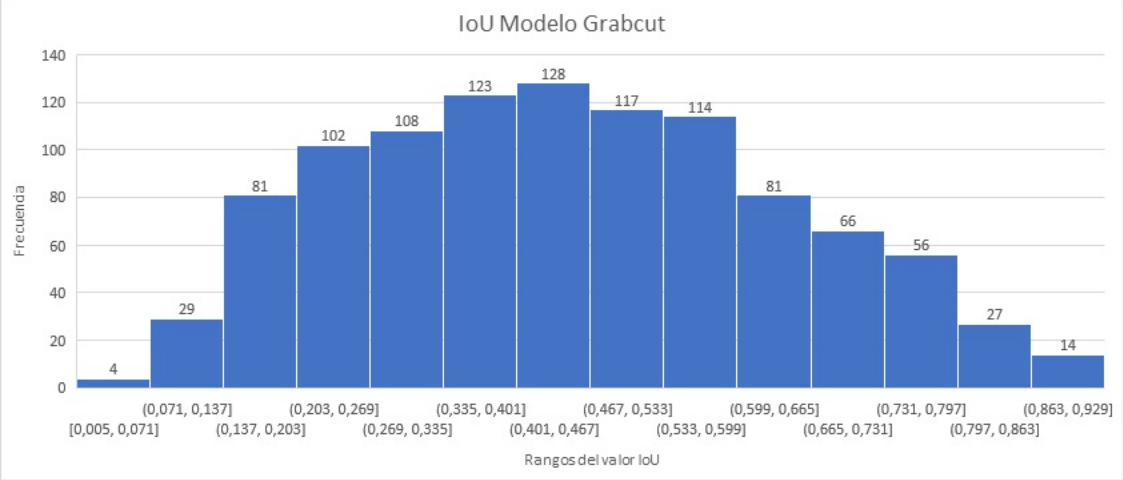


Figura 7.10: Resultado parámetro IoU para el modelo Grabcut (Fuente: Elaboración propia

extremo derecho del histograma, lo que indica que los valores mas bajos son menos frecuentes. Por el contrario, para el parámetro IoU se observa que los datos tienen mayor tendencia a los rangos centrales.

7.1.1.6. RGB

Los resultados obtenidos tras segmentar 1050 se presentan a continuación:

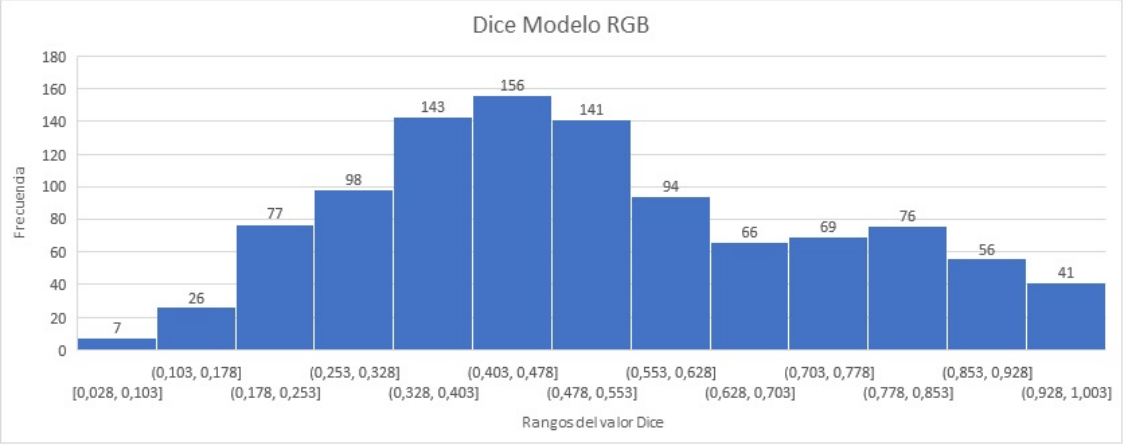


Figura 7.11: Resultado parámetro Dice para el modelo RGB (Fuente: Elaboración propia)

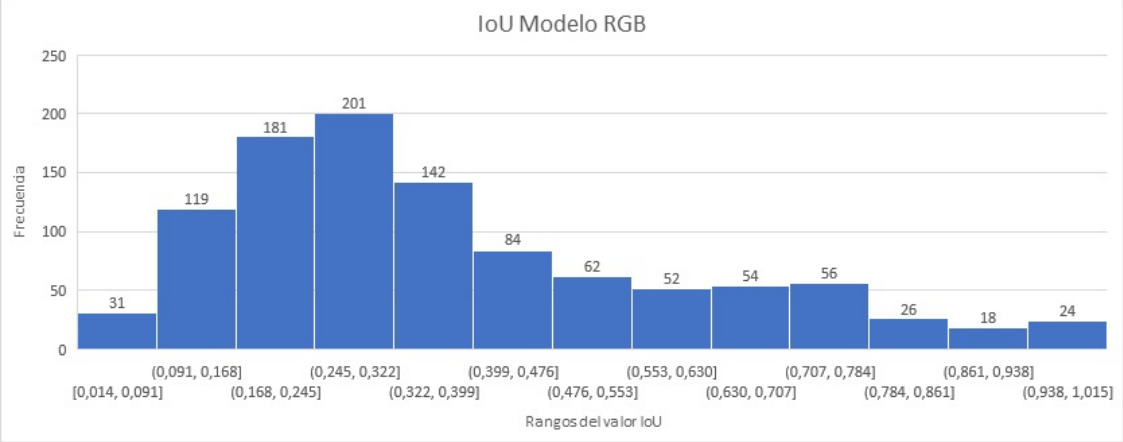


Figura 7.12: Resultado parámetro IoU para el modelo RGB (Fuente: Elaboración propia

Los resultados obtenidos para los parámetros Dice e IoU indican que el modelo de mezclas gaussianas es relativamente preciso para segmentar las imágenes. Sin embargo, hay una porción de imágenes que no fueron segmentadas tan bien. Estas imágenes pueden ser más difíciles de segmentar debido a que son más complejas o tienen menos contraste.

7.1.1.7. Watershed

Los resultados de los parámetros de precisión del modelos se presentan a continuación.

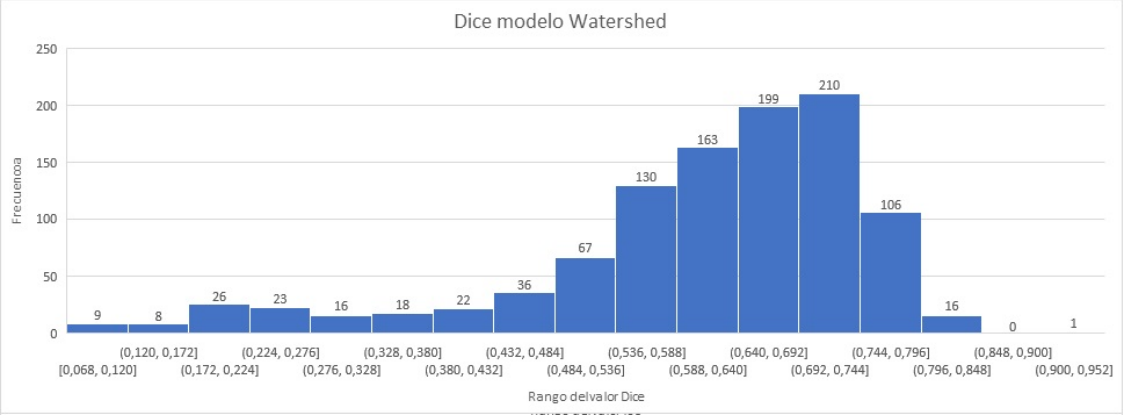


Figura 7.13: Resultado parámetro Dice para el modelo Watershed (Fuente: Elaboración propia)

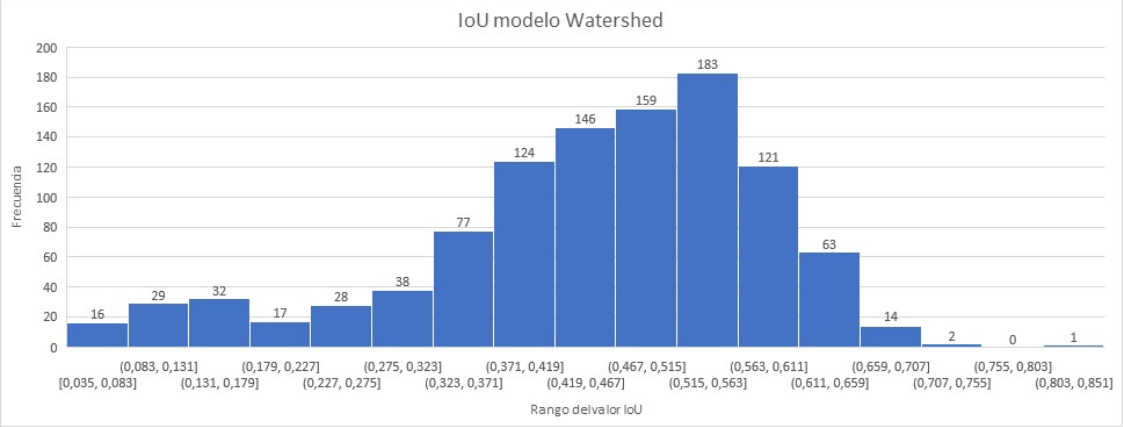


Figura 7.14: Resultado parámetro IoU para el modelo Watershed (Fuente: Elaboración propia

7.1.2. Resultados modelos supervisados

7.1.2.1. Segnet

El modelo genera las siguientes métricas, accuracy, loss. Obteniendo valores superiores a 0,7 e inferiores a 0.3 respectivamente al finalizar el entrenamiento.

```
Verifying training dataset
100%|██████████| 20/20 [00:00<00:00, 37.84it/s]
Dataset verified!
Epoch 1/6
512/512 [==============================] - 57s 100ms/step - loss: 0.5574 - accuracy: 0.7496
Epoch 2/6
512/512 [==============================] - 51s 99ms/step - loss: 0.4171 - accuracy: 0.8397
Epoch 3/6
512/512 [==============================] - 51s 99ms/step - loss: 0.3278 - accuracy: 0.8881
Epoch 4/6
512/512 [==============================] - 51s 100ms/step - loss: 0.2502 - accuracy: 0.9206
Epoch 5/6
512/512 [==============================] - 51s 99ms/step - loss: 0.1824 - accuracy: 0.9450
Epoch 6/6
512/512 [==============================] - 51s 100ms/step - loss: 0.1270 - accuracy: 0.9650
```

Figura 7.15: Métricas arquitectura SegNet

7.1.2.2. U-net

Después de implementar y entrenar el modelo con 741 imágenes, con 5 épocas los resultados gráficos y analíticos se presentan a continuación.

Figura 7.16: Entrenamiento inicial U-net

Los resultados finales de la precisión alcanzada por el modelo entrenado se presentan en la siguiente cuadro.

Acuraccy
0.8364

Cuadro 7.1: Resultados entrenamiento inicial modelo Unet

Después de obtener los resultado previos, se realizar el reentrenamiento del modelo con 1050 datos etiquetados y 5 épocas los resultados fueron los siguientes.

Figura 7.17: Reentrenamiento 1 modelo Unet

Los resultados de las métricas de precisión en el reentrenamiento son los siguientes.

Acuraccy
0.8517

Cuadro 7.2: Resultados entrenamiento 1 modelo Unet

A partir de las métricas obtenidas se puede evidenciar un leve incremento en el parámetro accuracy, sin embargo, el desempeño del modelo puede ser mejor, por esta razón se realizo un reentrenamiento por segunda vez, con imágenes 1050 y 10 épocas.

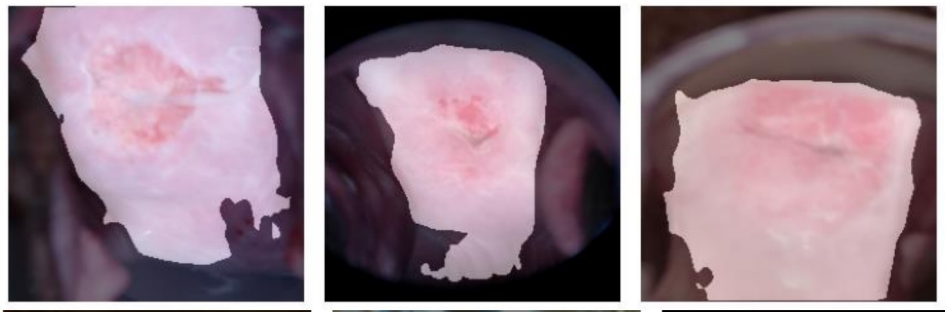


Figura 7.18: Reentrenamiento 2 modelo Unet

Al realizar el reentrenamiento los niveles de precisión alcanzados fueron los siguientes.

Acuraccy
0.9309

Cuadro 7.3: Resultados entrenamiento 2 modelo Unet

7.1.2.3. FPN

El entrenamiento inicial del modelo supervisado Feature Pyramid Network se realizó con 741 imagenes y 5 epocas, con estas condiciones se obtuvieron resultados en los que visualmente se puede identificar el cuello uterino, sin embargo, con el objetivo de determinar la calidad de la segmentación se añadieron los parámetros accuracy, IoU y Dice, para establecer la precisión del modelo al final del entrenamiento. Durante el entrenamiento inicial los resultados obtenidos fueron los que se presentan a continuación en el siguiente cuadro.

Acuraccy
0.8114

Cuadro 7.4: Resultados entrenamiento inicial modelo FPN

Posterior a ello se reentrena el modelo con 1050 datos etiquetados y 5 épocas.Los resultados obtenidos a nivel gráfico y analítico se presentan a continuación.

Figura 7.19: Ejemplo resultados reentrenamiento modelo FPN

Los resultados finales de la precisión alcanzada por el modelo entrenado se presentan en el siguiente cuadro.

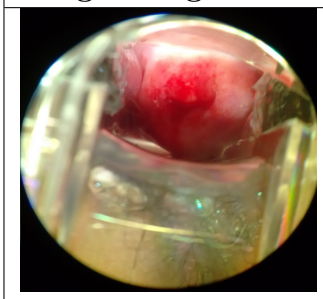
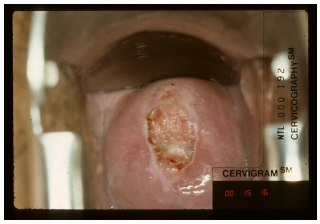
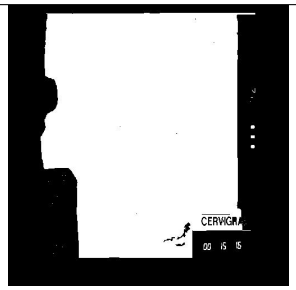
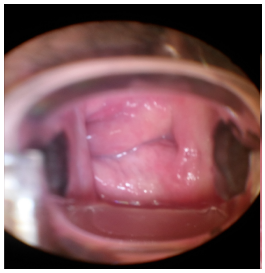
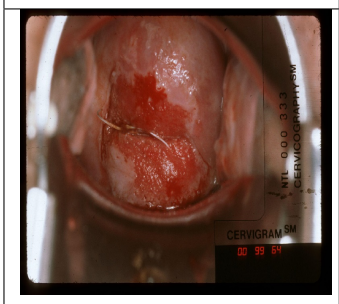
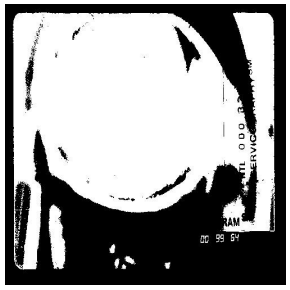
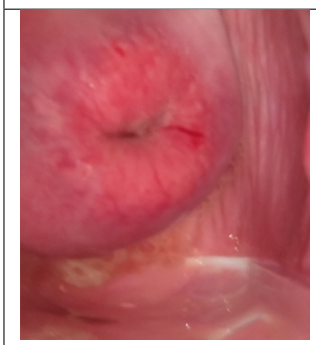
Acuraccy
0.9588

Cuadro 7.5: Resultados reentrenamiento modelo FPN

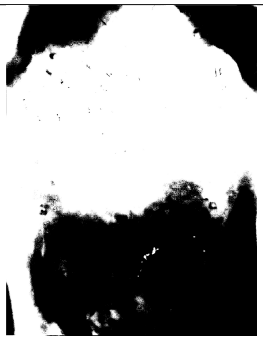
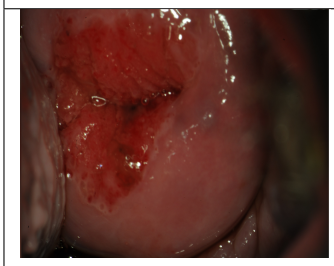
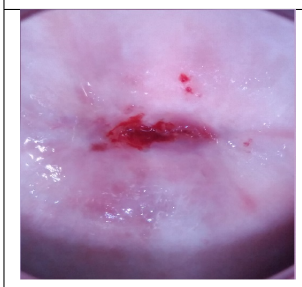
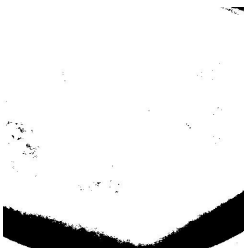
De los resultados del entrenamiento y reentrenamiento del modelo FPN se evidencia un incremento significativo para todos los parámetros de precisión, es decir que la predicción de las máscaras binarias mejoraron en un 18,16 % en el valor del accuracy, 29,66 % para el índice dice y de 55,73 % para el índice IoU. Con estos resultados es posible afirmar que este modelo es capaz de segmentar cualquier imagen de colposcopia con una precisión aproximada del 90 %.

7.2. Discusión

Después de presentar los resultados de cada modelo, es fundamental establecer criterios de evaluación para comparar el rendimiento de los algoritmos en la tarea de segmentar el cuello uterino. Estos criterios se basan en los parámetros de accuracy, coeficiente Dice e IoU, para ello, se construyó el siguiente cuadro que contiene las imágenes originales de colposcopia y las máscaras binarias generadas por modelos no supervisados, que varían a medida que el IoU se incrementa en 0.1. Al analizar los resultados de cada condición, se evidencia que para un coeficiente Dice $\geq$ 0,66 y IoU $\geq$ 0,5 se identifica únicamente el cuello uterino, mientras que en valores inferiores se observa que se segmenta el cuello uterino junto con objetos adicionales.

Imagen Original	Mascara Binaria	Dice	IoU
		0,192	0,106
		0,333	0,200
		0,461	0,300
		0,573	0,405
		0,668	0,501

Cuadro 7.6: Clasificación de criterios de evaluación parte 1

Imagen Original	Mascara Binaria	Dice	IoU
		0,750	0,600
		0,827	0,705
		0,888	0,799
		0,947	0,900
		1,0	1,0

Cuadro 7.7: Clasificación de criterios de evaluación parte 2

7.2.1. Modelos no supervisados

Para determinar el modelo no supervisado con los mejores resultados, se seleccionaron las 1050 imágenes del dataset, las cuales fueron procesadas por cada uno de los modelos para generar 1050 máscaras binarias por cada uno. Estas máscaras fueron evaluadas utilizando los parámetros IoU y Dice, y las puntuaciones asignadas a cada máscara de predicción fueron almacenadas en una hoja de cálculo de Excel. Posteriormente, los datos fueron filtrados basándose en el análisis del cuadro previo, donde se requería que el IoU ≥ 0.50 y el Dice ≥ 0.66. Al realizar este procedimiento, se obtuvieron los siguientes resultados.

Modelo	**Dice>0,66**	**IoU>0,50**
Whatershed	47,6 %	41,4 %
Grabcut	41,5 %	40,0 %
RGB	26,6 %	26,3 %
Otsu	25,5 %	24,0 %
K-means	19,9 %	19,2 %
Rango Umbral	16,0 %	15,8 %
Mezclas Gaussianas	12,8 %	12,3 %

Cuadro 7.8: Resultados generales modelos no supervisados (fuente: elaboración propia)

Después de analizar los resultados obtenidos para cada modelo no supervisado con los criterios mencionados anteriormente, IoU ≥ 0.5 y Dice ≥ 0.66, se obtuvo que los modelos con mejor rendimiento son Whatersehd y Grabcut, pues la predicción de las mascaras binarias para estos modelos tienen resultados notoriamente mejores respecto a los otros.

7.2.2. Modelos supervisados

Después de establecer los criterios de precisión aceptables para los modelos segmentación aplicados en imágenes de cuello uterino se presentan los resultados globales para definir cual se adapta en mayor medida a los parámetros de selección.

Modelo	**Accuracy**
SegNet	0,9605
FPN	0,9588
Unet	0,9303

Cuadro 7.9: Resultados generales modelos supervisados (fuente: elaboración propia)

El modelo que tiene un valor de desempeño más alto es SegNet, sin embargo, como se mostró anteriormente con valores mayores o iguales a 0.5 y 0.66, para IoU y Dice, se obtienen resultados favorables. En este caso todos los modelos tienen un desempeño cercano a 1, por lo tanto los tres pueden ser seleccionados.

Como respuesta al objetivo general de implementar técnicas de aprendizaje de máquina para la segmentación automática en imágenes de cuello uterino de colposcopia, con el fin de identificar las etapas de cáncer en la tercera fase del proyecto CITOBOT, se han seleccionado los modelos Whatersehd y SegNet. Los resultados de segmentación automática serán utilizados en la tercera fase del proyecto.

7.3. Entregable

En esta sección se mostrará el producto final entregado a los usuarios, este consiste en una aplicación que contiene los códigos de segmentación no supervisados. La razón por la cual no contiene los algoritmos supervisados, se debe estos algoritmos funcionan correctamente cuando se ejecutan en plataformas como google colaboratory o jupyter, sin embargo, cuando se programan en el IDLE de python,se generan errores relacionados a librerías o tiempos de carga en entrenamientos de los modelos, superando incluso las 7 horas en algunas pruebas.

La aplicación se diseño empleando la librería Tkinter debido a que es utilizada para realizar el diseño de interfaces gráficas de usuario o GUI. La biblioteca proporciona diferentes tipos de widgets, tal como botones, menús desplegables, campos de texto, entre otros. A continuación se muestra la aplicación de escritorio.

La figura anterior muestra la pagina de inicio de la API. En el panel izquierdo esta el recuadro “Imagen original” donde carga la imagen de cuello uterino sin ningún tipo de retoque, es decir, sin modificación de bordes, color, brillo, contraste, nitidez, etc. Para hacerlo, se debe hacer click sobre el botón “cargar imagen”, posterior a esto la aplicación tendrá acceso a las carpetas donde se han guardado previamente las imágenes a segmentar. Seleccionamos la foto de interés para cargar los datos.

Posteriormente se debe seleccionar el método a emplear, para esto debe elegir la opción no supervisados en el panel central, luego escoger los algoritmos de segmentación disponibles, algunos son: K-means, Watershed, mezclas Gausianas, etc. Una vez seleccionado el método, se debe dar click en el botón (Aceptar) para aplicar el método a la imagen cargada previamente. En la parte posterior del panel central. Finalmente la imagen segmentada carga en el recuadro derecho.

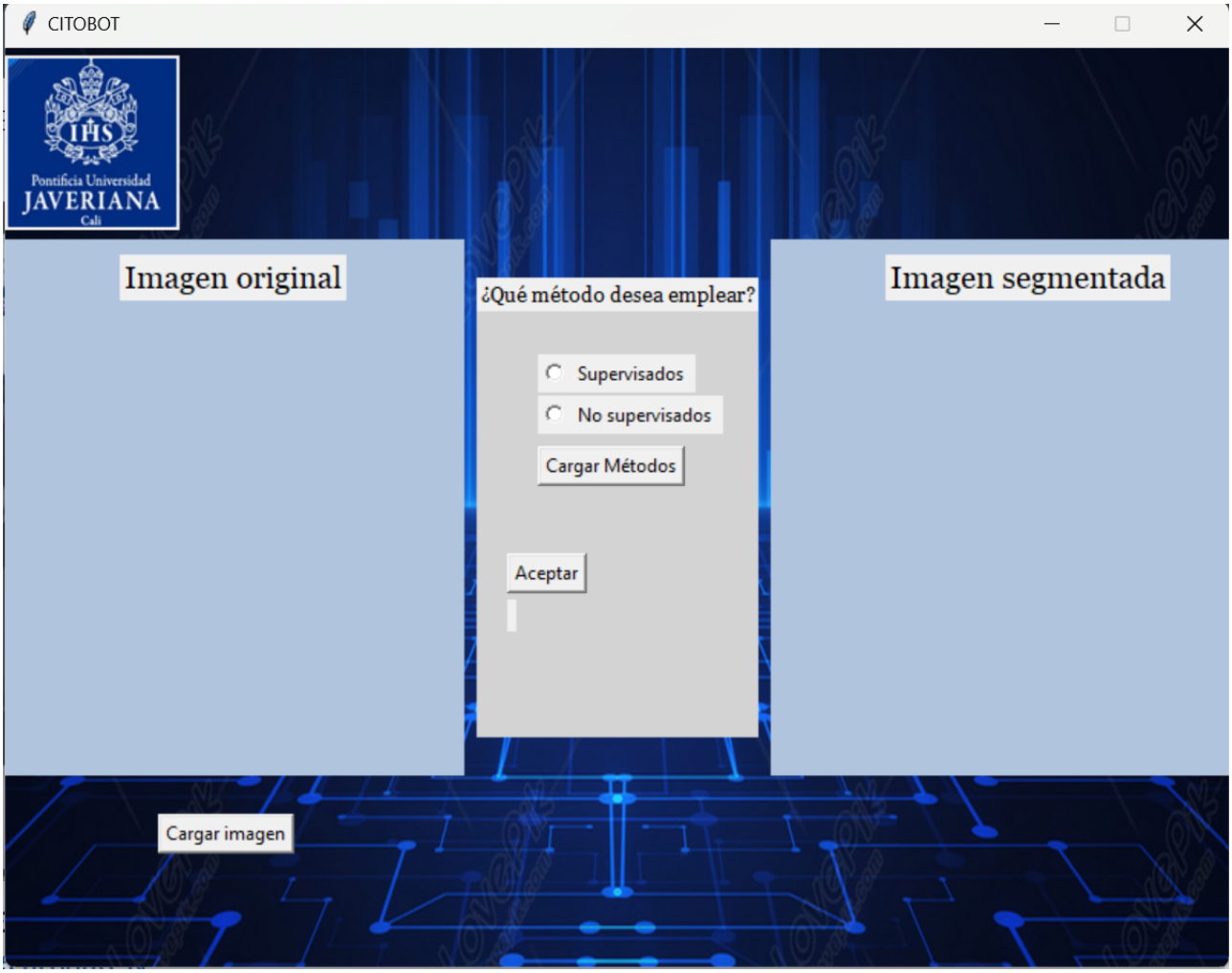


Figura 7.20: Página inicio API

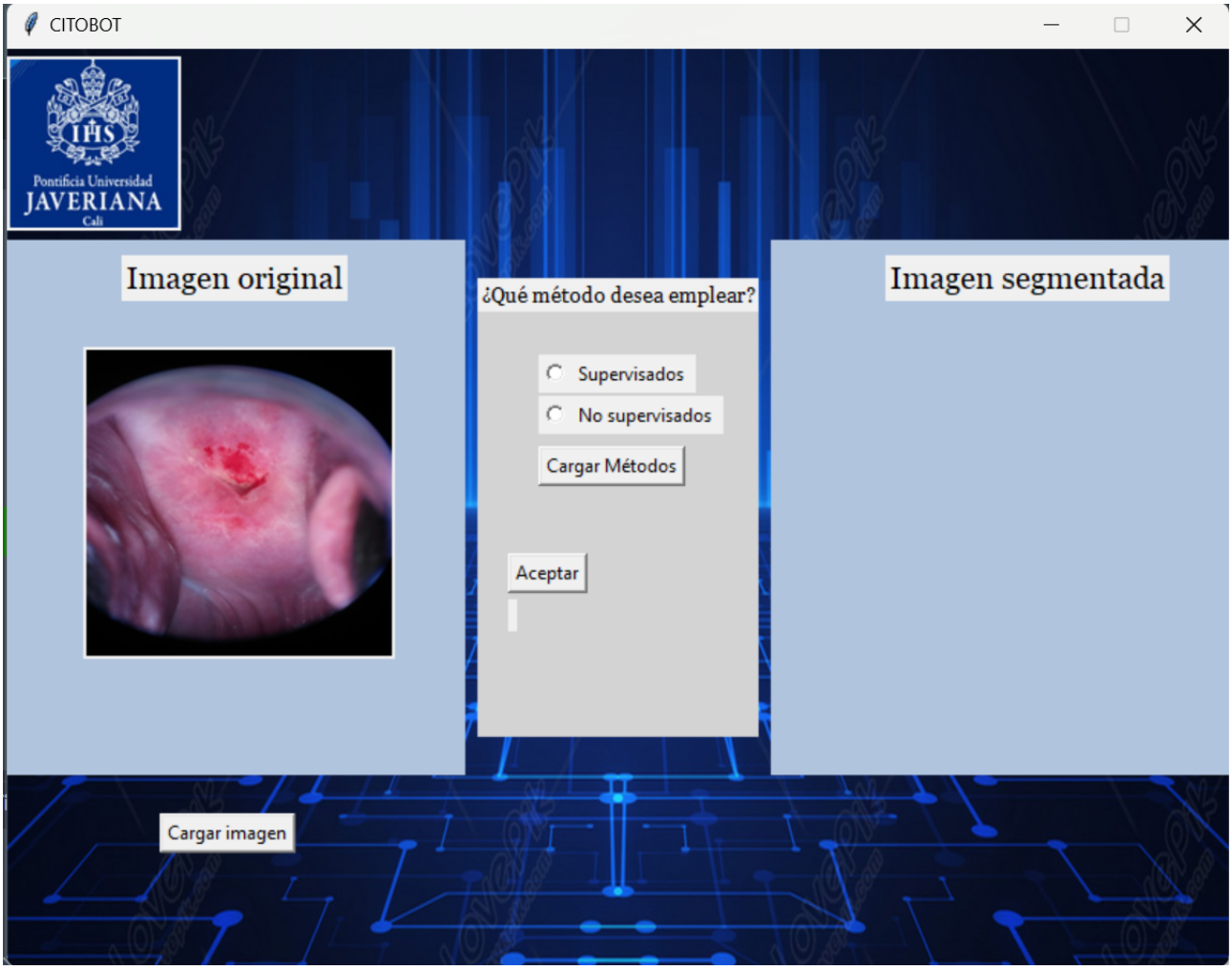


Figura 7.21: Cargar imagen API

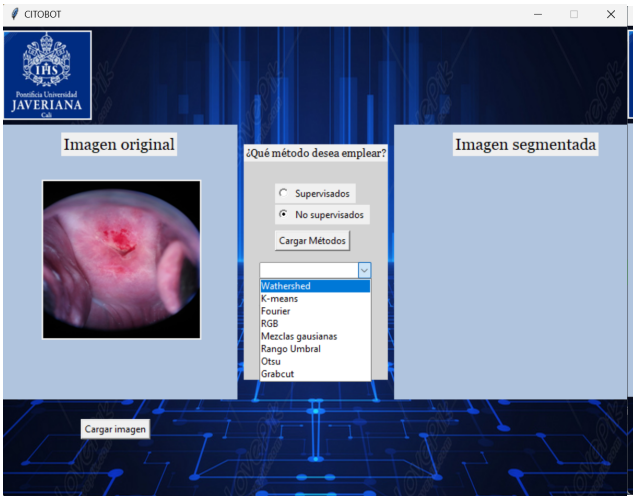


Figura 7.22: Menú API

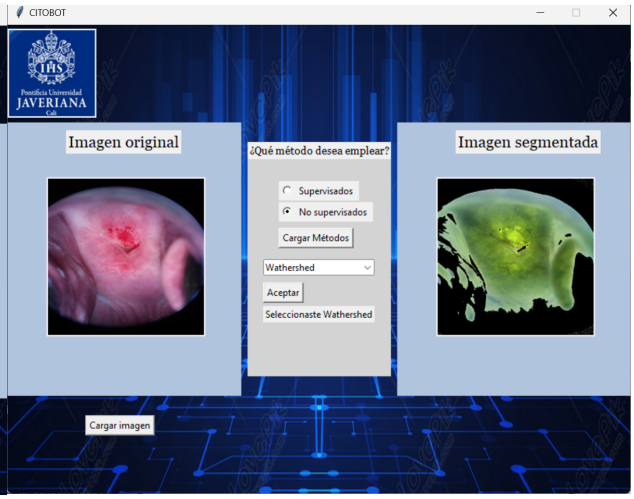


Figura 7.23: Segmentación Watershed

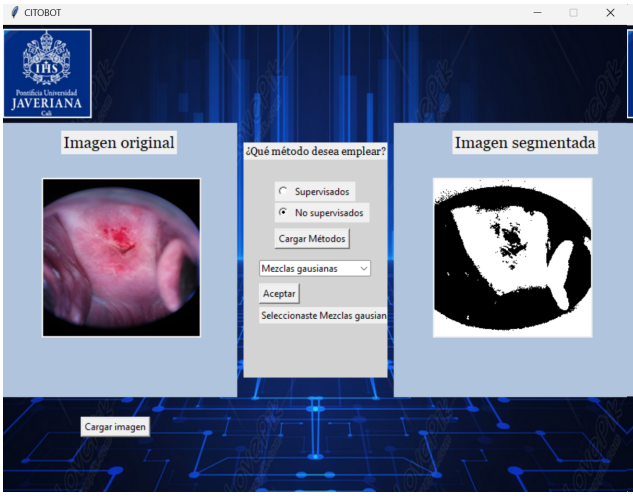


Figura 7.24: Segmentación Mezclas Gausianas

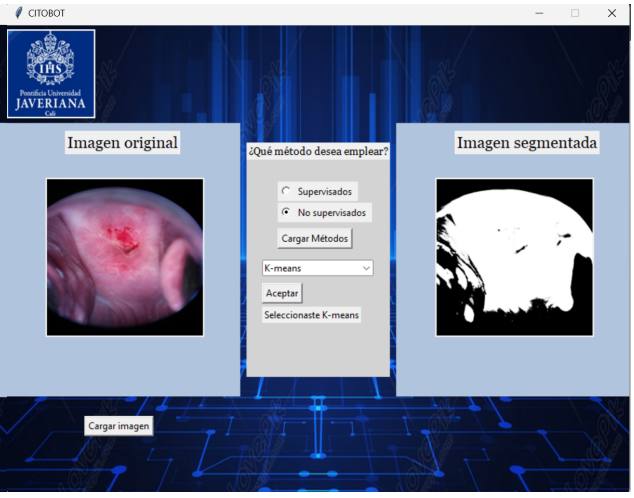


Figura 7.25: Segmentación K-means

Figura 7.26: Vistas de la GUI

Capítulo 8

Conclusiones

En este trabajo de grado nos propusimos a implementar la segmentación automática del cuello uterino en imágenes de colposcopia mediante técnicas de aprendizaje de maquina en la identificación de etapas de cáncer. Para llevarlo acabo se realizaron tres actividades principales, las cuales consistieron en organizar una base de datos donde se pudiera identificar la sección del cuello uterino, lo anterior se hacia de forma manual empleando software como krika, el cual permite editar imágenes de manera sencilla. El objetivo de organizar esta base de datos es poder emplear esta información para obtener gold standard y etiquetas, posterior y durante la etapa de segmentación automática. La base de datos es empleada en etapas posteriores a este trabajo de grado.

Seguidamente, se implementaron algoritmos de aprendizaje de máquina en python,que permitieron la segmentación del cuello uterino, los algoritmos empleados, fueron tanto, supervisados, tal como Segnet, FPN y Unet, y no supervisados, así como, grabcut, otsu, k-means, mezclas gausianas, entre otras. Cada código proporciona una mascara. En el contexto de la segmentación de imágenes, una "máscara" es una representación gráfica que se utiliza para identificar y separar áreas específicas de una imagen. Las máscaras son una herramienta fundamental en la segmentación de imágenes, ya que permiten resaltar o aislar regiones de interés dentro de una imagen.

Una máscara generalmente consiste en una imagen binaria en la que ciertas áreas están marcadas como "1"(o blanco) para representar las regiones de interés y otras áreas se marcan como "0"(o negro) para indicar áreas no deseadas. Estas áreas marcadas con "1"suelen corresponder a los objetos o características que se desean extraer de la imagen.

Posterior a la segmentación de las imágenes con las mascaras generadas por los algoritmos, se emplearon métricas para evaluar su rendimiento, tal como IoU, Dice, Accuracy y Loss los cuales son explicados con mayor detalle en secciones previas del documento. Con base a estas medidas se puede determinar que los modelos no supervisados con mejor rendimiento son grabcut y Watershed, mientras que los modelos supervisados todos tienen un rendimiento superior a 0.9 lo que lo hace apto para la aplicación del proyecto CITOBOT, para que los usuarios puedan acceder a esta tecnología se ha desarrollado una aplicación que permite el preprocesamiento de las imágenes de colposcopia con cada uno de los métodos implementados.

Como trabajos futuros, se propone el entrenamiento de los modelos supervisados con imágenes tomadas en los centros de salud que colaboran con el proyecto. Además, se propone unificar la etapa de preprocesamiento de las imágenes con la etapa de clasificación de las lesiones. De esta manera es posible obtener una herramienta robusta que soporte el diagnóstico de cáncer de cuello uterino en sus diferentes etapas.

Capítulo 9

Recomendaciones

Durante la etapa de investigación, se pudo evidenciar que han existido múltiples proyectos relacionados con la segmentación de imágenes médicas y exactamente cuello uterino para diversas aplicaciones, donde los resultados obtenidos han sido favorables en la mayoría de los casos, empleando únicamente algoritmos no supervisados, tal como k-means o k-medians, o solo algoritmos supervisados para realizar segmentación semántica, es decir, redes neuronales convoluciones. Sin embargo, gran parte de los resultados exitosos se deben a la etapa de adquisición, ya que se emplean protocolos para la toma de imágenes, es decir, se establece una posición exacta de la cámara, condiciones fijas de iluminación, único modelo de cámara para tomar las fotografías, etc. Lo anterior permite desarrollar un solo algoritmo para el preprocesamiento de los datos, tal como filtrado, realce de contraste, iluminación,etc.

Capítulo 10

Anexos

Anexo 1 – Función modelo Otsu

```
import matplotlib.pyplot as plt
import numpy as np
import cv2
import os
from PIL import Image
import numpy as np

def Otsu(img):
  imagen = cv2.imread(img)
  imagen_rgb = cv2.cvtColor(imagen,cv2.COLOR_BGR2RGB)
  imagen_gris =cv2.cvtColor(imagen,cv2.COLOR_BGR2GRAY)
  Histograma = cv2.calcHist([imagen_gris],[0],None,[256],[0,256])

  th, imagenbn = cv2.threshold(imagen_gris,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)

  return imagenbn
```

Anexo 2 – Función modelo Rango Umbral

```
import matplotlib.pyplot as plt
import numpy as np
import cv2
import os
from PIL import Image
import numpy as np

def RangoUmbral(img):
  imagen = cv2.imread(img)
  imagen_rgb = cv2.cvtColor(imagen,cv2.COLOR_BGR2RGB)
  imagen_gris =cv2.cvtColor(imagen,cv2.COLOR_BGR2GRAY)
```

```
  Histograma = cv2.calcHist([imagen_gris],[0],None,[256],[0,256])
  maximoH = max(Histograma)
  for i in range (0,len(Histograma)):
    if (Histograma [i] == maximoH) :
      indice = i
      break
  umbral_bajo, umbral_alto = indice  ,indice + 50
  imagenbn = cv2.inRange(imagen_gris, umbral_bajo,umbral_alto)

  return imagenbn
```

Anexo 3 – Función modelo K-means

```
from skimage import io
import cv2
import numpy as np
from skimage import color
from sklearn.cluster import KMeans
from PIL import Image
import os

def Kmeans(img,k):
  imagen = cv2.imread(img)
  # Convierte la imagen de BGR a RGB (OpenCV almacena las im genes en formato BGR)
  imagen_rgb = cv2.cvtColor(imagen, cv2.COLOR_BGR2RGB)
  imagen_rgb = cv2.resize(imagen_rgb, (500,500))

  # Obt n las dimensiones de la imagen
  filas, columnas, _ = imagen_rgb.shape

  # Reorganiza los pixeles para aplicar K-means
  imagen_reorganizada = imagen_rgb.reshape(-1, 3)

  # Aplica K-means
  k_means = KMeans(n_clusters=k, random_state=42)
  k_means.fit(imagen_reorganizada)
  etiquetas = k_means.labels_
  centros = k_means.cluster_centers_

  # Asigna los colores de los cl steres a cada pixel
  imagen_resultante = centros[etiquetas].reshape(filas, columnas, 3).astype(np.uint
```

```
    imagen_gris = cv2.cvtColor(imagen_resultante, cv2.COLOR_RGB2GRAY)

    return imagen_gris
```

Anexo 4 – Función modelo Grabcut

```
import numpy as np
import cv2
from matplotlib import pyplot as plt
from PIL import Image
import os
import pandas as pd

def segment_image_with_grabcut(image_path):
    # Cargar la imagen
    img = cv2.imread(image_path)

    # Redimensionar la imagen
    resized_img = cv2.resize(img, (500,500))

    # Crear una m scara inicial
    mask = np.zeros(resized_img.shape[:2], np.uint8)

    # Inicializar modelos de fondo y primer plano
    bgdModel = np.zeros((1, 65), np.float64)
    fgdModel = np.zeros((1, 65), np.float64)

    # Definir el rect ngulo de inicializaci n en la imagen redimensionada
    rect = (50, 0, 400, 450)

    # Aplicar el algoritmo GrabCut a la imagen redimensionada
    cv2.grabCut(resized_img, mask, rect, bgdModel, fgdModel, 5, cv2.GC_INIT_WITH_RE
    # Crear una m scara binaria a partir de la m scara resultante
    mask2 = np.where((mask == 2) | (mask == 0), 0, 1).astype('uint8')
    segmented = mask2.astype(bool)
    # Aplicar la m scara a la imagen redimensionada

    return segmented
```

Anexo 5 – Función modelo RGB

```
from matplotlib import image
import matplotlib.pyplot as plt
import numpy as np
import cv2
import pandas as pd
import csv
import os
from PIL import Image

def RGB(ruta_img):
  imagen = cv2.imread(ruta_img)
  imagen_rgb = cv2.cvtColor(imagen,cv2.COLOR_BGR2RGB)
  imagen_rgb = cv2.resize(imagen_rgb, (500,500))

  region = imagen_rgb[100:300,150:300,:]
  rmin, rmax = np.min(region[:,:,0].ravel()),np.max(region[:,:,0].ravel())
  gmin, gmax = np.min(region[:,:,1].ravel()),np.max(region[:,:,1].ravel())
  bmin, bmax = np.min(region[:,:,2].ravel()),np.max(region[:,:,2].ravel())

  umbral_bajo = np.array([rmin, gmin, bmin])
  umbral_alto = np.array([rmax, gmax, bmax])

  umbral = cv2.inRange(imagen_rgb,umbral_bajo,umbral_alto)
  segmented = umbral.astype(bool)

  return segmented
```

Anexo 6 – Función modelo Mezclas Gausianas

```
from PIL import Image
from skimage import io, segmentation, color
from sklearn.mixture import GaussianMixture
import numpy as np
import cv2
from matplotlib import pyplot as plt
from PIL import Image
import os
import pandas as pd

def M_Gausianas(img):
  imagen = cv2.imread(img)
  imagen_rgb = cv2.cvtColor(imagen,cv2.COLOR_BGR2RGB)
```

```
  imagen_gris =cv2.cvtColor(imagen,cv2.COLOR_BGR2GRAY)
  gmm = GaussianMixture(n_components=3,covariance_type = 'full',  init_params = 'km
  gmm.fit(imagen_gris.reshape(-1, 1))
  segmented = gmm.predict(imagen_gris.reshape(-1, 1))
  segmented = segmented.reshape(imagen_gris.shape)
  segmented = segmented.astype(bool)

  return segmented
```

Anexo 7 – Función modelo Whatershed

```
from skimage import io
import cv2
import os
import numpy as np
import matplotlib.pyplot as plt
from skimage import color
from PIL import Image

def watershed(ruta_imagen):
  #Imagen RGB
  img = cv2.imread(ruta_imagen)
  b, g, r = cv2.split(img)

  #Imagen Escala de grises
  img = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
  #Imagen escala de grises
  ret2,th2 = cv2.threshold(img,0,255,cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)

  #Remover ruido
  kernel = np.ones((3,3),np.uint8)
  opening = cv2.morphologyEx(th2, cv2.MORPH_OPEN, kernel, iterations=2)

  # Dilatamos para obtener las regiones que estamos seguros que pertenecen al fondo
  fondo = cv2.dilate(opening, kernel, iterations=3)

  # Aplicamos transformacion de distancia para hallar las regiones que estamos segu
  transformacioDistancia = cv2.distanceTransform(opening, cv2.DIST_L2, maskSize=5)
  w, primerPlano = cv2.threshold(transformacioDistancia, 0.7 * transformacioDistanc
  primerPlano = np.uint8(primerPlano)

  # Ahora hallamos las regiones sobre las que no estamos del todo seguros
```

```
  PartesDesconocidas = cv2.subtract(fondo, primerPlano)

  # Hallamos las componentes conectada
  ret, markers = cv2.connectedComponents(primerPlano)

  # Para que watershed no considere el fondo como una regi n desconocida,
  # tenemos que etiquetarla con un valor distinto
  # a 0, por lo que sumamos uno a los markers
  markers = markers + 1
  # Ahora s  , usamos el cero para denotar las regiones verdaderamente desconocidas
  markers[PartesDesconocidas == 255] = 0

  # Aplicamos watershed
  img_1 = cv2.imread(ruta_imagen)
  markers = cv2.watershed(img_1,markers)
  img_1[markers == -1] = [0, 0, 255]
  #plt.imshow(markers)

  inverted_image = cv2.bitwise_not(PartesDesconocidas)

  imagenfinal1 = b*inverted_image
  imagenfinal2 = g*inverted_image
  imagenfinal3 = r*inverted_image
  imagen_final = cv2.merge([imagenfinal1, imagenfinal2, imagenfinal3])

  return imagen_final
```

Anexo 8 – Código modelo SegNet

```
#Librer as
!pip install keras-segmentation
import cv2
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from google.colab import drive
import os
from google.oauth2.credentials import Credentials
from googleapiclient.discovery import build
from googleapiclient.errors import HttpError
import io
from google.colab import files
```

```
from IPython.display import Image
from googleapiclient.http import MediaIoBaseUpload
import sys
from pycocotools.coco import COCO
import urllib
import zipfile
from PIL import Image
from keras_segmentation.models.segnet import segnet
import shutil
import keras
drive.mount('/content/drive')

#imagenes de entramiento
!mkdir -p "/content/drive/My Drive/Mascaras_entrenamiento"
!mkdir -p "/content/drive/My Drive/Imagenes_entrenamiento"

drive_folder_path_mascaras = '/content/drive/MyDrive/Mask/'
drive_folder_path_imagenes = '/content/drive/MyDrive/Imagenes/'
image_files = os.listdir(drive_folder_path_imagenes)
image_files = sorted(image_files)
mask_files = os.listdir(drive_folder_path_mascaras)
mask_files = sorted(mask_files)

print(len(image_files),image_files)
print(len(mask_files),mask_files)

for i in image_files:
  image = cv2.imread(drive_folder_path_imagenes + i)
  image = cv2.imwrite("/content/drive/My Drive/Imagenes_entrenamiento/" + i,image )

n_classes = 2
input_height = 2448
input_width = 3264
epochs = 2
modelo = segnet(n_classes = n_classes,input_height = input_height,input_width = inp

modelo.train(train_images = '/content/drive/My Drive/Imagenes_entrenamiento', train
             epochs = epochs)

#para cada imagen
!mkdir -p "/content/drive/My Drive/MAS"
!mkdir -p "/content/drive/My Drive/IMG"
```

```
imagen1 = cv2.imread("/content/1043.jpg")
filas,columnas,profundidad = imagen1.shape
mascara_circular = np.zeros(imagen1.shape[:2], dtype='uint8')
cv2.circle(mascara_circular,(953,1800),800,(255,0,0),-1)
mascara = cv2.bitwise_and(imagen1, imagen1, mask=mascara_circular)
cv2.imwrite("Mascara1.jpg",mascara_circular)
plt.imshow(imagen1)
plt.show()
plt.imshow(mascara_circular)
plt.show()
plt.imshow(mascara)
plt.show()

imagen2 = cv2.imread("/content/1313.jpg")
filas,columnas,profundidad = imagen2.shape
mascara_circular2 = np.zeros(imagen2.shape[:2], dtype='uint8')
cv2.circle(mascara_circular2,(1765,2213),1388,(255,0,0),-1)
mascara2 = cv2.bitwise_and(imagen2, imagen2, mask=mascara_circular2)
cv2.imwrite("Mascara2.jpg",mascara_circular2)
plt.imshow(imagen2)
plt.show()
plt.imshow(mascara_circular2)
plt.show()
plt.imshow(mascara2)
plt.show()
...

#Guardar imagenes en drive

#Mascaras
cv2.imwrite('/content/drive/My Drive/MAS/Imagen1.png',mascara_circular)
cv2.imwrite('/content/drive/My Drive/MAS/Imagen2.png',mascara_circular2)
cv2.imwrite('/content/drive/My Drive/MAS/Imagen3.png',mascara_circular3)
cv2.imwrite('/content/drive/My Drive/MAS/Imagen4.png',mascara_circular4)
cv2.imwrite('/content/drive/My Drive/MAS/Imagen5.png',mascara_circular5)
cv2.imwrite('/content/drive/My Drive/MAS/Imagen6.png',mascara_circular6)
cv2.imwrite('/content/drive/My Drive/MAS/Imagen7.png',mascara_circular7)
cv2.imwrite('/content/drive/My Drive/MAS/Imagen8.png',mascara_circular8)
...

#IMAGENES Y CARPETA
```

```
cv2.imwrite('/content/drive/My Drive/IMG/Imagen1.png',imagen1)
cv2.imwrite('/content/drive/My Drive/IMG/Imagen2.png',imagen2)
cv2.imwrite('/content/drive/My Drive/IMG/Imagen3.png',imagen3)
cv2.imwrite('/content/drive/My Drive/IMG/Imagen4.png',imagen4)
cv2.imwrite('/content/drive/My Drive/IMG/Imagen5.png',imagen5)
cv2.imwrite('/content/drive/My Drive/IMG/Imagen6.png',imagen6)
cv2.imwrite('/content/drive/My Drive/IMG/Imagen7.png',imagen7)
cv2.imwrite('/content/drive/My Drive/IMG/Imagen8.png',imagen8)
...

#Redimensionar las imagenes
!mkdir -p "/content/drive/My Drive/Mascaras_entrenamiento_2"
!mkdir -p "/content/drive/My Drive/Imagenes_entrenamiento_2"

folder_path = '/content/drive/My Drive/IMG/'

#Redimensionar y normalizar imagenes y guardarlas en drive
for i in range(1,21):
  nombre_actual = "Imagen"+str(i)+'.png'
  image_path = '/content/drive/My Drive/IMG/'
  image_path = image_path + nombre_actual
  destination_folder = '/content/drive/My Drive/Imagenes_entrenamiento_2/'
  image = cv2.imread(image_path)
  resized_image = cv2.resize(image, (500,500))
  #resized_image = resized_image / np.max(resized_image)
  destination_path = destination_folder + nombre_actual
  cv2.imwrite(destination_path, resized_image)
  print(np.min(resized_image),np.max(resized_image))
  print(resized_image.shape)
  plt.imshow(resized_image)
  plt.show()

#Mascaras redimensionadas y a iadidas a drive
for i in range(1,21):
  nombre_actual = "Imagen"+str(i)+'.png'
  image_path = '/content/drive/My Drive/MAS/'
  image_path = image_path + nombre_actual
  destination_folder = '/content/drive/My Drive/Mascaras_entrenamiento_2/'
  image = cv2.imread(image_path,cv2.IMREAD_GRAYSCALE)
  resized_image = cv2.resize(image, (500,500))
  umbral, imagen_binaria = cv2.threshold(resized_image, 1, 1, cv2.THRESH_BINARY)
```

```
  #resized_image = resized_image / np.max(resized_image)
  destination_path = destination_folder + nombre_actual
  cv2.imwrite(destination_path, imagen_binaria)
  print(imagen_binaria.shape,np.max(imagen_binaria),np.min(imagen_binaria))
  plt.imshow(resized_image)
  plt.show()
  plt.imshow(imagen_binaria)
  plt.show()

#Entrenamiento
n_classes = 2
input_height = 500
input_width = 500
epochs = 6
modelo = segnet(n_classes = n_classes,input_height = input_height,input_width = inp

modelo.train(train_images = '/content/drive/MyDrive/Imagenes_entrenamiento_2',
             train_annotations = '/content/drive/My Drive/Mascaras_entrenamiento_2'
             epochs = epochs)

#Resultado del modelo
out = modelo.predict_segmentation(
    inp="/content/drive/MyDrive/Imagenes_entrenamiento_2/Imagen1.png"
)
Ruta_imagen = '/content/drive/MyDrive/Imagenes_entrenamiento_2/Imagen1.png'
imagen = Image.open(Ruta_imagen)
imagen = np.array(imagen)
out = np.array(out)
imagen = cv2.resize(imagen,(248,248))
plt.imshow(imagen)
plt.title('Imagen original')
plt.show()
plt.imshow(out)
plt.title('Mascara')
plt.show()

b, g, r = cv2.split(imagen)
MASK1 = b * out
MASK2 = g * out
MASK3 = r * out

imagen_segmentada = cv2.merge([MASK1, MASK2, MASK3])
```

```
plt.imshow(imagen_segmentada)
plt.title('imagen_segmentada')
plt.show()
```

Anexo 8 – Código modelo FPN

```
#Librerias
import numpy as np
import matplotlib.pyplot as plt
import os
import random
import torch
import segmentation_models_pytorch as smp
from torch import nn, optim
from torch.nn import functional as F
from torchvision import transforms as T
from torch.utils.data import DataLoader, Dataset, random_split
import PIL
from PIL import Image

#Path data
PATH = '/content/drive/MyDrive/Training_SM/Training_Unet'
TRAIN_PATH = '/content/drive/MyDrive/Training_SM/Training_Unet/Imagenes'
TRAIN_MASKS_PATH = '/content/drive/MyDrive/Training_SM/Training_Unet/Mask'
TEST_PATH = '/content/drive/MyDrive/Training_SM/Training_Unet/Type_1_test'

#Dataset
class Colposcopia_Dataset(Dataset):
    def __init__(self, data, masks=None, img_transforms=None, mask_transforms=None)
        '''
        data - train data path
        masks - train masks path
        '''
        self.train_data = data
        self.train_masks = masks
        self.img_transforms = img_transforms
        self.mask_transforms = mask_transforms
        self.images = sorted(os.listdir(self.train_data))
        self.masks = sorted(os.listdir(self.train_masks))
    def __len__(self):
        if self.train_masks is not None:
            assert len(self.images)==len(self.masks), 'not the same number of image
```

```
        return len(self.images)
    def __getitem__(self, idx):
        image_name = os.path.join(self.train_data, self.images[idx])
        img = Image.open(image_name)
        trans = T.ToTensor()
        if self.img_transforms is not None:
            img = self.img_transforms(img)
        else:
            img =trans(img)

        if self.train_masks is not None:
            mask_name = os.path.join(self.train_masks, self.masks[idx])
            mask = Image.open(mask_name)
            if self.mask_transforms is not None:
                mask = self.mask_transforms(mask)
            else:
                mask = trans(mask)
            mask_max = mask.max().item()
            mask /= mask_max
        else:
            return img
        return img, mask

#Transformaciones de im genes
transform_data = T.Compose([
                 T.Resize([224,224]),
                 T.ToTensor()])

transform_data_gray = T.Compose([
                 T.Resize([224,224]),
                 T.Grayscale(num_output_channels=1),
                 T.ToTensor()])

#Ordenar Dataset
full_dataset = Colposcopia_Dataset(TRAIN_PATH,
                           TRAIN_MASKS_PATH,
                           img_transforms=transform_data,
                           mask_transforms=transform_data_gray)

#Dividir dataset en entrenamieto y test
BATCH_SIZE = 3
TRAIN_SIZE = int(len(full_dataset)*0.8)
```

```
VAL_SIZE = len(full_dataset)-TRAIN_SIZE

print(TRAIN_SIZE,VAL_SIZE)

train_dataset, val_dataset = random_split(full_dataset, [TRAIN_SIZE, VAL_SIZE])
print(len(train_dataset), len(val_dataset))


train_loader = DataLoader(train_dataset, batch_size=BATCH_SIZE, shuffle=True)
val_loader = DataLoader(val_dataset, batch_size=BATCH_SIZE, shuffle=True)

#Modelo FPN
!pip install git+https://github.com/qubvel/segmentation_models.pytorch

#resnext50_32x4d, mit_b2, timm-gernet_s, efficientnet-b3, mobilenet_v2, resnet152,
ENCODER = 'resnet101'
ENCODER_WEIGHTS = 'imagenet'
ACTIVATION = 'softmax'

# create segmentation model with pretrained encoder
#Decoders= PAN, PSPNet, MAnet, Linknet, FPN, DeepLabV3, DeepLabV3Plus, Unet
model =smp.FPN(
    encoder_name=ENCODER,
    encoder_weights=ENCODER_WEIGHTS,
    classes=2,
    activation=ACTIVATION,
)

#Metricas
def accuracy(model, loader):
    correct = 0
    intersection = 0
    denom = 0
    union = 0
    total = 0
    cost = 0.
    model = model.to(device=device)
    with torch.no_grad():
        for x, y in loader:
            x = x.to(device=device, dtype = torch.float32)
            y = y.to(device=device, dtype = torch.long).squeeze(1)
            scores = model(x)
```

```
            cost += (F.cross_entropy(scores, y)).item()
            # standard accuracy not optimal
            preds = torch.argmax(scores, dim=1)
            correct += (preds == y).sum()
            total += torch.numel(preds)
            #dice coefficient
            intersection += (preds*y).sum()
            denom += (preds + y).sum()
            dice = 2*intersection/(denom + 1e-8)
            #intersection over union
            union += (preds + y - preds*y).sum()
            iou = (intersection)/(union + 1e-8)

        return cost/len(loader), float(correct)/total, dice, iou

#Learning rate
def find_lr(model, optimiser, start_val = 1e-6, end_val = 1, beta = 0.99, loader =
    n = len(loader) - 1
    factor = (end_val / start_val)**(1/n)
    lr = start_val
    optimiser.param_groups[0]['lr'] = lr #this allows you to update the learning ra
    avg_loss, loss, acc = 0., 0., 0.
    lowest_loss = 0.
    batch_num = 0
    losses = []
    log_lrs = []
    accuracies = []
    model = model.to(device=device)
    for i, (x, y) in enumerate(loader, start=1):
        x = x.to(device = device, dtype = torch.float32)
        y = y.to(device = device, dtype = torch.long).squeeze(1)
        optimiser.zero_grad()
        scores = model(x)
        cost = F.cross_entropy(input=scores, target=y)
        loss = beta*loss + (1-beta)*cost.item()
        #bias correction
        avg_loss = loss/(1 - beta**i)

        preds = torch.argmax(scores, dim=1)
        acc_ = (preds == y).sum()/torch.numel(scores)
#         acc = beta*acc + (1-beta)*acc_.item()
#         avg_acc = acc/(1 - beta**i)
```

```
        #if loss is massive stop
        if i > 1 and avg_loss > 4 * lowest_loss:
            print(f'from here{i, cost.item()}')
            return log_lrs, losses, accuracies
        if avg_loss < lowest_loss or i == 1:
            lowest_loss = avg_loss

        accuracies.append(acc_.item())
#         accuracies.append(avg_acc)
        losses.append(avg_loss)
        log_lrs.append(lr)
        #step
        cost.backward()
        optimiser.step()
        #update lr
        print(f'cost:{cost.item():.4f}, lr: {lr:.4f}, acc: {acc_.item():.4f}')
        lr *= factor
        optimiser.param_groups[0]['lr'] = lr
    return log_lrs, losses, accuracies

def train(model, optimiser, scheduler = None, epochs = 100, store_every = 25):
    model = model.to(device=device)
    for epoch in range(epochs):
        train_correct_num = 0
        train_total = 0
        train_cost_acum = 0.
        for mb, (x, y) in enumerate(train_loader, start=1):
            model.train()
            x = x.to(device=device, dtype=torch.float32)
            y = y.to(device=device, dtype=torch.long).squeeze(1)
            scores = model(x)
            cost = F.cross_entropy(input=scores, target=y)
            optimiser.zero_grad()
            cost.backward()
            optimiser.step()

            if scheduler:
                scheduler.step()

            train_predictions = torch.argmax(scores, dim=1)
            train_correct_num += (train_predictions == y).sum()
            train_total += torch.numel(train_predictions)
```

```
            train_cost_acum += cost.item()
            if mb%store_every == 0:
                val_cost, val_acc, dice, iou = accuracy(model, val_loader)
                train_acc = float(train_correct_num)/train_total
                train_cost_every = float(train_cost_acum)/mb
                print(f'epoch: {epoch}, mb: {mb}, train cost: {train_cost_every:.4f
                      f'train acc: {train_acc:.4f}, val acc: {val_acc:.4f},'
                      f'dice: {dice}, iou: {iou}')

#Entrenamiento
torch.manual_seed(42)
optimiser_unet = torch.optim.SGD(model.parameters(),
                                 lr=0.0001, momentum=0.95,
                                 weight_decay=0.001)

lg_lr, losses, accuracies = find_lr(model, optimiser_unet, start_val=1e-5, end_val=

# plot loss vs learning rate
f1, ax1 = plt.subplots(figsize=(12,10))
ax1.plot(lg_lr, losses)
ax1.set_xscale('log')
ax1.set_xlabel('lg_lr')
ax1.set_ylabel('losses')
plt.show()

# define the model and train with scheduler
torch.manual_seed(42)
epochs = 5
optimiser = torch.optim.SGD(model.parameters(),
                                 lr=0.01, momentum=0.95,
                                 weight_decay=1e-4)
scheduler = torch.optim.lr_scheduler.OneCycleLR(optimiser,
                                                max_lr = 1e-1,
                                                steps_per_epoch=len(train_loader),
                                                epochs=epochs, pct_start=0.43, div_
                                                three_phase=True)
train(model, optimiser, scheduler, epochs)

#Prueba del modelo
test_set = Colposcopia_Dataset(TEST_PATH, img_transforms=transform_data)
test_loader = DataLoader(test_set, batch_size=BATCH_SIZE, shuffle=True)
imgs_test = next(iter(test_loader))
```

```
imgs_test = imgs_test.to(device, dtype=torch.float32)
model = model.to(device)
with torch.no_grad():
    scores = model(imgs_test)
    preds = torch.argmax(scores, dim=1).float()

imgs_test = imgs_test.cpu()
preds = preds.cpu()
print(preds.shape)
plot_mini_bacth(imgs_test, preds.unsqueeze(1))
```

Anexo 9 – Código modelo Unet

```
#Librer as
import numpy as np
import matplotlib.pyplot as plt
import os
import random
import torch
from torch import nn, optim
from torch.nn import functional as F
from torchvision import transforms as T
from torch.utils.data import DataLoader, Dataset, random_split
import PIL
from PIL import Image

#PATH Datset and Dataloader
PATH = '/content/drive/MyDrive/Training_SM/Training_Unet'
TRAIN_PATH = '/content/drive/MyDrive/Training_SM/Training_Unet/Imagenes'
TRAIN_MASKS_PATH = '/content/drive/MyDrive/Training_SM/Training_Unet/Mask'
#TEST_PATH ='/content/drive/MyDrive/WHO/IARCImageBankColpo/Case 023/AAIR1.jpg'
TEST_PATH = '/content/drive/MyDrive/Training_SM/Training_Unet/Type_1_test'

#Dataloaders
class Colposcopia_Dataset(Dataset):
    def __init__(self, data, masks=None, img_transforms=None, mask_transforms=None)
        '''
        data - train data path
        masks - train masks path
        '''
        self.train_data = data
        self.train_masks = masks
```

```
        self.img_transforms = img_transforms
        self.mask_transforms = mask_transforms

        self.images = sorted(os.listdir(self.train_data))
        self.masks = sorted(os.listdir(self.train_masks))

    def __len__(self):
        if self.train_masks is not None:
            assert len(self.images)==len(self.masks), 'not the same number of image
        return len(self.images)

    def __getitem__(self, idx):
        image_name = os.path.join(self.train_data, self.images[idx])
        img = Image.open(image_name)
        trans = T.ToTensor()
        if self.img_transforms is not None:
            img = self.img_transforms(img)
        else:
            img =trans(img)

        if self.train_masks is not None:
            mask_name = os.path.join(self.train_masks, self.masks[idx])
            mask = Image.open(mask_name)
            if self.mask_transforms is not None:
                mask = self.mask_transforms(mask)
            else:
                mask = trans(mask)

            mask_max = mask.max().item()
            mask /= mask_max
        else:
            return img

        return img, mask

#Transformaciones de imagenes
transform_data = T.Compose([
                 T.Resize([224,224]),
                 T.ToTensor()])

transform_data_gray = T.Compose([
```

```
                    T.Resize([224,224]),
                    T.Grayscale(num_output_channels=1),
                    T.ToTensor()])

#Ordenar Dataset
full_dataset = Colposcopia_Dataset(TRAIN_PATH,
                                   TRAIN_MASKS_PATH,
                                   img_transforms=transform_data,
                                   mask_transforms=transform_data_gray)

#Dividir imagenes en entrenamiento y test
BATCH_SIZE = 3
TRAIN_SIZE = int(len(full_dataset)*0.8)
VAL_SIZE = len(full_dataset)-TRAIN_SIZE

print(TRAIN_SIZE,VAL_SIZE)

train_dataset, val_dataset = random_split(full_dataset, [TRAIN_SIZE, VAL_SIZE])
print(len(train_dataset), len(val_dataset))


train_loader = DataLoader(train_dataset, batch_size=BATCH_SIZE, shuffle=True)
val_loader = DataLoader(val_dataset, batch_size=BATCH_SIZE, shuffle=True)

#Modelo Unet
class Conv_3_k(nn.Module):
    def __init__(self, channels_in, channels_out):
        super().__init__()
        self.conv1 = nn.Conv2d(channels_in, channels_out, kernel_size=3, stride=1,
    def forward(self, x):
        return self.conv1(x)

class Double_Conv(nn.Module):
    '''
    Double convolution block for U-Net
    '''
    def __init__(self, channels_in, channels_out):
        super().__init__()
        self.double_conv = nn.Sequential(
                                   Conv_3_k(channels_in, channels_out),
                                   nn.BatchNorm2d(channels_out),
                                   nn.ReLU(),
```

```
                                Conv_3_k(channels_out, channels_out),
                                nn.BatchNorm2d(channels_out),
                                nn.ReLU(),
                                 )
    def forward(self, x):
        return self.double_conv(x)

class Down_Conv(nn.Module):
    '''
    Down convolution part
    '''
    def __init__(self, channels_in, channels_out):
        super().__init__()
        self.encoder = nn.Sequential(
                        nn.MaxPool2d(2,2),
                        Double_Conv(channels_in, channels_out)
                        )
    def forward(self, x):
        return self.encoder(x)

class Up_Conv(nn.Module):
    '''
    Up convolution part
    '''
    def __init__(self,channels_in, channels_out):
        super().__init__()
        self.upsample_layer = nn.Sequential(
                        nn.Upsample(scale_factor=2, mode='bicubic'),
                        nn.Conv2d(channels_in, channels_in//2, kernel_size=1, strid
                        )
        self.decoder = Double_Conv(channels_in, channels_out)

    def forward(self, x1, x2):
        '''
        x1 - upsampled volume
        x2 - volume from down sample to concatenate
        '''
        x1 = self.upsample_layer(x1)
        x = torch.cat([x2, x1],dim=1)
        return self.decoder(x)
```

```
class UNET(nn.Module):
    '''
    UNET model
    '''
    def __init__(self, channels_in, channels, num_classes):
        super().__init__()
        self.first_conv = Double_Conv(channels_in, channels) #64, 224, 224
        self.down_conv1 = Down_Conv(channels, 2*channels) # 128, 112, 112
        self.down_conv2 = Down_Conv(2*channels, 4*channels) # 256, 56, 56
        self.down_conv3 = Down_Conv(4*channels, 8*channels) # 512, 28, 28

        self.middle_conv = Down_Conv(8*channels, 16*channels) # 1024, 14, 14

        self.up_conv1 = Up_Conv(16*channels, 8*channels)
        self.up_conv2 = Up_Conv(8*channels, 4*channels)
        self.up_conv3 = Up_Conv(4*channels, 2*channels)
        self.up_conv4 = Up_Conv(2*channels, channels)

        self.last_conv = nn.Conv2d(channels, num_classes, kernel_size=1, stride=1)

    def forward(self, x):
        x1 = self.first_conv(x)
        x2 = self.down_conv1(x1)
        x3 = self.down_conv2(x2)
        x4 = self.down_conv3(x3)

        x5 = self.middle_conv(x4)

        u1 = self.up_conv1(x5, x4)
        u2 = self.up_conv2(u1, x3)
        u3 = self.up_conv3(u2, x2)
        u4 = self.up_conv4(u3, x1)

        return self.last_conv(u4)

#Presicion modelo
def accuracy(model, loader):
    correct = 0
    intersection = 0
    denom = 0
    union = 0
    total = 0
```

```
    cost = 0.
    model = model.to(device=device)
    with torch.no_grad():
        for x, y in loader:
            x = x.to(device=device, dtype = torch.float32)
            y = y.to(device=device, dtype = torch.long).squeeze(1)
            scores = model(x)
            cost += (F.cross_entropy(scores, y)).item()
            # standard accuracy not optimal
            preds = torch.argmax(scores, dim=1)
            correct += (preds == y).sum()
            total += torch.numel(preds)
            #dice coefficient
            intersection += (preds*y).sum()
            denom += (preds + y).sum()
            dice = 2*intersection/(denom + 1e-8)
            #intersection over union
            union += (preds + y - preds*y).sum()
            iou = (intersection)/(union + 1e-8)

        return cost/len(loader), float(correct)/total, dice, iou

#Learning rate
def find_lr(model, optimiser, start_val = 1e-6, end_val = 1, beta = 0.99, loader =
    n = len(loader) - 1
    factor = (end_val / start_val)**(1/n)
    lr = start_val
    optimiser.param_groups[0]['lr'] = lr #this allows you to update the learning ra
    avg_loss, loss, acc = 0., 0., 0.
    lowest_loss = 0.
    batch_num = 0
    losses = []
    log_lrs = []
    accuracies = []
    model = model.to(device=device)
    for i, (x, y) in enumerate(loader, start=1):
        x = x.to(device = device, dtype = torch.float32)
        y = y.to(device = device, dtype = torch.long).squeeze(1)
        optimiser.zero_grad()
        scores = model(x)
        cost = F.cross_entropy(input=scores, target=y)
        loss = beta*loss + (1-beta)*cost.item()
```

```
        #bias correction
        avg_loss = loss/(1 - beta**i)

        preds = torch.argmax(scores, dim=1)
        acc_ = (preds == y).sum()/torch.numel(scores)
#         acc = beta*acc + (1-beta)*acc_.item()
#         avg_acc = acc/(1 - beta**i)
        #if loss is massive stop
        if i > 1 and avg_loss > 4 * lowest_loss:
            print(f'from here{i, cost.item()}')
            return log_lrs, losses, accuracies
        if avg_loss < lowest_loss or i == 1:
            lowest_loss = avg_loss

        accuracies.append(acc_.item())
#         accuracies.append(avg_acc)
        losses.append(avg_loss)
        log_lrs.append(lr)
        #step
        cost.backward()
        optimiser.step()
        #update lr
        print(f'cost:{cost.item():.4f}, lr: {lr:.4f}, acc: {acc_.item():.4f}')
        lr *= factor
        optimiser.param_groups[0]['lr'] = lr
    return log_lrs, losses, accuracies

def train(model, optimiser, scheduler = None, epochs = 100, store_every = 25):
    model = model.to(device=device)
    for epoch in range(epochs):
        train_correct_num = 0
        train_total = 0
        train_cost_acum = 0.
        for mb, (x, y) in enumerate(train_loader, start=1):
            model.train()
            x = x.to(device=device, dtype=torch.float32)
            y = y.to(device=device, dtype=torch.long).squeeze(1)
            scores = model(x)
            cost = F.cross_entropy(input=scores, target=y)
            optimiser.zero_grad()
            cost.backward()
            optimiser.step()
```

```
            if scheduler:
                scheduler.step()

            train_predictions = torch.argmax(scores, dim=1)
            train_correct_num += (train_predictions == y).sum()
            train_total += torch.numel(train_predictions)
            train_cost_acum += cost.item()
            if mb%store_every == 0:
                val_cost, val_acc, dice, iou = accuracy(model, val_loader)
                train_acc = float(train_correct_num)/train_total
                train_cost_every = float(train_cost_acum)/mb
                print(f'epoch: {epoch}, mb: {mb}, train cost: {train_cost_every:.4f
                      f'train acc: {train_acc:.4f}, val acc: {val_acc:.4f},'
                      f'dice: {dice}, iou: {iou}')

#Modelo de entrenamiento
torch.manual_seed(2)
model = UNET(3, 10, 2)
optimiser_unet = torch.optim.SGD(model.parameters(),
                                 lr=0.0001, momentum=0.95,
                                 weight_decay=0.001)

lg_lr, losses, accuracies = find_lr(model, optimiser_unet, start_val=1e-5, end_val=

# plot loss vs learning rate
f1, ax1 = plt.subplots(figsize=(20,10))
ax1.plot(lg_lr, losses)
ax1.set_xscale('log')
plt.show()

# define the model and train with scheduler
torch.manual_seed(42)
model = UNET(3, 4, 2)
epochs = 10
optimiser_unet = torch.optim.SGD(model.parameters(),
                                 lr=0.01, momentum=0.95,
                                 weight_decay=1e-4)
scheduler = torch.optim.lr_scheduler.OneCycleLR(optimiser_unet,
                                                max_lr = 1e-1,
                                                steps_per_epoch=len(train_loader),
                                                epochs=epochs, pct_start=0.43, div_
```

```
                                                    three_phase=True)
train(model, optimiser_unet, scheduler, epochs)

#Prueba del modelo
test_set = Colposcopia_Dataset(TEST_PATH, img_transforms=transform_data)
test_loader = DataLoader(test_set, batch_size=BATCH_SIZE, shuffle=True)
imgs_test = next(iter(test_loader))
imgs_test = imgs_test.to(device, dtype=torch.float32)
model = model.to(device)
with torch.no_grad():
    scores = model(imgs_test)
    preds = torch.argmax(scores, dim=1).float()

imgs_test = imgs_test.cpu()
preds = preds.cpu()
print(preds.shape)
plot_mini_bacth(imgs_test, preds.unsqueeze(1))
```

Anexo 10 – Código API

```
from tkinter import *
import tkinter as tk
from tkinter import filedialog
from PIL import Image, ImageTk
from tkinter import ttk
import cv2
import matplotlib.pyplot as plt
import Wathershed
import Kmeans
import Fourier
import Mezclas_gausianas
import rgb
import Otsu
import time
import numpy as np


#raiz es un objeto, el cual contiene el frame y los widget

ruta_imagen = None
```

```
def cargar_imagen():
    global ruta_imagen
    ruta_imagen = filedialog.askopenfilename(initialdir="/", title="Seleccionar ima
    if ruta_imagen:
        #print(ruta_imagen)
        #print(type(ruta_imagen))
        imagen = Image.open(ruta_imagen)
        imagen = imagen.resize( ( 200, 200 ) )
        foto = ImageTk.PhotoImage( imagen )
        img1 = Label( frame_imagen_original, image = foto )
        img1.image = foto
        img1.place(x = 50,y = 70)


def obtener_seleccion( iteraciones = [ ]):
    global ruta_imagen
    seleccion = opcion.get()
    opciones_noSupervisados = [ "Wathershed","K-means","Fourier","RGB","Mezclas gau
    opciones_Supervisados = [ "Segnet","U-net"]
    #print(f"Seleccion actual:{seleccion}")
    print(f"Esta es la ruta actual de la imagen {ruta_imagen}")
    print(type(ruta_imagen))
    if seleccion == 2:
            lista_NoSupervisados = ttk.Combobox(frame_opciones, textvariable = comb
            lista_NoSupervisados.pack()
            lista_NoSupervisados.place( x = 20 , y = 150 )
            seleccion_lista = combo_var.get()
            etiqueta_mensaje.config( text = f"Seleccionaste {seleccion_lista}" )
    else:
            lista_Supervisados = ttk.Combobox(frame_opciones, textvariable = combo_
            lista_Supervisados.pack()
            lista_Supervisados.place( x = 20, y =150 )
            seleccion_lista = combo_var.get()
            etiqueta_mensaje.config( text = f"Seleccionaste {seleccion_lista}" )
    iteraciones.append(seleccion_lista)
    print(f"Metodo: {seleccion_lista}")
    print(iteraciones)
    if len(iteraciones) > 1:
        if iteraciones[-1] == iteraciones[-2]:
            del iteraciones[-1]
            print(iteraciones)
    print("------------------------------------------------------------------")
```

```
if iteraciones[-1] == "Wathershed":
    Img_segmentada_watershed = Wathershed.watershed(ruta_imagen)
    print(type(Img_segmentada_watershed))
    f,c,p = Img_segmentada_watershed.shape
    print(f,c,p)
    img_pil = Image.fromarray(Img_segmentada_watershed)
    img_pil = img_pil.resize( ( 200, 200 ) )
    img_tk = ImageTk.PhotoImage(img_pil)
    etiqueta = Label( frame_imagen_segmentada, image = img_tk )
    etiqueta.pack()
    #etiqueta.config( image = img_tk )
    etiqueta.image = img_tk
    etiqueta.place(x = 50,y = 70)
if iteraciones[-1] == "K-means":
    Img_segmentada_kmeans = Kmeans.KMEANS(ruta_imagen)
    img_pil = Image.fromarray(Img_segmentada_kmeans)
    img_pil = img_pil.resize( ( 200, 200 ) )
    img_tk = ImageTk.PhotoImage(img_pil)
    etiqueta = Label( frame_imagen_segmentada, image = img_tk )
    etiqueta.pack()
    etiqueta.image = img_tk
    etiqueta.place(x = 50,y = 70)
if iteraciones[-1] =="Fourier":
    Img_segmentada_fourier = Fourier.FOURIER(ruta_imagen)
    img_pil = Image.fromarray(Img_segmentada_fourier)
    img_pil = img_pil.resize( ( 200, 200 ) )
    img_tk = ImageTk.PhotoImage(img_pil)
    etiqueta = Label( frame_imagen_segmentada, image = img_tk )
    etiqueta.pack()
    etiqueta.image = img_tk
    etiqueta.place(x = 50,y = 70)
if iteraciones[-1] == "RGB":
    Img_segmentada_RGB = rgb.RGB(ruta_imagen)
    img_pil = Image.fromarray(Img_segmentada_RGB)
    img_pil = img_pil.resize( ( 200, 200 ) )
    img_tk = ImageTk.PhotoImage(img_pil)
    etiqueta = Label( frame_imagen_segmentada, image = img_tk )
    etiqueta.pack()
    etiqueta.image = img_tk
    etiqueta.place(x = 50,y = 70)
if iteraciones[-1] == "Mezclas gausianas":
```

```
        Img_segmentada_Mezclas_gausianas = Mezclas_gausianas.M_Gausianas(ruta_image
        img_pil = Image.fromarray(Img_segmentada_Mezclas_gausianas)
        img_pil = img_pil.resize( ( 200, 200 ) )
        img_tk = ImageTk.PhotoImage(img_pil)
        etiqueta = Label( frame_imagen_segmentada, image = img_tk )
        etiqueta.pack()
        etiqueta.image = img_tk
        etiqueta.place(x = 50,y = 70)
    if iteraciones[-1] =="Rango Umbral":
        Img_segmentada_Rango_umbral = Otsu.RangoUmbral(ruta_imagen)
        img_pil = Image.fromarray(Img_segmentada_Rango_umbral)
        img_pil = img_pil.resize( ( 200, 200 ) )
        img_tk = ImageTk.PhotoImage(img_pil)
        etiqueta = Label( frame_imagen_segmentada, image = img_tk )
        etiqueta.pack()
        etiqueta.image = img_tk
        etiqueta.place(x = 50,y = 70)
    if iteraciones[-1] =="Otsu":
        Img_segmentada_otsu = Otsu.Otsu(ruta_imagen)
        img_pil = Image.fromarray(Img_segmentada_otsu)
        img_pil = img_pil.resize( ( 200, 200 ) )
        img_tk = ImageTk.PhotoImage(img_pil)
        etiqueta = Label( frame_imagen_segmentada, image = img_tk )
        etiqueta.pack()
        etiqueta.image = img_tk
        etiqueta.place(x = 50,y = 70)



    return seleccion_lista

raiz = Tk()

#Opciones del menu desplegable
imagen_de_fondo = Image.open("C:/Users/Santiago H/OneDrive/Escritorio/Aplicacion/im
imagen_de_fondo = ImageTk.PhotoImage(imagen_de_fondo)
fondo_label = tk.Label(raiz, image=imagen_de_fondo)
fondo_label.place(x=0, y=0, relwidth=1, relheight=1)


logo_javeriana = Image.open("C:/Users/Santiago H/OneDrive/Escritorio/Aplicacion/PUJ
nuevo_ancho = 110
```

```
nuevo_alto = 110
imagen_redimensionada = logo_javeriana.resize((nuevo_ancho, nuevo_alto), Image.ANTI
logo_javeriana = ImageTk.PhotoImage(imagen_redimensionada)
etiqueta = tk.Label(raiz, image=logo_javeriana)
etiqueta.place(x = 0, y = 5, anchor="nw")

raiz.title("CITOBOT")
raiz.resizable(False,False)
raiz.geometry("800x600")
frame_imagen_original = Frame()
frame_imagen_segmentada = Frame()


color_rgb = (134, 207, 190)
color_hex = "#{:02x}{:02x}{:02x}".format(color_rgb[0], color_rgb[1], color_rgb[2])
raiz.configure(bg=color_hex)

label1 = Label( frame_imagen_original, text = "Imagen original",font=("Georgia"))
label1.place(x =75 ,y = 10)

label2 = Label(frame_imagen_segmentada, text = "Imagen segmentada",font = ("Georgia
label2.place(x = 75, y = 10)

#Frame de color rosado con dimensiones de 300 * 300, el frame debe empaquetarce en
frame_imagen_original.pack( side = "left" )
frame_imagen_original.config(bg = "lightgray")
frame_imagen_original.config(width = "300",height = "350")

#Frame azul para imagenes segmentadas
frame_imagen_segmentada.pack( side = "right" )
frame_imagen_segmentada.config( bg ="lightgray" )
frame_imagen_segmentada.config(width = "300",height = "350")

#Boton para cargar la imagen al frame de imagen original
boton_cargar_imagen = Button(raiz, text = "Cargar imagen",command = cargar_imagen)
boton_cargar_imagen.pack()
boton_cargar_imagen.place(x = 100, y = 500)

#Frame para las opciones supervisado o no supervisado
frame_opciones = Frame(raiz,bg = "white")
frame_opciones.place(relx=0.5, rely=0.5, relwidth=0.23, relheight=0.5,anchor = "cen
```

```
metodos = Label( frame_opciones, text = "  Qu   m todo desea emplear?" ,font = ("G
metodos.place( x = 0 ,y = 0)

opcion = IntVar()
opcion_1 = Radiobutton(frame_opciones, text = " Supervisados ", variable = opcion,
value = 1)
opcion_1.pack()
opcion_1.place( x =40 ,y =50 )
opcion_2 = Radiobutton(frame_opciones, text = " No supervisados ", variable = opcio
value = 2)
opcion_2.pack()
opcion_2.place( x = 40,y = 77)

#Variable para almacenar los valor del combobox
combo_var = StringVar()

# Boton para obtener valor de la variable "opcion"
valor_opcion = Button(frame_opciones, text = "Cargar M todos", command = obtener_s
valor_opcion.pack()
valor_opcion.place( x =40 ,y =110 )

#Boton para mostrar el metodo seleccionado
seleccion_actual = Button(frame_opciones, text = "Aceptar", command = obtener_selec
seleccion_actual.pack()
seleccion_actual.place( x = 20, y = 180 )

#Etiqueta posterior al boton acaeptar
etiqueta_mensaje = Label(frame_opciones, text="")
etiqueta_mensaje.pack()
etiqueta_mensaje.place( x = 20, y = 210 )

raiz.mainloop()
```

Bibliografía

[1] Centro internacional de investigaciones sobre el cáncer, oganización mundial de la salud, cancer today. [Online]. Available: https://gco.iarc.fr/today/online-analysis-multi-bars?v=2020&mode=cancer&mode_population=countries&population=900&populations=170&key=total&sex=2&cancer=39&type=0&statistic=5&prevalence=0&population_group=0&ages_group%5B%5D=0&ages_group%5B%5D=17&nb_items=10&group_cancer=0&include_nmsc=0&include_nmsc_other=1&type_multiple=%257B%2522inc%2522%253Afalse%252C%2522mort%2522%253Atrue%252C%2522prev%2522%253Afalse%257D&orientation=horizontal&type_sort=0&type_nb_items=%257B%2522top%2522%253Atrue%252C%2522bottom%2522%253Afalse%257D#collapse-others

[2] S. learn. Clustering.

[3] J. Zhang. (2019) Unet — line by line explanation.

[4] R. G. K. H. B. H. S. B. Tsung-Yi Lin, Piotr Dollar, "Feature pyramid networks for object detection," *Computer Vision Foundation*, p. 2117–2121, 2017.

[5] D. V. A. S. Saul Olivera, Marta Baguer, "Segmentacion de la región acetoblanca en colposcopías del cervix/ acetowhite region segmentation in cervix colposcopies," *Ciencias Matematicas*, vol. 31, no. 1, pp. 171–181, 2017.

[6] P. L. J. R. J. Guerrero, J. Rivas, "Red neuronal artificial para detectar lesiones precancerosas en el cuello uterino," *Revista INGENIERÍA UC*, vol. 15, no. 1, pp. 14–27, 2008.

[7] R. T. S. J. I. P. B. S. O.-A. Mario Alejandro Bravo Ortíz, Harold Brayan Arteaga Arteaga, "Clasificación de cáncer cervical usando redes neuronales convolucionales, transferencia de aprendizaje y aumento de datos," *RevistaEIA*, vol. 18, no. 35, pp. 1–12, 2021.

[8] K. D. A. P. K. A. C. A. R. L. M. S. S. A. Zhiyun Xue, Peng Guo, "A deep clustering method for analyzing uterine cervix images across imaging devices," *2021 IEEE 34th International Symposium on Computer-Based Medical Systems (CBMS)*, 2021.

[9] M. E. G. Mayorga, "Implementacion de metodos para reconocimiento de imagenes para el diagnostico de cancer cervicouterino," Maestría, Universidad Autonoma del Estado de México, 2016.

[10] SociedadAmericanaDeCáncer. (2022) Prueba de vph. [Online]. Available: https://www.cancer.org/es/cancer/cancer-de-cuello-uterino/deteccion-diagnostico-clasificacion-por-etapas/pruebas-de-deteccion/prueba-de-vph.html

[11] InstitutoNacionalDelCáncer. Colposcopia. [Online]. Available: https://www.cancer.gov/espanol/publicaciones/diccionarios/diccionario-cancer/def/colposcopia

[12] MayoClinic. (2022) Infección por vph - sintomas y causas - mayo clinic. [Online]. Available: https://www.mayoclinic.org/es-es/diseases-conditions/hpv-infection/symptoms-causes/syc-20351596#:~:text=Los%20factores%20de%20riesgo%20para%20la%20infecci%C3%B3n%20por,VPH%2C%20como%20duchas%20p%C3%BAblicas%20o%20piscinas%2C%20podr%C3%ADa%20

[13] GobiernoDeMéxico. (2022) Cáncer de cuello uterino. [Online]. Available: https://www.gob.mx/salud/acciones-y-programas/cancer-de-cuello-uterino#:~:text=El%20c%C3%A1ncer%20de%20cuello%20uterino%20es%20la%20segunda,Paraguay%2C%20Guyana%2C%20Bolivia%2C%20Honduras%2C%20Venezuela%2C%20Nicaragua%20y%20Surinam.

[14] OrganizaciónPanamericanaDeLaSalud. (01 Febrero 2019) El cáncer cervicouterino es el tercero más frecuente entre las mujeres de américa latina y caribe, pero se puede prevenir. [Online]. Available: https://www3.paho.org/hq/index.php?option=com_content&view=article&id=14947:cervical-cancer-is-the-third-most-common-cancer-among-women-in-latin-america-and-the-caribbean-but-it-can-be-prevented&Itemid=1926&lang=es#gsc.tab=0

[15] *MINISTERIO DE LA PROTECCIÓNSOCIAL INSTITUTONACIONAL DE CANCEROLOGÍA. Recomendaciones para la tamización de neoplasias del cuello uterino en mujeres sin antecedentes de patología cervical (preinvasora o invasora) en Colombia*, 2007.

[16] MinisterioDeSalud. (2007) Ministerio de la protección social instituto nacional de cancerología. [Online]. Available: https://minsalud.gov.co/sites/rid/Lists/BibliotecaDigital/RIDE/IA/INCA/Guia-tamizacion-cuello-uterino.pdf.

[17] L. Bailey. (Abril 2022) Comprender los resultados anormales en las pruebas de detección del cáncer cervical. [Online]. Available: https://es.familydoctor.org/comprender-los-resultados-anormales-en-las-pruebas-de-deteccion-del-cancer-cervical/

[18] RedJurista. (18 Mayo 2016) ¿cuánto tiempo se puede demorar una eps en asignar una cita con un médico general y un especialista? [Online]. Available: https://www.redjurista.com/NewsPaper/35/salud/1011/cuanto-tiempo-se-puede-demorar-una-eps-en-asignar-una-cita-con-un-medico-general-y-un-especialista

[19] SuperintendenciaNacionalDeSalud, "Indicadores de oportunidad reportados por las entidades administradoras de planes de beneficios a la superintendencia nacional de salud (sns) durante la vigencia 2012-2013. seguimiento al cumplimiento del decreto 019 de 2012." [Online]. Available: https://www.minsalud.gov.co/sites/rid/Lists/BibliotecaDigital/RIDE/IA/SSA/Articulo%208.pdf

[20] R. T. L. Fernández, "Ruido y calidad en imágenes médicas," Pregrado, Facultad de ingeniería eléctrica Departamento de telecomunicaciones y electrónica. Universidad central Marta Abreu de las villas, 2008.

[21] M. T. V. Francisco Ochoa, Diana Guarneros, "Infección por virus del papiloma humano en mujeres y su prevención," *Gaceta Mexicana de Oncología*, vol. 14, no. 3, p. 157–163, 2015.

[22] H. G. A. Mesa, "Sensibilidad y especificidad de la colposcopía," *Medicgraphic*, vol. 2, no. 3, 2010.

[23] OrganzaciónMundialDeLaSalud. Cáncer cervicouterino. [Online]. Available: https://www.who.int/es/news-room/fact-sheets/detail/cervical-cancer#:~:text=El%20c%C3%A1ncer%20de%20cuello%20uterino,bajos%20y%20medianos%20(1).

[24] InstitutoNacionalDelCáncer. Detección del cáncer de cuello uterino. [Online]. Available: https://www.cancer.gov/espanol/tipos/cuello-uterino/deteccion#:~:text=Se%20introduce%20un%20esp%C3%A9culo%20en,de%20VPH%20que%20causan%20c%C3%A1ncer.

[25] Centro internacional de investigaciones sobre el cáncer, oganización mundial de la salud, bases de los procedimientos para el examen colposcópico. [Online]. Available: https://screening.iarc.fr/colpochap.php?lang=3&chap=4#:~:text=El%20ingrediente%20clave%20en%20la,coagula%20y%20despeja%20el%20moco.

[26] *Organización panamericana de la salud, SECCIÓN 6: PROCEDIMIENTOS PARA LA TOMA DE LA MUESTRA Y SU ENVÍO AL LABORATORIO.* PAHO, ch. 6.

[27] MayoClinic. Prueba del vph, riesgos. [Online]. Available: https://www.mayoclinic.org/es-es/tests-procedures/hpv-test/about/pac-20394355#:~:text=Un%20resultado%20falso%20negativo%20significa,o%20en%20los%20procedimientos%20adecuados.

[28] J. G. Fabián Andrés Giraldo, Elizabeth León, "Caracterización de flujos de datos usando algoritmos de agrupamiento," *Revista Recnura*, vol. 17, no. 37, p. 153, 2013.

[29] L. G. P. Huamán. (2020) Estudio de la segmentación semántica para la navegación autónoma de un vehículo que circula en las calles de la provincia de huamanga.

[30] R. S. C. Rocamora. (2019) Navegación y conducción autónoma de vehículos con geometría ackermann.

[31] H. R. L. y. P. B. A. Mauricio Barrios Barrios, "Segmentación de imágenes de tomografía computarizada en pacientes que han sufrido un accidente cerebro vascular," *Prospect*, vol. 9, no. 2, pp. 48–53, 2011.

[32] K. A. y. G. H. Guo-Qing Wei, "Real-time visual servoing for laparoscopic surgery. controlling robot motion with color image segmentation," *IEEE*, vol. 16, no. 1, pp. 40–45, 1997.

[33] InstitutoNacionalDelCáncer. Cuello uterino. [Online]. Available: https://www.cancer.gov/espanol/publicaciones/diccionarios/diccionario-cancer/def/cuello-uterino

[34] OrganizacionPanamericadaDeLaSalud. (18 Diciembre 2018) Virus del papiloma humano (vph). [Online]. Available: https://www3.paho.org/hq/index.php?option=com_content&view=article&id=14873:sti-human-papilloma-virus-hpv&Itemid=3670&lang=es#gsc.tab=0

[35] D. R. José Luis Sagarduy, Blanca Estela Pérez, "Vista de conocimiento y creencias sobre la prueba de papanicolaou en estudiantes universitarios," *Psicología y salud*, vol. 22, no. 2, 2012.

[36] InstitutoNacionalDelCáncer. ¿qué es cáncer de cuello uterino (cervical)? [Online]. Available: https://www.cancer.org/es/cancer/cancer-de-cuello-uterino/acerca/que-es-cancer-de-cuello-uterino.html#:~:text=El%20exoc%C3%A9rvix%20(o%20ectoc%C3%A9rvix)%20es,est%C3%A1%20cubierto%20de%20c%C3%A9lulas%20escamosas

[37] A. Z. I. R. M. S. B. R. M. . E. L. G. Y. Schmolling Guinovart, J.J. Barquín Solera, "Cell anomalies in the cervix and subsequent pre-cancerous lesions in a health area," *Aten primaria*, vol. 29, no. 4, pp. 223–229, 2002.

[38] F. Skoda, P. Adam, *Knowledge Discovery in Big Data from Astronomy and Earth Observation: Astrogeoinformatics. En Chapter 12 - Learning in Big Data: Introduction to Machine Learning.* Elsevier Gezondheidszorg, 2020.

[39] D. O. M. Thomas W. Edgar, *Research Methods for Cyber Security.* Elsevier Gezondheidszorg, 2017.

[40] S. Hurson, A. R. Wu, *AI and Cloud Computing*, 1st ed. Academic Press, 2021, vol. 127.

[41] G. H. Yann LeCun, Yoshua Bengio, "Deep learning," *Nature*, vol. 521, p. 436–444, 2015.

[42] A. K. V. K. Pang Ning Tan, Michael Steinbach, *Introduction data mining.* Addison-Wesley Companion Book, 2006.

[43] E. M. E. L. Santiago Romaní, Pilar Sobrevilla, "CaracterizaciÓn del color del citoplasma en imÁgenes de citologia cervico-vaginal," *ESTYLF*, 2010.

[44] D. K. M. A. C. J. G. L. H. N. M. Teutsch, "Fully convolutional region proposal networks for multispectral person detection. 2017 ieee conference on computer vision and pattern recognition workshops (cvprw)," *IEEE*, 2017.

[45] J. Y. Dongju Liu, "Otsu method and k-means," *IEEE*, 2009.

[46] Z. M. Z. W. I. Muhamad Rizal Mohamed razali, Nazatul Sabariah Ahmad, "Region of adaptive threshold segmentation between mean, median and otsu threshold for dental age assessment," *IEEE*, 2014.

[47] "Sistema de enseñanza para la técnica de agrupamiento k-means," *Universidad Autonoma de Hidalgo*, 2021.

[48] "Segmentación de objetos basada en el modelo de mezcla gaussiana y campos aleatorios condicionales," *IEE*, pp. 900–904, 2016.

[49] "SegmentaciÓn de imÁgenes a color basada en el algoritmo de grabcut," *Universidad central de Venenzuela*, 2012.

[50] "Método de aprendizaje no supervisado para la segmentación de plantas y hojas," *IEEE*, 2017.

[51] "Unsupervised rgb-d image segmentation by multi-layer clustering," *IEEE*, 2017.

[52] "Watershed: un algoritmo eficiente y flexible para segmentación de imágenes de geles 2-de," *Universidad Nacional Mayor de San Marcos*, 2010.

[53] G. T. Y. L. Juntao Jiang, Xiyu Chen, "Vig-unet: Vision graph neural networks for medical image segmentation," *IEEE*, 2023.

[54] V. C. S. S. Aswathy A. L, "Cascaded 3d unet architecture for segmenting the covid-19 infection from lung ct volume," *Scientific Reports*, 2022.

[55] Q. V. Mingxing Tan, Ruoming Pang, "Efficientdet: Scalable and efficient object detection," *Computer Vision Foundation*, p. 10781–10785, 2020.

[56] "Generalized intersection over union: A metric and a loss for bounding box regression," *IEEE*, 2019.

[57] "Generalized overlap measures for evaluation and validation in medical image analysis," *IEEE*, 2006.

[58] "¿qué es el accuracy?" 2019.

[59] "Métricas de evaluación de modelos en el aprendizaje automático," 2023.

[60] O. P. Gustavo Lorca, José Arzola, "Segmentación de imágenes médicas digitales mediante técnicas de clustering," *Aporte Santiaguino*, vol. 3, no. 1, pp. 108–116, 2010.

[61] M. E. G. Mayorga, "Detección de cáncer cérvico-uterino mediante red neuronal función de base radial," Maestría, Instituto Politécnico Nacional México, 2009.