UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE CUENCA

CARRERA DE ELECTRÓNICA Y AUTOMATIZACIÓN

CLASIFICADOR DE CÉLULAS SANGUÍNEAS USANDO PROCESAMIENTO DE IMÁGENES

Trabajo de titulación previo a la obtención del
título de Ingeniero en Electrónica y Automatización

AUTOR: EDWIN ADRIAN TENEN QUIZHPI
ERIKA PATRICIA ALVARADO ANDRADE

TUTOR: ING. JUAN PAUL INGA ORTEGA, MsC

Cuenca – Ecuador
2023

CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN

Nosotros, Edwin Adrián Tenén Quizhpi con documento de identificación N° 0106629066 y Erika Patricia Alvarado Andrade con documento de identificación N° 0150287472; manifestamos que:

Somos los autores y responsables del presente trabajo; y, autorizamos a que sin fines de lucro la Universidad Politécnica Salesiana pueda usar, difundir, reproducir o publicar de manera total o parcial el presente trabajo de titulación.

Cuenca, 30 de agosto del 2023

Atentamente,

Edwin Adrián Tenén Quizhpi

0106629066

Erika Patricia Alvarado Andrade

0150287472

CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE TITULACIÓN A LA UNIVERSIDAD POLITÉCNICA SALESIANA

Nosotros, Edwin Adrián Tenén Quizhpi con documento de identificación N° 0106629066 y Erika Patricia Alvarado Andrade con documento de identificación N° 0150287472, expresamos nuestra voluntad y por medio del presente documento cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del Proyecto técnico: "Clasificador de células sanguíneas usando procesamiento de imágenes", el cual ha sido desarrollado para optar por el título de: Ingeniero en Electrónica y Automatización, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En concordancia con lo manifestado, suscribimos este documento en el momento que hacemos la entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Cuenca, 30 de agosto del 2023

Atentamente,

Edwin Adrián Tenén Quizhpi

0106629066

Erika Patricia Alvarado Andrade

0150287472

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN

Yo, Juan Paúl Inga Ortega con documento de identificación N° 0104166491, docente de la Universidad Politécnica Salesiana, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: CLASIFICADOR DE CÉLULAS SANGUÍNEAS USANDO PROCESAMIENTO DE IMÁGENES, realizado por Edwin Adrián Tenén Quizhpi con documento de identificación N° 0106629066 y Erika Patricia Alvarado Andrade con documento de identificación N° 0150287472, obteniendo como resultado final el trabajo de titulación bajo la opción Proyecto técnico que cumple con todos los requisitos determinados por la Universidad Politécnica Salesiana.

Cuenca, 30 de agosto del 2023

Atentamente,

Ing. Juan Paúl Inga Ortega
0104166491

AGRADECIMIENTOS

Agradecimiento de Adrián Tenen

Quiero expresar mi sincero agradecimiento a todas las personas que han contribuido de manera significativa en la realización de esta tesis. En primer lugar, deseo reconocer y expresar mi gratitud al Ing. Juan Inga, por su dedicación, paciencia, apoyo y compromiso que han sido fundamentales para la culminación exitosa de este proyecto Además, quiero agradecer profundamente a mi familia por el apoyo inquebrantable. Su amor y aliento han sido un faro de luz a lo largo de este viaje académico.

A la coautora de esta tesis, Erika, quien compartió conmigo este camino y contribuyó de manera significativa en la investigación y redacción de esta tesis, te agradezco por tu compromiso y paciencia.

Agradecimiento de Erika Alvarado

En primer lugar, quiero expresar mi más sincero agradecimiento a Dios por haberme dado la fuerza, la sabiduría y la perseverancia necesaria para llevar a cabo este trabajo de titulación. También quiero agradecer al Ing. Juan Inga, tutor de este trabajo, por su valiosa orientación y apoyo en este proceso. Sus consejos y dirección fueron esenciales para dar forma a esta investigación. Agradezco a mi familia por su amor y apoyo. Su fe en mí han sido un faro en los momentos difíciles.

No puedo dejar de expresar mi profunda gratitud hacia el coautor de este trabajo quien ha sido mi apoyo constante, tanto en mi vida personal como en mi recorrido académico. Su inquebrantable apoyo, colaboración y amor han sido pilares fundamentales en cada etapa de esta investigación.

DEDICATORIAS

Dedicatoria de Adrian Tenen

A mi amada familia, quienes han sido mi apoyo inquebrantable a lo largo de este emocionante viaje académico. Sus sacrificios, amor y aliento constante han sido la brújula que me ha guiado hasta este punto.

A mi novia Erika, mi compañera en este desafiante proyecto de tesis. Tu dedicación, inteligencia y compromiso con esta investigación han sido un faro de inspiración para mí. Juntos hemos enfrentado obstáculos, celebrado victorias y compartido momentos inolvidables a lo largo de este recorrido.

Dedicatoria de Erika Alvarado

Dedico este trabajo de titulación a mis padres, quienes han sido mi apoyo a lo largo de mi vida. Su amor y sacrificio constante me han impulsado a superar desafíos y alcanzar mis metas. A mi familia y amigos, por su paciencia, comprensión y palabras de aliento durante este viaje académico. A mis profesores y mentores, por compartir su sabiduría y guiarme en el camino del aprendizaje.

Dedico este trabajo de titulación de manera especial a Adrián, quien no solo es mi compañero en el desarrollo de esta tesis, sino también en el hermoso viaje de la vida. Por su apoyo constante, dedicación y contribuciones han dejado una huella profunda en cada página de este trabajo. Juntos hemos superado desafíos, celebrado logros y compartido el amor por el aprendizaje y la investigación

Índice general

Índice de figuras

Índice de tablas

Resumen

Este trabajo está enfocado a la evaluación de técnicas para procesamiento de imágenes. Se realizó una revisión de alrededor de 25 técnicas diferentes con el fin de seleccionar las más adecuadas para conteo y clasificación de células.

De esta manera, se aplicó un análisis basado en dos enfoques: el primero desarrollado desde la configuración de la red neuronal convolucional personalizada y un segundo enfoque empleando una red neuronal basada en YOLO v5. De esta forma, el primer enfoque busca identificar y clasificar células sanguíneas y el segundo enfoque se empleó para realizar un conteo de células.

Luego de un análisis comparativo, se determinó que la combinación de ambas técnicas ofrecía un sistema integral que permitía tanto el conteo preciso de células sanguíneas como la clasificación efectiva de glóbulos blancos en una sola imagen.

Entonces, para evaluar el desempeño de las metodologías propuestas, se creó una interfaz visual que simplifica el uso del sistema para el usuario. Esta interfaz permitió realizar pruebas de funcionamiento utilizando diversos bancos de imágenes y evaluar la precisión de los resultados obtenidos.

Por tanto, este trabajo muestra una combinación efectiva de técnicas para el tratamiento de imágenes para el conteo e identificación de clases de células sanguíneas, demostrando su utilidad potencial en aplicaciones médicas y de diagnóstico.

Palabras clave: Células Sanguíneas Humanas; Conteo y Clasificación; Python; Redes Neuronales Convolucionales; Técnicas de procesamiento de imágenes; YOLO v5;

Abstract

The central theme of this work is the application of image processing techniques. A review of about 25 different techniques waas conducted to identify the most appropriate ones for cell counting and classification.

In this way, an examination was performed using two separate methods: the he primary one developed from the construction of the personalized convolutional neural network and a second approach using a convolutional neural network based on YOLO v5. Thus, the first approach seeks to identify and classify blood cells. The second one was used to perform cell counting.

Following a comparative analysis, it was established that the fusion of both methods offered a comprehensive system that allowed both accurate blood cell counting and effective white blood cell sorting in a single image.

Then, to assess the effectiveness of the proposed methodologies, a visual interface was created to streamline the utilization of the system for the user. This interface allowed to perform performance tests using various image banks and o assess the precision of the obtained results.

Thus, this work shows an effective combination of techniques for image analysis for counting and classification of blood cells, demonstrating its potential usefulness in medical and diagnostic applications.

Keywords: Convolutional Neuronal Networks; Counting and Classification; Human Blood Cells; Image Processing; Python; YOLO V5;

Antecedentes

La clasificación de las células sanguíneas es un tema con gran importancia en la medicina debido a que los análisis de sangre pueden detectar diversas enfermedades y establecer el tratamiento correcto [1]. En muchos de los casos la clasificación de las células aún se realiza de forma manual [2], siendo este método lento e ineficiente. Existen varios métodos de clasificación manual de las células sanguíneas como la citometría del flujo y el recuento de células haciendo uso de la rejilla del hemocitómetro, que son usados en la actualidad, que a pesar de ser los más confiables, son muy poco precisos [2].

Una herramienta para aumentar la precisión y la eficiencia en la clasificación de células es la automatización del proceso que tiene como objetivo ahorrar tiempo, reducir errores, aumentar la eficiencia y la precisión, y en consecuencia el correcto diagnóstico médico. El uso del procesamiento de imágenes basado en redes neuronales artificiales es una de las principales soluciones. Autores de varias partes del mundo han tomado esta técnica para dar solución a este problema [3].

De acuerdo con lo mencionado con anterioridad, hay diversos trabajos relacionados con la automatización de procesos de clasificación de las células sanguíneas a través del procesamiento de imágenes, por ejemplo en [4] se propone la doble convolución. Este trabajo propone realizar una estructura conectada con dos capas, una oculta y otra de salida. Esta capa convolucional obtiene características de la imagen de entrada con el uso de la matriz Kernel, dando como resultado una imagen convolucionada y mediante esto se puede encontrar ciertos patrones en las imágenes o características para mejorar la clasificación de los glóbulos blancos.

Khashman en su artículo[3] menciona la posibilidad de uso de un filtro Wavelet en el procesamiento de imágenes. Usa el método Otsu para un prepresamiento de

imágenes, esta forma es la más eficaz a la hora de procesar imágenes, sin embargo, necesita ejecutarse en tiempo real y realizar el cálculo de umbral, es un tanto complejo. También se menciona un método de eliminación del fondo del microscopio, y mediante la segmentación identificar las distintas células. Para la clasificación utiliza una red neuronal que identifica patrones globales. Este método identifica características de rotación, tamaño y morfologías de cada célula.

En el caso del trabajo de Ammar [5], se usa una base de datos públicos con aproximadamente, 12400 imágenes, mientras que el segundo contiene alrededor de 17.000 imágenes, este grupo de datos se utilizó para hallar la mejor combinación entre algoritmos de aprendizaje y CNN)). En este trabajo proponen un modelo híbrido basado en la obtención de características haciendo uso de las CNN y AdaboostM1 que es un algoritmo para la toma de decisiones, entrenando las redes para luego ser probadas por varios modelos con los datos iniciales.

En los trabajos de Yuanyuan Xu [6] y Abel Tessema [7], proponen un preprocesamiento de imágenes. Xu plantea un análisis de imágenes de fase en coordenadas polares utilizando el software MATLAB. La red neuronal que usó es LeNet-5, realiza la clasificación de los tipos de núcleos de las células. Tessema, plantea la clasificación automática basada en el análisis cuantitativo de células sanguíneas utilizando como modelo la red neuronal YOLOv2. El preprocesamiento de imagen lo realiza por pasos, mientras la red se entrena la imagen de entrada se procesa para tener una mejor resolución. Realiza el análisis de los parámetros morfológicos y se hace el cálculo del recuento. Finalmente, se realiza ya la detección y clasificación de células.

Entre otros trabajos, se identifica la tesis de maestría en ciencias computacionales realizada por Martha Coral [8] que realiza un preprocesamiento para detectar los bordes de las células y de esta manera obtiene las características geométricas, de textura, etc. Estas características fueron usadas como datos iniciales para el proceso de análisis de datos para reconocer diferentes células sanguíneas.

En el caso de buscar pre-diagnósticos, se identifica la tesis de Jiménez [9], trabajo que implementa un proceso de clasificación de patologías en eritrocitos destinado al diagnóstico de enfermedades usando el reconocimiento de patrones en

imágenes y el cual se realizó un modelo incremental, para el desarrollo del proyecto hace uso del softaware Python el cual ofrece varias librerías para el manejo de imágenes y su procesamiento.

El software de programación usado varía con cada artículo, sin embargo, de acuerdo a la investigación el mejor software para el uso de redes neuronales y clasificadores es Python, Tessema hace uso de este software en su trabajo, siendo muy eficaz. En nuestro proyecto se usara Python en su versión más reciente, ya que ofrece nuevas librerías que facilitarían el proceso de clasificación de células sanguíneas.

Justificación

La clasificación de células sanguíneas presenta algunos desafíos y problemas, generando un bajo porcentaje de precisión y exactitud. Las células sanguíneas pueden tener diferentes formas, tamaños y características, lo que puede dificultar la identificación de algunos tipos de células. Además, la variabilidad en la apariencia celular puede aumentar cuando las células están afectadas por alguna enfermedad o trastorno. En una muestra de sangre, puede haber otros componentes que pueden interferir con la clasificación de células. Por ejemplo, las fibras y las proteínas pueden dificultar la visualización de algunas células y hacer que se confundan con otras. La clasificación de células sanguíneas a menudo implica el uso de microscopios y técnicas de tinción que requieren habilidades técnicas y experiencia. Por lo tanto, los errores humanos pueden ocurrir durante el proceso de clasificación, lo que puede afectar la precisión del diagnóstico. Aunque la tecnología ha mejorado significativamente la capacidad de clasificar células sanguíneas, todavía existen algunas limitaciones. Por ejemplo, algunos tipos de células pueden ser difíciles de identificar mediante técnicas de tinción convencionales, lo que puede hacer que se pierdan en la clasificación.

En general, aunque la clasificación manual de células sanguíneas es una herramienta útil en el diagnóstico médico, también tiene sus limitaciones y puede presentar algunos desafíos y problemas. Los problemas de la clasificación de células se pueden superar mediante la creación de un sistema automático que pueda clasificar las células sanguíneas humanas utilizando el método de procesamiento de imágenes, el cual es posible con el uso de redes neuronales.

En este sentido, la empresa Narváez Soluciones Integrales (NSI), es una empresa privada con sede en Ecuador que ofrece servicios de comercialización y mantenimiento de equipos médicos, ha buscado establecer un convenio con

la Universidad Politécnica Salesiana, para plantear un sistema automático de clasificación de células sanguíneas, el requerimiento de este sistema radica en la necesidad de obtener una mayor exactitud y precisión en sus resultados en análisis de células sanguíneas y en consecuencia los diagnósticos médicos sean los correctos.

Este sistema de clasificación ha sido ya desarrollado en varias partes del mundo y varios de estos proyectos están publicados en diferentes bases de datos. Al analizar algunos artículos publicados se encuentran autores como Tiwari [4] y Ammar [5] los cuales mediante el uso de diferentes redes neuronales y con bases de datos públicas realizan la clasificación de células sanguíneas mediante procesamiento de imágenes con una precisión entre el 85 y 90 %, lo cual evidencia una mejora en comparación a una clasificación manual. Tessema [7] y Xu [6] en sus artículos, siendo los más actuales, mencionan que al tener una base de datos pública, las imágenes no son de la mejor calidad, siendo un riesgo al momento de realizar la clasificación, por esto, previo a la clasificación, realiza una preparación previa de las imágenes para abordar cuestiones relacionadas con la resolución, la iluminación, y otros aspectos similares que pueden afectar la clasificación de células. Sin embargo, de todos los artículos revisados, muy pocos usan el preprocesamiento de imágenes y existen muy pocos elaborados en Latinoamérica.

Dado lo anteriormente expuesto, este trabajo consiste en diseñar un sistema clasificador de células sanguíneas a través de procesamiento de imágenes.

La importancia de este trabajo es elevar la precisión en el proceso de identificación de clases de células sanguíneas, lo cual tiene un impacto directo en el diagnóstico médico. Entonces, este proyecto busca abordar ciertas limitaciones y desafíos específicos que han sido identificados en la literatura.

En primer lugar, a diferencia de algunos proyectos previos que se basan en bases de datos públicas con imágenes de calidad variable, este trabajo reconoce la importancia de abordar problemas relacionados con la resolución, la iluminación y otros aspectos que pueden afectar la clasificación de células. Esto implica un enfoque en la preparación previa de las imágenes para asegurar la calidad de la información, lo que ayudaría a aumentar la precisión del sistema.

Otra distinción importante de este trabajo con respecto a otros proyectos

similares es que no se limita a utilizar una sola técnica de clasificación, sino que combina dos técnicas diferentes para obtener un clasificador más robusto y efectivo. Esta combinación de técnicas presenta un enfoque más completo en la resolución del problema de clasificación. Cada técnica tienen sus propias ventajas y limitaciones, y la fusión de ellas ayuda a compensar las deficiencias individuales de cada una.

Además de las distinciones previamente mencionadas, este proyecto se destaca de otros por la inclusión de una interfaz gráfica, lo que proporciona una ventaja significativa en términos de accesibilidad y usabilidad para los profesionales médicos. Una ventaja de tener una interfaz gráfica, es que facilita el proceso de captura de datos y exhibición de resultados. Esto se traduce en un ahorro de tiempo significativo para el personal médico, quienes pueden utilizar el sistema de manera más efectiva y dedicar más tiempo a la validación de los resultados y la acción basada en ellos.

Objetivos

Objetivo General

- Evaluar dos técnicas de clasificación de células sanguíneas a través del procesamiento de imágenes.

Objetivos específicos:

- Realizar el estado del arte de la clasificación de células por fenotipo, morfología (color, tamaño -área y perímetro) y de sistemas de clasificación de células sanguíneas mediante procesamiento de imágenes y redes neuronales.
- Seleccionar dos técnicas de procesamiento de imágenes basadas en redes neuronales para su aplicación en la clasificación de células sanguíneas.
- Implementar las técnicas de procesamiento de imágenes elegidas para la clasificación de células sanguíneas a partir de un banco de imágenes de acceso público.
- Contrastar y comparar el desempeño las técnicas utilizadas en la clasificación de células sanguíneas en base a criterios de exactitud, precisión, tiempo de procesamiento, eficiencia, estabilidad y escalabilidad.

Introducción

La evaluación de técnicas de clasificación de células sanguíneas usando procesamiento de imágenes surge como una solución técnica sólida para la clasificación precisa y eficiente de células sanguíneas. Esto es fundamental para obtener diagnósticos precisos y monitorear la evolución de condiciones médicas. Así, en lugar de depender de la observación visual, se busca automatizar este proceso a través del uso de CNN, que han sido precisamente creadas para el procesamiento de imágenes, se ha comprobado su eficacia en una gran variedad de usos, que incluye la clasificación en imágenes médicas.

Para lograr con el objetivo planteado, se ha creado una red neuronal convolucional personalizada que está adaptada específicamente a la tarea de clasificación de células sanguíneas. Además, se usa YOLOv5 para obtener un conteo de células sanguíneas en imágenes.

Una parte integral de este trabajo es la aplicación de una interfaz gráfica que facilita a las personas que lo usen, especialmente a los profesionales médicos, interactuar con estas herramientas de procesamiento de imágenes de manera intuitiva y eficiente.

Es importante destacar que a lo largo de esta investigación, se han enfrentado desafíos técnicos significativos, como el requerimiento de un banco de imágenes de óptima calidad y bien etiquetadas, y la optimización de parámetros. Se reconoce que la efectividad de las soluciones desarrolladas puede variar según las características de las imágenes utilizadas y que esta tecnología debe complementar la experiencia y el juicio de los expertos en hematología.

Capítulo 1

Conceptualización y Estado del Arte

Este capítulo se centra en proporcionar una base sólida de conocimiento para comprender los elementos clave que influyen en el trabajo de procesamiento en imágenes para la clasificación de células sanguíneas. Aquí, se abordará las definiciones esenciales de las distintas células sanguíneas, considerando sus características morfológicas y funcionales. Además, se expondrá las resoluciones de imágenes utilizadas en el contexto de este estudio, dado que juegan un rol esencial en la calidad y precisión de la clasificación. Por último, se examinarán los avances en redes neuronales convolucionales y cómo han revolucionado el procesamiento de imágenes en aplicaciones médicas, destacando su relevancia para este enfoque específico.

1.1. Células sanguíneas

La sangre es considerada un tejido líquido que se encuentra en los seres vivos. Está compuesta por células (glóbulos), que se encuentran disueltos en un líquido llamado plasma sanguíneo. Aquí se encuentran tres tipos, glóbulos blancos, rojos y la llamada plaqueta.

Las células sanguíneas tienen diferentes formas y características morfológicas que les permiten cumplir sus funciones específicas en el cuerpo [1].

1.1.1. Glóbulos rojos (Eritrocitos)

El rol que cumplen estos glóbulos es el de llevar el oxígeno que se encuentran en los pulmones hacia los tejidos y llevar a los pulmones dióxido de carbono, para su expulsión.

Estos, se generan dentro de la médula ósea, teniendo una vida útil de aproximadamente 120 días en circulación. Durante este tiempo, se someten a cambios estructurales y metabólicos que les permiten mantener su forma y función. La morfología del glóbulo rojo normal es de forma de disco bicóncavo, lo que les permite tener una mayor superficie de contacto con el oxígeno en los pulmones. Tienen un diámetro de aproximadamente 7,5 micrómetros y un espesor de 2 micrómetros. Tiene como color característico, el rojo debido a la hemoglobina. Los eritrocitos no tienen núcleo, lo que les da más espacio para contener hemoglobina y les permite ser más flexibles [10].

La forma normal del eritrocito se ve alterada cuando se encuentra alguna patología, como se observa en la figura 1.1. En cuanto a los trastornos relacionados con los glóbulos rojos, incluye anemia, que es una condición en la que hay una reducción en la cantidad de eritrocitos, y policitemia, que es una condición en la que hay un aumento eritrocitos. Ambas condiciones pueden tener una variedad de causas, incluyendo deficiencias nutricionales, enfermedades crónicas y trastornos genéticos [11].

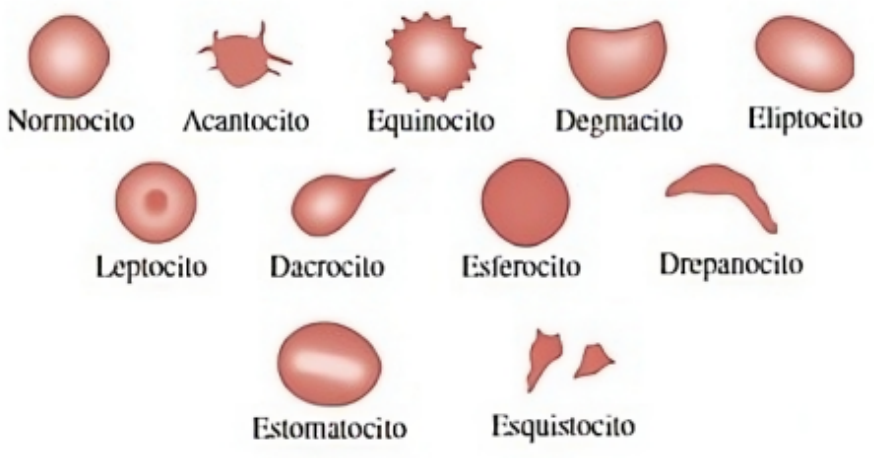

Figura 1.1: Patologías de los glóbulos rojos [12].

Eritroblastos

Los eritroblastos son las células que dan origen a los eritrocitos y se encargan en producir y madurar en la médula ósea. Además, durante su desarrollo, experimentan cambios morfológicos y funcionales.

Estas células progenitoras son importantes en la producción de hemoglobina, esencial en transportar el oxígeno en la sangre. A medida que los eritroblastos maduran, van acumulando hemoglobina en su citoplasma y, simultáneamente, reducen su tamaño. Estos atraviesan diferentes etapas, que son el eritroblasto basófilo, policromatófilo y el ortocromático. Una vez que alcanzan esta última etapa, expulsa el núcleo y se transforma en reticulocitos, proceso mostrado en la figura 1.2, que posteriormente se convierten en eritrocitos maduros, los cuales son los glóbulos rojos funcionales [13].

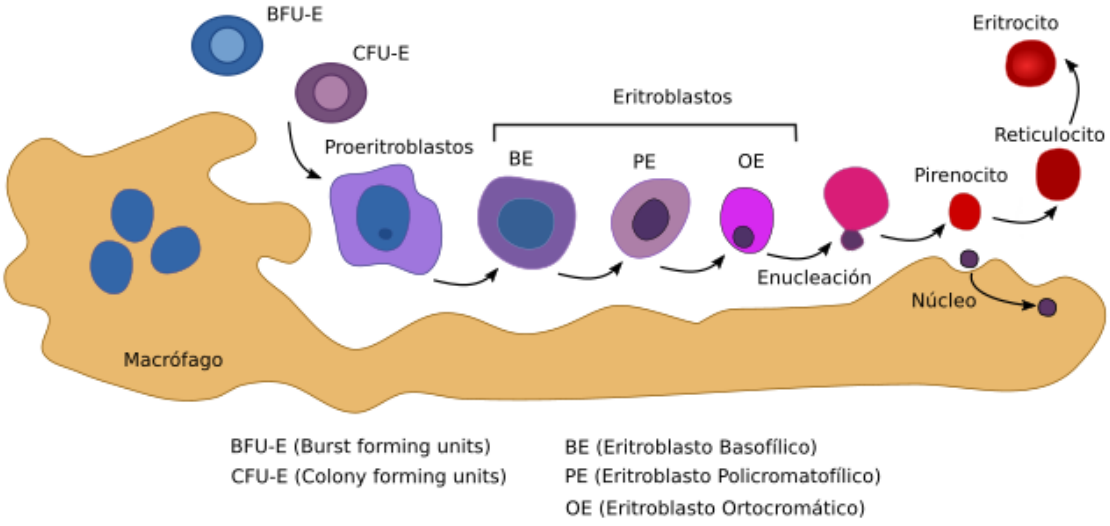

Figura 1.2: Maduración del Eritroblasto [14].

1.1.2. Glóbulos blancos (Leucocitos)

Estos glóbulos se encuentran en el sistema inmunológico, cumplen un rol fundamental en la protección del cuerpo contra infecciones y enfermedades. Existen diferentes glóbulos blancos, cada uno tiene una función específica en la defensa.

La médula ósea es el lugar donde se generan los glóbulos blancos, que posteriormente se difunden por todo el cuerpo mediante el sistema circulatorio.

Los niveles de glóbulos blancos pueden variar según diferentes factores y ciertos medicamentos. La cantidad de glóbulos blancos indica la presencia de una enfermedad o de infecciones, por lo que es importante que se controlen regularmente mediante análisis de sangre.

Existen diferentes tipos de leucocitos diferenciados por su forma y tamaño (Fig. 1.3). Algunos tienen forma de esfera (linfocitos), otros tienen forma de huso (monocitos), y otros tienen forma irregular (neutrófilos, eosinófilos y basófilos). El tamaño de los leucocitos varía entre 7 y 20 micrómetros y poseen un núcleo y el tamaño del núcleo es una característica importante para su clasificación. Los neutrófilos, eosinófilos y basófilos tienen un núcleo con forma de segmento, mientras que los linfocitos y los monocitos tienen un núcleo redondeado u ovalado [11].

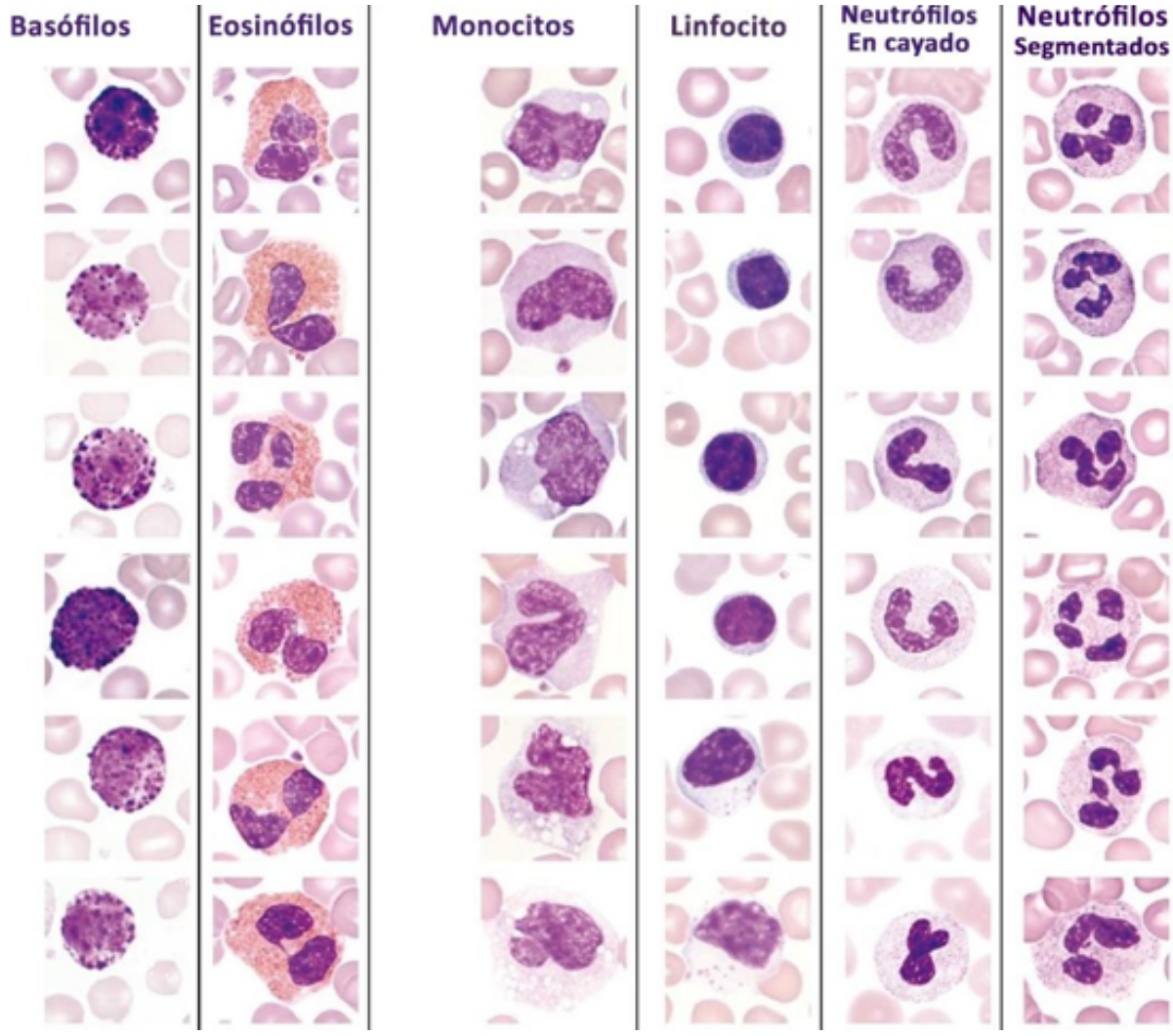

Figura 1.3: Morfología de los diferentes tipos de glóbulos blancos [15].

Basófilo

Un basófilo es un tipo de leucocito granulocito que está tanto en la sangre como en los tejidos. Su función principal es participar en la respuesta inflamatoria y en defender los organismos contra infecciones parasitarias. Los basófilos liberan sustancias químicas, fundamental en la respuesta alérgica y en regular la coagulación sanguínea.

En cuanto a su morfología, los basófilos son células redondeadas con un núcleo lobulado y lleno de gránulos basófilos dentro del citoplasma, así mostrada en la figura 1.4. Estos gránulos contienen las sustancias histamina, heparina, proteoglicanos y enzimas lisosomales. Los gránulos basófilos se tiñen de azul oscuro con colorantes básicos, como el azul de metileno, lo que les da su nombre [16].

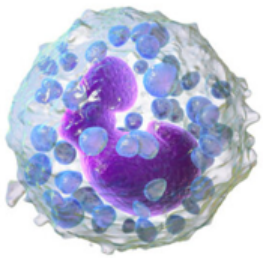

Figura 1.4: Basófilos [17].

Linfocito

Un linfocito juega un papel esencial en el sistema inmunológico. Su función es defender al organismo contra infecciones y enfermedades. Los linfocitos son responsables de reconocer y eliminar agentes patógenos, como bacterias, virus y células cancerosas, mediante respuestas inmunitarias específicas.

En cuanto a su morfología, son células pequeñas y redondeadas, que tienen un núcleo redondeado y grande que se encuentra en gran parte de la célula. El citoplasma es escaso y contiene pocos gránulos, como se observa en la figura 1.5. Además, presentan una alta relación núcleo citoplasma, lo que les da una apariencia de célula en anillo. Los linfocitos se pueden clasificar en diferentes subtipos, como linfocitos B, linfocitos T, cada uno con características y funciones específicas en la respuesta inmunológica [16].

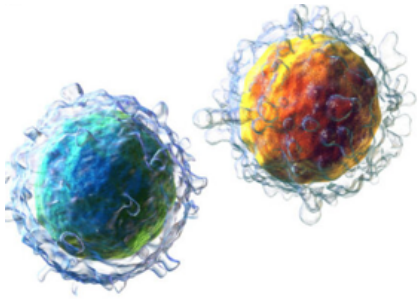

Figura 1.5: Linfocitos [18].

Neutrófilo

Es el más abundante y están presentes en la defensa contra infecciones. Son células fagocíticas, capaces de ingerir y destruir bacterias y otros microorganismos invasores.

La figura 1.6 muestra que los neutrófilos son células redondeadas con un núcleo lobulado que generalmente presenta de 2 a 5 lóbulos conectados por finos filamentos. El citoplasma de los neutrófilos contiene gránulos específicos que se tiñen de manera característica con colorantes ácidos y básicos. Estos gránulos se dividen en gránulos primarios o llamados azurófilos, gránulos secundarios o también conocidos como específicos y gránulos terciarios o gelatinasa [16].

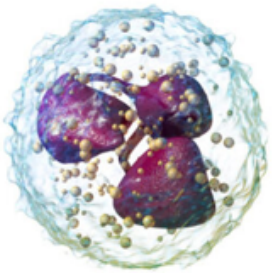

Figura 1.6: Neutrófilo [19].

Eosinófilo

Un eosinófilo ayuda en la defensa del organismo contra infecciones parasitarias, alergias y en la respuesta inflamatoria. Su función principal es la fagocitosis y la destrucción de parásitos, así como la regulación de las respuestas alérgicas.

Los eosinófilos son células redondeadas con un núcleo bilobulado, lleno de gránulos eosinófilos dentro del citoplasma, mostrados en la figura 1.7. Estos gránulos se tiñen de color rojo-anaranjado con colorantes ácidos, como la eosina, lo que les da su nombre. Los gránulos eosinófilos contienen sustancias como enzimas, proteínas tóxicas y mediadores inflamatorios, que se liberan durante la respuesta inmunitaria [16].

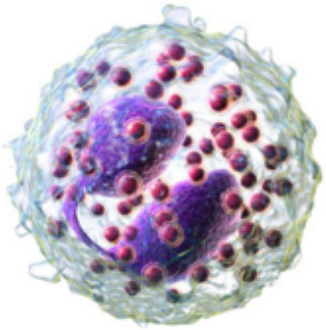

Figura 1.7: Eosinófilos [20].

Monocito

Su función principal es la presentación de antígenos a otras células del sistema inmunológico. Los monocitos son células grandes y redondeadas con un núcleo ovalado o en forma de riñón (Fig. 1.8). Puede contener gránulos azurófilos y su citoplasma es abundante. Los monocitos se desplazan en la sangre y cuando se activan, pueden dirigirse hacia los tejidos y diferenciarse en macrófagos o dendríticas. Estas células eliminan patógenos y células dañadas [16].

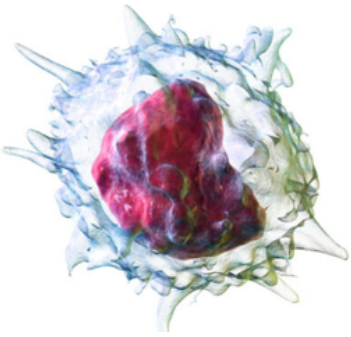

Figura 1.8: Monocito [21].

1.1.3. Plaquetas(Trombocitos)

Son células sanguíneas que tienen como principal función la formación de coágulos sanguíneos para detener el sangrado. Las plaquetas permanecen en estado inactivo hasta que son necesarias para formar un coágulo.

Cuando ocurre una herida, las plaquetas se activan y forman un tapón que para la hemorragia. Además, liberan sustancias químicas que limita el derrame excesivo de sangre [1].

Es importante destacar que la cantidad de plaquetas pueden variar debido a varios factores, como trastornos de la médula ósea, medicamentos o una dieta pobre en nutrientes esenciales. La cantidad de plaquetas puede ser indicativo de una enfermedad o condición (Fig. 1.9). Tienen forma de esfera y tienen un diámetro de aproximadamente 2-3 micrómetros.

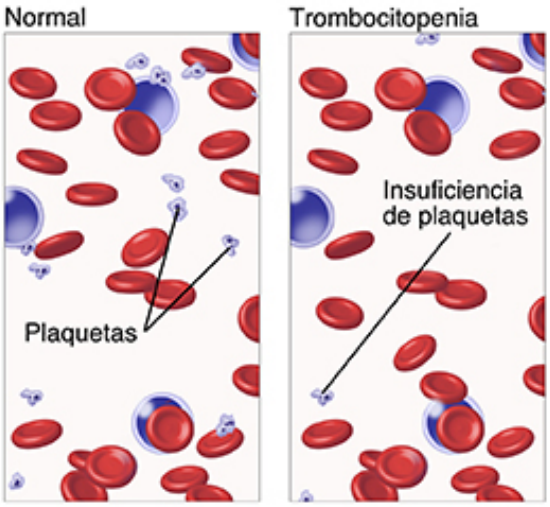

Figura 1.9: Insuficiencia de plaquetas [22].

1.2. Tinción en células sanguíneas

La tinción en células sanguíneas es una técnica usada en la hematología y la medicina. Esta técnica ayuda a estudiar las células que se encuentran en diferentes muestras de sangre. Este proceso aplica colorantes o tintes específicos a las células sanguíneas con el propósito de resaltar diferentes componentes celulares y características, lo que facilita su observación y análisis bajo un microscopio.

1.2.1. La tinción de Wright

Sirve para teñir y visualizar diferentes tipos de células sanguíneas. Esta técnica usa una mezcla de colorantes básicos y ácidos para teñir las células y resaltar sus características morfológicas (Fig.1.10). Esta tinción facilita observar la estructura y dimensiones de los glóbulos rojos, diferentes tipos de glóbulos blancos. También permite visualizar las plaquetas y evaluar su morfología [23].

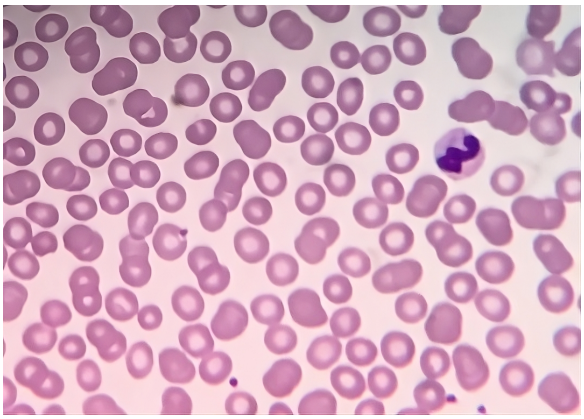

Figura 1.10: Tinción de Wright [23].

1.2.2. La tinción de Reticulocitos

Es una técnica para visualizar y contar los reticulocitos, que son glóbulos rojos inmaduros (Fig. 1.11). Se basa en el uso de tintes, como el azul metileno o de cresilo, que unen a los ribosomas presentes en los reticulocitos.

La tinción de reticulocitos permite identificar y contar los reticulocitos en una muestra de sangre, lo que resulta valioso en el diagnóstico y monitoreo de trastornos de la médula ósea, anemias y otras condiciones [23].

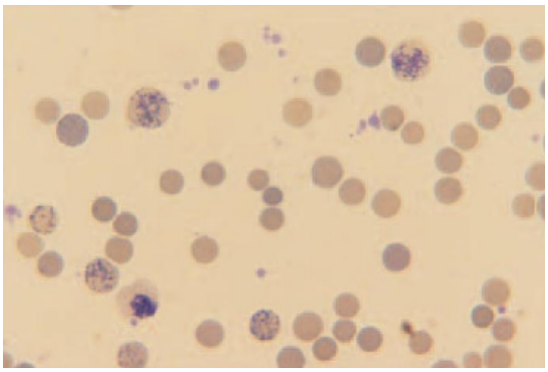

Figura 1.11: Tinción de Reticulocitos [23].

1.2.3. La tinción de Giemsa

Tiñe diferentes tipos de células sanguíneas y estructuras. En la figura 1.12 muestra la tinción de Giemsa basada en una combinación de tintes o colorantes, como la eosina. Estas permiten visualizar y diferenciar diferentes componentes celulares, como los núcleos, los citoplasmas y las estructuras intracelulares. Es especialmente útil para la observación de células sanguíneas, ya que permite identificar y clasificar diferentes tipos de glóbulos blancos y detectar anomalías en su morfología, como inclusiones intracelulares, cuerpos de inclusión y cambios en la coloración [23].

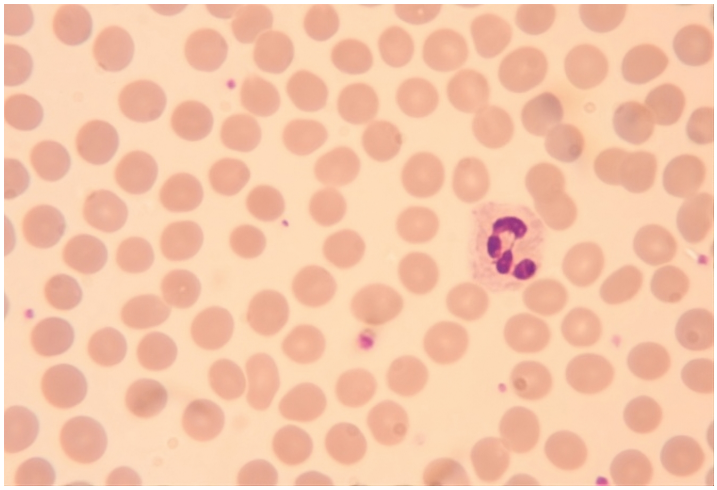

Figura 1.12: Tinción de Giemsa [24].

1.2.4. La tinción de Wright-Giemsa

Es utilizada en la hematología para teñir las células sanguíneas y permitir su visualización y diferenciación. Esta técnica combina los colorantes de Wright y Giemsa para obtener una tinción más completa y diferenciar con mayor claridad los diferentes tipos de células sanguíneas. El colorante de Wright tiñe los núcleos de las células de color azul oscuro, mientras que el colorante de Giemsa tiñe el citoplasma de las células de diferentes tonalidades, como rosa, violeta o azul claro (Fig.1.13). Esta combinación de colorantes permite identificar y distinguir los diferentes tipos de células sanguíneas [25].

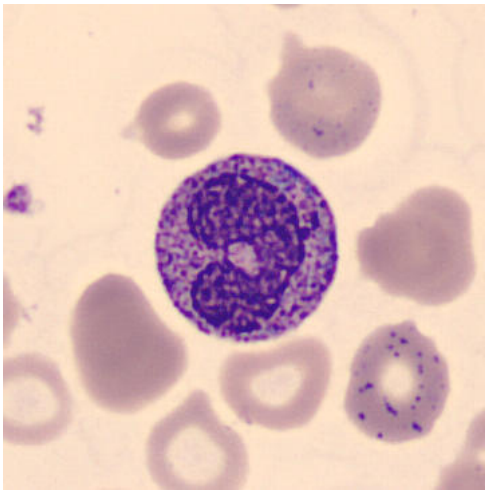

Figura 1.13: Tinción de Wright-Giemsa [26].

1.3. Imagen de entrada

Antes de procesar una imagen, es crucial tener en cuenta diversos parámetros que influyen en su calidad e interpretación. Entre estos parámetros se encuentran el brillo, calidad, contraste, formato y otros factores relevantes. Estos aspectos son fundamentales para asegurar una interpretación precisa y efectiva de la imagen, así como obtener resultados confiables en el análisis posterior.

1.3.1. Formato de imagen

Unos de los parámetros que se considera para la obtención de imagen de entrada es el formato de imagen, debido a que puede afectar considerablemente al funcionamiento y resultados según el contexto en el que se vaya a utilizar. Al comprender las características y aplicaciones específicas de los diferentes formatos de imagen, se puede aprovechar al máximo el potencial de la imagen y garantizar su eficacia. Cada formato de imagen tiene ventajas y desventajas en cuanto a la calidad, tamaño de archivo, compatibilidad y funcionalidad [27].

El autor Boutell dice que el formato PNG es especialmente adecuado para imágenes que requieren alta calidad y precisión, como gráficos, logotipos y capturas de pantalla. Ofrece una representación precisa de los colores y detalles de la imagen, sin introducir artefactos o pérdida de calidad [28].

Otro formato es el TIFF, que es un archivo flexible que admite gran variedad de características. Puede manejar imágenes a color, en grises, imagen de gran resolución, imágenes de múltiples páginas y meta datos asociados. Además, el formato TIFF es compatible con diferentes tipos de datos. En escala de grises puede manejar 8 bits, a color de 24 bits e imágenes de alta profundidad de bits. Esto lo hace una elección popular en aplicaciones que necesitan una representación precisa y detallada de la imagen [29].

Wallace en [30] habla del JPG ampliamente utilizado debido a su algoritmo de compresión con pérdida. Este algoritmo reduce el tamaño del archivo al eliminar información redundante y detalles menos perceptibles en la imagen. Esto resulta en una reducción significativa en el tamaño del archivo, lo que facilita el almacenamiento y la transmisión de imágenes [30].

En la figura 1.14 se exponen imágenes con los formatos antes indicados al igual que sus principales datos. Esto permite comparar los formatos teniendo en cuenta que en JPG el tamaño es el más bajo con 2.2 MB, mientras que el más alto es el TIFF con 21.4 MB, con una profundidad de 32 bits siendo la más alta de los formatos.

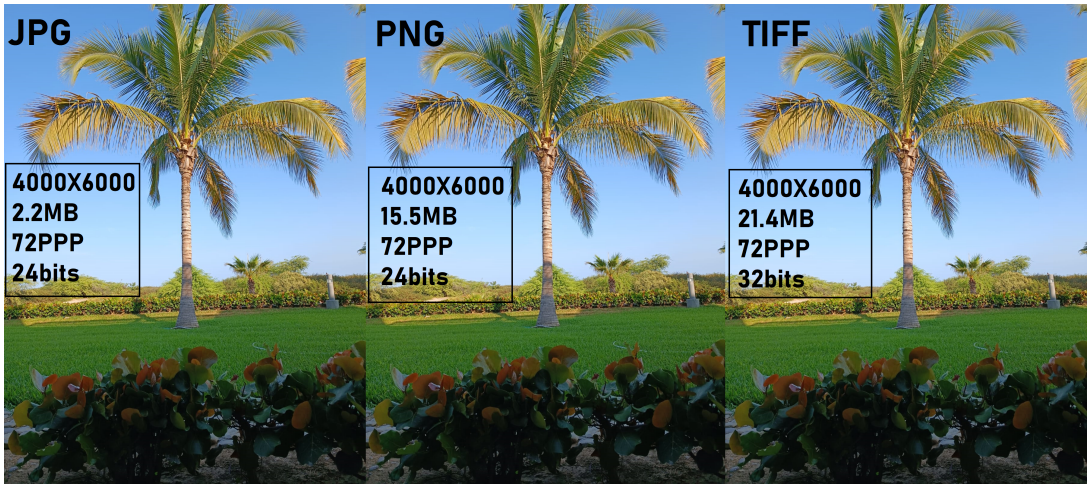

Figura 1.14: Formato de imagen (PNG, JPG, TIFF) [Fuente: Autor].

Elegir el formato adecuado, dependerá de las necesidades específicas, ya que cada uno de ellos presenta sus propias ventajas. Por lo tanto, es fundamental realizar una evaluación antes de seleccionar uno.

1.3.2. Resolución espacial

La resolución espacial indica del nivel de detalle o nitidez que puede capturarse en una imagen o visualizar en un dispositivo de visualización [31]. Entonces, cuando se habla de resolución espacial, implica la cantidad de píxeles en las dimensiones tanto horizontal como vertical de una imagen. Así, cuando la resolución espacial sea mayor en una imagen, mayor será el detalle que se puede capturar o visualizar.

Las imágenes con una resolución más alta tienen más píxeles, lo que permite representar con mayor precisión los detalles finos y las características de la imagen. Por el contrario, las imágenes con una resolución más baja tienen menos píxeles y, por lo tanto, pueden parecer más borrosas o menos detalladas.

Bovik [32] también menciona que la profundidad de bits es la información que se almacena para los píxeles de la imagen. Estos, se miden en bits y determina la cantidad de colores o niveles de gris que se pueden representar. Por ejemplo, una imagen de 8 bits puede representar 256 niveles de color, mientras que una imagen de 24 bits puede representar 2^{24} niveles de colores.

La figura 1.14 mostró que el formato de TIFF tiene una profundidad de 32 bits que es 2^{32} esto indica que la de 32 bits puede representar una amplia gama de colores. Pero se debe tener en cuenta que esto hace que la imagen pese más, lo que sería una dificultad a la hora de entrenar imágenes. En el entrenamiento se necesitan miles de imágenes y usar imágenes de 32 bits implicaría que el archivo de imágenes es muy pesado, lo que va a requerir más tiempo para entrenar en comparación con una imagen de menor profundidad de bits, como una de 8 bits o 16 bits. La razón principal es que las imágenes de 32 bits contienen más información y, por lo tanto, requieren más cálculos y capacidad de almacenamiento durante el proceso de entrenamiento.

1.3.3. Contraste, Brillo

Los siguientes autores presentan las perspectivas sobre el contraste y el brillo en el procesamiento de imágenes.

Gózales [27], dice que el contraste es la diferencia de intensidad entre áreas claras y oscuras. Un alto contraste significa que hay una diferencia entre los valores

de intensidad, lo que entrega una imagen más nítida y con una claridad de detalles mayor. Pratt [33], habla que el brillo se relaciona con la percepción de la intensidad de la luz y mencionan técnicas específicas. Estas técnicas se hablarán más abajo como un preprocesamiento de imagen. Burger [34], presentan técnicas específicas para ajustar estos parámetros y se destaca la importancia de considerar factores subjetivos y contextuales al hacerlo. Además, se proporciona una explicación sobre la respuesta del ojo humano a los cambios de intensidad de iluminación y la influencia del fondo en la percepción del brillo.

Aunque, en los trabajos antes indicados se mencionan algunos aspectos relacionados con el contraste y el brillo, no se especifican los valores óptimos para lograr un mejor contraste o brillo en las imágenes. Esto se debe a que no existe un valor universalmente óptimo para estos parámetros, ya que su adecuación depende del contenido específico de la imagen, el propósito del procesamiento y las preferencias subjetivas del observador. El contraste y el brillo son características que varían según el contexto y los objetivos específicos de cada imagen. Lo que puede considerarse un buen contraste o brillo en un caso particular puede no serlo en otro [31].

En caso de la figura 1.15 se puede utilizar en diferentes ámbitos, en donde se puede cambiar los valores para observar más el objeto que se quiera visualizar. Es necesario hacer pruebas con diferentes valores y técnicas para encontrar el ajuste que produzca los resultados deseados en cada caso particular.

Figura 1.15: Parámetros (Brillo, Contraste) [Fuente: Autor].

1.4. Preprocesamiento

El tratamiento de imágenes se extiende a múltiples campos, como la identificación de patrones o elementos en imágenes médicas, satélites, reconocimiento facial en sistemas de seguridad, mejoramiento de la calidad en fotografía digital, entre otras.

El objetivo del preprocesamiento es mejorar la resolución de imágenes. Esto se realiza con técnicas y algoritmos matemáticos que ayudan a modificar, analizar y procesar las imágenes de forma automatizada y eficiente. El procesamiento de imágenes es muy útil para la eliminación del ruido, lo que hace posible obtener imágenes más claras y detalladas. Permite extraer información útil de las imágenes, patrones y características específicas. Ayuda en la automatización de tareas que de otra manera serían difíciles de realizar manualmente, lo que ahorra tiempo y esfuerzo.

Existen diversas técnicas de procesamiento que se usan en función del objetivo específico que se busca lograr [35].

1.4.1. Ecualización de histograma

La ecualización de histograma es una función matemática aplicada a la imagen para ajustar la distribución en escala de grises y mejorar su contraste. En términos simples, la función de ecualización de histograma se utiliza para transformar la intensidad de la imagen original en otros valores de intensidad en la imagen mejorada.

De esta forma, la función de ecualización se calcula a partir del histograma de la imagen que se va a analizar y su ecuación de distribución acumulativa (CDF), que muestra la probabilidad acumulada de encontrar un nivel de gris en la imagen. Mediante CDF, se puede calcular la función de transformación de intensidad que se utiliza para mapear los niveles de gris originales a nuevos valores de intensidad. Al aplicar la ecualización de histograma, los niveles de intensidad se amplían y se comprimen los niveles intermedios, mejorando así la claridad y contraste de la imagen. El resultado es una apariencia más nítida y con detalles más claros, lo que facilita su interpretación y análisis [35].

Al aplicar la ecualización del histograma al canal de intensidad se logra

expandir el rango de tonos de gris, incrementando el contraste. Este es un método simple y eficaz para realzar imágenes en escala de grises. Por otro lado, la ecualización de histograma en RGB se aplica a cada canal de color por separado, lo que permite ajustar la distribución de histograma en cada canal individualmente. Esto puede ser útil cuando se desea resaltar o corregir ciertos colores en la imagen [36].

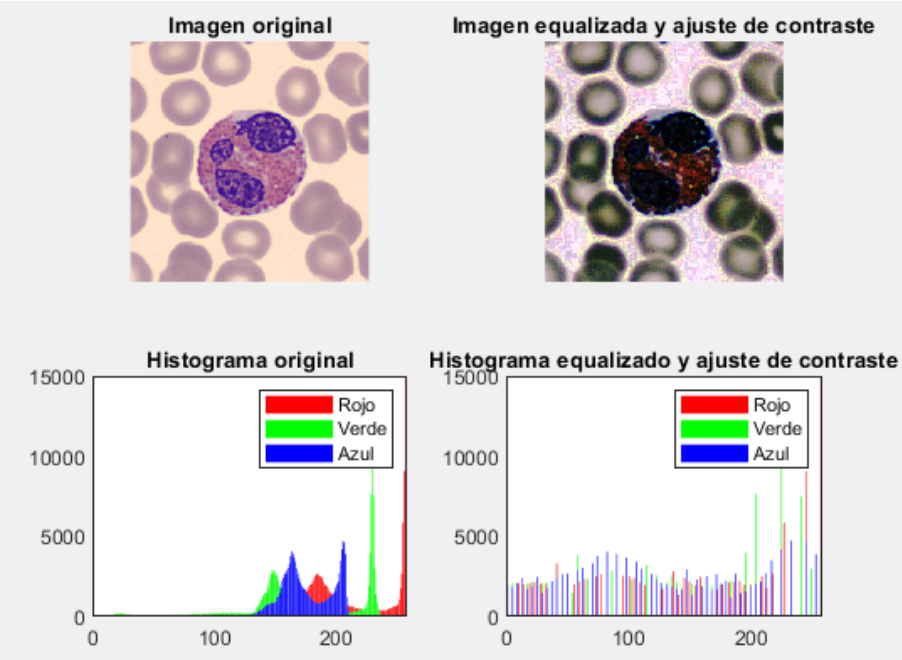

Figura 1.16: Ecualización de histograma [Fuente: Autor].

La figura 1.16 muestra la imagen que ha sido sometida a un proceso de ecualización de histograma y ajuste de contraste. Mostrando que los valores de intensidad en los canales RGB se han redistribuido de mejor manera a comparación del histograma original, logrando una representación más uniforme de las intensidades en toda la imagen. Los colores se vuelven más vivos y los detalles se destacan más, demostrando que la imagen ha experimentado una mejora en su calidad visual y en el contraste.

1.4.2. Filtro de la mediana

Es un filtro no lineal que es usado para quitar el ruido presente en una imagen A diferencia del filtro de la media que usa el promedio, este filtro sustituye el píxel por la mediana de los píxeles adyacentes.

Este enfoque es eficaz para suprimir el ruido de sal y pimienta vista en la figura 1.17 y es caracterizado por tener píxeles con valores muy altos o bajos en comparación con sus vecinos. Estos valores atípicos pueden causar distorsiones significativas en la imagen y afectar su calidad visual.

Este filtro funciona seleccionando el valor mediano de en una vecindad determinada, alrededor de cada píxel de la imagen. El tamaño de la vecindad se elige según la cantidad y el tipo de ruido en la imagen. El valor mediano es seleccionado como el valor que se utilizará para reemplazar el píxel original.

Este procedimiento es aplicado a todos los píxeles, dando como resultado una imagen filtrada con menor ruido impulsivo [31].

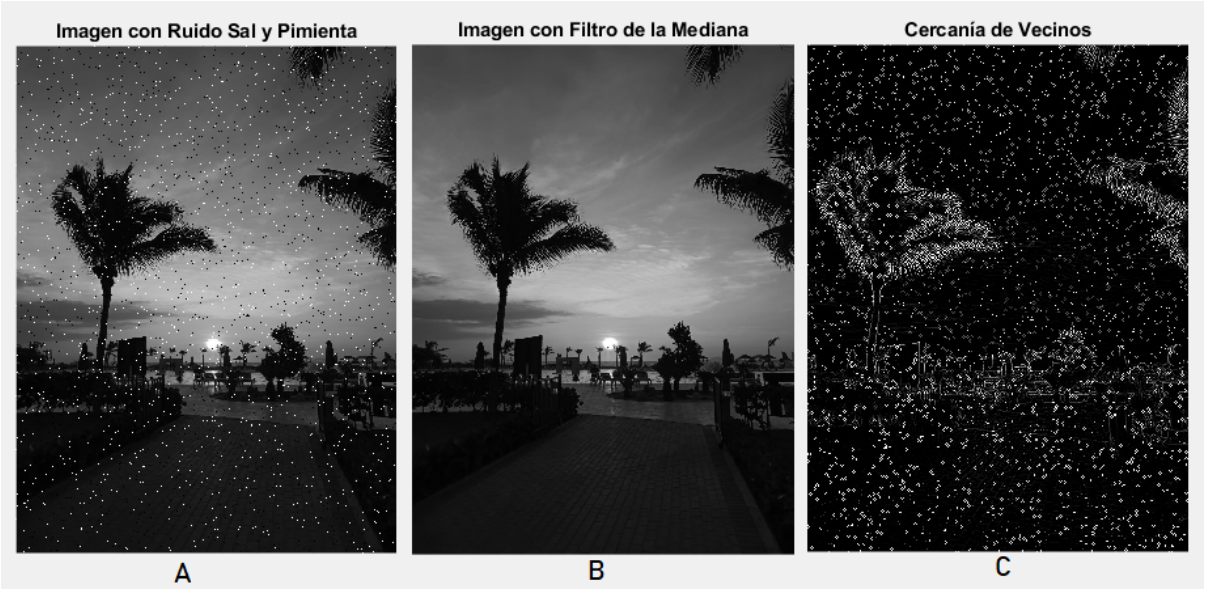

Figura 1.17: Filtro de la mediana [Fuente: Autor].

La imagen C de cercanía de vecinos mostrada en la figura anterior, muestra píxeles oscuros y claros. Los píxeles más oscuros indican que los valores de los vecinos son muy similares al valor mediano del píxel seleccionado. En cuanto a los píxeles más claros, indican que los valores de los vecinos son significativamente diferentes al valor del píxel seleccionado.

La imagen B muestra la imagen luego haber aplicado el filtro, mostrando una mejora en términos de reducción de ruido, mostrando una imagen de mejor calidad y más limpia.

1.4.3. Eliminación de fondo

La eliminación de fondo mediante segmentación del umbral es bastante utilizado para separar el objeto de interés del fondo en una imagen. Esta técnica se basa en que los píxeles del fondo y objeto presentan características distintas, como intensidad, color y otras propiedades.

En la segmentación del umbral, se establece un umbral o rango de umbrales que divide la imagen en dos regiones, una región correspondiente al objeto de interés y otra región correspondiente al fondo. Los píxeles cuya intensidad está por encima o por debajo del umbral se asignan a la región del objeto o del fondo, respectivamente. La elección del umbral puede realizarse de diferentes maneras. Se puede seleccionar un valor fijo basado en el conocimiento previo del problema o utilizar técnicas automáticas de selección de umbrales, como el método de Otsu que intenta determinar el umbral ideal que incrementa al máximo la varianza entre ambas zonas. (Fig. 1.18).

Aunque la segmentación del umbral es una técnica simple, su efectividad puede depender de algunos factores, como calidad, el contraste del fondo y objeto, y la elección adecuada del umbral [31].

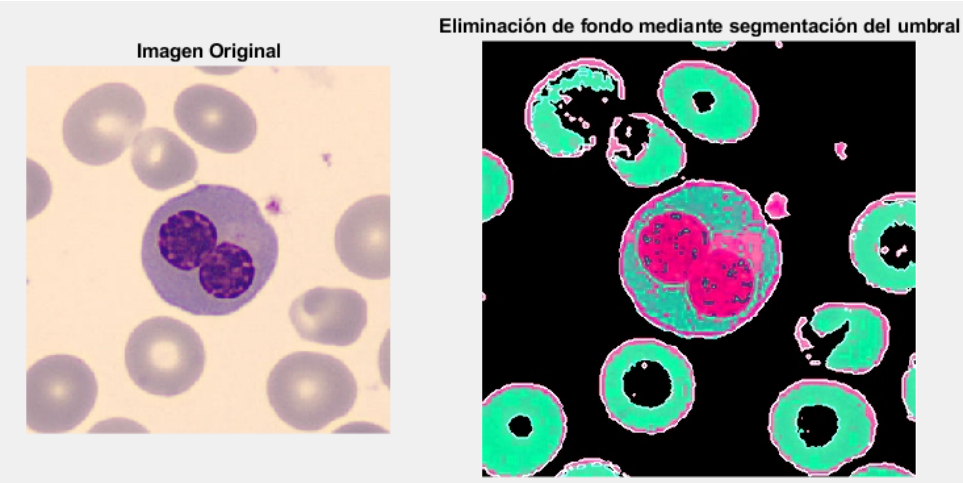

Figura 1.18: Eliminación de fondo [Fuente: Autor].

1.4.4. Filtro Laplaciano

El filtro es ampliamente utilizado para procesar imágenes, para mejorar bordes y destacar las transiciones abruptas de intensidad en una imagen. Se basa en el

operador Laplaciano, que mide la segunda derivada espacial de la función. Para aplicar este filtro, se convoluciona la imagen original con una máscara o kernel que representa el operador Laplaciano. Esta máscara tiene valores positivos en el centro y valores negativos en los bordes, lo que resalta los cambios bruscos de intensidad. La convolución con el filtro Laplaciano produce una imagen resultante en la que los bordes y las transiciones de intensidad se realzan, lo que hace que los bordes sean más visibles y destacados [27].

Sin embargo, es importante tener en cuenta que este filtro también puede aumentar el ruido en algunos aspectos mostrados en la figura 1.19. Aunque también muestra una mejora en cuanto a la calidad y otros aspectos.

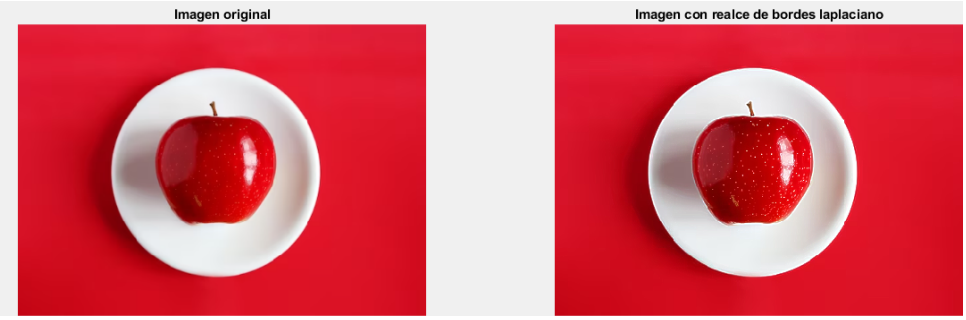

Figura 1.19: Filtro Laplaciano [Fuente: Autor].

1.5. Redes Neuronales Artificiales

Las Redes Neuronales Artificiales (RNA), se refieren a sistemas informáticos que simulan la operación neuronal del cerebro. Consisten en elementos de procesamiento denominados neuronas artificiales, organizadas en capas y se enlazan mediante conexiones ponderadas.

Las RNA se emplean en resolver problemas complejos en aprendizaje automático, detección de patrones, clasificación, regresión, detección de imágenes, procesamiento de voz, y otros más. Estas redes son capaces de aprender con ejemplos y generalizar, para realizar clasificaciones o predicción en datos nuevos [37].

1.5.1. Red neuronal convolucional

Una red neuronal convolucional (CNN, del inglés *Convolutional Neural Network*), es un modelo de red neuronal profunda (en inglés *deep learning*) especialmente diseñado para el tratamiento de imágenes. Una CNN consiste en capas conectadas en secuencia, cada una de las cuales está diseñada para obtener ciertos rasgos de la imagen de entrada. Las capas más tempranas suelen incluir filtros convolucionales, que escanean la imagen para detectar patrones, características o bordes y contornos. A medida que las capas avanzan, las características detectadas se combinan en patrones más complejos, como formas y texturas [38].

Las CNN han probado ser extremadamente eficiente en clasificar y detectar objetos en imágenes, y son de gran utilidad como la identificación de enfermedades médicas, como la clasificación de células (Fig. 1.20) y la conducción autónoma de vehículos.

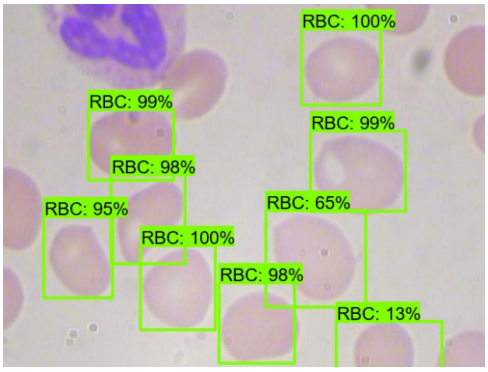

Figura 1.20: Clasificación de células mediante red neuronal [39].

Entrenar una CNN para clasificar células sanguíneas, se necesitará una gran cantidad de imágenes etiquetadas de células en diferentes estados y condiciones relevantes. Estas imágenes se utilizarán para entrenar la red neuronal, y la red aprenderá a reconocer las características comunes de las células en cada categoría.

La ventaja de la CNN es que aprende patrones en los datos y va mejorando su precisión en la tarea asignada a medida que se les suministran más datos. Estas se pueden escalar y controlar altas cantidades de datos y problemas más complejos, haciéndolas adecuadas para aplicaciones empresariales y científicas a gran escala [40].

1.5.2. Red neuronal convolucional profunda

Estas redes se refieren a modelos de aprendizaje automático compuestos por varias capas ocultas. Estas capas posibilitan la obtención de características y conceptos cada vez más complejos y abstractos a partir de información de entrada. Este red emplea algoritmos de aprendizaje que ajustan los sesgos y pesos de cada capa. Esto posibilita que la red aprenda a identificar patrones y ejecutar tareas, como clasificar imágenes, traducción automática o reconocimiento de voz.

La capacidad de estas redes neuronales profundas para adquirir y representar aspectos de alto nivel en los datos de entrada la hace especialmente eficiente en problemas de procesamiento de imágenes [41].

1.5.3. YOLO

YOLO (*del inglés, You Only Look Once*) es un modelo de red neuronal profunda, usado para identificar elementos de manera rápida presentes en imágenes o secuencias de video. A diferencia de otros métodos que utilizan espacios de interés, YOLO hace predicciones de detección después de una sola evaluación directa de la imagen a través de una CNN.

La arquitectura de YOLO realiza una división en celdas de la imagen, donde cada una predice los cuadros delimitadores y la posible clase para los objetos contenidos en esas cajas. Las cajas delimitadoras se definen por su posición y tamaño, y se asigna una puntuación de confianza para indicar la probabilidad de que contengan un objeto. Mientras se realiza el entrenamiento, YOLO adapta la red con una función de pérdidas que compara las predicciones con las etiquetas de los objetos reales. Esto incrementa la certeza en la identificación de objetos y rasgos distintivos.

Una vez entrenado, YOLO puede utilizarse para identificar objetos en nuevos datos de entrada. La red analiza la imagen de entrada y produce las cajas delimitadoras junto con las certezas de las clases de los objetos identificados. Estas detecciones sirven para tareas como seguimiento de objetos, detectar objetos o análisis de escenas [42].

YOLO V5

YOLOv5 es una versión mejorada y más eficiente del algoritmo YOLO para identificar objetos en imágenes o vídeos. Está basado en la arquitectura de redes (CNN) para realizar detecciones en tiempo real.

Esta arquitectura consta de varias capas convolucionales que obtienen aspectos de las imágenes y generan predicciones de las cajas delimitadoras y las clases que se encuentran en las imágenes, (Fig. 1.21). Las capas están seguidas por capas de agrupación y capas de detección. YOLOv5 tiene la capacidad de realizar detecciones a distintas escalas. Esto se logra utilizando una técnica llamada (FPN, del inglés Feature Pyramid Network), la cual combina características y utiliza múltiples escalas de análisis para identificar objetos de distintos tamaños.

Además, durante el entrenamiento, YOLOv5 emplea métodos de expansión de datos, como recortes aleatorios, giros y cambios de escala, para mejorar la robustez y la generalización del modelo. Estas técnicas permiten que el modelo sea más capacitado para la identificación de objetos en diferentes situaciones y condiciones [42].

Figura 1.21: Clasificación de objetos mediante YOLO [43].

Capítulo 2

Técnicas de Procesamiento de Imágenes para la Clasificación de Células Sanguíneas

Este capítulo muestra la elección e implementación de dos técnicas para el diseño de un clasificador células sanguíneas, las cuales están basadas en CNN . El objetivo principal es explorar las ventajas y el potencial de estas técnicas, resaltando su capacidad para proporcionar resultados precisos y eficientes.

En primer lugar, se realizó un análisis completo de la literatura referente al empleo de redes neuronales que se emplean para procesar imágenes y para la clasificación de células sanguíneas. Se analizaron diferentes enfoques y arquitecturas de redes neuronales utilizadas en investigaciones previas, destacando sus fortalezas y limitaciones.

Luego, se procedió a seleccionar dos técnicas de procesamiento de imágenes basadas en redes neuronales consideradas adecuadas en clasificar células sanguíneas. Esta selección se basó en criterios como la precisión de los resultados, el tiempo de ejecución y la viabilidad de implementación en entornos clínicos.

Una vez seleccionadas las técnicas, se procedió con su implementación. A continuación se detallarán las acciones realizadas para procesar las imágenes de células sanguíneas, así como la configuración y el entrenamiento de las redes.

2.1. Revisión y Selección de Técnicas

Para establecer los métodos de procesamiento más apropiados, se consideró cuidadosamente el tipo de red neuronal idónea para la clasificación de imágenes. Según el estudio realizado, se concluyó que las redes neuronales convolucionales muestran un rendimiento superior cuando se trabaja con entradas de imágenes en comparando diversas formas de redes neuronales [44].

2.1.1. Técnicas de Procesamiento basadas en redes neuronales revisadas

Clasificación glóbulos blancos con Red Neuronal Convolucional (Técnica 1)

Es el modelo que usa la biblioteca Keras y se define como una secuencia de capas que se acumulan una sobre la otra conformando la red. Este modelo establece la estructura de una CNN con capas de agrupación (*Max Pooling*), normalización, completamente conectadas, de eliminación (*dropout*) y principalmente con tres capas convolucionales.

En la mayor parte de las capas ocultas, la activación realiza con la función de Unidad Lineal Rectificada (ReLU, *del inglés Rectified Lineal Unit*), excepto en una capa que utiliza LeakyReLU que es una versión mejorada a la anterior y se usa para evitar la perdida de las unidades.

El modelo es entrenado para categorizar la información de entrada en una de las 4 categorías posibles.

Se procederá a examinar con detalle las capas que conforman esta red neuronal.

- Capa 1: Primera capa convolucional. Cuenta con 32 filtros. Su activación es mediante ReLU. Toma una entrada (100, 100, 3) que representa la imagen RGB de 100x100 píxeles.

- Capa 2: Esta es una capa de agrupación que minimiza la amplitud en el espacio de la salida de la capa anterior mediante un factor de 2. Ayuda a reducir el costo computacional y extraer características importantes.

- Capa 3 y 4: Segunda capa convolucional con 64 filtros y un kernel de 5x5. Su activación es mediante ReLU y capa de agrupación para reducir la dimensión espacial.
- Capa 5 y 6: Tercera capa de convolución con 128 filtros y kernel de 3x3. Su activación es mediante ReLU y capa de agrupación adicional
- Capa 7: Capa que normaliza la salida de la capa anterior. Ayuda a acelerar el entrenamiento y estabilizar la red.
- Capa 8: Esta capa aplanada convierte los datos de entrada en un vector de una dimensión para poder enlazarlo a capas completamente conectadas.
- Capa 9: Capa completamente conectada con 64 unidades y activación LeakyReLU.
- Capa 10: Capa de eliminación que desactiva aleatoriamente el 50 % de las unidades para evitar el sobreajuste.
- Capa 11: Otra capa completamente conectada con 32 unidades y activación ReLU.
- Capa 12: Capa de eliminación adicional con una tasa del 40
- Capa 13: Otra capa completamente conectada con 16 unidades y activación ReLU.
- Capa 14: Capa de eliminación adicional con una tasa del 40
- Capa 15: Capa de salida con 4 unidades y función de suavización *(softmax)*. Se utiliza para identificar en la entrada una de las 4 clases.

La figura 2.1 es una descripción del diseño de la red neuronal y sus capas antes detalladas. Esta información se obtiene de la ayuda del paquete de la red neuronal implementada en Python.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 98, 98, 32)	896
max_pooling2d (MaxPooling2D)	(None, 49, 49, 32)	0
conv2d_1 (Conv2D)	(None, 45, 45, 64)	51264
max_pooling2d_1 (MaxPooling 2D)	(None, 22, 22, 64)	0
conv2d_2 (Conv2D)	(None, 20, 20, 128)	73856
max_pooling2d_2 (MaxPooling 2D)	(None, 10, 10, 128)	0
batch_normalization (BatchN ormalization)	(None, 10, 10, 128)	512
flatten (Flatten)	(None, 12800)	0
dense (Dense)	(None, 64)	819264
dropout (Dropout)	(None, 64)	0
dense_1 (Dense)	(None, 32)	2080
dropout_1 (Dropout)	(None, 32)	0
dense_2 (Dense)	(None, 16)	528
dropout 2 (Dropout)	(None, 16)	0

Figura 2.1: Descripción del modelo [Fuente: Autor].

La figura 2.2 es un ejemplo de las imágenes de entrada utilizadas en esta red neuronal para la clasificación. Estas son redimensionadas para que tengan un formato de 100x100 píxeles y están en formato RGB. El tamaño de 100x100 píxeles indica la resolución de la imagen y RGB se refiere al modelo de color utilizado para representar los colores en la imagen.

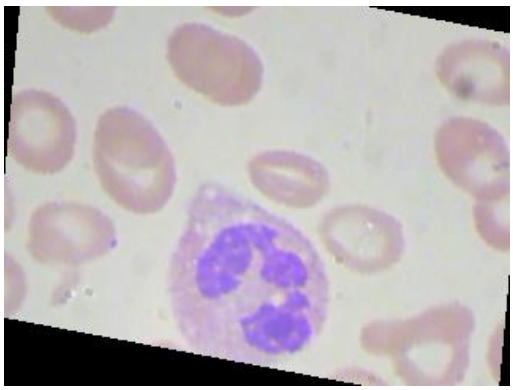

Figura 2.2: Imagen de entrada [45].

Para el entrenamiento se realiza una conversión de la información en matrices, es decir, los datos se actualizarán para contener matrices de tipo NumPy en lugar de los objetos originales. Esto permite realizar operaciones y cálculos matemáticos de manera más eficiente en estas matrices utilizando las funciones y métodos proporcionados por NumPy. El código empleado para esto es el siguiente:

```
X_train = np.array(X_train)
X_test = np.array(X_test)
```

Luego, se realiza una normalización que se realiza para normalizar las magnitudes de los píxeles de las imágenes para que estén en el intervalo entre 0 y 1, lo que facilita el procesamiento y mejora el rendimiento del modelo. El código empleado para esto es el siguiente:

```
X_train = X_train /255
X_test = X_test /255
```

Para configurar el proceso de entrenamiento se toman en varios argumentos para especificar la metodología que se seguirá para realizar el entrenamiento del modelo.

optimizer: optimizador «adam», es un enfoque eficaz de optimización que se fundamenta en la adaptación de estadísticas de segundo y primer orden.

loss: utiliza una función de pérdida que es empleada frecuentemente en situaciones de clasificación que involucran varias clases.

metrics: utiliza la métrica precisión, que es usa comúnmente en problemas de categorización y evalúa la cantidad de predicciones acertadas.

Para entrenar el modelo se ha configurado en 30 la cantidad de épocas para entrenar. Los parámetros detallados se los colocan de la siguiente manera:

```
KerasModel.compile(optimizer ='adam',
loss='sparse_categorical_crossentropy', metrics=['accuracy'])
ThisModel = KerasModel.fit(X_train, y_train, epochs=30, verbose=1)
```

Después de completar el proceso de entrenamiento, el cual demora alrededor de 2 minutos y medio por época y en total aproximadamente 1 hora y media, se obtiene una precisión del 0.49 o del 49 %.

En otras palabras, el modelo ha aprendido a predecir correctamente el 49 % de las muestras en la base de prueba. Esto significa que, en promedio, ha acertado aproximadamente la mitad de las clasificaciones.

Tabla 2.1: Precisión y Perdida Obtenida con la Técnica 1.

Pérdida en Conjunto de Prueba:	2.170181512
Precisión en Conjunto de Prueba:	49.8994767 %

Es importante tener en cuenta que este modelo de clasificación está específicamente diseñado para clasificar únicamente cuatro tipos de glóbulos blancos. Por lo tanto, su capacidad para distinguir y clasificar con precisión otros tipos de células sanguíneas puede ser limitada. Esto se debe a que cada tipo de célula sanguínea posee características y patrones únicos que podrían requerir enfoques de modelado y entrenamiento diferentes.

En este sentido, es fundamental tener en consideración que si el objetivo es clasificar diferentes tipos de células sanguíneas aparte de los leucocitos, es posible que se opte por un enfoque diferente. Esto podría implicar adaptar el modelo actual para incorporar nuevas clases o incluso desarrollar un nuevo modelo específicamente diseñado para abordar esas clases adicionales.

Clasificación glóbulos blancos con Red Neuronal Convolucional (Técnica 2)

Esta técnica es una representación de red neuronal convolcional para obtener atributos de las imágenes, compuesta por varias capas de normalización por lotes, agrupación para disminuir la dimensionalidad espacial y evidentemente capas de convolución. Luego, las características se pasan a través de capas llamadas totalmente

conectadas, con capas de eliminación para la regularización, antes de llegar a la capa final con activación de función de suavización *(softmax)*.

- Primera capa convoluvional: Se utilizan 16 filtros de 3x3, y se especifica activación ReLU. La forma de entrada es (120, 120, 3), lo que indica imágenes de tamaño 120x120 píxeles con 3 canales de color RGB. Seguida de una capa de agrupación.
- Segunda capa convolucional: Se utilizan 32 filtros de tamaño 3x3, y se activa con ReLU. Capa de agrupación después de la segunda capa convolucional.
- Tercera capa convolucional: Se utilizan 64 filtros de tamaño 3x3,y se activa con ReLU. Capa de agrupación después de la tercera capa convolucional.
- Cuarta capa convolucional: Se utilizan 128 filtros de tamaño 3x3, y se activa con ReLU. Capa de agrupación después de la cuarta capa convolucional.
- Quinta capa convolucional: Se utilizan 256 filtros de tamaño 3x3, y activación ReLU.

En cada capa convolucional se agrega una para normalizar la salida de la capa antecesora y estabilizar el entrenamiento.

Se adicionan capas totalmente conectadas al modelo:

- Primera capa completamente conectada: Tiene 256 unidades y utiliza activación ReLU. Capa de eliminación para reducir el sobreajuste.
- Segunda capa completamente conectada: Tiene 128 unidades y activación ReLU. Capa de eliminación adicional.
- Tercera capa completamente conectada: Posee 128 unidades y se activa con ReLU. Capa de eliminación adicional.

Capa de salida: Tiene cuatro unidades que corresponden a las clases de salida *(EOSINOPHIL, LYMPHOCYTE, MONOCYTE y NEUTROPHIL)*. La activación utilizada es de suavización *(softmax)* para obtener probabilidades.

La figura 2.3 presenta el diseño de la red neuronal, obtenida del paquete de la red implementada en python.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 118, 118, 16)	448
conv2d_1 (Conv2D)	(None, 116, 116, 16)	2320
batch_normalization (BatchNormalization)	(None, 116, 116, 16)	64
max_pooling2d (MaxPooling2D)	(None, 58, 58, 16)	0
conv2d_2 (Conv2D)	(None, 56, 56, 32)	4640
conv2d_3 (Conv2D)	(None, 54, 54, 32)	9248
batch_normalization_1 (BatchNormalization)	(None, 54, 54, 32)	128
max_pooling2d_1 (MaxPooling2D)	(None, 27, 27, 32)	0
dropout (Dropout)	(None, 27, 27, 32)	0
conv2d_4 (Conv2D)	(None, 25, 25, 64)	18496
batch_normalization_2 (BatchNormalization)	(None, 25, 25, 64)	256

Layer (type)	Output Shape	Param #
max_pooling2d_2 (MaxPooling2D)	(None, 12, 12, 64)	0
dropout_1 (Dropout)	(None, 12, 12, 64)	0
conv2d_5 (Conv2D)	(None, 10, 10, 128)	73856
batch_normalization_3 (BatchNormalization)	(None, 10, 10, 128)	512
max_pooling2d_3 (MaxPooling2D)	(None, 5, 5, 128)	0
dropout_2 (Dropout)	(None, 5, 5, 128)	0
conv2d_6 (Conv2D)	(None, 3, 3, 256)	295168
batch_normalization_4 (BatchNormalization)	(None, 3, 3, 256)	1024
max_pooling2d_4 (MaxPooling2D)	(None, 1, 1, 256)	0
dropout_3 (Dropout)	(None, 1, 1, 256)	0

Layer (type)	Output Shape	Param #
flatten (Flatten)	(None, 256)	0
dense (Dense)	(None, 256)	65792
dropout_4 (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 128)	32896
dropout_5 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 128)	16512
dropout_6 (Dropout)	(None, 128)	0
dense_3 (Dense)	(None, 4)	516

Total params: 521,876
Trainable params: 520,884
Non-trainable params: 992

Figura 2.3: Descripción del modelo [Fuente: Autor].

La figura 2.4 es un ejemplo de las imágenes de entrada utilizadas en esta red neuronal para la clasificación. Estas imágenes son redimensionadas para que tengan un formato de 120x120 píxeles y están en formato RGB. El tamaño de 120x120 píxeles indica la resolución de la imagen y RGB se refiere al modelo de color usado para representar los colores en la imagen.

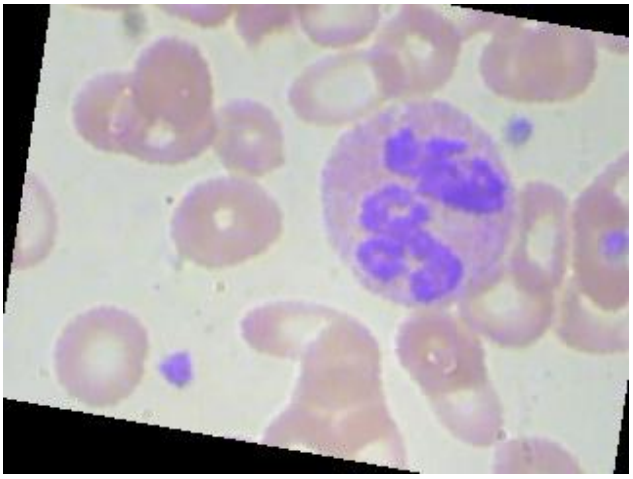

Figura 2.4: Imagen de entrada [45].

Antes de comenzar el entrenamiento, se realizan una serie de pasos para preparar los datos de imagen, se empieza por redimensionar, normalizar y convertir los datos a una matriz. Estos pasos son importantes para garantizar que estén en un adecuado formato y listos para ser procesados por el modelo de entrenamiento. Al redimensionar, normalizar y convertir los datos a una matriz, se mejora el rendimiento

y se facilita la manipulación de los datos durante el entrenamiento. A continuación se muestra el código empleado para realizar estos pasos:

```
image = cv2.resize(image, [120,120])
X.append(image/255.0)
y.append(label)
X = np.array(X, dtype = 'float32')
y = np.array(y, dtype = 'int32')
```

Así mismo, se crea un generador de datos de imágenes que realiza una variedad de transformaciones aleatorias en las imágenes, como rotaciones, desplazamientos, cortes, zoom y volteos, su objetivo es refinar los sistemas de aprendizaje automático utilizando datos de entrenamiento adicionales. Esto se implementa con el código mostrado a continuación:

```
    datagen = ImageDataGenerator(
  rotation_range=20,
  width_shift_range=0.1,
  height_shift_range=0.1,
  shear_range=0.1,
  zoom_range=0.2,
  horizontal_flip=True,
  vertical_flip=True
)
```

Luego se procede con la configuración, puntos de control, detención temprana y disminución gradual de la velocidad de aprendizaje durante el entrenamiento de un modelo en Keras. Estas técnicas ayudan a guardar los mejores pesos del modelo, parar el entrenamiento en caso de estancamiento y reducir la tasa de aprendizaje cuando se estanca el progreso. Para esto se implementa el siguiente código:

```
  checkpoint = ModelCheckpoint(
      filepath='model-{epoch:02d}-{val_accuracy:.2f}.h5',
      save_weights_only=True,
      monitor='val_accuracy',
      mode='max',
      save_best_only=True)
earlystop = EarlyStopping(
      monitor='val_loss', patience=3, restore_best_weights=True)
lrReduction = ReduceLROnPlateau(
    monitor = 'val_accuracy',
    patience = 2,
    verbose = 1,
    factor = 0.3,
    min_lr = 0.00001)
```

Al finalizar el entrenamiento, que duró alrededor de una hora en total, se logró una precisión del 78 % como se muestra en la tabla 2.2 . Este valor de precisión muestra un avance notable con respecto a la técnica previamente revisada. Cada época de entrenamiento tomó aproximadamente 3 minutos y medio, y se realizaron 24 épocas en total.

Tabla 2.2: Pérdida y Precisión Obtenida con la Técnica 2.

Perdida del Modelo :	1.065824389
Precisión del Modelo:	78.55227589 %

El modelo actual fue entrenado con muestras de cuatro tipos diferentes de glóbulos blancos: eosinófilos, monocitos, linfocitos y neutrófilos. A diferencia del modelo anterior, este nuevo modelo tiene un mayor número de capas convolucionales para el entrenamiento. Como resultado, se observó un aumento significativo en la precisión del modelo. Al analizar la figura 2.5, se puede apreciar que al principio del entrenamiento, se produce una pérdida en la precisión del modelo. Esto significa que en las primeras etapas, el modelo no logra predecir con alta exactitud y su rendimiento es relativamente bajo. Sin embargo, mientras el entrenamiento continúa,

se puede apreciar un aumento gradual en la precisión del modelo. Esto hace notar que el modelo está perfeccionando su desempeño a partir de los datos y mejorando sus parámetros internos para optimizar su capacidad de hacer predicciones precisas. Con cada época de entrenamiento, la precisión aumenta y el modelo se vuelve más competente en la tarea de clasificación de los glóbulos blancos.

La mayor precisión obtenida muestra que la red entrenada detecta con mayor precisión diferentes tipos de glóbulos blancos en comparación con el modelo anterior. Esto sugiere que las capas convolucionales adicionales permitieron una representación más efectiva de las imágenes de los glóbulos blancos, lo que llevó a una mejora en la habilidad del modelo para discernir entre los diferentes tipos celulares.

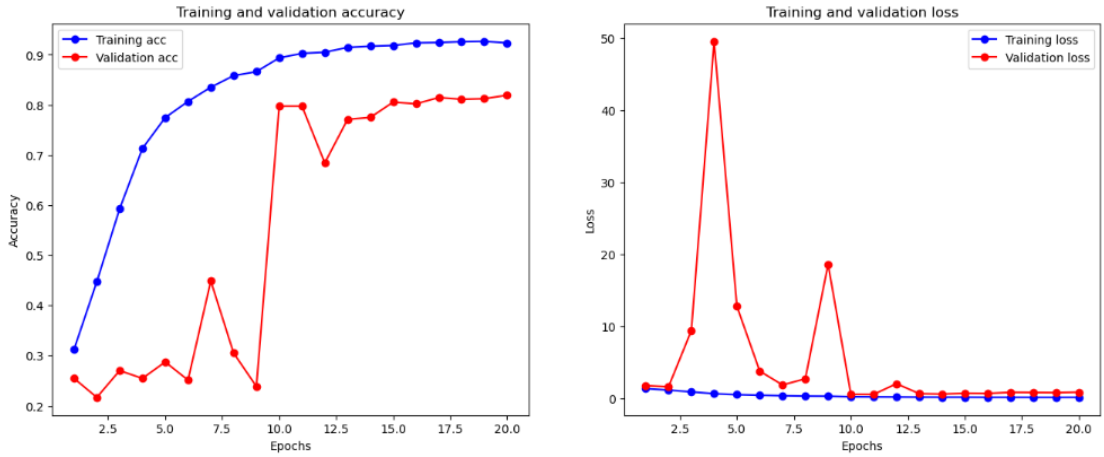

Figura 2.5: Gráficas de perdida y precisión del modelo [Fuente: Autor].

Clasificación de siete tipos de células sanguíneas con CNN (Técnica 3)

Este modelo describe el entrenamiento y la evaluación de una arquitectura de CNN con el propósito de clasificar imágenes que representan diversos tipos de células sanguíneas en 7 categorías distintas. Se empleó un banco de datos formado por más de 12.000 imágenes que habían sido previamente etiquetadas y correspondían a diferentes tipos de células sanguíneas. El modelo posee capas de convolución, de agrupamiento y densas totalmente conectadas. A continuación me detalla cada capa empleada en esta técnica.

- Capa de entrada: Esta capa acepta imágenes de 300x300 píxeles en (RGB).
- Primera capa convolucional: con 32 filtros de 3x3 y se activa con ReLU.

- Capa de agrupación: Se aplica para disminuir la resolución espacial, seleccionando el valor máximo en cada ventana.
- Otra capa convolucional: con filtros similares y activación ReLU.
- Otra capa de agrupación: Se utiliza nuevamente para disminuir aún más la resolución espacial.
- Capa de eliminación: Se introduce para evitar el sobreajuste, desactivando aleatoriamente una parte de las unidades de entrada.
- Capa densa: Se crea esta capa con 128 unidades y activación ReLU.
- Capa de aplanamiento: Alista la información de entrada para las capas densamente conectadas.
- Capa de salida: Posee un número de unidades, que representa las clases que se deben clasificar.

La figura 2.6 presenta un resumen de la arquitectura de la red neuronal.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
rescaling (Rescaling)        (None, 300, 300, 3)       0
_________________________________________________________________
conv2d (Conv2D)              (None, 298, 298, 32)      896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 149, 149, 32)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 147, 147, 32)      9248
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 73, 73, 32)        0
_________________________________________________________________
dropout (Dropout)            (None, 73, 73, 32)        0
_________________________________________________________________
dense (Dense)                (None, 73, 73, 128)       4224
_________________________________________________________________
flatten (Flatten)            (None, 682112)            0
_________________________________________________________________
dense_1 (Dense)              (None, 8)                 5456904
=================================================================
Total params: 5,471,272
Trainable params: 5,471,272
Non-trainable params: 0
_________________________________________________________________
```

Figura 2.6: Descripción del modelo [Fuente: Autor].

La figura 2.7 es un ejemplo de las imágenes empleadas en la fase de aprendizaje de la red, provienen de una base de datos público de Kaggle, las imágenes cambian su tamaño a 300 x 300 píxeles.

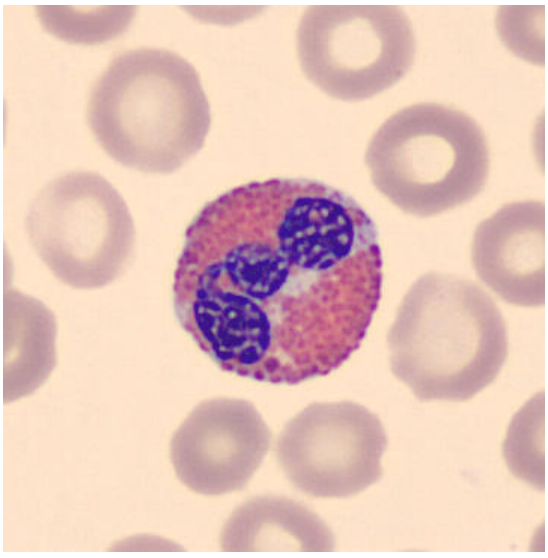

Figura 2.7: Imagen de entrada [26].

Se extrae el 80 % de las imágenes para entrenamiento del modelo CNN que realizará en épocas e iteraciones. Para esto se emplea el siguiente código:

```
train_ds = tf.keras.utils.image_dataset_from_directory(
data_directory,
validation_split=0.2,
subset="training",
seed=123,
image_size=(img_height, img_width),  batch_size=batch_size)
```

El conjunto de entrenamiento separado del de validación asegura que no haya contaminación entre ambos y se pueda evaluar el desempeño real.

Para validación se separa el 20 % de las imágenes. Estos datos no se usan en el entrenamiento. El código necesario para esto, es el siguiente:

```
val_ds = tf.keras.utils.image_dataset_from_directory(
data_directory,
validation_split=0.2,
subset="validation",
seed=123,
image_size=(img_height, img_width),
batch_size=batch_size))
```

Esto permite monitorear métricas como precisión durante el entrenamiento para detectar sobreajuste y determinar cuándo detener el entrenamiento.

Se aplica el optimizador *Adam* para modificar gradualmente los pesos del modelo para minimizar la pérdida de categoría. La función de perdida es apropiada para situaciones que abordan situaciones de identificación de varias clases, y cuando las etiquetas toman valores categóricos. Estos parámetros se ajustan de esta manera:

```
model.compile(
optimizer='adam',
loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
  metrics=['accuracy'])
```

Se entrenó un modelo durante 15 épocas, cada una con 855 muestras. Se empleó validación cruzada para evaluar la pérdida y la precisión en los datos de entrenamiento y en los de validación. En el entrenamiento la pérdida disminuyó desde 1.07 en la primera época hasta 0.05 en la época 15, lo cual señala que el sistema adquirió un aprendizaje efectivo. Con respecto a la precisión de entrenamiento, esta creció de 0.66 a 0.98, mostrado en la figura 2.8, confirmando un buen ajuste en el entrenamiento. Obteniendo un 0.87 de validación de precisión.

```
855/855 [==============================] - 64s 65ms/step - loss: 1.0739 - accuracy: 0.6639 - val
_loss: 0.5758 - val_accuracy: 0.7820
Epoch 2/15
855/855 [==============================] - 42s 49ms/step - loss: 0.4137 - accuracy: 0.8609 - val
_loss: 0.4702 - val_accuracy: 0.8327
Epoch 14/15
855/855 [==============================] - 40s 47ms/step - loss: 0.0465 - accuracy: 0.9843 - val
_loss: 0.6217 - val_accuracy: 0.8625
Epoch 15/15
855/855 [==============================] - 40s 47ms/step - loss: 0.0513 - accuracy: 0.9822 - val
_loss: 0.5763 - val_accuracy: 0.8847
```

Figura 2.8: Valor de precisión obtenida [Fuente: Autor].

En conjunto, la reducción de la pérdida en el entrenamiento y el aumento en la precisión reflejan el progreso del aprendizaje del modelo. Las métricas en validación son buenas, pero podrían mejorar la capacidad de generalización, probablemente requiriendo más datos de validación o ajustando la arquitectura.

Analizando la figura 2.9 que muestra los valores de pérdida, se puede observar cómo la pérdida de entrenamiento comienza alrededor de 1.2 y disminuye constantemente hasta aproximadamente 0.1 en la época 15. Esto indica que el modelo está captando efectivamente las pautas presentes en datos de entrenamiento. Mientras que la pérdida de validación comienza en 0.6 y su disminución es más irregular, fluctuando entre 0.6 y 0.4. También se observan en las épocas 8 y 10, picos de crecimiento. A pesar de estas fluctuaciones, la tendencia general sigue siendo

decreciente, lo cual es una señal positiva.

A lo largo del entrenamiento, las pérdidas de entrenamiento y validación se mantienen relativamente constantes. Esto sugiere que no hay un exceso de ajuste significativo, ya que ambas curvas disminuyen. Sin embargo, persiste una brecha considerable entre ambas pérdidas.

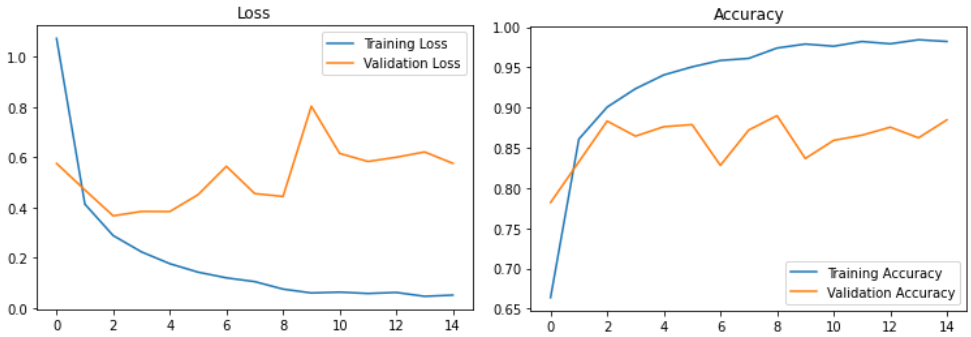

Figura 2.9: Gráficas de perdida y precisión del modelo [Fuente: Autor].

En lo que se refiere al gráfico de precisión (Fig. 2.9), el rendimiento en entrenamiento comienza en 0.66 y aumenta de manera constante hasta casi 0.95 en la última época. Esto confirma que está aprendiendo correctamente las pautas de los datos de entrenamiento. Por su parte, la precisión en validación inicia en 0.76 y sigue una tendencia similar, aunque con variabilidad, oscilando entre 0.76 y 0.9.

Esto sugiere que el modelo no está logrando generalizar de manera óptima hacia nuevos datos no vistos durante el entrenamiento.

Reconocimiento de 7 tipos de células usando CNN (Técnica 4)

El objetivo de esta técnica es construir un modelo para clasificar imágenes de 7 tipos diferentes de células sanguíneas (basófilos, eosinófilos, eritroblastos, linfocitos, monocitos, neutrófilos y plaquetas).

Esta red posee capas de convolución, de agrupación, densas y de salida con función de suavización. Se procederá a examinar con detalle las capas que conforman esta red neuronal.

- Capa de entrada del modelo con una forma de (300, 300, 3), que representa imágenes de 300x300 píxeles con 3 canales de color (RGB).

- Capa convolucional con 32 filtros de 3x3, se activa con ReLU y un relleno para mantener la forma de entrada. Se utiliza la misma forma de entrada especificada en la capa de entrada.
- Capa de agrupación de 2x2.
- Capa que aplana la información de entrada para ser procesada por las capas densamente conectadas.
- Capa densa de 64 unidades y se activa con ReLU.
- Otra capa densa de 64 unidades y se activa con ReLU.
- Capa de salida del modelo, con 8 unidades y se activa con *softmax* que genera una disposición de posibilidad entre las 7 posibles salidas.

La figura 2.10 presenta un resumen del diseño de la red neuronal explicada previamente.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 300, 300, 32)      896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 150, 150, 32)      0
_________________________________________________________________
flatten (Flatten)            (None, 720000)            0
_________________________________________________________________
dense (Dense)                (None, 64)                46080064
_________________________________________________________________
dense_1 (Dense)              (None, 64)                4160
_________________________________________________________________
dense_2 (Dense)              (None, 8)                 520
=================================================================
Total params: 46,085,640
Trainable params: 46,085,640
Non-trainable params: 0
_________________________________________________________________
```

Figura 2.10: Descripción del modelo [Fuente: Autor].

Al igual que la técnica 3 se usa la misma base, compuesta por más de 12.000 imágenes que habían sido previamente etiquetadas y correspondían a diferentes tipos

de células sanguíneas. La figura 2.9 pertenece al conjunto de datos tomada.

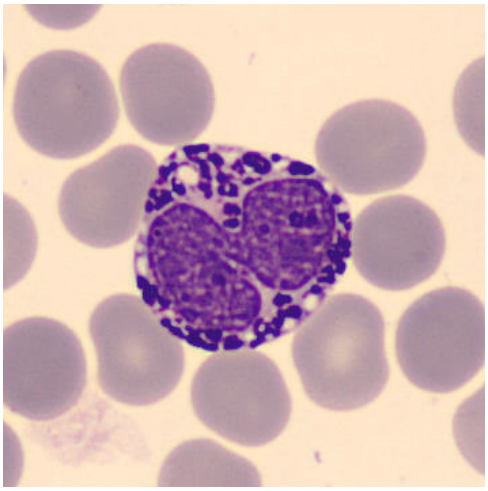

Figura 2.11: Imagen de entrada [26].

En esta técnica utiliza un código que diseña un objeto con el propósito de preprocesar y generar lotes de imágenes destinadas al entrenamiento. Se especifica la orden de normalizar las imágenes al dividir sus valores de píxeles por 255, con el objetivo de llevar estos valores a un rango entre 0 y 1. También se especifica que el 30 % de las imágenes se reservará para tareas de validación.

La función *flow_from_directory* es usada posteriormente para crear la base de datos de entrenamiento. Esta función toma como entrada un directorio predeterminado, carga las imágenes presentes en él y las redimensiona a dimensiones de 300x300 píxeles. Además, las etiquetas se convierten en vectores categóricos, el orden de las imágenes se mezcla y se asignan nombres a las clases correspondientes.

En la etapa de creación de los datos, se configura el tamaño de lote en 32 imágenes, y se especifica que solamente se deben emplear las imágenes de entrenamiento, excluyendo aquellas destinadas a validación. Todos estos pasos se los realiza con el siguiente código:

```
image_generator = ImageDataGenerator(rescale=1.0/255,
validation_split=0.3)
train_dataset = image_generator.flow_from_directory(batch_size=32,
 directory='/input/blood-cells-image-dataset/bloodcells_dataset',
        target_size=(300, 300),
        subset="training",
        class_mode='categorical',
        shuffle=True,
        classes=('basophil',
          'eosinophil',
          'erythroblast',
          'lymphocyte',
          'monocyte',
          'neutrophil',
          'platelet') )
```

De manera similar al conjunto de entrenamiento, para la validación, se indica que las imágenes deben ser leídas desde el directorio de datos y redimensionadas a un tamaño de 300x300 píxeles. Además, las etiquetas se convierten en vectores categóricos.

La diferencia es que aquí se configura el nombre como *validation*, lo que implica que solo se seleccionarán las imágenes de validación que habían sido previamente separadas. Se determina un tamaño de lote de 32 imágenes para el procesamiento, las imágenes son mezcladas y se mantienen los mismos nombres de clases. De igual manera el código es el siguiente:

```
validation_dataset=image_generator.flow_from_directory(batch_size=32,
 directory='/input/blood-cells-image-dataset/bloodcells_dataset',
        target_size=(300, 300),
        subset="validation",
        class_mode='categorical',
        shuffle=True,
        classes=('basophil',
            'eosinophil',
            'erythroblast',
            'lymphocyte',
            'monocyte',
             'neutrophil',
            'platelet'))
```

Se entrenó el modelo durante 15 iteraciones del entrenamiento de una red neuronal. En cada iteración, se entrenan 374 lotes y se registran medidas de pérdida y precisión para ambos conjuntos después de cada iteración.

Inicialmente, cuando los pesos son aleatorios, la pérdida es alta (3.45) y la precisión es baja (0.58) en la base de entrenamiento. Mientras avanza el entrenamiento, disminuye la pérdida y exactitud aumenta, lo que sugiere que el modelo está capturando los aspectos de las imágenes. La pérdida y la exactitud en validación también muestran mejoras, aunque a un ritmo más gradual que en el de entrenamiento. Esto señala que el sistema está generalizando efectivamente y está funcionando eficazmente.

En la iteración 15, el conjunto de entrenamiento, muestra una pérdida de 0.08 y una exactitud del 97 %, mientras que en el conjunto de validación, estos valores son de 0.86 y 0.82 (Fig.2.12). Estos valores indican que el modelo es sólido y no está sufriendo de sobre ajuste.

```
374/374 [==============================] - 193s 498ms/step - loss: 3.4576 - accuracy: 0.5847 - val_loss: 0.7516 - val_accuracy:
0.7241
Epoch 2/15
374/374 [==============================] - 89s 239ms/step - loss: 0.5617 - accuracy: 0.8027 - val_loss: 0.6392 - val_accuracy: 0.
```

```
Epoch 14/15
374/374 [==============================] - 89s 239ms/step - loss: 0.0584 - accuracy: 0.9826 - val_loss: 0.6203 - val_accuracy: 0.
8521
Epoch 15/15
374/374 [==============================] - 87s 233ms/step - loss: 0.0821 - accuracy: 0.9730 - val_loss: 0.8602 - val_accuracy: 0.
8220
```

Figura 2.12: Valores de precisión obtenido [Fuente: Autor].

Durante el proceso de entrenamiento, la figura ilustra un rápido aumento en la precisión inicial en las primeras etapas, llegando a aproximadamente un 95 %. No obstante, después de ese punto, no se ve una mejora. En cuanto a la gráfica de pérdida durante el entrenamiento, se observa una disminución en las etapas iniciales, llegando a un valor cercano a 0.1, y posteriormente permanece constante.

También, la figura muestra que la tasa de acierto en la validación sigue un patrón similar a la del entrenamiento, con un ascenso rápido al comienzo que llega alrededor del 90 % de precisión, seguido de un crecimiento más lento. La representación gráfica de la pérdida en la validación también experimenta una caída rápida al inicio, alcanzando aproximadamente 0.2, y luego decrece de manera gradual.

La figura 2.13 revela que la red se adapta adecuadamente, logrando elevadas tasas de precisión y reducidas pérdidas en ambas bases de datos. La presencia de sobre ajuste parece ser mínima, dado que no se aprecia una diferencia sustancial entre las métricas de entrenamiento y validación. Las gráficas en validación muestran mejoras leves cuando aumentan las etapas, mostrando que la red podría beneficiarse de un entrenamiento adicional.

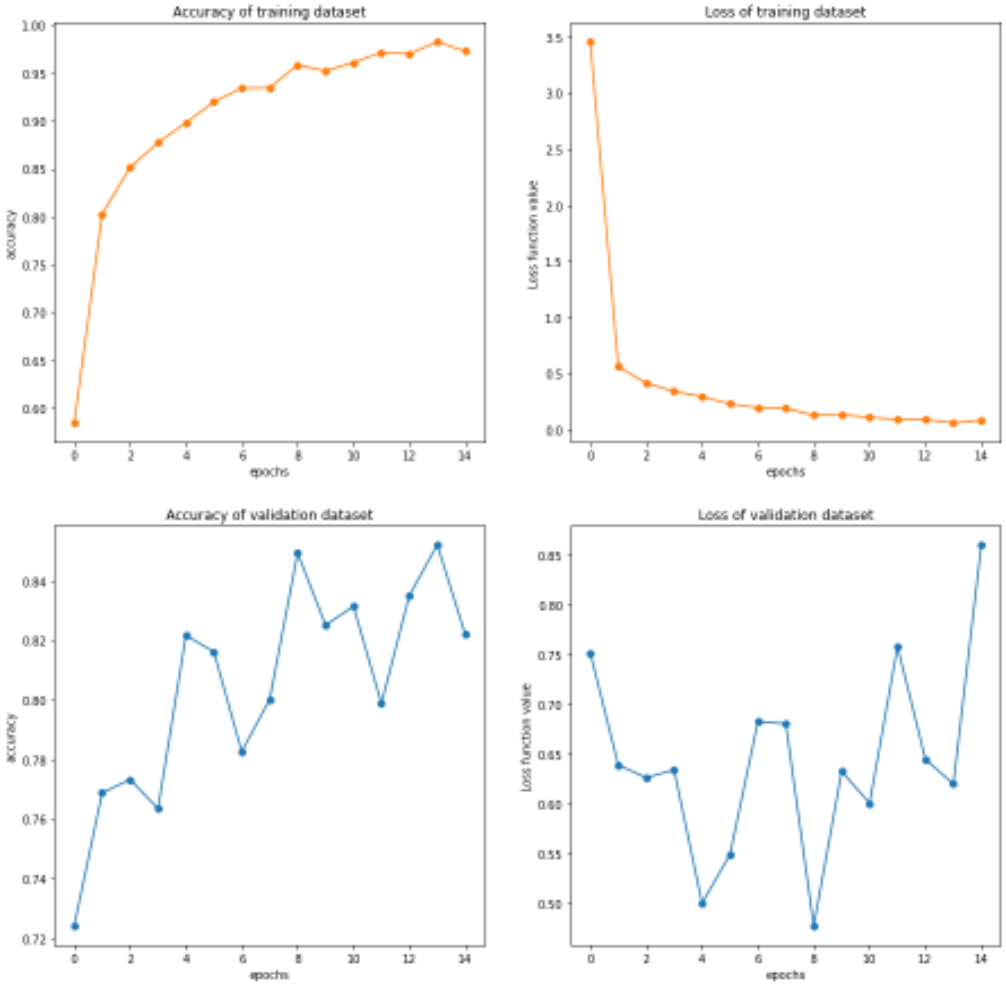

Figura 2.13: Gráficas de perdida y precisión de la técnica 4 [Fuente: Autor].

Clasificación glóbulos blancos con PyTorch ResNet18 (Técnica 5)

Esta técnica se basa en el modelo ResNet18. Es uno de los modelos ResNet más simples y se compone de 18 capas en total. Está formado por varias capas de convolución, de agrupación y totalmente conectadas para la clasificación final. Cada bloque residual en ResNet18 está conformada de dos capas de convolucionales con un enlace que añade la entrada a la salida de las capas convolucionales. Esto permite que la red aprenda los cambios residuales en las características.

Para trabajar con este modelo se necesitan varias librerías de PyTorch:

```
import torch.nn as nn
import torch.nn.functional as F
from tourchvision.models import ResNet18_Weights
```

La definición del modelo es la siguiente:

```
class Net(nn.Module):
    def __init__(self, freeze=True):
        super(Net, self).__init__()
        self.resnet = torchvision.models.resnet18(
        weights=ResNet18_Weights.DEFAULT)
        if freeze:
            self.freeze_resnet()
        output_features = self.resnet.fc.in_features
        self.resnet.fc = nn.Linear(output_features, 4)

    def freeze_resnet(self):
        for param in self.resnet.parameters():
            param.requires_grad = False

    def forward(self, x):
        x = self.resnet(x)
        return x
```

A continuación se describe la definición y el funcionamiento del modelo REsNet18.

- class Net(nn.Module): Define una nueva clase llamada Net que hereda de nn. Module, la clase base para todas las definiciones de modelos de PyTorch.
- def __init__(self, freeze=True): Define la forma en que se genera una instancia de la clase Net. Tiene el argumento opcional congelar, que controla si se congelarán los pesos de la ResNet-18.

- self.resnet = torchvision.models.resnet18(weights=ResNet18_Weights.DEFAULT): Aquí sé instancia el modelo ResNet-18 desde torchvision.models, utilizando los pesos preentrenados por defecto (en inglés, "weights=ResNet18_Weights.DEFAULT"). Esto genera el componente principal del modelo, que será la base para el nuevo modelo personalizado.

- if freeze: Esto verifica si la opción congelado es verdadera. Si es verdadera, llama al método self.freeze_resnet().

- self.freeze_resnet(self): Este método itera a través de los parámetros del modelo ResNet-18 y establece requires_grad en Falso para congelar los pesos y evitar que se actualicen durante el entrenamiento.

- output_features = self.resnet.fc.in_features: Aquí se obtiene la cantidad de características de entrada de la capa completamente conectada en la ResNet-18.

- self.resnet.fc = nn.Linear(output_features, 4): Reemplaza la capa completamente conectada existente en ResNet-18 con una nueva capa completamente conectada con 4 unidades de salida. Esto adapta el modelo a una tarea que implica clasificación en 4 clases diferentes.

- def forward(self, x): Define cómo se propagan los datos a través del modelo.

- x = self.resnet(x): Aplica la arquitectura de ResNet-18 al tensor de entrada x para extraer características.

- return x: Devuelve el tensor de salida después de pasar por la capa completamente conectada con 4 unidades.

Este código define un modelo personalizado basado en ResNet-18 con la capacidad de congelar sus pesos si es necesario y adaptar la capa completamente conectada para la clasificación en 4 clases.

El entrenamiento del modelo fue realizado con un total de 30 épocas, se calcula y se muestra varias métricas de rendimiento (pérdida, precisión y puntaje, F1 macro) después de completar una época de entrenamiento y validación del modelo. Estas

métricas ayudan a evaluar qué tan bien se está desempeñando el modelo en la tarea de clasificación.

Luego de culminadas todas las épocas se obtienen gráficas de comparación de valores de perdida y precisión de cada época (Fig. 2.14, 2.15).

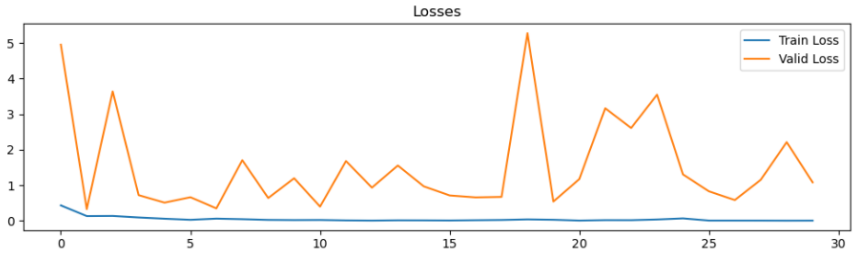

Figura 2.14: Gráfica de pérdida en cada época [Fuente: Autor].

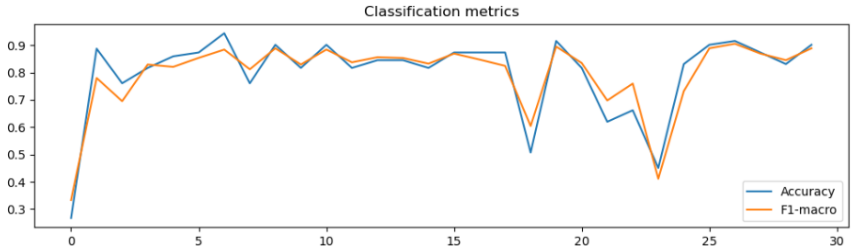

Figura 2.15: Gráfica de los valores de métricas obtenidos en cada época [Fuente: Autor].

Al finalizar el entrenamiento de este modelo de red neuronal diseñado para clasificar glóbulos blancos en cuatro clases diferentes, se calcula un promedio de perdida y precisión en validación. Estas métricas ofrecen una evaluación de sí, la red está realizando la clasificación en datos no vistos previamente. La pérdida promedio muestra la desviación de la predicción de la red de las etiquetas reales en promedio, mientras que la precisión muestra las predicciones correctas en comparación con las predicciones ejecutadas en el conjunto de validación. Estas son fundamentales para analizar el rendimiento y ajustar la red en el proceso de entrenamiento.

Tabla 2.3: Precisión Obtenida con la Técnica 5.

Perdida del Modelo :	1.3418196134
Precisión del Modelo:	87.615601 %
Macro F1:	0.8792936578

Recuento y detección de células sanguíneas basado en YOLOV5 (Técnica 6)

YOLO v5 es una versión más reciente y optimizada de la serie YOLO. Aunque no está oficialmente relacionada con los originales desarrolladores, este modelo continúa con el enfoque de clasificación de objetos, pero con mejoras significativas en cuanto a precisión y rapidez en comparación con anteriores versiones.

Este enfoque comienza descargando el repositorio oficial de YOLOv5 desde GitHub, al entorno en el que se está ejecutando el programa. Al hacerlo, se adquieren todos los archivos y recursos esenciales para utilizar y desarrollar con la arquitectura YOLOv5 en este entorno. Esta descarga se hace con el siguiente código:

```
git clone  'https://github.com/ultralytics/yolov5.git'
```

Para poder usar YOLOv5, es necesario instalar y cargar una variedad de bibliotecas que son requeridas por el modelo. Estas bibliotecas son esenciales para que YOLOv5 funcione correctamente. Esto se lo realiza con el siguiente código:

```
import os, sys, random, shutil
from glob import glob
import pandas as pd
from shutil import copyfile
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
from matplotlib import patches
import numpy as np
import os
```

Al descargar el modelo de YOLOv5, se obtendrá un archivo llamado "train.py". Este archivo es una parte importante del proceso, es un código que permite ajustar los parámetros necesarios para que el modelo pueda aprender a detectar objetos en tiempo real, en este caso, células sanguíneas y sus tipos específicos.

Para entrenar el modelo, se necesitan pasar algunos parámetros

1. Ruta de tren.py
2. –img <tamaño de imagen (int)>
3. –batch <tamaño del lote (int)>
4. –épocas <No. de Épocas a entrenar (int) >
5. –data <Ruta del archivo yaml de su conjunto de datos>
6. –cfg <ruta del archivo yaml de su modelo yolo preferido>
7. –name <"nombre de su modelo guardado después del entrenamiento»

```
python /yolov5/train.py
--img 416 --batch 8 --epochs 100
--data/yolov5/bcc-kaggle.yaml
--cfg /yolov5/models/yolov5m.yaml --name BCCM
```

En cada etapa de entrenamiento, se proporciona información sobre la precisión (P), la recuperación (R) y el promedio de precisión del área bajo la curva (mAP) para diferentes clases de objetos. También se mencionan la cantidad de imágenes y etiquetas en la base de datos, los resultados obtenidos se muestran en la tabla 2.4

Tabla 2.4: Resultados de Métricas Obtenidas en la Técnica 6.

Época 0/2 :	0.237	Recall:0.258	mAP: 0.0086
Época 1/2 :	0.2239	Recall:0.205	mAP: 0.00263
Época 2/2 :	0.2214	Recall:0.206	mAP: 0.00609

Recuento y detección de células sanguíneas basado en YOLOV8 (Técnica 7)

YOLOv8 es la versión más reciente del modelo YOLO (*del inglés You Only Look Once*) desarrollado por Ultralytics. Este modelo es diseñado para ampliar y mejorar las capacidades de las versiones anteriores de YOLO.

Cuando se lleva a cabo una clasificación utilizando el modelo YOLOV8, el primer paso importante es revisar el archivo con extensión .yaml. Este archivo contiene información importante, como la ubicación de las imágenes, la cantidad de etiquetas disponibles y los nombres asociados a esas etiquetas como se muestra a continuación.

```
train: ../train/images
val: ../valid/images
nc: 3
names: ['Platelets', 'RBC', 'WBC']
```

Examinar y configurar el tamaño de la imagen es un paso importante por algunas razones.

Impacto en el rendimiento del modelo: el tamaño de la imagen puede afectar significativamente el rendimiento de los modelos. Los modelos entrenados en imágenes más pequeñas pueden tener dificultades para clasificar elementos pequeños.

Restricciones de memoria: entrenar modelos de clasificación en imágenes grandes puede requerir mucha memoria, lo que puede ser una limitación para algunas configuraciones de hardware.

Capacidad de generalización: los modelos generalmente se entrenan en un rango específico de tamaños de imagen y es posible que no funcionen bien en imágenes fuera de ese rango.

El siguiente código genera un conjunto de resoluciones únicas (ancho y alto) para una lista de imágenes.

Este código usa la biblioteca OpenCV para leer imágenes de una lista de rutas de imágenes, extraer su resolución (ancho y alto) y agregarla a una lista llamada resoluciones.

Luego, el código crea un conjunto de resoluciones únicas a partir de la lista de resoluciones. Un conjunto es una colección de elementos únicos, por lo que cualquier resolución duplicada en la lista se eliminará del conjunto.

```
resolutions = []
for image_path in image_paths:
    img = cv2.imread(str(image_path))
    h, w, _ = img.shape
    resolutions.append((w, h))
unique_resolutions = set(resolutions)
```

Luego de tener definido el tamaño de cada imagen se genera un conjunto de imágenes seleccionadas aleatoriamente con cuadros delimitadores y etiquetas de clase basadas en los archivos de etiquetas correspondientes, un ejemplo de esto esta en la figura 2.16.

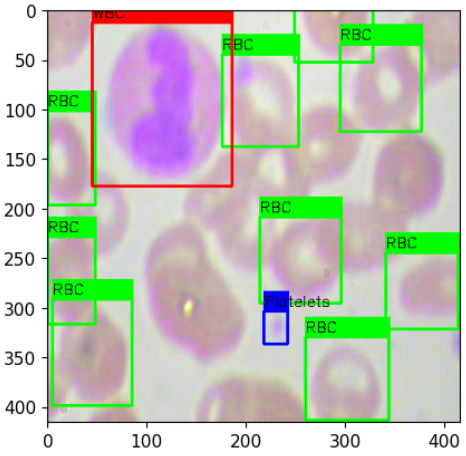

Figura 2.16: Imagen de entrada con cajas delimitadoras [Fuente: Autor].

Una vez completada la revisión y configuración del conjunto de imágenes de entrada, el siguiente paso consiste en descargar el modelo YOLOV8 desde GitHub para su posterior utilización en la tarea de clasificación de células sanguíneas, para esto se implementa el siguiente código.

```
baseline_model = YOLO('yolov8x.pt')
```

Una vez descargado el modelo, se procede con el entrenamiento, para el cual se deben configurar los siguientes parámetros:

```
params = { 'data':'data.yaml', 'imgsz':416,
    'epochs':25, 'optimizer':'Adam',
    'pretrained':True, 'lr0':0.1,
    'lrf': 0.00001, 'cos_lr': True}
results = baseline_model.train(**params)
```

Una vez culminado el entrenamiento se presentan los resultados alcanzados

luego de 25 épocas de entrenamiento en la tabla 2.5.

Tabla 2.5: Resultados de Métricas Obtenidas en la Técnica 7.

General:	Precisión:	90.1 %	Recall:0.787	mAP: 0.438
Plaquetas:	Precisión:	72.4 %	Recall:0.683	mAP: 0.284
RBC:	Precisión:	99.3 %	Recall:0.756	mAP: 0.443
WBC:	Precisión:	98.6 %	Recall:0.923	mAP: 0.586

2.2. Técnicas Seleccionadas de Clasificación

En esta sección, se profundiza el proceso de selección que se realizó entre una amplia variedad de aproximadamente 25 métodos distintos. Cabe destacar que en la sección previa, se abordaron con detalle 7 de las 25 técnicas analizadas. Las técnicas restantes, aunque no se presentaron en esa sección, comparten similitudes sustanciales en términos de estructura y funcionamiento.

Esta fase de selección no solo resalta la importancia de encontrar enfoques efectivos, sino que también la necesidad de optar por aquellos modelos que mejor se adapten al contexto de clasificación de células sanguíneas. A través de un riguroso análisis comparativo, se busca las dos técnicas más prometedoras para abordar este desafío.

2.2.1. Primer técnica seleccionada: Clasificación de células sanguíneas en cinco tipos con redes neuronales convolucionales

Entre las diversas alternativas de técnicas de clasificación evaluadas, se ha optado por esta técnica, en particular que destaca por varias cualidades superiores. En primer lugar, esta técnica sobresale por su precisión, superando a las demás opciones en términos de resultados confiables y exactitud en la clasificación. Además, su tiempo de entrenamiento optimizado contribuye significativamente a un proceso más eficiente y ágil.

Uno de los aspectos más destacados de esta técnica es la habilidad para clasificar las células sanguíneas en cinco categorías distintas, a comparación de

las demás técnicas que solo clasifican en cuatro tipos de glóbulos blancos. Esta habilidad para categorizar con precisión una amplia variedad de tipos celulares no solo demuestra su robustez, sino que también brinda una eficacia mejorada al sistema.

Esta técnica construye un modelo automatizado para clasificar los 5 tipos diferentes de células sanguíneas (basófilos, eosinófilos, linfocitos, monocitos, y neutrófilos).

A esta primera técnica seleccionada se le aplicó un proceso de refinamiento y optimización con el objetivo de potenciar su desempeño y lograr avances significativos en la tarea de clasificación de células. A través de una serie de ajustes estratégicos y mejoras, se buscó elevar la calidad y precisión de los resultados obtenidos.

Mediante este enfoque de refinamiento, se examinaron detenidamente los componentes clave de la técnica y se implementaron modificaciones dirigidas a superar las limitaciones previas. Esto involucró la identificación y resolución de posibles puntos débiles que pudieran afectar la precisión de la clasificación.

La primera fase de mejora consiste en aplicar un proceso de preprocesamiento a las imágenes de entrada de la red. Esta medida se adoptó con el propósito de potenciar la capacidad del sistema para extraer de manera más efectiva las características más pertinentes de las imágenes durante el entrenamiento.

El primer paso del preprocesamiento es realizar un proceso de ecualización del histograma de la imagen, esto se lo realiza en escala de colores RGB, lo que permite mejorar el contraste y la distribución de intensidades en una imagen.

El proceso de ecualización de histograma se lo realiza con el siguiente código:

```
equalized = cv2.cvtColor(image, cv2.COLOR_BGR2YCrCb)
equalized[:, :, 1] = cv2.equalizeHist(equalized[:, :, 1])
equalized = cv2.cvtColor(equalized, cv2.COLOR_YCrCb2BGR)
```

El siguiente paso en el proceso de preprocesamiento implica la aplicación del filtro de la media. En este caso, se empleó un tamaño de kernel de 3x3 para su implementación. Con esta técnica se logra una suavización en la imagen al calcular

el valor promedio de intensidad en una región pequeña y adyacente de píxeles. Esto contribuye a reducir el ruido y las irregularidades presentes en la imagen, al tiempo que preserva las características y detalles esenciales de la misma.

El código empleado para realizar este filtro es el siguiente:

```
blurred = cv2.blur(equalized, (3, 3))
```

Continuando con el proceso, el tercer paso implica llevar a cabo la eliminación del fondo de la imagen. Esta fase es esencial en la mejora de la calidad como la importancia de los elementos de interés presentes en la imagen.

El código implementado es el siguiente:

```
gray = cv2.cvtColor(blurred, cv2.COLOR_BGR2GRAY)
threshold = cv2.threshold(gray,0,255,
    cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
resultado = cv2.bitwise_and(blurred, blurred, mask=threshold)
```

Para culminar con el proceso, se procedió a la aplicación de un filtro laplaciano como último paso. La inclusión de este filtro agrega un componente crucial para realzar aún más la claridad de la imagen.

La utilización del filtro laplaciano cumple un papel importante en la detección de bordes y características importantes presentes en la imagen. Al resaltar los cambios abruptos en la intensidad, el filtro laplaciano contribuye a destacar los contornos y detalles esenciales de los objetos en la imagen.

```
laplacian = cv2.Laplacian(resultado, cv2.CV_64F)
sharpened = cv2.convertScaleAbs(resultado - laplacian)
```

La figura 2.17 muestra un ejemplo de imagen prepocesada.

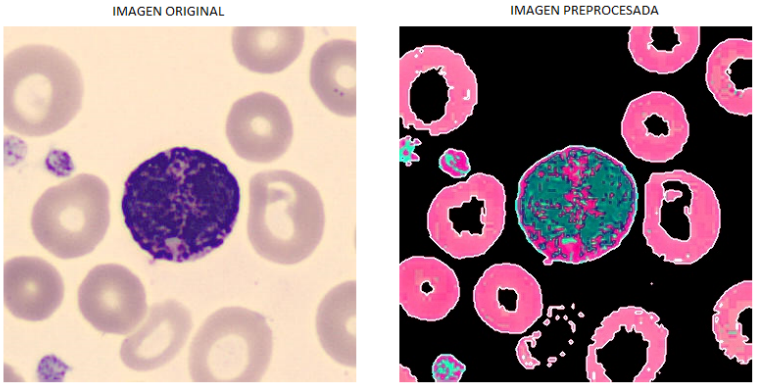

Figura 2.17: Comparación de la imagen original con la preprocesada [Fuente: Autor].

Ahora, una vez que todos estos pasos de preprocesamiento han sido completados, las imágenes se encuentran en un estado preprocesado y están listas para su uso como entradas en la red neuronal convolucional.

Este proceso de organización es fundamental para sostener la claridad y la coherencia en el conjunto de datos. Cada imagen preprocesada se aloja en su carpeta designada, identificada por la etiqueta que corresponde a la categoría a la que pertenece.

Una vez que se completó el proceso de preprocesamiento de las imágenes, estas se combinaron con las imágenes originales y otras bases de datos de imágenes. Esta unión se realizó con el fin de incrementar significativamente el número de imágenes disponibles para entrenar el modelo.

La integración de las imágenes preprocesadas con las originales y otras bases de datos contribuye a enriquecer y diversificar los de datos de entrenamiento. Con esto, se logra que la red neuronal convolucional tenga acceso a una gran variedad de ejemplos visuales, lo que puede mejorar su capacidad para generalizar y clasificar de manera más precisa en diversas situaciones.

Así luego de estos pasos previos, la figura 2.18, detalla el proceso aplicado para el entrenamiento de la red.

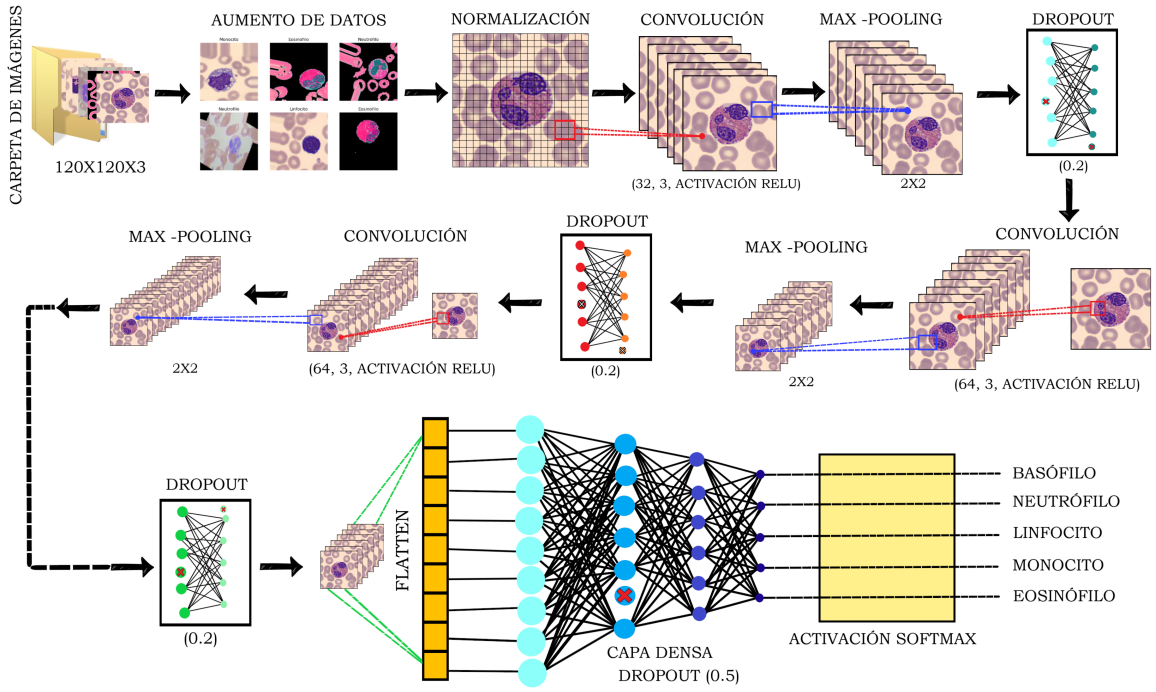

Figura 2.18: Diagrama de bloque de la red neuronal CNN [Fuente: Autor].

Se realiza un aumento de datos en conjuntos de imágenes, lo que implica aplicar transformaciones aleatorias a las imágenes originales para crear variantes aumentadas que enriquece y diversifica la base de datos del entrenamiento (Fig.2.19.

```
data_augmentation = tf.keras.preprocessing.image.ImageDataGenerator(
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest')
```

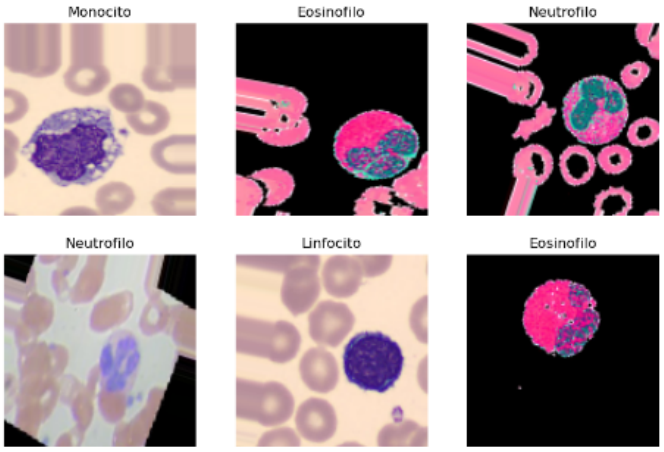

Figura 2.19: Imágenes de entrada a la red neuronal [Fuente: Autor].

Tras finalizar el análisis y la configuración de la base de imágenes de entrada, sigue la configuración de la red. Este diseño consta de un total de 15 capas, lo que representa un incremento comparando con la técnica de referencia que posee solamente 10 capas, la figura 2.20 muestra un resumen del modelo. Este incremento ha sido implementado con el propósito de optimizar y elevar el rendimiento global de la red.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling (Rescaling)       (None, 120, 120, 3)       0

 conv2d (Conv2D)             (None, 118, 118, 32)      896

 max_pooling2d (MaxPooling2D  (None, 59, 59, 32)       0
 )

 dropout (Dropout)           (None, 59, 59, 32)        0

 conv2d_1 (Conv2D)           (None, 57, 57, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 28, 28, 64)       0
 2D)

 dropout_1 (Dropout)         (None, 28, 28, 64)        0

 conv2d_2 (Conv2D)           (None, 26, 26, 64)        36928

 max_pooling2d_2 (MaxPooling  (None, 13, 13, 64)       0
 2D)

 dropout_2 (Dropout)         (None, 13, 13, 64)        0

 flatten (Flatten)           (None, 10816)             0

 dense (Dense)               (None, 128)               1384576

 dropout_3 (Dropout)         (None, 128)               0

 dense_1 (Dense)             (None, 7)                 903

=================================================================
Total params: 1,441,799
Trainable params: 1,441,799
Non-trainable params: 0
_________________________________________________________________
```

Figura 2.20: Estructura de la red neuronal [Fuente: Autor].

Al incorporar capas adicionales a la arquitectura, se busca lograr una mayor capacidad de aprendizaje y adaptabilidad de la red ante patrones complejos y rasgos sutiles presentes en los datos. Esta estrategia de aumentar el número de capas tiene como objetivo fortalecer la precisión y generalización del modelo, permitiéndole abordar de manera más efectiva la variabilidad inherente en los datos de entrada.

Una vez completada la etapa de entrenamiento y obtenidos los valores de precisión y pérdida del modelo, se lleva a cabo la acción de guardar el modelo. Esta acción es esencial para poder emplear el modelo en la realización de predicciones sobre nuevas imágenes de células sanguíneas.

Guardar el modelo implica conservar todos los parámetros, pesos y configuraciones de la red neuronal. Al hacerlo, se crea una representación guardada del modelo que puede ser utilizada en cualquier momento para realizar predicciones precisas y confiables sobre datos nuevos o no vistos previamente.

El modelo se guarda con una extensión.h5 lo que permite conservar el diseño, pesos y parámetros aprendidos durante el entrenamiento, lo que garantiza que el modelo pueda ser recuperado y utilizado posteriormente sin perder la capacidad en realizar predicciones sobre datos nuevos.

Para realizar la predicción de clases en nuevas imágenes de células sanguíneas, se siguen varios pasos esenciales:

1. Cargar el Modelo: En primer lugar, se carga el modelo previamente guardado que ha sido entrenado para realizar clasificaciones precisas en imágenes de células sanguíneas.

```
ruta_del_modelo_guardado = "clf_celulas3.h5"
modelo_cargado = load_model(ruta_del_modelo_guardado)
```

2. Cargar las Imágenes: Se procede a cargar las imágenes que se desean predecir. Estas nuevas imágenes deben ser proporcionadas al modelo para obtener las predicciones de clase correspondientes.

```
carpeta_imagenes = "C:/Users/Andres/Desktop/Prueba"
lista_archivos = os.listdir(carpeta_imagenes)
for archivo in lista_archivos:
    ruta_imagen = os.path.join(carpeta_imagenes, archivo)
```

3. Procesar las Imágenes: Las imágenes cargadas inicialmente podrían tener tamaños diferentes a las imágenes de entrenamiento utilizadas en el modelo. Por lo tanto, es importante procesar estas imágenes para ajustar su tamaño a la dimensión específica con la que el modelo fue entrenado. Esto asegura que las imágenes sean compatibles con el diseño de la red neuronal.

```
imagen = image.load_img(ruta_imagen, target_size=(120, 120))
imagen_array = image.img_to_array(imagen)
imagen_preprocesada = np.expand_dims(imagen_array, axis=0)
```

4. Cargar las Clases: Las clases que corresponden a las diferentes categorías de células sanguíneas deben estar previamente definidas y cargadas. Estas clases se utilizan para asignar etiquetas a las predicciones generadas por el modelo.

```
etiquetas = {'Basofilo': 0, 'Eosinofilo': 1,
'Linfocito': 2, 'Monocito': 3, 'Neutrofilo': 4}
```

5. Realizar Predicciones: Finalmente, con el modelo cargado, las imágenes procesadas y las clases disponibles, se lleva a cabo la fase de predicción. Se aplica el modelo a las imágenes procesadas y se obtienen las predicciones de clase correspondientes para cada una de ellas (Fig. 2.21).

```
predicciones = modelo_cargado.predict(imagen_preprocesada)
index_clase_predicha = np.argmax(predicciones)
clase_predicha = index_to_class[index_clase_predicha]
```

```
1/1 [==============================] - 0s 156ms/step
Imagen: BA_114899.jpg - Clase predicha: Basofilo
1/1 [==============================] - 0s 23ms/step
Imagen: BA_115323.jpg - Clase predicha: Basofilo
1/1 [==============================] - 0s 28ms/step
Imagen: BA_115777.jpg - Clase predicha: Basofilo
1/1 [==============================] - 0s 27ms/step
Imagen: BA_116477.jpg - Clase predicha: Basofilo
1/1 [==============================] - 0s 25ms/step
Imagen: BA_119131.jpg - Clase predicha: Basofilo
1/1 [==============================] - 0s 33ms/step
Imagen: BA_8742.jpg - Clase predicha: Eritroblasto
1/1 [==============================] - 0s 27ms/step
Imagen: EO_866120.jpg - Clase predicha: Eosinofilo
```

Figura 2.21: Clases predichas en un conjunto de imágenes [Fuente: Autor].

2.2.2. Segunda técnica seleccionada: Recuento y detección de células sanguíneas basado en YOLOV5

La elección de la segunda técnica se fundamenta principalmente en una característica adicional que ofrece que es llevar a cabo el recuento de células. Esta funcionalidad adicional representa un factor clave en la selección de esta técnica en particular.

Acerca de recuento de células sanguíneas se revisaron técnicas basadas en YOLOV5, YOLOV8 y ResNet18. Se ha seleccionado un método de clasificación de células sanguíneas basada en YOLOv5.

La primera razón de esta decisión, es que esta ténica ofrece eficiencia computacional. Este modelo es más liviano y optimizado en comparación con YOLOv8 y ResNet-18, lo que significa que requiere menos recursos computacionales para su ejecución.

Además, ofrece una mayor velocidad de detección, lo que es crucial en aplicaciones en tiempo real, ha demostrado ser más rápido que YOLOv8 y ResNet-18 en términos de tiempo de respuesta, lo que permite una detección y clasificación en tiempo real más efectiva.

En cuanto a la precisión, YOLOv5 ha demostrado resultados comparables con YOLOv8 y ResNet-18 en la clasificación de células sanguíneas. Aunque estas técnicas son modelos de última generación con buen rendimiento, YOLOv5 ha logrado una precisión similar al tiempo que ofrece ventajas en eficiencia y velocidad.

Finalmente, la facilidad de uso y la flexibilidad también han influido en la

elección.

La técnica de clasificación escogida se la implementó en Google Colab, ya que proporciona recursos de hardware como GPU y TPU de forma gratuita, que ayuda a acelerar el proceso de entrenamiento, que requiere cálculos intensivos.

Para llevar a cabo efectivamente el proceso con YOLOv5, resulta fundamental tener una base de datos previamente anotados de manera adecuada. Para esto, se realizo una mejora en los datos para el entrenamiento, dado que inicialmente se contaba solamente con 264 imágenes para la fase de entrenamiento y 270 para validación. Esto, sin duda, afectaba negativamente la exactitud en la detección de glóbulos.

Al implementar un nuevo conjunto de datos, se amplió el conjunto a un total de 769 imágenes para entrenar y 96 imágenes para la validación. Este cambio ha tenido un impacto considerable, ya que experimento un aumento significativo en la precisión general de la detección de glóbulos.

Para facilitar este proceso, se almacenan estas imágenes en Google Drive y se ha concedido acceso a estos recursos desde el entorno del notebook.

```
from google.colab import drive
drive.mount('/content/drive')
```

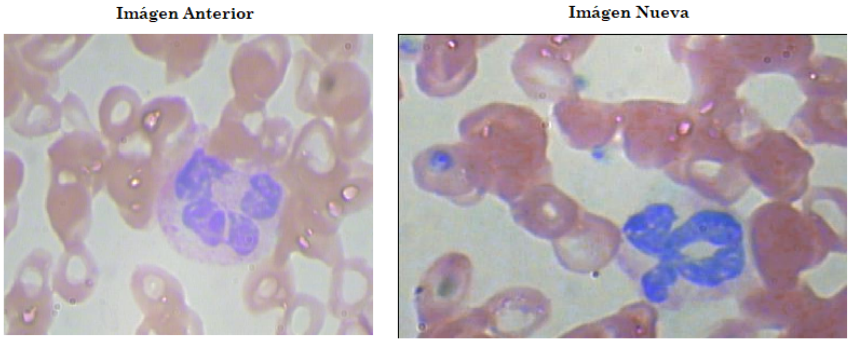

Figura 2.22: Imagen anterior y nueva de célula sanguínea [Fuente: Autor].

Luego de tener las imágenes correctamente cargada se debe descargar o clonar el modelo YoloV5 desde Github con el siguiente código.

```
git clone  'https://github.com/ultralytics/yolov5.git'
```

Una vez que se descargo exitosamente el modelo YOLOv5 en el entorno de ejecución, automáticamente se crea una carpeta que contendrá todos los recursos esenciales requeridos para llevar a cabo tanto el entrenamiento como las etapas subsiguientes de conteo y clasificación de células sanguíneas.

Dentro de esta carpeta, un elemento de particular importancia es el archivo .yaml que contiene la siguiente información.

```
train: ../train/images
val: ../valid/images
nc: 3
names: ['Platelets', 'RBC', 'WBC']
```

Este archivo almacena de manera centralizada la información vital para el proceso, incluyendo las rutas a las imágenes de entrenamiento y validación, así como los nombres y números correspondientes a las etiquetas.

Dentro de esta carpeta también se debe revisar que contenga el archivo train.py el cual se usa para entrenar la red con la base base datos y por último para el entrenamiento es necesario tener una carpeta llamada models donde se encuentran las variantes de YOLOv5 (s, m, l, n, x) representan diferentes configuraciones o tamaños de modelos dentro de la arquitectura YOLOv5. Cada una de estas variantes se ha diseñado para adaptarse a diferentes necesidades y requisitos de aplicaciones específicas en términos de velocidad, precisión y recursos de hardware.

Durante la fase de entrenamiento, se configura las ubicaciones esenciales de los archivos clave para el proceso. Primeramente, se debe especificar la ubicación del archivo train.py, que es el script encargado de llevar a cabo el entrenamiento. Además, definir la ubicación del archivo .yaml que contiene información crucial como las rutas a la base de datos de validación y entrenamiento, así como los nombres y números correspondientes a las etiquetas.

Adicionalmente, se elige la configuración del modelo YOLOv5 que se acople de mejor manera. En este caso, se optó por la variante Yolov5m, la cual ofrece

un equilibrio óptimo entre rendimiento y recursos computacionales. Esta elección permite obtener resultados satisfactorios de detección sin requerir una carga excesiva en términos de capacidad de procesamiento.

También se debe especificar la dimensión de la imagen, en el conjunto original es de 640, un total de 300 épocas y el numero de lotes de 8, tal como se muestra en el siguiente comando.

```
|python /yolov5/train.py --img 640 --batch 8
--epochs 300 --data /yolov5/bcc-kaggle.yaml
--cfg /yolov5/models/yolov5s.yaml --name BCCM
```

Al cambiar el conjunto el tamaño cambio y es de 416, con el cambio de base se realizó un total de 100 épocas y el número de lotes (batch) de 8, mostrado a continuación.

```
python /yolov5/train.py
--img 416 --batch 8 --epochs 100
--data /yolov5/bcc-kaggle.yaml
--cfg /yolov5/models/yolov5s.yaml --name BCCM
```

El resultado que proporciona, luego de 300 épocas, son valores de precisión, tasa de verdaderos positivos y valor del punto de presión de cada clase de célula sanguínea.

Una vez completado el proceso de entrenamiento y obtenidos los resultados, se avanza hacia la etapa de prueba del modelo, que implica realizar predicciones o clasificaciones en nuevas imágenes. Para llevar a cabo esta fase, es esencial asegurarse de tener el archivo "detect.py"disponible en la carpeta que contiene el modelo YOLOv5. Este archivo cumple una función esencial en la implementación de las predicciones.

En esta fase, se sigue un proceso sistemático: se proporciona la ruta de las imágenes que se desean analizar, lo que permite al modelo hacer predicciones precisas. Luego, se carga los pesos de la red previamente guardado al entrenar. Esta

combinación de pasos permite aplicar el modelo entrenado en nuevas situaciones, lo que resulta en la capacidad de identificar y clasificar con precisión los objetos de interés en las imágenes, en este caso, las células sanguíneas.

```
|python /yolov5/detect.py
--source /Imagenes_Celulas_Sanguineas/BloodImage_00002.jpg
--weights /yolov5/runs/train/BCCM15/weights/best.pt
```

En la Figura 2.23, se visualiza la imagen de entrada para la clasificación, así como la imagen resultante que se obtiene después de que se completa la clasificación.

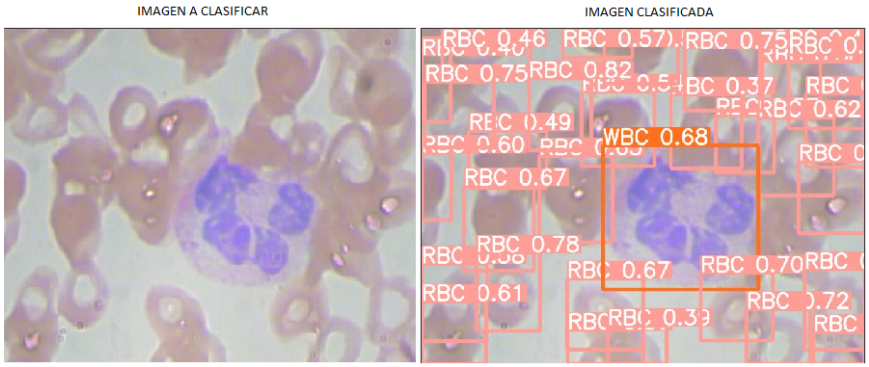

Figura 2.23: Imagen clasificada [Fuente: Autor].

2.2.3. Unión de técnicas para conteo y clasificación automático de células sanguíneas

Para lograr una mejora significativa y crear un clasificador integral se optó para la implementación de clasificación de células sanguíneas combinando de manera sinérgica las dos técnicas previamente explicadas. Estas técnicas se seleccionaron debido a sus notables ventajas en precisión de clasificación y conteo de células sanguíneas.

En primer lugar, se emplea la segunda técnica, explicada en la subsección anterior, para analizar imágenes de células sanguíneas y determinar el recuento de células presentes en la muestra. Con esta información, se segmenta y aísla un glóbulo blanco específico en otra imagen. Luego, utilizando la primera técnica, se identifica con precisión el tipo de glóbulo blanco al que pertenece.

Con el objetivo de integrar las técnicas de manera eficiente, se procedió inicialmente a efectuar modificaciones en el archivo detect.py. Dicho archivo desempeña un papel fundamental al realizar el conteo de células sanguíneas presentes en las imágenes. Para lograr un aislamiento preciso de uno o varios glóbulos blancos presentes en la imagen, se requiere la obtención precisa de las coordenadas de los límites del cuadro que enmarcan dichos glóbulos.

Para esto se implementó un código que va a iterar sobre todos los resultados de la detección de células de la imagen o un conjunto de células sanguíneas, obteniendo para cada detección las coordenadas que definen el rectángulo delimitador y la clase predicha. Si la clase predicha es 'WBC', significa que detectó un glóbulo blanco en la imagen. De esta manera, se guarda en el archivo las coordenadas de todos los glóbulos blancos detectados en cada imagen, junto con el nombre de la imagen correspondiente. Cuando no se encuentre glóbulos blancos en la imagen no guardará ningún archivo .txt en la carpeta. El siguiente es el código empleado en el archivo detect.py para obtener las coordenadas de las cajas delimitadoras de los glóbulos blancos presentes en cada imagen.

```
for *xyxy, conf, cls in reversed(det):
     x_min, y_min, x_max, y_max = map(int, xyxy)
     class_name = names[int(cls)]
     if class_name == 'WBC':
        coords_file.write(f"{current_image_name},{x_min},{y_min},
        {x_max},{y_max},{class_name}\n")
```

Una vez cambiado el detect.py lo que se hace es cargar el nuevo archivo ya modificado para que guarde las coordenadas con los límites del cuadro de los glóbulos blancos, en este caso se colocó con el nombre de detect1.py mostrado a continuación.

```
python C:\\Users\\Andres\\Desktop\\yolov5_conteo\\detect1.py ...
```

Tras cargar el archivo modificado, se obtiene como resultado el recuento de las células sanguíneas en conjunto con las imágenes que presentan los cuadros

delimitadores pertinentes, en la figura 2.24 se muestra un ejemplo de como se agregan las cajas delimitadoras a la imagen.

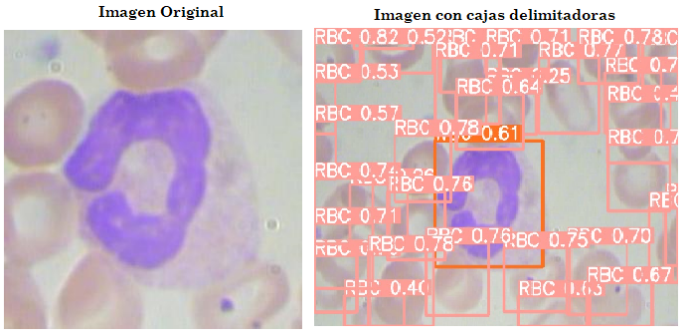

Figura 2.24: Imagen original y con cajas delimitadoras [Fuente: Autor].

Además de esta información, se genera un archivo en formato .txt donde se almacenan las coordenadas de las cajas delimitadoras asociadas a los glóbulos blancos presentes en cada imagen. En el siguiente cuadro se muestra un ejemplo de como se guardan las coordenadas.

```
BASOFILO (1),98,131,246,265,WBC
EOSINOFILO (1),91,98,269,253,WBC
EOSINOFILO (2),90,91,258,258,WBC
IMAGEN (1),370,1,416,59,WBC
IMAGEN (10),425,303,539,420,WBC
```

Después de adquirir las coordenadas precisas de las cajas delimitadoras que rodean a cada glóbulo blanco en las imágenes contadas mediante la técnica YOLOv5, se inicia el proceso de recorte de las imágenes. El objetivo es aislar exclusivamente los glóbulos blancos. Este proceso sigue varios de pasos:

- Paso 1: Se obtiene la ruta donde se encuentran las carpetas que contienen las coordenadas de las regiones de interés previamente detectadas en las imágenes.
- Paso 2: Teniendo las carpetas con las coordenadas, se obtiene la ruta de la carpeta más reciente, basándose en la fecha de creación. Esto se hace para que cuando se realice la clasificación, tome la carpeta indicada correspondiente a la última versión o iteración de las detecciones.

- Paso 3: Dentro de esta carpeta más reciente con la última interacción, se lee el archivo .txt que contiene las coordenadas de las regiones de interés por cada imagen.
- Paso 4: Se crea un diccionario para almacenar las coordenadas. El diccionario funciona como una tabla de búsqueda que toma las coordenadas por nombre de imagen. Esto facilita luego iterar sobre las imágenes y sus coordenadas para recortar las regiones deseadas de cada una.
- Paso 5: Se define el tamaño deseado para las imágenes recortadas, en este caso 300x300 píxeles. Se coloca estas dimensiones debido a que las imágenes recortadas con una menor dimensión no se lograba apreciar completamente. En este caso lo que se hizo es agrandar el cuadro de recorte para poder tener una imagen más amplia y que se pueda visualizar de mejor manera el glóbulo blanco.
- Paso 6: En este paso se toma la fecha y hora actual para generar una carpeta única para guardar los recortes.
- Paso 7: Se crea la ruta completa para esta nueva carpeta que almacenará los recortes.
- Paso 8: Se itera sobre las imágenes y sus correspondientes coordenadas leídas previamente.
- Paso 9: Para cada imagen, se calcula el centro de la región de interés original según sus coordenadas.
- Paso 10: Se calculan nuevas coordenadas alrededor de este centro para obtener una región de 300x300 píxeles.
- Paso 11: Se recorta esa región de 300x300 de la imagen original.
- Paso 12: Dentro de la carpeta única creada, se crea una subcarpeta para guardar las imágenes recortadas.
- Paso 13: Finalmente, se guardan las imágenes recortadas en esta subcarpeta para su posterior uso en la clasificación.

La figura 2.25 es un ejemplo de una imagen recortada al rededor del glóbulo blanco.

Figura 2.25: Imagen Recortada [Fuente: Autor].

Una vez completado el recorte de las imágenes y obtenidas las representaciones exclusivas de los glóbulos blancos, se avanza a la siguiente etapa que es la implementación de la técnica de clasificación de células sanguíneas mediante una red neuronal convolucional (CNN).

Esta fase tiene como objetivo determinar la categoría específica de glóbulo blanco a la que pertenece cada imagen. En este proceso, las imágenes recortadas anteriormente se emplean como entrada para la red neuronal.

La figura 2.26 es un representación gráfica del proceso que se lleva a cabo para el conteo y clasificación de células sanguíneas haciendo uso de las dos técnicas antes expuestas.

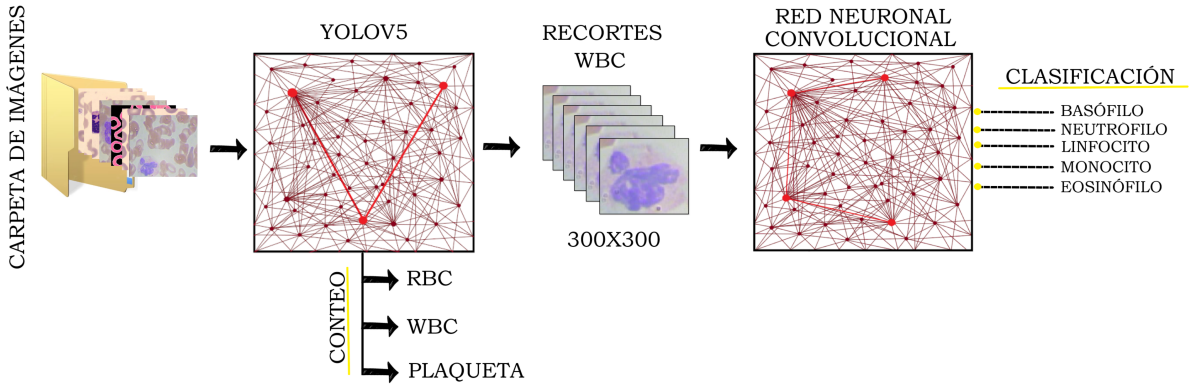

Figura 2.26: Diagrama de bloque de la red completa [Fuente: Autor].

Capítulo 3

Resultados Obtenidos y Diseño de Interfaz Gráfica para Pruebas de Funcionamiento

Este capítulo, presenta los resultados alcanzados en los diferentes procesos de entrenamiento en cada una de las técnicas seleccionadas y expuestas en el capitulo 2.

También, se detallará el diseño de una Interfaz Gráfica de Usuario (*del inglés Graphical User Interface, GUI)*) especialmente creada para las pruebas de operación de las técnicas de clasificación de células sanguíneas. A lo largo de este capítulo, se expone el diseño de la interfaz, las consideraciones ergonómicas, las funciones clave que ofrece y su contribución a la mejora del proceso de clasificación celular.

También, se explica cómo la interfaz gráfica actúa como un puente entre las complejas técnicas de clasificación de células sanguíneas y los usuarios finales, que pueden ser médicos, investigadores o profesionales de laboratorio. Se destacará cómo las herramientas de interacción y visualización brindan a los usuarios un control más efectivo sobre los parámetros de operación y les permiten analizar los resultados de manera más precisa.

3.1. Resultados Obtenidos

3.1.1. Resultados Obtenidos en Entrenamiento en la Técnica de Clasificación

En el capítulo 2 se mostró el proceso de entrenamiento de la red configurada en este clasificador, a continuación se muestran los resultados obtenidos de este proceso. Así, para el entrenamiento, se llevaron a cabo un total de 20 épocas de entrenamiento. Cada una de estas épocas se completó en aproximadamente 400 segundos, lo que se traduce en una duración aproximada de 6 minutos. En conjunto, este proceso de entrenamiento consumió dos horas en total lo que representa una mejora considerable en comparación con la técnica de referencia seleccionada. En dicha técnica, el entrenamiento de la red requería alrededor de 6 horas. La reducción en el tiempo de entrenamiento, de seis horas a dos horas, demuestra una mayor eficiencia en el proceso de optimización de la red.

Al concluir el entrenamiento, se calculan los valores de precisión y pérdida. Además, se generan gráficas comparativas que representan la evolución de estos valores a lo largo de cada una de las épocas realizadas.

Estos resultados brindan una visión integral del rendimiento del modelo a medida que se ajusta durante el entrenamiento.

En la tabla 3.1 se muestran los resultados obtenidos al final del entrenamiento y en la figura 3.1 se muestran las gráficas de perdida y precisión obtenidos en cada época.

Tabla 3.1: Valores de precisión y perdida obtenidos en el entrenamiento.

Valor mínimo de pérdida del conjunto de datos de entrenamiento:	0.2564
Valor mínimo de pérdida del conjunto de datos de validación:	0.1056
Precisión máxima del conjunto de datos de entrenamiento:	91.44 %
Precisión máxima del conjunto de datos de validación:	96.64 %

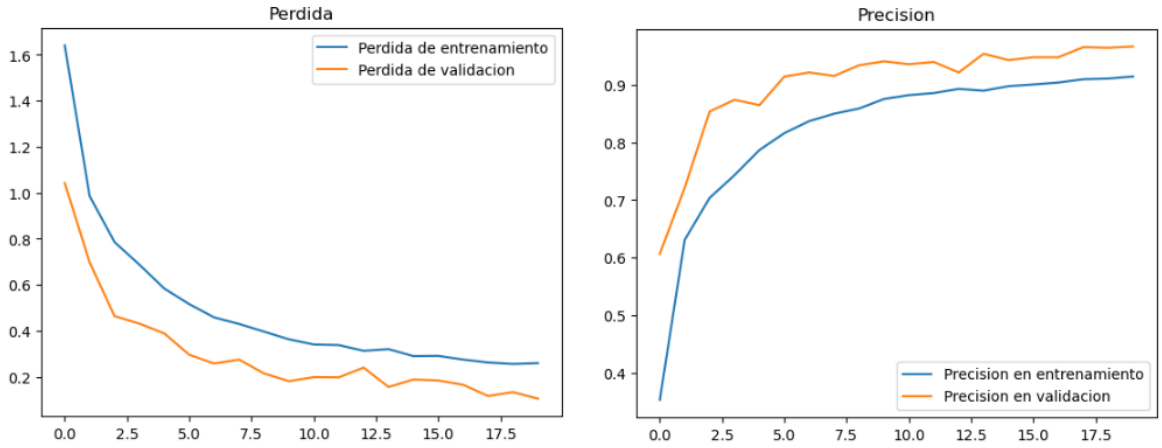

Figura 3.1: Gráfica de precisión y perdida.

La precisión lograda en las imágenes de validación experimentó una mejora notable en contraste con el modelo de referencia. En este modelo se obtuvo un porcentaje de precisión de 96,64 %, mientras el modelo anterior obtuvo un 87,5 % de precisión en la clasificación. Este avance se atribuye tanto al incremento de capas en la red y a la disminución en las dimensiones de las imágenes.

Esta ampliación contribuyó a su habilidad para detectar patrones y aspectos más sutiles presentes en los datos. Esta expansión permitió que el modelo aprendiera representaciones más complejas y refinadas, lo que se tradujo en un aumento en la precisión de las predicciones en el conjunto de validación.

Además, la reducción del tamaño de las imágenes a 120x120 píxeles en comparación con las imágenes de referencia de 300x300 píxeles también desempeñó un papel fundamental. La disminución del tamaño ayudó a concentrar la atención de la red en detalles más relevantes y significativos de las imágenes, lo que mejoró su capacidad para identificar y clasificar con precisión.

3.1.2. Resultados Obtenidos en Entrenamiento en la técnica de conteo

En el capítulo 2 se muestra el entrenamiento del modelo YOLO V5 utilizada para el contador, aquí se muestran los resultados alcanzados.

En términos de precisión, se observa que los valores de clasificación total

muestran un rendimiento de 0.126 (12.6 %), mientras que para las plaquetas se alcanza una precisión de 0.147 (14.7 %), en el caso de los glóbulos rojos se registra un valor de 0.106 (10.6 %), y finalmente, en la clasificación de glóbulos blancos se observa una precisión de 0.125 (12.5 %). Estos resultados revelan un nivel de precisión notablemente bajo, lo que motivó la decisión de realizar una modificación en el conjunto de imágenes que se usa para entrenar la red neuronal.

Tabla 3.2: Valores de métricas obtenidas en el entrenamiento con la base de datos original.

Todas las clases:	Precisión:	0.126	Recall:	0.784	mAP:	0.155
Plaquetas:	Precisión:	0.147	Recall:	0.511	mAP:	0.156
RBC:	Precisión:	0.106	Recall:	0.627	mAP:	0.113
WBC:	Precisión:	0.125	Recall:	0.973	mAP:	0.197

Tras implementar la modificación en el conjunto de datos, los resultados han experimentado una marcada mejoría en la precisión. Comparando con los valores previos, la precisión en la clasificación total ha incrementado significativamente hasta alcanzar un notable 0.85 (85 %). Del mismo modo, los resultados específicos también revelan mejoras sustanciales: en el caso de las plaquetas, se ha logrado una precisión del 0.8 (80 %), en los glóbulos rojos se registra un impresionante 0.783 (78.3 %), y, finalmente, con un incremento notable, los glóbulos blancos presentan ahora una precisión sobresaliente de 0.968 (96.8 %). Estos resultados demuestran un cambio sustancial en la precisión de las clasificaciones, subrayando la efectividad de la modificación realizada en la base de datos de entrenamiento.

Tabla 3.3: Valores de métricas obtenidas en el entrenamiento con la base de datos nueva.

Todas las clases:	Precisión:	0.85	Recall:	0.905	mAP:	0.923
Plaquetas:	Precisión:	0.8	Recall:	0.892	mAP:	0.981
RBC:	Precisión:	0.783	Recall:	0.822	mAP:	0.887
WBC:	Precisión:	0.968	Recall:	1	mAP:	0.981

La figura 3.2, representa la precisión en función de la confianza para la clasificación de células sanguíneas. Esto permite evaluar directamente la calidad del modelo de detección en cuanto a su precisión a diferentes niveles de confianza. La confianza se relaciona con el nivel de seguridad que tiene el modelo en relación con sus predicciones. Un nivel de confianza de 0 el modelo carece de certeza, un nivel

de confianza de 1 lleva a una certeza máxima. Mientras que, la precisión mide la capacidad del modelo de hacer predicciones correctas.

La curva de plaquetas muestra una lenta escalada de precisión con la confianza, destacando la dificultad del modelo para categorizar con certeza plaquetas en niveles de confianza bajos. La curva de glóbulos rojos presenta un crecimiento constante pero con algunas variaciones en el rango de confianza entre 0.8 y 0.85. Aun así, la precisión se eleva rápidamente a 1.0 con una confianza de 0.89, evidenciando su capacidad de lograr clasificaciones precisas en niveles más altos de confianza.

Por otro lado, la curva de precisión de los glóbulos blancos muestra mejoras graduales y consistentes a medida que la confianza aumenta. Comienza con un modesto 0.1 a una confianza de 0.2 y aumenta a 0.5 en 0.79 de confianza, alcanzando rápidamente una precisión perfecta de 1.0.

Esta revelación enfatiza la necesidad de altos niveles de confianza para obtener una clasificación precisa y confiable de todas las células sanguíneas en su conjunto, aunque vale mencionar que este umbral puede variar entre las diferentes categorías.

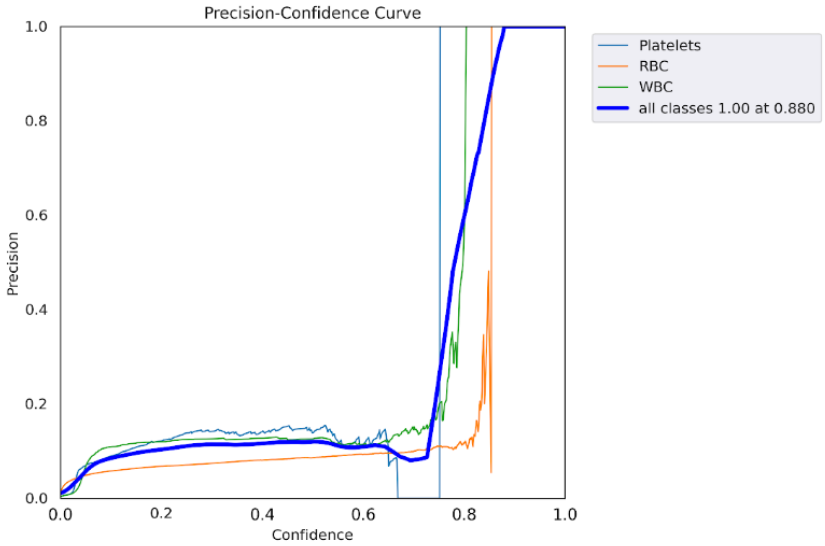

Figura 3.2: Proceso de entrenamiento con YoloV5 con base de datos original [Fuente: Autor].

La figura 3.3 presenta la relación entre la confianza en la clasificación de diferentes tipos de células sanguíneas y la precisión de dichas clasificaciones.

Comenzando con la curva de plaquetas, se observa que la precisión incrementa gradualmente mientras aumenta la confianza, pero con pequeñas fluctuaciones en

los primeros puntos. A medida que la confianza alcanza valores más altos, las fluctuaciones en la precisión también son mayores, indicando cierta variabilidad en la clasificación. La curva de glóbulos rojos (RBC) muestra una relación más estable entre la confianza y la precisión. La precisión aumenta de manera constante a medida que aumenta la confianza, con fluctuaciones apenas perceptibles.

La curva de glóbulos blancos (WBC) presenta una fase inicial de rápida mejora en la precisión con respecto a la confianza. Sin embargo, a partir de un punto cercano a (0.1, 0.95), la precisión se mantiene constante independientemente de la confianza. Esto podría indicar que una vez que la confianza alcanza cierto nivel, la precisión de la clasificación de glóbulos blancos es alta.

La última curva, que representa todas las clases combinadas, la precisión aumenta de manera significativa a medida que la confianza aumenta hasta aproximadamente (0.4, 0.8), donde alcanza un nivel alto pero no máximo. A partir de este punto, la precisión sigue aumentando, aunque de manera más gradual, hasta alcanzar una precisión perfecta (1.0) cuando la confianza llega a 1.0. Esto sugiere que a medida que la confianza en la clasificación aumenta, la precisión también aumenta de manera constante.

Al comparar con el gráfico (3.2) que mostraba precisiones altas en niveles altos de confianza. Este gráfico muestra una mayor precisión en niveles de confianza más bajos, indicando que el sistema es mas cuidadoso al asignar etiquetas y solo lo hace cuando está seguro de que la clasificación es correcta.

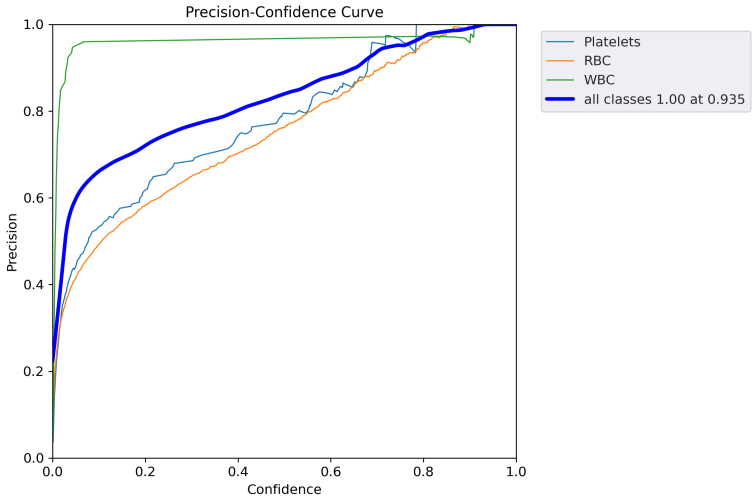

Figura 3.3: Gráfica de precisión con base de datos actual.

3.2. Desarrollo de Interfaz Gráfica

El proceso de desarrollo de la interfaz gráfica implica la creación de cuatro ventanas distintas: una ventana principal y cuatro adicionales, cada una correspondiente a una técnica específica y a la combinación sinérgica de ambas, y la última se usa para realizar un nuevo entrenamiento de la red.

3.2.1. Ventana Para Técnica de Clasificación

El objetivo de esta ventana es tener una aplicación de clasificación de células sanguíneas con una interfaz gráfica, en la cual los usuarios pueden cargar imágenes individuales o carpetas de imágenes para predecir y mostrar los tipos de células sanguíneas correspondientes.

Para esto se implementa una interfaz gráfica de usuario (GUI)) utilizando la biblioteca Tkinter en Python. La cual se muestra en la figura 3.4. Tiene varias funciones:

En primer lugar, brinda al usuario la capacidad de elegir un modelo de predicción desde un menú desplegable.

Además, permite al usuario cargar una imagen individual para predecir el tipo

de célula sanguínea presente en ella. Una vez realizada la predicción utilizando el modelo escogido del menú desplegable, muestra la imagen cargada junto con la clase de célula sanguínea que se ha predicho.

De igual manera, ofrece la opción de cargar una carpeta que contiene múltiples imágenes. Se procesa estas imágenes utilizando el mismo modelo de predicción y muestra los resultados de las predicciones en forma de una lista, detallando los tipos de células sanguíneas correspondientes a cada imagen.

La GUI también incorpora un botón diseñado para borrar los resultados anteriores, lo que contribuye a mantener el espacio de trabajo organizado y sin información redundante.

Figura 3.4: Interfaz Gráfica Completa.

3.2.2. Ventana Para Técnica de Conteo

En esta ventana el objetivo es tener una interfaz gráfica, en la cual los usuarios pueden seleccionar una carpeta de imágenes y ejecutar el proceso YOLOv5 para la detección de células, mostrando la salida en una ventana de (GUI) interactiva.

Para cumplir con este objetivo se implementa una interfaz gráfica en Python utilizando la biblioteca Tkinter mostrada en la figura 3.5, para ejecutar la detección de

células utilizando YOLOv5.

La (GUI) permite a los usuarios seleccionar una carpeta de imágenes y ejecutar la detección y conteo. A continuación, se detalla su funcionamiento:

Muestra una etiqueta (“Carpeta de imágenes:”) y un campo de etiqueta vacío que se llenará con la ruta de la carpeta seleccionada por el usuario.

El botón Seleccionar Carpeta permite al usuario buscar y seleccionar una carpeta de imágenes.

El botón Ejecutar YOLOv5 que inicia el proceso de detección en las imágenes de la carpeta seleccionada.

Muestra un área de texto desplazable donde se imprime la salida de la ejecución del proceso.

También hay un botón “Limpiar Resultados” para borrar el contenido del área de texto.

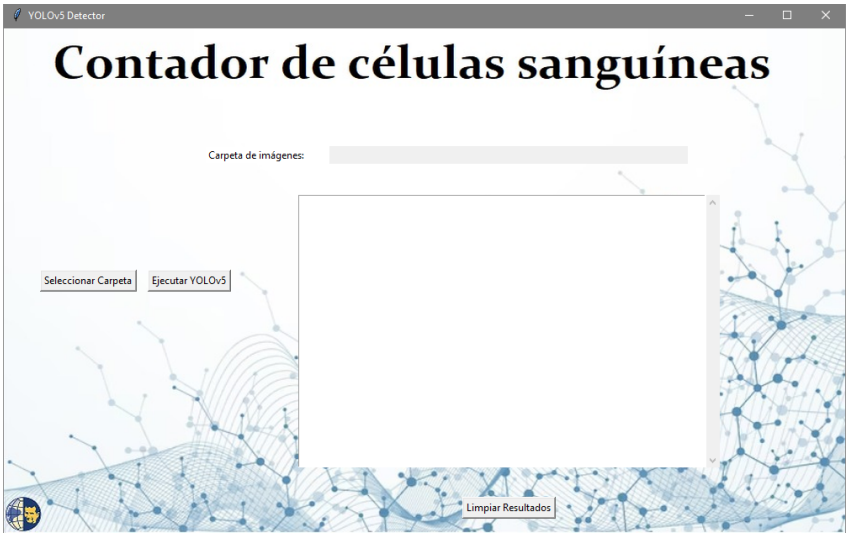

Figura 3.5: Interfaz Gráfica Completa.

3.2.3. Ventana Para la Unión de Técnicas

En esta ventana el objetivo es tener una interfaz, en la cual los usuarios pueden cargar una carpeta de imágenes, ejecutar un proceso de detección y análisis utilizando YOLOv5 y un modelo de clasificación de células sanguíneas, y muestra los resultados en una ventana de GUI interactiva.

Para lograr esto se implementa una interfaz gráfica en Python usando la

biblioteca Tkinter para realizar análisis de imágenes 3.6. La GUI permite a los usuarios cargar una carpeta de imágenes, ejecutar un proceso de detección utilizando YOLOv5, recortar regiones de interés de las imágenes detectadas y predecir las clases de las regiones recortadas usando un modelo de clasificación de células sanguíneas. Se detalla su funcionamiento, a continuación:

Brinda al usuario la capacidad de elegir un modelo de predicción desde un menú desplegable.

El botón "Cargar Carpeta" permite al usuario buscar y seleccionar una carpeta que contiene imágenes para el análisis.

El botón "Ejecutar Proceso" inicia el proceso de detección de células en las imágenes de la carpeta seleccionada. Se recortan las regiones de interés y se redimensionan al tamaño deseado. Luego, se guardan las imágenes recortadas y se realizan predicciones para cada imagen recortada. Las predicciones se muestran en el área de texto y en la otra se muestra los resultados de detección o conteo.

El botón "Limpiar resultados" borra el contenido de las áreas de texto.

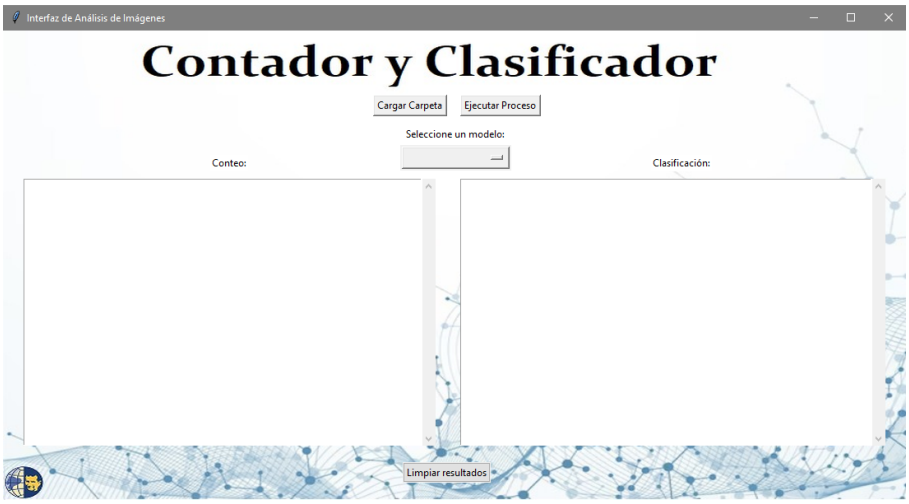

Figura 3.6: Interfaz Gráfica Completa.

3.2.4. Ventana de Entrenamiento

En esta ventana el objetivo es tener una interfaz gráfica mostrada en la figura 3.7 que permite cargar una carpeta de imágenes, entrenar un modelo de red neuronal en base a esas imágenes y luego guardar el modelo entrenado.

La interfaz tiene lo siguientes elementos:

Botón ”Seleccionar Base de Entrenamiento”: Al pulsar en este botón, se abre un cuadro de diálogo para seleccionar una carpeta que contiene imágenes de entrenamiento y validación.

Botón ”Entrenar”: Al pulsar en este botón, se inicia el proceso de entrenamiento del modelo (CNN) con las imágenes de las carpetas cargadas.

Cuadro de texto: aquí se muestra información y resultados sobre el proceso de entrenamiento y los pasos realizados. Puede mostrar mensajes como "Proceso en ejecución...", "Modelo guardado como:", y detalles sobre la precisión de entrenamiento y validación.

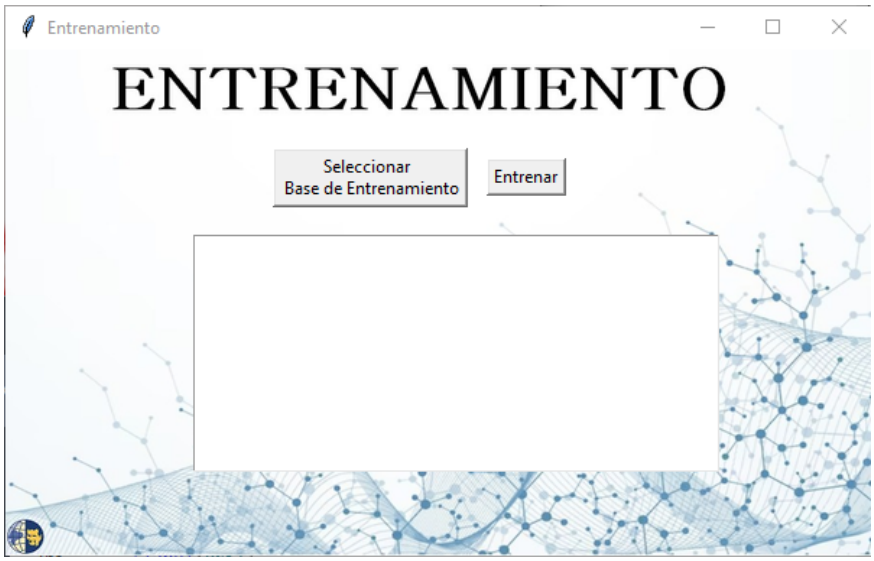

Figura 3.7: Interfaz Gráfica Completa.

3.2.5. Ventana Principal

En esta ventana el objetivo es crear una interfaz principal que permite al usuario abrir tres tipos de interfaces secundarias: Técnica de Conteo, Técnica de Clasificación y de Unión de Técnicas.

Esta interfaz, como se muestra en la figura 3.8, cuenta con tres botones en la ventana principal: abrir contador, abrir clasificador y abrir contador/clasificador. Cuando se hace clic en el botón “Ejecutar Contador”, se abre una ventana secundaria que contiene la interfaz de la Técnica de Conteo. El botón “Ejecutar Clasificador”, abre una ventana secundaria que contiene la interfaz para realizar la clasificación de las células que se encuentren en la imagen de un frotis sanguíneo. Para la ejecución

de conteo y clasificación celular de un banco de imágenes de frotis sanguíneos, se usa el botón "Ejecutar Contador/Clasificador". La ventana secundaria que se abre permite seleccionar el modelo entrenado de CNN para el análisis correspondiente y la operación del sistema corresponde a la unión de las dos acciones anteriores. Cuando se hace clic en el botón "Entrenar Red Neuronal ", se abre una ventana secundaria que contiene la interfaz de Entrenamiento. Se permite guardar el modelo entrenado para trabajar en análisis cuando se lo considere necesario.

Figura 3.8: Interfaz Gráfica Completa.

3.3. Pruebas de Funcionamiento en Interfaz Gráfica

En esta sección, se exponen pruebas de rendimiento para evaluar el desempeño de las dos técnicas seleccionadas, la red neuronal convolucional para la clasificación, y YOLOv5, empleado para el conteo de células sanguíneas.

Se evalúa la precisión, la velocidad o tiempo, la estabilidad, escalabilidad de las dos redes seleccionadas. También se evalúa las dos técnicas en conjunto, aprovechando las fortalezas de cada uno. Estas pruebas permitirán determinar cuál técnica o combinación de técnicas es más adecuada para desarrollar un sistema automático confiable para el recuento de células sanguíneas. Los resultados proporcionarán información valiosa para guiar el desarrollo de futuros sistemas de diagnóstico asistidos por IA en el campo de la hematología.

3.3.1. Pruebas de Funcionamiento para la técnica de clasificación

Matriz de confusión

Esta matriz es útil para analizar el desempeño y la precisión del clasificador basado en redes neuronales convolucionales. La matriz muestra la cantidad de predicciones correctas e incorrectas para cada clase. Esto permite rápidamente identificar errores sistemáticos, como una clase que es consistentemente confundida con otra.

Para el análisis de la figura 3.9, se debe tomar en cuenta que se realizó una prueba de predicción con una base de datos pública de 10298 imágenes etiquetadas con los diferentes tipos de células sanguíneas. Es decir, se dispone de la clase real de cada imagen ingresada y se obtendrá una predicción para cada una de las imágenes, la matriz de confusión se realiza con base en la comparación de la clase real con la clase predicha de cada imagen.

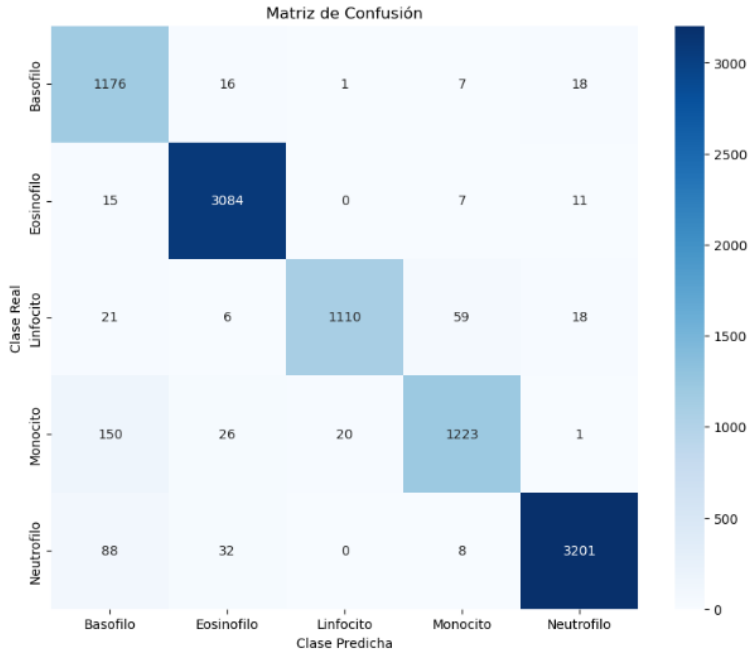

Figura 3.9: Matriz de confusión para el clasificador de células sanguíneas.

Entonces, de acuerdo con los datos de la diagonal principal de la matriz de confusión antes expuesta, de las 1360 predicciones como Basófilo, 1176 fueron correctas, representando un 86 % de precisión. Del 14 % incorrecto, 150 fueron confundidas como Monocitos (11 % del total) y 88 como Neutrófilos (6 % del total). Las

confusiones restantes fueron 21 como Linfocitos (2 % del total) y 15 como Eosinófilos (1 % del total). La baja precisión de esta clase puede estar afectada debido a que algunas imágenes dentro de esta carpeta tienen una baja resolución y no se puede apreciar bien la clase a la que pertenece.

De las 3164 predicciones como Eosinófilo, 3084 fueron correctas, representando un 97 % de precisión. Del 3 % incorrecto, 32 fueron confundidas como Neutrófilos (1 % del total), 26 como Monocitos (1 % del total), 16 como Basófilos (0.5 % del total), 6 como Linfocitos (0.2 % del total). La precisión de Eosinófilo fue muy alta, de un 97 %, con muy pocas confusiones con otras clases. El mayor porcentaje de error se dio con los Neutrófilos, representando el 1 % del total de predicciones. Esto indica que la red categoriza bien los Eosinófilos, distinguiéndolos correctamente en la gran mayoría de los casos. Con apenas un 3 % de error, no se requerirían mayores ajustes para esta clase.

De las 1131 predicciones como Linfocito, 1110 fueron correctas, representando un 98 % de precisión. Del 2 % incorrecto, 20 fueron confundidas como Monocitos (2 % del total), 1 como Basófilo (0.1 % del total). No hubo confusiones con Neutrófilos ni Eosinófilos.

De las 1317 predicciones como Monocito, 1223 fueron correctas, representando un 93 % de precisión. Del 7 % incorrecto, 59 fueron confundidas como Linfocitos (4 % del total), 8 como Neutrófilos (0.6 % del total), 7 tanto como Eosinófilos como Basófilos (0.5 % en cada caso).

De las 3248 predicciones como Neutrófilo, 3201 fueron correctas, representando un 99 % de precisión. Del 1 % incorrecto, 18 fueron confundidas tanto con Linfocitos como con Basófilos (0.5 % en cada caso), 11 con Eosinófilos (0.3 %) y 1 con Monocitos (0.03 %).

En resumen, la precisión de Neutrófilo fue del 99 %, con solo un 1 % de error distribuido entre confusiones con otras clases. La precisión de Eosinófilo también fue muy alta, de 97 %, con un 3 % de error principalmente con Neutrófilos (1 %). La clase Linfocito tuvo un 98 % de precisión y un 2 % de error distribuido entre Monocitos (2 %) y Basófilos (0.1 %). La precisión de Monocito fue de 93 %, con un 7 % de error, mayormente confusiones con Linfocitos (4 %). Finalmente, Basófilo tuvo la

precisión más baja, de 86 %, con un 14 % de error distribuido principalmente entre Monocitos (11 %) y Neutrófilos (6 %). Por lo tanto, las clases con mayor precisión fueron Neutrófilo y Eosinófilo. Linfocito y Monocito tuvieron buena precisión con algo de confusión entre ellos. Basófilo mostró el rendimiento más bajo, con dificultades para distinguirlo de Monocitos y Neutrófilos.

Estabilidad

La estabilidad es una métrica que debe evaluarse al desarrollar un clasificador automático de células sanguíneas. Para comprobar la estabilidad se analizará un mismo conjunto de imágenes, la cual será subida 20 veces. Esto permitirá evaluar la estabilidad del modelo al realizar pruebas repetidamente con los mismos datos. Al hacer esto se verificará si el modelo muestra resultados similares o si existe una alta variabilidad en las métricas de rendimiento entre cada prueba.

En caso de ser un clasificador inestable, puede producir resultados muy diferentes en distintos conjuntos de datos, incluso si provienen del mismo conjunto de imágenes, esto resultaría en un modelo poco confiable en entornos clínicos reales.

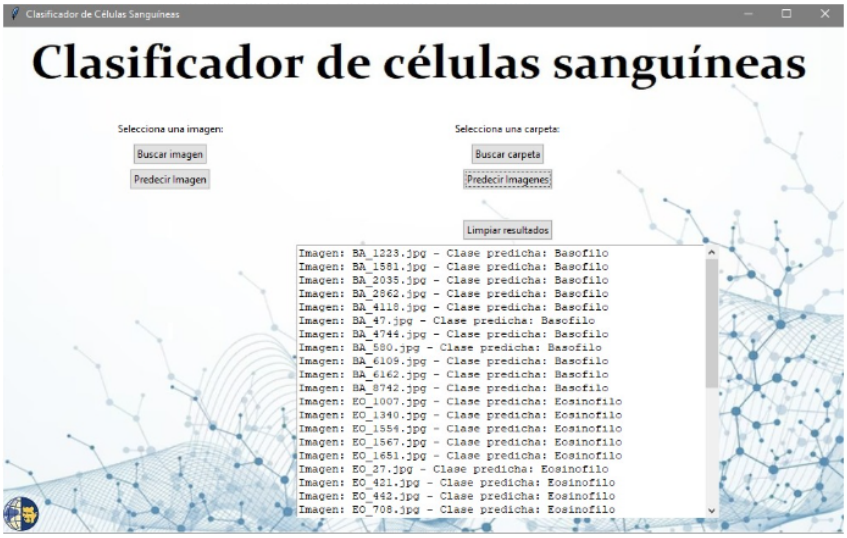

Figura 3.10: Primera prueba (Carpeta 1).

Figura 3.11: Vigésima prueba (Carpeta 1).

El análisis de estabilidad revela un patrón constante en las predicciones realizadas por el sistema. En ambas ocasiones, se observa que las imágenes son clasificadas con una precisión del 100 %, lo que sugiere una consistencia muy alta en la predicción de las clases correspondientes. Este nivel de precisión constante podría indicar que la red obtuvo un aprendizaje efectivo en cuanto a las características distintivas de las diferentes clases presentes en las imágenes, permitiéndole realizar predicciones coherentes y correctas en cada iteración.

Al comparar los resultados entre la primera (Fig.3.10) y la vigésima vez (Fig.3.11) que se subió la carpeta de datos, se nota que no existe variación en las clases predichas para ninguna de las imágenes. Esto subraya aún más la estabilidad del modelo en su capacidad para realizar predicciones coherentes y precisas. La ausencia de cambios en las predicciones podría indicar que las imágenes y las características que la red utiliza para realizar las clasificaciones son altamente consistentes y bien definidas. Esto también sugiere que el modelo no está siendo afectado por factores externos o aleatorios que podrían influir en sus predicciones.

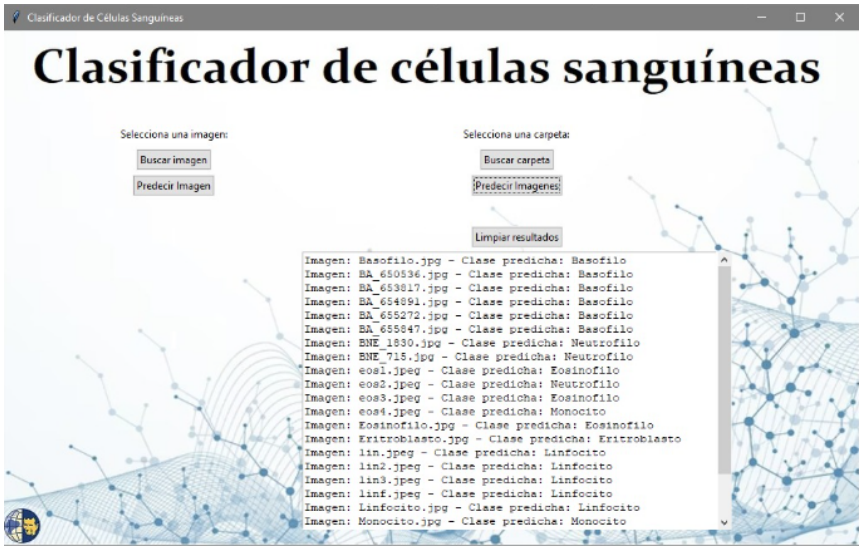

Figura 3.12: Primera prueba (Carpeta 2).

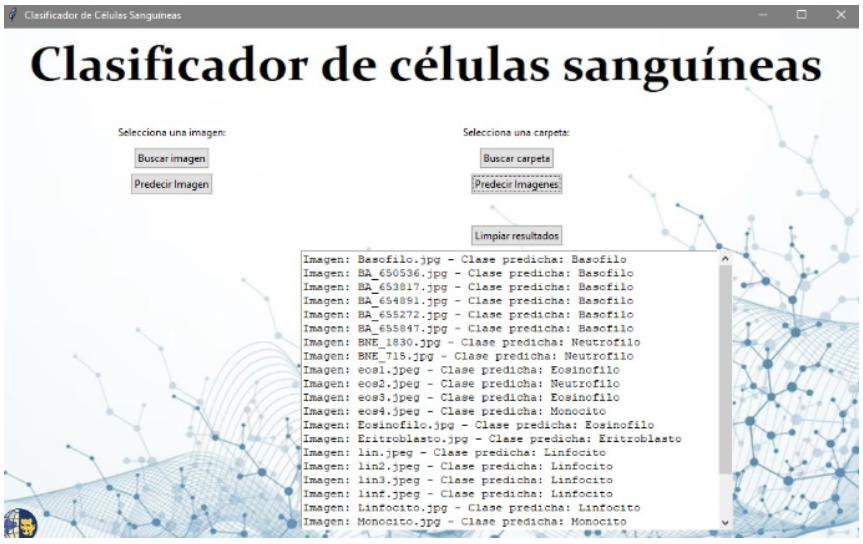

Figura 3.13: Vigésima prueba (Carpeta 2).

Se realizó el mismo procedimiento para un conjunto de imágenes diferentes (3.12),(3.13). Mostrando consistencia en las predicciones a lo largo de las 20 iteraciones, indicando que el modelo no solo ha aprendido las características específicas de la primera carpeta de datos, sino que también ha capturado patrones relevantes en la segunda carpeta. La falta de variabilidad en las predicciones y la precisión constante sugieren que el modelo no está cayendo en trampas de ajuste excesivo.

Escalabilidad

Este tipo de prueba sirve para asegurarse de que el clasificador pueda manejar grandes volúmenes de datos sin degradar su rendimiento o volverse significativamente más lento. En el ámbito médico, donde se utilizan clasificadores de células sanguíneas para diagnosticar enfermedades, es esencial que el sistema sea capaz de manejar eficazmente una amplia gama de muestras, ya que la cantidad de datos puede ser bastante alta.

Realizar una prueba de escalabilidad implica alimentar gradualmente más datos al clasificador y observar cómo responde. Si el clasificador sigue manteniendo un rendimiento aceptable y el tiempo de procesamiento se mantiene dentro de límites aceptables a medida que los datos aumentan, se puede decir que el clasificador es escalable.

En la figura (3.14) se observa, como clasifica perfectamente las 7 imágenes de las diferentes clases de células sanguíneas en la primera prueba, es un indicio positivo de su capacidad inicial para reconocer y distinguir entre estas clases. Sin embargo, es importante tener en cuenta que esta prueba inicial se realizó en una base de datos relativamente pequeño.

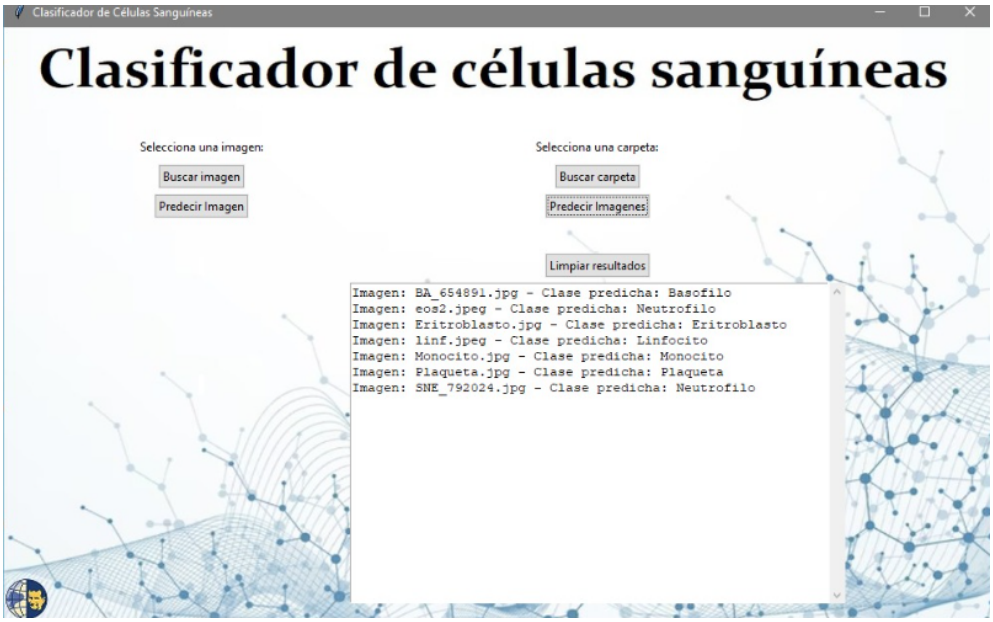

Figura 3.14: Clasificación de 7 imágenes.

Un error en la clasificación durante la prueba con 10 imágenes de diversas

clases (3.15), donde un eosinófilo fue clasificado incorrectamente como un neutrófilo, podría tener implicaciones interesantes para la escalabilidad del clasificador. Este error podría indicar que el clasificador está enfrentando cierta dificultad cuando se le presentan datos más variados. Considerando que la imagen sacada de una base de datos en cuestión tiene muy poca resolución.

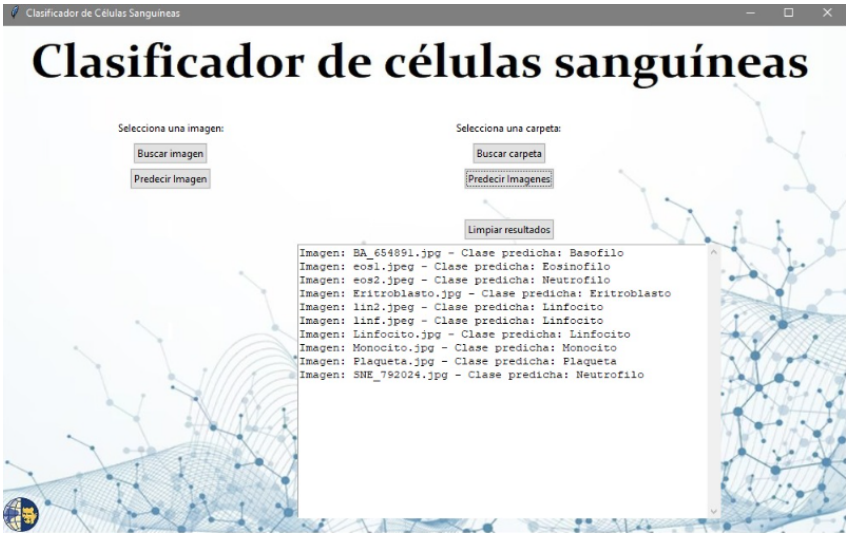

Figura 3.15: Clasificación de 10 imágenes.

En esta prueba el clasificador cometió 4 errores al clasificar un conjunto de 30 imágenes de diversas clases (Fig.3.16). Esto brinda información valiosa sobre la escalabilidad y la capacidad de generalización del sistema. Al aumentar el número de datos y enfrentarlo a una mayor variedad de imágenes, es natural que surjan más desafíos para el clasificador. Sin embargo, el hecho de que solo se hayan producido unos pocos errores en comparación con el número total de imágenes es una señal positiva.

Este resultado sugiere que el clasificador sigue siendo capaz de manejar una mayor cantidad de datos mientras mantiene un nivel razonable de precisión. La aparición de algunos errores podría deberse a la complejidad adicional introducida por las imágenes.

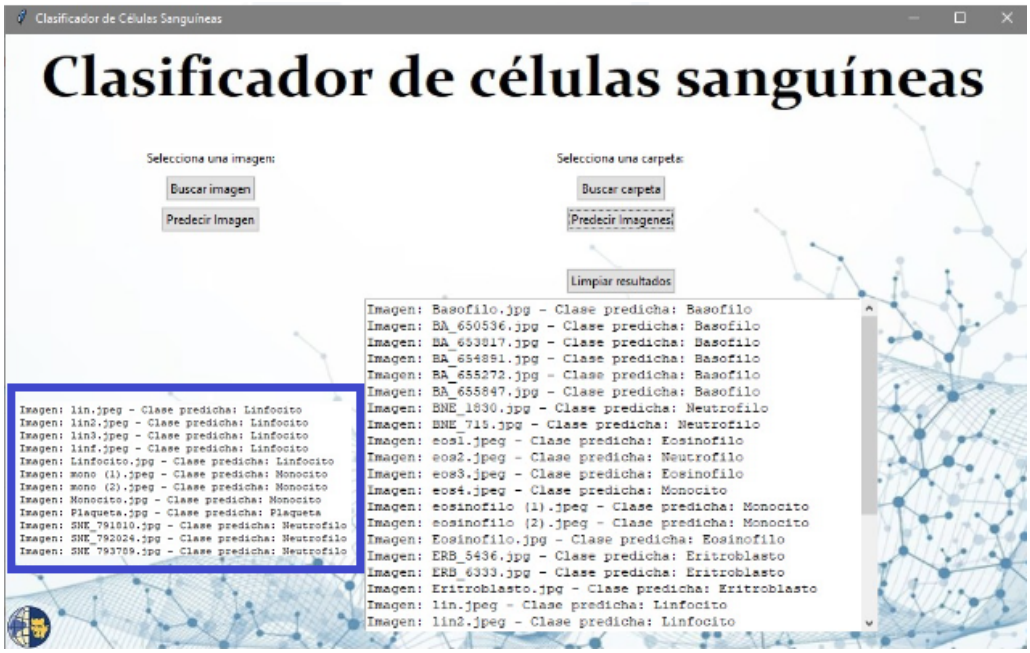

Figura 3.16: Clasificación de 30 imágenes.

Se evaluaron 500 imágenes de diversas clases y se obtuvo una clasificación perfecta para 20 basófilos en la gráfica (3.17) es una señal positiva en términos de escalabilidad. La capacidad del clasificador para reconocer correctamente una cantidad de muestras significativa, incluso en una base de datos más grande, sugiere que el sistema está demostrando cierto grado de robustez y capacidad de manejar volúmenes mayores de datos.

Figura 3.17: Clasificación de 500 imágenes.

En la figura 3.18 se observaron 9 errores en la clasificación de 500 imágenes de diversas clases. Lo que proporciona información adicional sobre la escalabilidad y la capacidad del clasificador para manejar una cantidad de datos más extensa y variada. Aunque es un resultado positivo que el clasificador haya presentado solo 9 errores en una prueba con 500 imágenes, es importante considerar el contexto y las implicaciones de estos errores.

Si el número de errores aumento en comparación con la prueba anterior sugiere que mientras que el número de datos aumenta, el clasificador puede enfrentar más desafíos en la generalización a través de diferentes clases y condiciones de imagen. Esto no necesariamente significa que el clasificador sea insuficiente, pero indica que podría ser necesario ajustar más el modelo, con respecto a las imágenes de entrada. Las imágenes de baja resolución suelen contener menos detalles y características distintivas, lo que puede dificultar la tarea de un clasificador para identificar patrones y características relevantes.

En términos generales, el hecho de que el clasificador haya logrado clasificar con éxito la mayoría de las imágenes en las pruebas, sugiere que el clasificador tiene un grado de utilidad y eficacia en la tarea de clasificación de células sanguíneas.

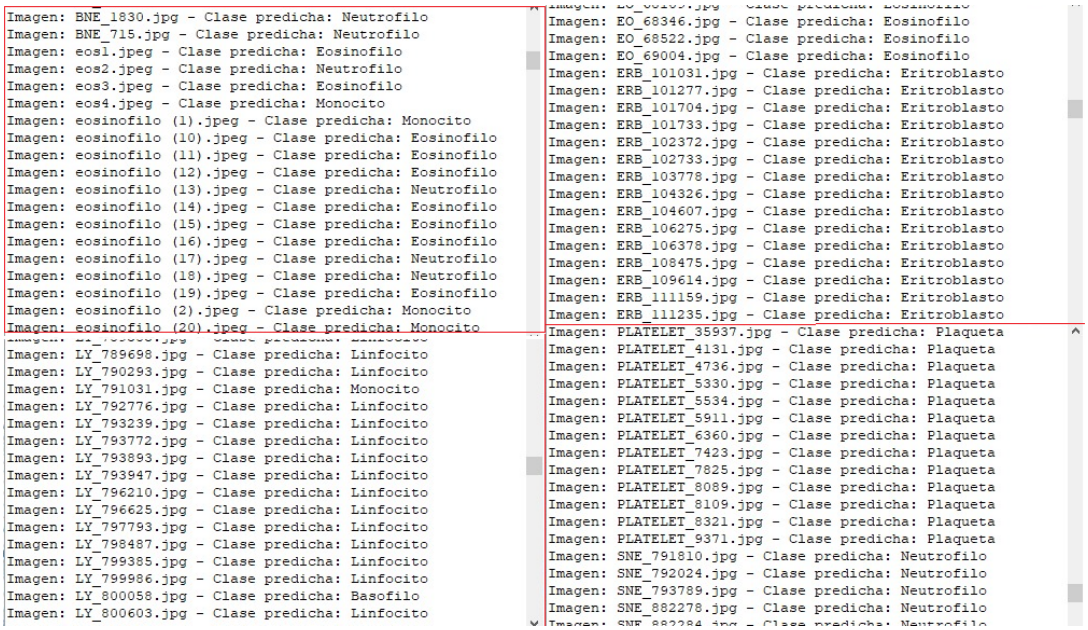

Figura 3.18: Predicciones de la clasificación de las 500 imágenes.

Tiempo

Realizar una prueba de tiempo en un clasificador de células sanguíneas es esencial para evaluar su eficiencia y rendimiento en términos de velocidad de procesamiento. Esta prueba implica medir cuánto tiempo le toma al clasificador analizar una determinada cantidad de muestras de células sanguíneas y emitir resultados. Conocer el tiempo que lleva al clasificador analizar una muestra de células sanguíneas ayuda a determinar cuán eficiente es el proceso de análisis. Si el tiempo es corto, significa que el clasificador puede procesar más muestras en un período de tiempo dado, lo que es especialmente crucial en entornos clínicos donde se deben analizar grandes volúmenes de muestras diariamente.

El entrenamiento del clasificador de células sanguíneas se realizó a lo largo de 20 épocas (3.19). Cada época implica un entrenamiento completo en el que la red ajusta sus pesos para mejorar su rendimiento. En total, el entrenamiento duró aproximadamente 8370 segundos, lo que equivale alrededor de 2 horas y 19 minutos.

El uso de dimensiones más bajas para las imágenes de células sanguíneas contribuyó a reducir el tiempo de entrenamiento en comparación con dimensiones más grandes. Esto se evidencia en el tiempo promedio por época, que varía entre aproximadamente 400 y 500 segundos, a comparación con tiempos más largos que podrían ocurrir con imágenes de resolución más alta. Esta disminución en el tiempo de entrenamiento se debe a que las imágenes más pequeñas requieren menos cálculos y memoria para procesar, lo que agiliza el proceso de entrenamiento del modelo.

```
Epoch 14/20
197/197 [==============================] - 406s 2s/step - loss: 0.3208 - accuracy: 0.8897 - val_loss: 0.1564 - val_accuracy: 0.
9539
Epoch 15/20
197/197 [==============================] - 406s 2s/step - loss: 0.2905 - accuracy: 0.8976 - val_loss: 0.1891 - val_accuracy: 0.
9429
Epoch 16/20
197/197 [==============================] - 417s 2s/step - loss: 0.2915 - accuracy: 0.9006 - val_loss: 0.1848 - val_accuracy: 0.
9479
Epoch 17/20
197/197 [==============================] - 479s 2s/step - loss: 0.2755 - accuracy: 0.9039 - val_loss: 0.1658 - val_accuracy: 0.
9479
Epoch 18/20
197/197 [==============================] - 413s 2s/step - loss: 0.2628 - accuracy: 0.9096 - val_loss: 0.1173 - val_accuracy: 0.
9654
Epoch 19/20
197/197 [==============================] - 407s 2s/step - loss: 0.2564 - accuracy: 0.9109 - val_loss: 0.1343 - val_accuracy: 0.
9643
Epoch 20/20
197/197 [==============================] - 412s 2s/step - loss: 0.2600 - accuracy: 0.9144 - val_loss: 0.1056 - val_accuracy: 0.
9664
```

Figura 3.19: Tiempo en el entrenamiento.

Los tiempos que se ven en la gráfica (3.20) se refieren al tiempo que le toma al clasificador procesar una muestra individual durante la etapa de clasificación. Esto indica el tiempo que lleva al clasificador procesar una única muestra. En general, se busca que estos tiempos sean lo más bajos posibles, ya que un tiempo de procesamiento más rápido significa que el clasificador puede analizar muestras más rápidamente.

Los tiempos de procesamiento individual que se muestran están bien. Indican que el clasificador está realizando las predicciones de manera eficiente y en un período de tiempo corto. Esto es bueno, ya que se busca implementar el clasificador en aplicaciones que requieren respuestas rápidas, como diagnósticos médicos en tiempo real. En promedio, cada predicción lleva alrededor de 30 a 50 milisegundos y si se toma un conjunto de datos grande de 500 imágenes tendría una duración de 20 segundos, dando un tiempo muy bajo y eficiente para un conjunto grande de imágenes. Teniendo en cuenta que los tiempos reales pueden variar según el hardware utilizado.

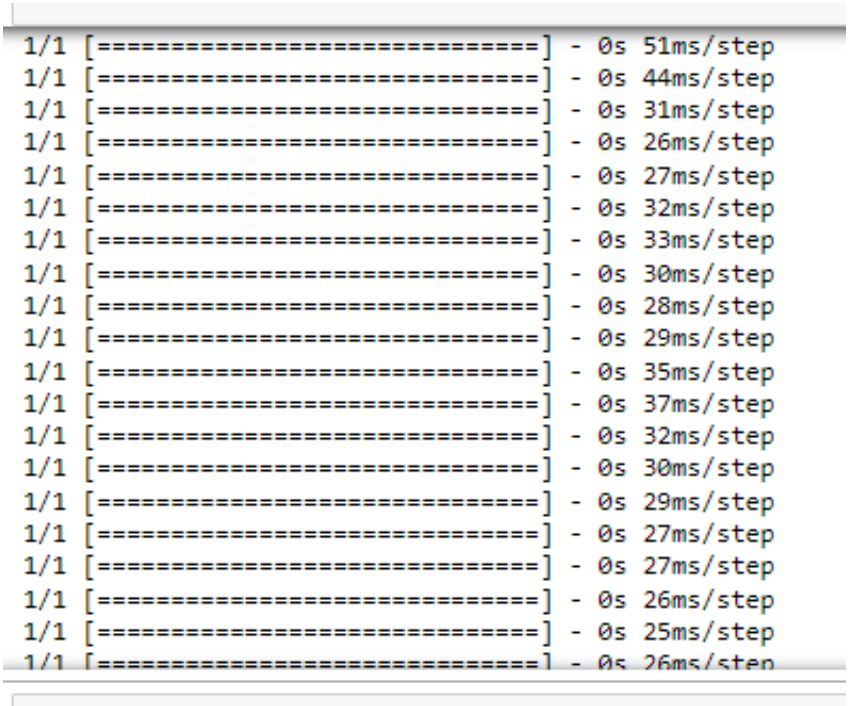

Figura 3.20: Tiempo en predecir.

Hablando de la estabilidad con respecto al tiempo que se demoró en clasificar, se observó que al subir el mismo conjunto de datos 20 veces. Los tiempos de predicción en todas las 20 repeticiones son similares y no varían drásticamente entre sí, esto indica que el clasificador está mostrando una cierta estabilidad en su rendimiento. Estos tiempos consistentes sugieren que el proceso de clasificación es predecible y que el modelo no está siendo afectado por fluctuaciones aleatorias o comportamientos inesperados.

En el caso de la escalabilidad, si el tiempo de predicción es relativamente constante a pesar de aumentar la cantidad de imágenes, indica que el sistema es capaz de manejar un mayor volumen de datos sin una degradación significativa en su velocidad de procesamiento. Esto muestra que el clasificador está diseñado de manera eficiente y puede acoplarse a una carga más grande sin experimentar un aumento considerable en los tiempos de respuesta. Una buena escalabilidad en términos de tiempo es esencial para garantizar que el clasificador sea práctico y útil en situaciones del mundo real donde la cantidad de datos puede ser aún más grande.

3.3.2. Pruebas de Funcionamiento para la técnica de contador

Estas pruebas de funcionamiento son importantes para garantizar la efectividad del contador basado en YOLOv5. Permiten analizar la precisión de la red en la tarea de contar objetos, identificar posibles errores como detecciones incorrectas y asegurar que el conteo sea preciso y coherente. Además, estas pruebas también ayudan a medir el rendimiento en parámetros de velocidad y eficiencia, asegurando que pueda procesar imágenes en tiempo real sin demoras significativas.

Adicionalmente, las pruebas de funcionamiento va a permitir ajustar el contador basado para adaptarse a diferentes escenarios y condiciones específicas. Al recopilar datos de prueba en una variedad de entornos, se puede mejorar la robustez del modelo y su capacidad para manejar situaciones diversas. Estas pruebas también son buenas para validar cualquier modificación que se realizó al modelo, asegurando que los cambios introducidos no afecten negativamente su desempeño y que sigan cumpliendo con los requisitos de precisión y eficiencia.

Matriz de confusión

Esta matriz representa la clasificación de células sanguíneas en tres categorías plaqueta, glóbulos rojos, glóbulos blancos. En el eje x se encuentran las clases reales, mientras que en el eje y se encuentran las clases que predijo la red. Cada celda contiene la proporción de cómo las células de la clase real fueron clasificadas en la clase predicha.

En esta matriz, se encuentra un background (3.21), que es la región de la imagen que no contiene las características específicas del objeto que se está tratando de identificar o clasificar, en este caso las células sanguíneas. En este caso, como no se obtuvo datos dedicados para el fondo, se toma a los glóbulos rojos como fondo, ya que están en gran parte de la imagen, por lo que muestra un background de 93 % con la clase predicha RBC.

Las plaquetas se están clasificando con alta precisión en su mayoría, ya que el 95 % de las plaquetas reales se clasifican correctamente como plaquetas predichas. La confusión entre plaquetas y glóbulos blancos es baja (1 %), pero existe cierta confusión con los glóbulos rojos (RBC) con un 4 %. Esto podría ser debido a ciertas similitudes morfológicas entre plaquetas y glóbulos rojos.

Los glóbulos rojos (RBC) se clasifican en gran medida con precisión (94 %), pero hay una confusión notable con las plaquetas (7 %). Esto sugiere que algunas características visuales compartidas entre plaquetas y glóbulos rojos pueden estar afectando la clasificación.

Los glóbulos blancos se clasifican de manera sobresaliente (100 %) como WBC, lo que indica que la diferenciación entre glóbulos blancos y las otras clases es clara. Sin embargo, la confusión con los glóbulos rojos (RBC) y las plaquetas es nula y baja respectivamente, lo que indica una precisión alta en la clasificación de glóbulos blancos.

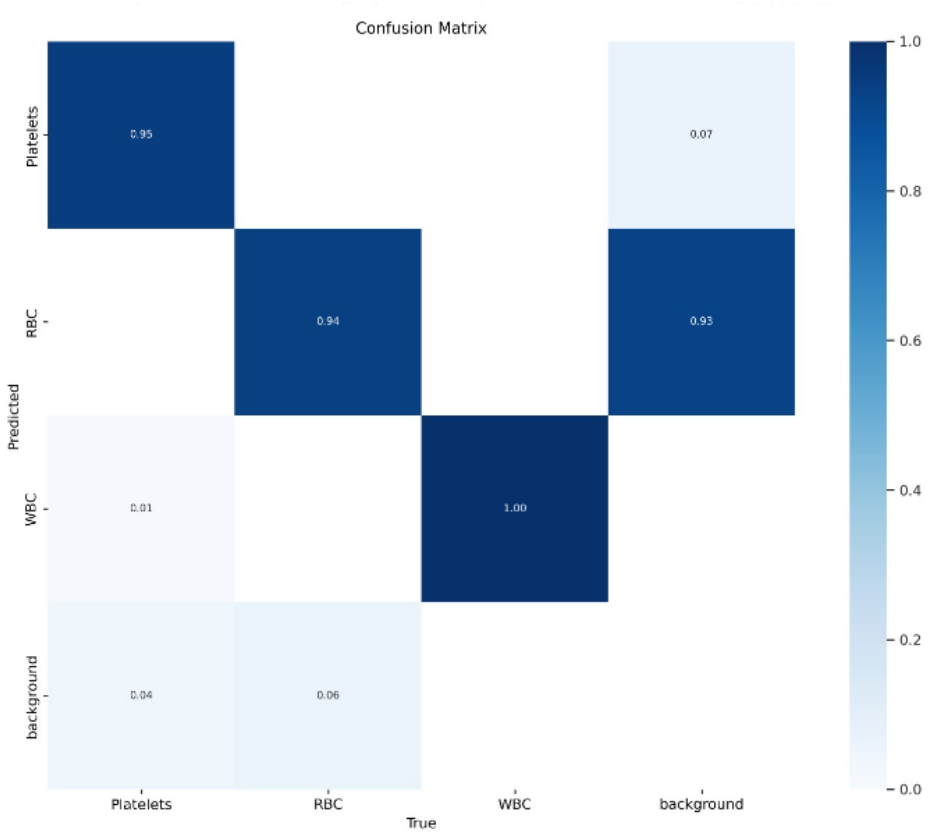

Figura 3.21: Matriz de confusión del contador.

Estabilidad

Una prueba de estabilidad tiene como objetivo proporcionar información valiosa sobre cómo el sistema contador de glóbulos rojos basado en YOLOv5 se comporta en situaciones de uso repetido. Los resultados ayudan a determinar si el sistema es adecuado para aplicaciones médicas y de diagnóstico, asegurando que sea confiable, preciso y consistente en la detección y el conteo de glóbulos rojos en muestras sanguíneas.

En esta evaluación, se ha aplicado repetidamente en el contador de células sanguíneas un conjunto específico de imágenes de muestras sanguíneas. Esta repetición se ha llevado a cabo 20 veces con el propósito de realizar una prueba de estabilidad. La esencia de esta prueba radica en observar cómo el sistema responde y se comporta en términos de rendimiento y resultados a medida que se somete a la misma tarea en múltiples ocasiones.

Además, esta prueba también busca evaluar la precisión y coherencia de

los valores resultantes en cada iteración. Esto implica verificar si el sistema genera recuentos de glóbulos rojos consistentes en todas las repeticiones, lo que es fundamental para confiar en los datos generados por la tecnología.

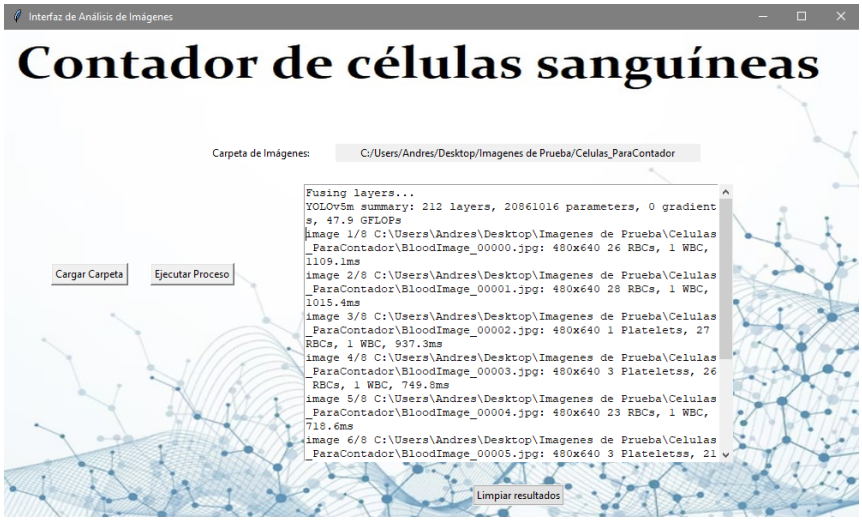

Figura 3.22: Primera prueba.

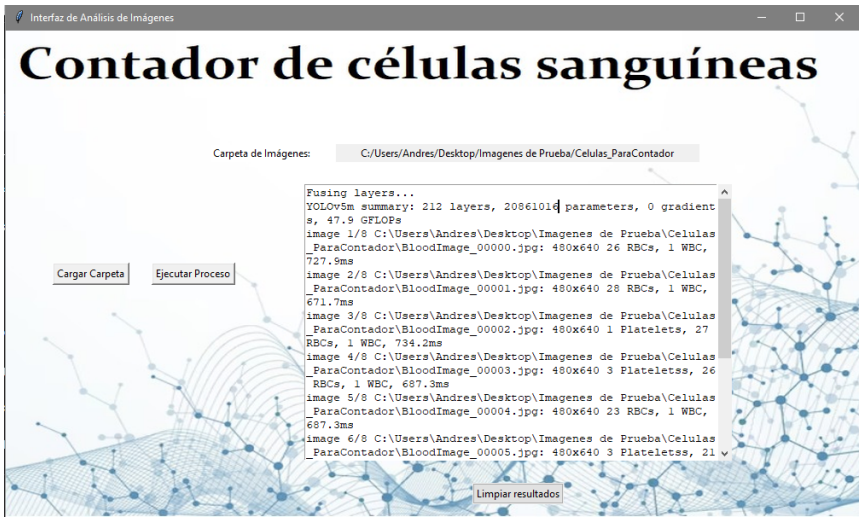

Figura 3.23: Vigésima prueba.

Los resultados obtenidos ofrecen una perspectiva sobre la estabilidad del contador de células sanguíneas en las 20 repeticiones de la prueba.

En términos de tiempo de procesamiento, se observa que se ha mantenido en niveles consistentes en las diferentes iteraciones. Si bien existen pequeñas fluctuaciones, la variación es relativamente baja. Esto sugiere que el sistema mantiene un rendimiento generalmente constante a pesar de las repeticiones.

Al analizar los tiempos de procesamiento en cada repetición, se evidencian diferencias entre ellas, pero estas no son considerables. Estas variaciones pueden deberse a factores externos como la carga del sistema en momentos específicos o el uso de recursos específicos en cada caso.

En cuanto a la coherencia de los resultados de detección, se puede apreciar que los recuentos de células sanguíneas en las imágenes se mantienen relativamente constantes. Aunque hay leves diferencias en algunos conteos, estas no indican una variación significativa en la precisión del sistema.

Se observa que el tiempo de inferencia (la etapa principal de procesamiento) puede variar en algunas repeticiones. Esto podría ser resultado de optimizaciones internas del modelo YOLOv5 o de la gestión de recursos durante la inferencia. A pesar de estas variaciones, el rendimiento general del sistema parece mantenerse estable.

Los resultados muestran que el contador de células sanguíneas muestra una estabilidad aceptable en cuanto a tiempo de respuesta y resultados de detección a lo largo de las 20 repeticiones de la prueba. Las fluctuaciones son moderadas y no indican un deterioro significativo del rendimiento.

Al observar las imágenes procesadas, la misma imagen se sometió al proceso en la primera y la vigésima repetición de la prueba. Notablemente, los resultados obtenidos presentan similitudes significativas, lo que indica un nivel notable de estabilidad en el desempeño del contador. Esta coherencia en los resultados a lo largo de repeticiones consecutivas resalta la robustez del sistema y sugiere que el contador mantiene una consistencia apreciable en su funcionalidad.

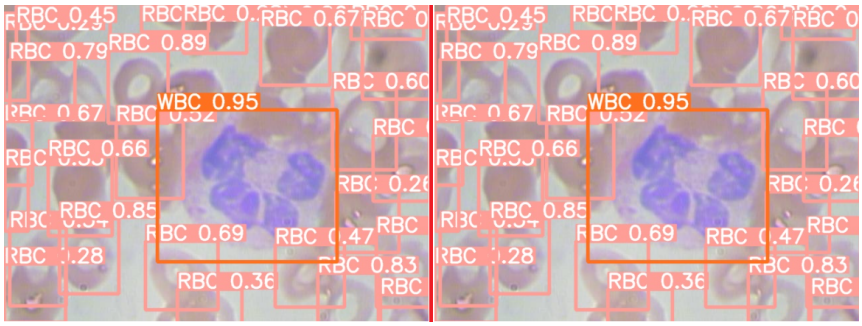

Figura 3.24: Imágenes en la primera y vigésima prueba.

Escalabilidad

La evaluación de la escalabilidad de un contador de células sanguíneas basado en YOLOv5 implica comprender cómo el sistema responde y se adapta al aumentar la cantidad de trabajo que debe manejar. En esencia, se trata de observar cómo el sistema mantiene su rendimiento y eficiencia a medida que se enfrenta a un mayor volumen de datos y tareas a procesar. Esta evaluación proporciona información crucial sobre la capacidad del contador para manejar mayores demandas y tamaños de muestras, lo que es esencial para su viabilidad en aplicaciones médicas y de diagnóstico.

El proceso de evaluación de la escalabilidad involucró tres pruebas distintas, cada una diseñada para comprender cómo el sistema se adapta a diferentes niveles de carga de trabajo y volúmenes de datos.

En la primera prueba, se utilizó una carpeta con 10 imágenes de muestras sanguíneas. El objetivo fue observar cómo el contador manejaba una carga de trabajo relativamente pequeña. Esta prueba permitió identificar si el sistema podía mantener su eficiencia y precisión incluso en situaciones de baja demanda.

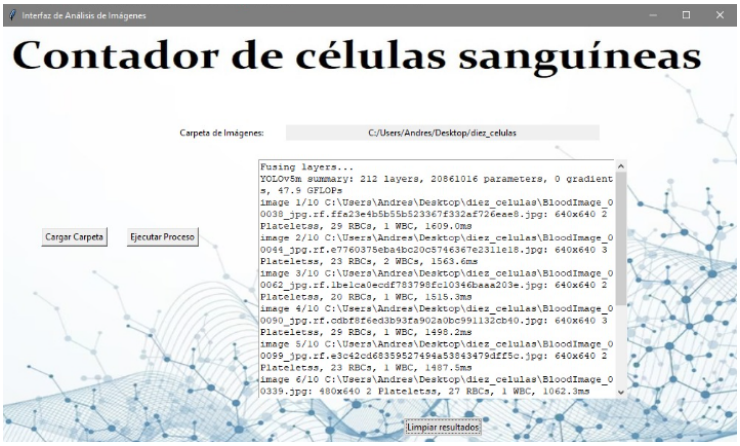

Figura 3.25: Prueba con diez imágenes.

Los resultados proporcionados de la prueba con diez imágenes ofrecen una visión sobre la respuesta del contador de células sanguíneas ante una carga de trabajo relativamente pequeña.

Se puede notar que el tiempo de procesamiento (2124.5 ms en el caso más alto) es considerable en comparación con la cantidad de imágenes procesadas. Esta observación sugiere que el sistema está funcionando de manera eficiente en términos

de tiempo de respuesta.

Los resultados de detección revelan recuentos consistentes en la mayoría de las imágenes. La presencia de variaciones en los recuentos podría atribuirse a las particularidades de cada muestra. La detección de componentes como plaquetas y glóbulos rojos se mantiene, lo que indica la capacidad del sistema para adaptarse a diferentes tipos de células sanguíneas.

En términos de escalabilidad, los resultados sugieren que el contador es capaz de manejar una carga de trabajo limitada de diez imágenes de manera eficiente y precisa.

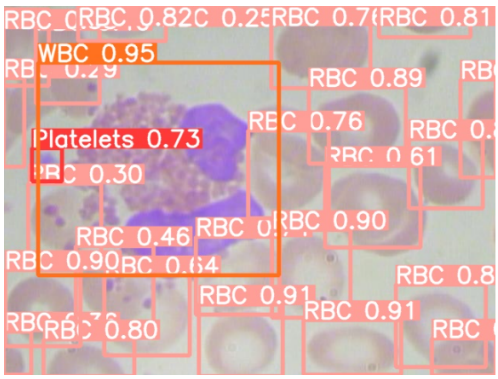

Figura 3.26: Resultado de la prueba con diez imágenes.

En la segunda prueba, el sistema se sometió a un conjunto de 70 imágenes. Aquí, la intención era explorar cómo el contador respondía a una carga de trabajo moderada. El enfoque estaba en verificar si el sistema podía mantener un rendimiento constante y resultados precisos a medida que la cantidad de datos aumentaba.

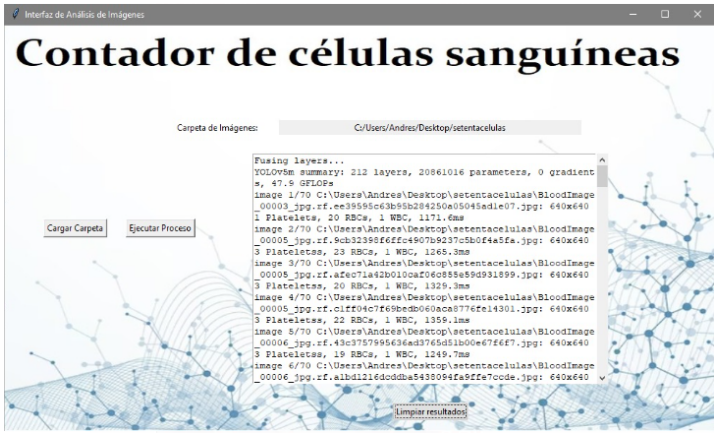

Figura 3.27: Prueba con setenta imágenes.

La evaluación de escalabilidad del contador de células sanguíneas basado en YOLOv5 se llevó a cabo mediante una prueba con setenta imágenes. Los resultados proporcionan una visión de cómo el sistema se comporta mientras se aumenta el volumen de datos. En términos de tiempo de procesamiento, se observa un incremento en comparación con la prueba anterior con diez imágenes, pero este aumento no es exponencialmente significativo, lo que sugiere una capacidad razonable de manejar una mayor carga de trabajo.

En cuanto a la precisión de detección, la mayoría de los resultados son coherentes y precisos, indicando la robustez del sistema ante una cantidad mayor de imágenes. Aunque algunas imágenes pueden mostrar variaciones en los recuentos de células, estas diferencias son esperables debido a las propias características biológicas de las muestras. En resumen, la prueba con setenta imágenes refleja que el contador de células sanguíneas mantiene su desempeño en situaciones de mayor demanda, brindando información valiosa para considerar en futuras expansiones del volumen de datos a procesar.

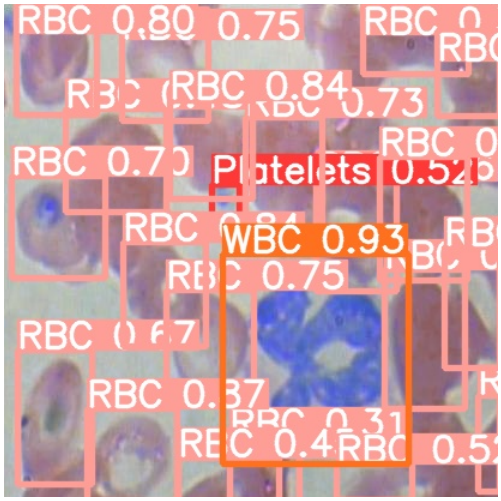

Figura 3.28: Imagen resultado de la prueba con setenta imágenes.

Finalmente, en la tercera prueba, se incrementó significativamente la carga de trabajo al utilizar un conjunto de 300 imágenes. Este escenario buscaba evaluar los límites de la escalabilidad del sistema. La pregunta clave era si el contador sería capaz de gestionar de manera eficiente una carga de trabajo más intensa sin degradar su rendimiento o precisión.

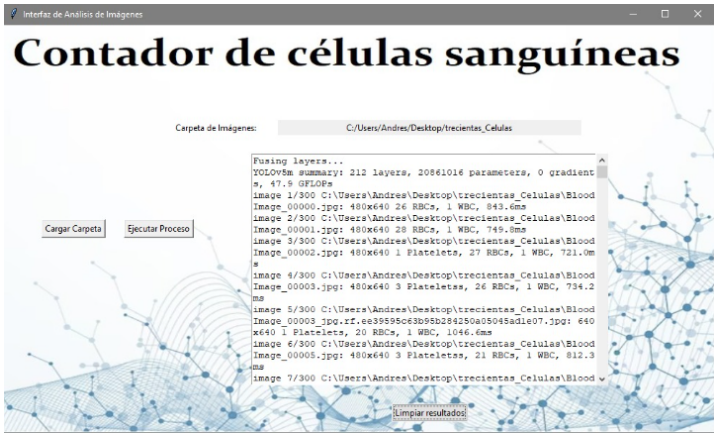

Figura 3.29: Prueba con trescientas imágenes.

Los resultados obtenidos en términos de tiempo de procesamiento, se observa un aumento en comparación con las pruebas anteriores con 10 y 70 imágenes. Este aumento en el tiempo de procesamiento era esperable debido a la mayor cantidad de imágenes y la complejidad asociada. A pesar de este incremento, el sistema aún demuestra una capacidad razonable para manejar una carga de trabajo considerable sin un aumento exponencialmente significativo en el tiempo de respuesta.

La precisión de detección se mantiene en niveles coherentes con las pruebas previas. Aunque pueden existir algunas variaciones en los recuentos de células en algunas imágenes, estas discrepancias son comprensibles debido a las diferencias naturales en las muestras biológicas. La capacidad del sistema para detectar y contar diferentes tipos de células sanguíneas, incluidas las plaquetas y los glóbulos rojos, sigue siendo evidente en esta prueba de mayor envergadura.

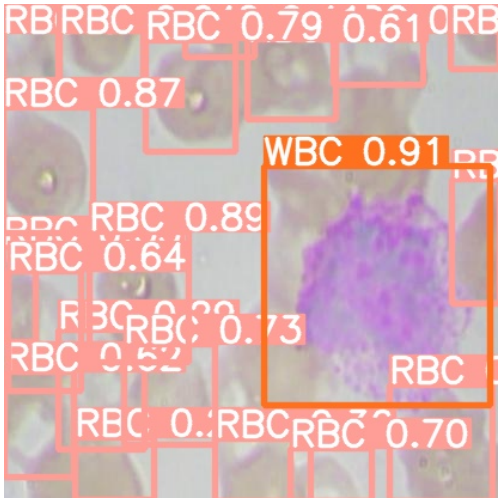

Figura 3.30: Imagen resultado de la prueba con trescientas imágenes.

Mientras que el número de imágenes incrementa, se aprecia un aumento gradual en el tiempo de procesamiento del contador de células sanguíneas. Sin embargo, a pesar de este incremento, el sistema logra mantener su capacidad para detectar y contar células sanguíneas con un alto grado de precisión. Este comportamiento sugiere que el contador posee una adecuada escalabilidad y se presenta como una solución viable para escenarios en los que se manejen volúmenes crecientes de datos de imágenes.

Tiempo

Realizar pruebas detalladas y examinar el tiempo de procesamiento en un contador automatizado de células sanguíneas que se basa en modelos como YOLOv5 es un paso esencial para maximizar su eficiencia y aplicabilidad. Este enfoque se fundamenta en la necesidad de garantizar un rendimiento óptimo del sistema. Al conocer con precisión cuánto tiempo lleva procesar las imágenes de células sanguíneas, se pueden tomar decisiones informadas para ajustar y afinar el funcionamiento del contador

La evaluación del tiempo de procesamiento proporciona una comprensión completa del rendimiento en cuanto a velocidad y agilidad. Esta información es crucial para optimizar los recursos de hardware y la asignación de capacidad de procesamiento, asegurando que el contador pueda funcionar con eficiencia incluso en situaciones de alta demanda.

El análisis de los tiempos de entrenamiento proporciona una visión esclarecedora sobre la eficiencia y la practicidad del modelo. Se pudo observar que el entrenamiento se extendió por lo menos 45 minutos y 16 segundos. Esta duración demuestra que la red pudo alcanzar un grado alto de precisión en un período de tiempo relativamente razonable.

Este hecho tiene implicaciones significativas en la viabilidad y aplicabilidad del modelo en escenarios prácticos. La capacidad de lograr una alta precisión en un tiempo de entrenamiento relativamente corto sugiere que el modelo es adecuado para tareas que requieren una detección eficiente y precisa de objetos en imágenes. En situaciones en las que el tiempo es un factor crítico, como la identificación en tiempo

real o el análisis de grandes conjuntos de datos, esta eficiencia temporal se convierte en un atributo esencial.

```
YOLOv5m summary: 212 layers, 20861016 parameters, 0 gradients
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 5/5 [00:06<00:00,  1.39s/it]
                   all         73        967       0.85      0.905      0.923      0.629
             Platelets         73         76        0.8      0.892      0.901      0.487
                   RBC         73        819      0.783      0.822      0.887      0.621
                   WBC         73         72      0.968          1      0.981       0.78
Results saved to drive/MyDrive/Working/yolov5/runs/train/BCCM23
CPU times: user 27.2 s, sys: 2.99 s, total: 30.2 s
Wall time: 45min 16s
```

Figura 3.31: Tiempo empleado en el entrenamiento.

En cuanto al análisis de tiempo de predicción, en las tres pruebas, realizadas para el análisis de escalabilidad, se observa un patrón consistente en cuanto a los tiempos de procesamiento. En la prueba inicial con diez imágenes, se logra un tiempo de procesamiento eficiente de alrededor de, 2124.5 ms, indicando una respuesta rápida ante una carga de trabajo relativamente pequeña.

```
image 1/10 C:\Users\Andres\Desktop\diez_celulas\BloodImage_0
0038_jpg.rf.ffa23e4b5b55b523367f332af726eae8.jpg: 640x640 2
Plateletss, 29 RBCs, 1 WBC, 1515.3ms
image 2/10 C:\Users\Andres\Desktop\diez_celulas\BloodImage_0
0044_jpg.rf.e7760375eba4bc20c5746367e231le18.jpg: 640x640 3
Plateletss, 23 RBCs, 2 WBCs, 1530.9ms
image 3/10 C:\Users\Andres\Desktop\diez_celulas\BloodImage_0
0062_jpg.rf.lbelca0ecdf783798fc10346baaa203e.jpg: 640x640 2
Plateletss, 20 RBCs, 1 WBC, 1452.8ms
image 4/10 C:\Users\Andres\Desktop\diez_celulas\BloodImage_0
0090_jpg.rf.cdbf8f6ed3b93fa902a0bc991132cb40.jpg: 640x640 3
Plateletss, 29 RBCs, 1 WBC, 1499.6ms
image 5/10 C:\Users\Andres\Desktop\diez_celulas\BloodImage_0
0099_jpg.rf.e3c42cd68359527494a53843479dff5c.jpg: 640x640 2
Plateletss, 23 RBCs, 1 WBC, 1515.3ms
image 6/10 C:\Users\Andres\Desktop\diez_celulas\BloodImage_0
0339.jpg: 480x640 2 Plateletss, 27 RBCs, 1 WBC, 1140.4ms
```

Figura 3.32: Tiempo empleado en la predicción de diez imágenes.

Mientras la cantidad de imágenes se aumenta a setenta en la segunda prueba, se nota un incremento gradual en los tiempos de procesamiento, pero el sistema sigue manteniendo una velocidad aceptable, lo que sugiere una escalabilidad adecuada para volúmenes mayores de datos.

```
image 1/70 C:\Users\Andres\Desktop\setentacelulas\BloodImage
_00003_jpg.rf.ee39595c63b95b284250a05045ad1e07.jpg: 640x640
1 Platelets, 20 RBCs, 1 WBC, 1086.4ms
image 2/70 C:\Users\Andres\Desktop\setentacelulas\BloodImage
_00005_jpg.rf.9cb32398f6ffc4907b9237c5b0f4a5fa.jpg: 640x640
3 Plateletss, 23 RBCs, 1 WBC, 1066.9ms
image 3/70 C:\Users\Andres\Desktop\setentacelulas\BloodImage
_00005_jpg.rf.afec71a42b010caf06c855e59d931899.jpg: 640x640
3 Plateletss, 20 RBCs, 1 WBC, 1109.7ms
image 4/70 C:\Users\Andres\Desktop\setentacelulas\BloodImage
_00005_jpg.rf.c1ff04c7f69bedb060aca8776fe14301.jpg: 640x640
3 Plateletss, 22 RBCs, 1 WBC, 1182.7ms
image 5/70 C:\Users\Andres\Desktop\setentacelulas\BloodImage
_00006_jpg.rf.43c3757995636ad3765d51b00e67f6f7.jpg: 640x640
3 Plateletss, 19 RBCs, 1 WBC, 973.0ms
```

Figura 3.33: Tiempo empleado en la predicción de setenta imágenes.

La tercera prueba, con 300 imágenes, sigue esta tendencia con un aumento en el tiempo de procesamiento, aunque el sistema demuestra nuevamente su capacidad para manejar cargas de trabajo más pesadas sin comprometer gravemente su rendimiento. En resumen, las pruebas indican que el contador de células sanguíneas mantiene una respuesta rápida y eficiente en diferentes escalas de trabajo, lo que lo hace prometedor para aplicaciones futuras en análisis de imágenes de gran volumen.

```
image 37/300 C:\Users\Andres\Desktop\trecientas_Celulas\Bloo
dImage_00023_jpg.rf.3e5daaba06ae3dd0a806db15b2ec245a.jpg: 64
0x640 1 Platelets, 26 RBCs, 1 WBC, 1413.6ms
image 38/300 C:\Users\Andres\Desktop\trecientas_Celulas\Bloo
dImage_00023_jpg.rf.acd46cf05f5b1c4c8d9c5255711c7434.jpg: 64
0x640 1 Platelets, 24 RBCs, 1 WBC, 1421.5ms
image 39/300 C:\Users\Andres\Desktop\trecientas_Celulas\Bloo
dImage_00023_jpg.rf.d004696fe9eac6dd2182059bfcee9dae.jpg: 64
0x640 1 Platelets, 22 RBCs, 1 WBC, 1374.7ms
image 40/300 C:\Users\Andres\Desktop\trecientas_Celulas\Bloo
dImage_00024.jpg: 480x640 23 RBCs, 1 WBC, 749.8ms
image 41/300 C:\Users\Andres\Desktop\trecientas_Celulas\Bloo
dImage_00024_jpg.rf.5ad4a325d8f4d9347e2f4c797dedc2f6.jpg: 64
0x640 18 RBCs, 1 WBC, 937.3ms
```

Figura 3.34: Tiempo empleado en la predicción de diez imágenes.

3.3.3. Pruebas de Funcionamiento para las dos técnicas en conjunto

Dado que el sistema de conteo y clasificación opera como una integración de ambas técnicas, se evitará la redundancia de llevar a cabo análisis adicionales sobre precisión, estabilidad, escalabilidad y tiempo en esta configuración conjunta. Cada una de estas técnicas ya ha sido evaluadas de manera individual en etapas previas, haciendo innecesaria una revisión duplicada en el contexto de esta operación conjunta.

Las próximas pruebas consistirán en la aplicación del proceso de conteo y clasificación a diversas imágenes de células sanguíneas, permitiendo un análisis

detenido de los resultados obtenidos. Se prestará especial atención al recorte efectuado en cada imagen después del proceso de conteo, el cual constituye un paso crucial para su posterior clasificación.

Se procedió a analizar los resultados obtenidos de las ocho imágenes de células sanguíneas procesadas. Cada imagen fue evaluada en términos de dimensiones (480x640) y el conteo de componentes sanguíneos presentes. En las imágenes, se observó una variabilidad en los recuentos de glóbulos rojos (RBC) y plaquetas (Platelets), así como la presencia constante de un glóbulo blanco (WBC). Los tiempos de procesamiento también se registraron, oscilando entre, 1060.2 ms y 1172.3 ms para cada imagen, lo que sugiere un rango relativamente constante en los tiempos de inferencia.

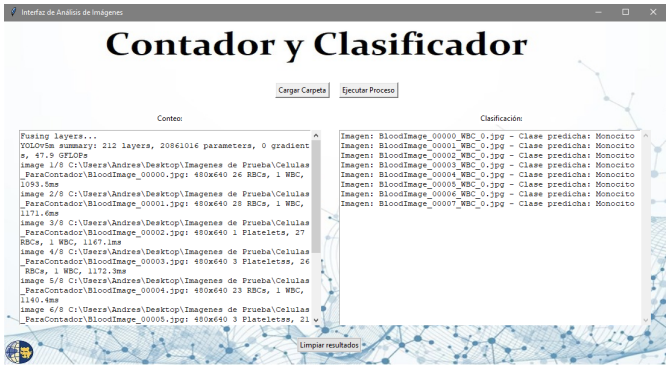

Figura 3.35: Conteo y Clasificación de Imágenes.

Después de llevar a cabo el proceso de conteo de células, el modelo procede a realizar un recorte de la imagen. Este recorte se enfoca en el glóbulo blanco, utilizando las coordenadas precisas obtenidas a partir de la caja delimitadora generada durante el proceso de detección.

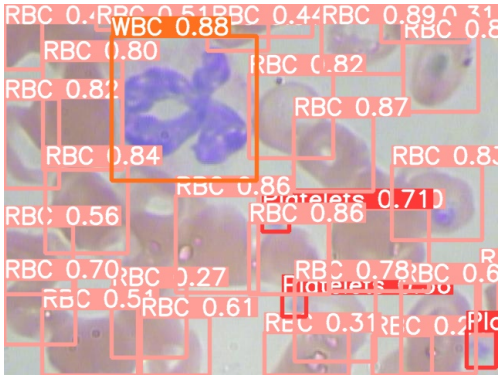

Figura 3.36: Imagen con cajas delimitadoras.

```
BloodImage_00000,230,159,500,387,WBC
BloodImage_00001,71,305,288,480,WBC
BloodImage_00002,340,2,534,113,WBC
BloodImage_00003,139,41,327,228,WBC
BloodImage_00004,98,119,323,327,WBC
BloodImage_00005,232,275,446,480,WBC
BloodImage_00006,108,185,286,356,WBC
BloodImage_00007,196,80,410,295,WBC
```

Es evidente que al ampliar el área de recorte, se logra una mejor apreciación del glóbulo blanco en la imagen.

Figura 3.37: Imagen recortada.

La fase de clasificación después del conteo también fue analizada. En cada caso, se presentó una clase predicha: "Monocito". Esto indica que el sistema identificó y clasificó con éxito la presencia de monocitos en todas las imágenes evaluadas. Este nivel de precisión en la clasificación es una indicación positiva de la robustez y efectividad del modelo.

En conjunto, los resultados reflejan una capacidad constante para detectar y contar diferentes tipos de células sanguíneas en imágenes con tiempos de procesamiento coherentes. La clasificación exitosa de monocitos en todas las imágenes muestra una prometedora funcionalidad en el proceso de conteo y clasificación de células sanguíneas.

3.3.4. Prueba de Funcionamiento para el entrenamiento

Para pruebas de funcionamiento en la ventana de entrenamiento, se necesita un conjunto de datos que incluya un número suficiente de imágenes de células sanguíneas junto con sus respectivas etiquetas. Esta base de datos se cargará a través del botón "Seleccionar Base de Entrenamiento". Una vez que se haya cargado la carpeta, se puede iniciar el proceso haciendo clic en el botón "Entrenar". Al presionar este botón, aparecerá una ventana de advertencia, mostrado en la figura 3.38, que le informará que el proceso tomará aproximadamente 3 horas y que es importante no apagar el equipo ni cerrar el programa durante este tiempo.

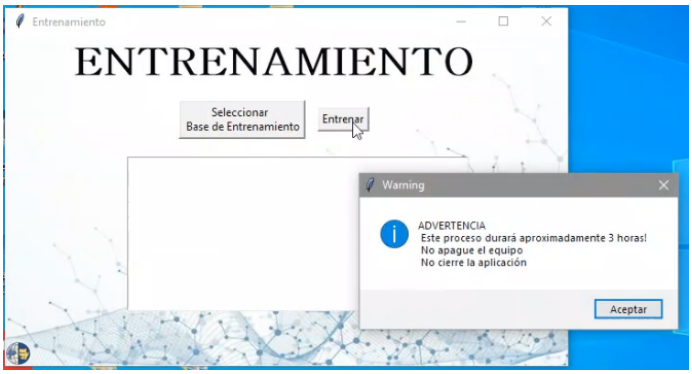

Figura 3.38: Ventana de Advertencia.

Una vez finalizado el proceso de entrenamiento, que tiene una duración de aproximadamente 3 horas, se mostrará una ventana emergente con un mensaje que indica que el entrenamiento se ha completado satisfactoriamente (Fig.3.39).

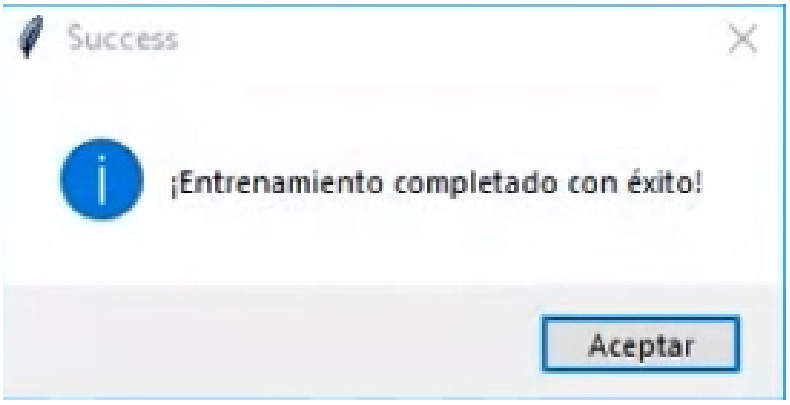

Figura 3.39: Mensaje Entrenamiento Completado.

Además, en el cuadro de texto se exhibirá la precisión obtenida tanto en el

proceso de entrenamiento y validación, junto con el nombre con el que se ha guardado el modelo entrenado (Fig.3.40).

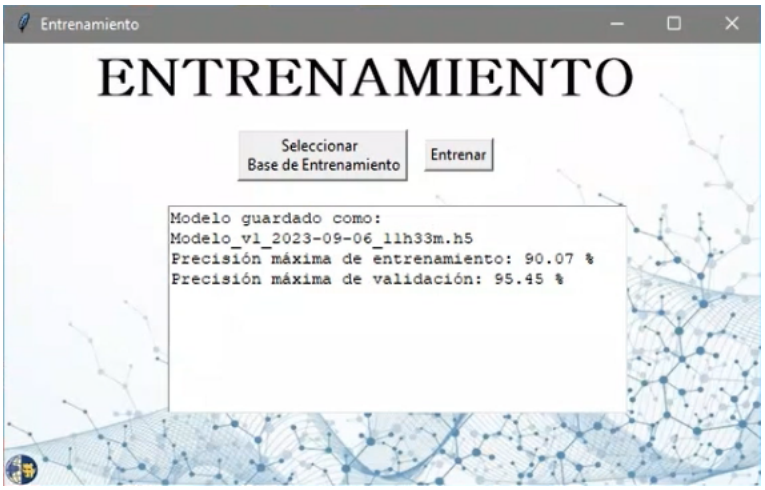

Figura 3.40: Precisión Obtenida.

Capítulo 4

Conclusiones y Trabajos Futuros

4.1. Conclusiones

Las pruebas de evaluación de las dos técnicas seleccionadas evidencian la capacidad positiva para manejar incrementos graduales en la cantidad de muestras de imágenes de células sanguíneas. Si bien, se detectaron leves variaciones en los tiempos de respuesta, la precisión se mantuvo en niveles elevados en ambas técnicas, indicando adaptabilidad ante aumentos en la demanda de procesamiento.

Respecto a la estabilidad, las 20 pruebas reiteradas sobre los mismos datos mostraron que tanto la clasificación como el conteo lograron una consistencia significativa en sus resultados. Las fluctuaciones fueron mínimas entre ejecuciones sucesivas, demostrando solidez de los modelos ante usos repetidos.

Las matrices de confusión, en el caso de la clasificación, mostró altos niveles de precisión en eosinófilos (97 %), neutrófilos (99 %), linfocitos (98 %) y otras células, con algunas confusiones entre tipos. Por su parte, YOLOv5, muestra buena precisión en plaquetas (95 %) y glóbulos rojos (94 %), con cierta confusión entre ambos. Ambas técnicas demostraron altos niveles de precisión en la categorización de distintos tipos de células. De acuerdo con esto, es necesario siempre considerar la matriz de confusión para validar los resultados ya que, aunque se detectaron confusiones entre algunas clases, la precisión general fue elevada.

En cuanto a los tiempos de procesamiento, en las dos técnicas, mostraron eficiencia y velocidad adecuadas. En la clasificación, los tiempos individuales por

imagen fueron rápidos, del orden de milisegundos. En el conteo, los tiempos de entrenamiento también resultaron apropiados para la detección eficiente de los distintos tipos de células. En general, ambas técnicas presentan velocidades de procesamiento aceptables para un análisis rápido y eficiente de imágenes de células sanguíneas.

Este estudio muestra la efectividad y versatilidad de la unión de técnicas de procesamiento de imágenes. La evaluación exhaustiva de diversas técnicas permitió la selección de enfoques óptimos para el conteo y la clasificación de células sanguíneas. El uso de una red neuronal convolucional personalizada y la arquitectura YOLOv5 destacaron por su capacidad para abordar ambas tareas de manera precisa y eficiente.

La integración de estas técnicas en un sistema conjunto ofreció resultados prometedores, logrando un conteo confiable de células sanguíneas y una clasificación precisa de glóbulos blancos en una única imagen.

La creación de una interfaz gráfica amigable para las pruebas de funcionamiento facilitó la interacción con el sistema, lo que podría tener implicaciones significativas en la práctica clínica y el diagnóstico médico.

Este trabajo también subraya la importancia de la selección cuidadosa de técnicas de procesamiento de imágenes según los requisitos específicos de la aplicación.

Se resalta el valor de la colaboración entre enfoques basados en redes neuronales convolucionales y arquitecturas especializadas como YOLOv5 para abordar desafíos complejos en el análisis de imágenes médicas.

Este estudio establece el fundamento para investigaciones y aplicaciones futuras en el diagnóstico automatizado y la interpretación de imágenes médicas, mejorando la precisión y eficiencia de la atención médica.

4.2. Recomendaciones

Los resultados de esta investigación ofrecen valiosas recomendaciones para futuros investigadores y profesionales que se dedican al procesamiento de imágenes médicas y la automatización del análisis en este ámbito.

En primer lugar, se destaca la importancia de seleccionar con precisión las técnicas más adecuadas para abordar problemas específicos en imágenes médicas. La elección de enfoques debe basarse en una evaluación exhaustiva y comparativa, teniendo en cuenta los requisitos únicos de cada aplicación.

Además, se recomienda la integración de múltiples enfoques en una solución conjunta. En este sentido, los resultados muestran que la unión de técnicas, como las usadas en este caso, puede ofrecer resultados más completos y precisos. Esta sinergia entre diferentes técnicas puede llevar a mejorar la calidad de los resultados.

4.3. Trabajos Futuros

Dentro del marco de esta investigación, se contemplan varias direcciones para futuros trabajos que podrían ampliar y mejorar aún más el campo de estudio:

- Mejora de la precisión: Aunque la combinación de las técnicas arrojó resultados prometedores, es esencial continuar refinando los algoritmos y las arquitecturas utilizadas para aumentar la precisión del conteo y la clasificación de células sanguíneas. Esto podría implicar ajustes en los hiperparámetros, la exploración de nuevas arquitecturas o la implementación de técnicas de regularización.

- Ampliación del conjunto de datos: La calidad y diversidad del conjunto de datos utilizado para entrenar y probar el sistema tienen un impacto significativo en su desempeño. Ampliar el conjunto de datos con una mayor variedad de imágenes y tipos de células sanguíneas podría ayudar a mejorar la generalización y la robustez del sistema.

- Validación clínica: Para que el sistema tenga un impacto clínico real, es necesario someterlo a rigurosas pruebas y validaciones en entornos clínicos reales. Esto implica colaborar con profesionales médicos y comparar los resultados del sistema con evaluaciones humanas para verificar su precisión y utilidad en situaciones prácticas.

- Optimización del rendimiento: La eficiencia computacional es crucial para la aplicación práctica de sistemas de procesamiento de imágenes en tiempo real.

Futuros trabajos podrían centrarse en optimizar los modelos y los algoritmos para disminuir el tiempos y tener una mejora en la velocidad de respuesta del sistema.

- Interfaz de usuario mejorada: La interfaz utilizada en las pruebas de funcionamiento podría ser objeto de mejoras para hacerla más intuitiva y amigable para los usuarios finales, como profesionales médicos. La retroalimentación de los usuarios podría ser de gran utilidad, con el objetivo de hacer que la interfaz sea más efectiva y accesible.

- Exploración de otras aplicaciones médicas: Las técnicas desarrolladas podrían aplicarse en otros contextos médicos, como la detección de otras anomalías celulares o estructuras en imágenes médicas. Explorar estas posibilidades podría ampliar el alcance y la relevancia del trabajo.

Glosario

CDF Función de distribución Acumulativa .

CNN Convolutional Neural Network (Red Neuronal Convolucional).

DCLNN Double Convolutional Layer Neural Networks (Red neuronal de capa de doble convolución).

DCNN Deep Convolutional Neural Network (Red neuronal convolucional con plantilla dinámica).

DNN Deep Neural Network (Red Neuronal Profunda).

FPN Red Piramidal de Funciones – Feature Pyramid Network.

GITEL Grupo de Investigación en Telecomunicaciones y Telemática .

GUI Graphical User Interface (Interfaz Gráfica de Usuario).

JPG Joint Photographic Experts Group.

NSI Narváez Soluciones Integrales.

PNG Gráficos de Red Portátiles – Portable Network Graphics.

RBC Red Blood Cell (Glóbulo Rojo de la Sangre).

ReLU Rectificied Lineal Unit.

RGB Red, Green, Blue.

RNA Redes Neuronales Artificiales – Artificial Neural Networks.

TIFF Formato de Archivo de Imágenes con Etiquetas – Tag Image File Format.

UPS Universidad Politécnica Salesiana.

WBC White Blood Cell (Glóbulo Blanco de la Sangre).

YOLO You Only Look Once.

Referencias

[1] M. D. Romero, «Manual del hemograma y el frotis de sangre periferica.,» págs. 20-259, 2013.

[2] E. Brambila, R. Castillo-Guerra y P. Lozano-Zarain, «Comparación entre tres métodos manuales empleados en la cuenta diferencial de leucocitos respecto a un equipo automatizado,» *Bioquimia*, vol. 28, n.º 3, págs. 4-12, 2003.

[3] A. Khashman, «IBCIS: Intelligent blood cell identification system,» *PROGRESS IN NATURAL SCIENCE-MATERIALS INTERNATIONAL*, vol. 18, n.º 10, págs. 1309-1314, oct. de 2008, ISSN: 1002-0071. DOI: 10.1016/j.pnsc.2008.03.026.

[4] P. Tiwari, J. Qian, Q. Li et al., «Detection of subtype blood cells using deep learning,» *COGNITIVE SYSTEMS RESEARCH*, vol. 52, págs. 1036-1044, dic. de 2018, ISSN: 1389-0417. DOI: 10.1016/j.cogsys.2018.08.022.

[5] M. Ammar, M. E. H. Daho, K. Harrar y A. Laidi, «Feature Extraction using CNN for Peripheral Blood Cells Recognition,» *EAI ENDORSED TRANSACTIONS ON SCALABLE INFORMATION SYSTEMS*, vol. 9, n.º 34, 2022, ISSN: 2032-9407. DOI: 10.4108/eai.20-10-2021.171548.

[6] Y. Xu, S. Xue, Y. Zou, J. Liao, Y. Sun e Y. Wang, «Fast classification and recognition method of blood cells using deep learning based on wrapped phase in polar coordinate,» *OPTIK*, vol. 261, jul. de 2022, ISSN: 0030-4026. DOI: 10.1016/j.ijleo.2022.169175.

[7] A. W. Tessema, M. A. Mohammed, G. L. Simegn y T. C. Kwa, «Quantitative analysis of blood cells from microscopic images using convolutional neural network,» *MEDICAL & BIOLOGICAL ENGINEERING & COMPUTING*, vol. 59, n.º 1, págs. 143-152, ene. de 2021, ISSN: 0140-0118. DOI: 10.1007/s11517-020-02291-w.

[8] M. Coral, G. Domínguez, D. J. Antonio y G. Bernal, «Obtención de Características de Subtipos de Leucemia en Imágenes Digitales de Células Sanguíneas para su Clasificación,» 2008.

[9] M. J. López, «Segmentación y cuantificación en imágenes de células sanguíneas denominadas plaquetas,» feb. de 2017. dirección: http://tesis.ipn.mx:8080/xmlui/handle/123456789/21394.

[10] D. J. S. Gonzalez y N. I. T. Bahena, «Practicas de histologia,» pág. 400, 2007.

[11] L. Gartner y J. Hiatt, «Biología Celular e Histología,» pág. 414, 2014.

[12] D. Harmening, «Clinical hematology and fundamentals of hemostasis,» pág. 730, 2002.

[13] B. Ernest y B. Brian, *Hematologia*, págs. 1-1036.

[14] J. W. Weisel y R. I. Litvinov, «Red blood cells: the forgotten player in hemostasis and thrombosis,» *Journal of Thrombosis and Haemostasis*, vol. 17, págs. 271-282, 2 feb. de 2019, ISSN: 15387836. DOI: 10.1111/JTH.14360.

[15] B. R. D. Jhon Jairo Jacqueline Carr, «Atlas de Hematología Clínica,» ene. de 2010.

[16] A. K. Abul, L. Andrew y P. Shiv, *Inmunologia Celular y Molecular- Abul K. Abbas*, vol. Sexta Edición, págs. 1-552.

[17] «Blausen0077Basophil2- Genotipia,» dirección: https://genotipia/genetica_medica_news/urticaria-vibratoria-adgre2/blausen_0077_basophil2-jpg/.

[18] B. Ingelsson, D. Söderberg, T. Strid et al., «Lymphocytes eject interferogenic mitochondrial DNA webs in response to CpG and non-CpG oligodeoxynucleotides of class C,» *Proceedings of the National Academy of Sciences of the United States of America*, vol. 115, E478-E487, 3 ene. de 2018, ISSN: 10916490. DOI: 10.1073/PNAS.1711950115.

[19] «Los neutrófilos: No sólo actores pasivos en la inmunidad - MiSistemaInmune,» dirección: https://www.misistemainmune.es/inmunologia/componentes/los-neutrofilos-no-solo-actores-pasivos-en-la-inmunidad.

[20] «Síndrome hipereosinofílico - NetMD® | Connect Healthcare,» dirección: https://netmd.org/alergologia-e-inmunologia-clinica/alergologia-e-inmunologia-articulos/sindrome-hipereosinofilico.

[21] «Álbumes 102+ Foto Que Es Monocitos En La Sangre El último,» dirección: https://dinosenglish.edu.vn/que-es-monocitos-en-la-sangre-1690665551813488/.

[22] «PLAQUETAS EN SANGRE – Valores Normales, Altas, Bajas, y Tipos.,» dirección: https://www.analiticadesangre.net/plaquetas/.

[23] B. J. Bain, «Blood Cells: A Practical Guide: Fourth Edition,» *Blood Cells: A Practical Guide: Fourth Edition*, págs. 1-476, oct. de 2007. DOI: 10.1002/9780470987551. dirección: https://onlinelibrary.wiley.com/doi/book/10.1002/9780470987551.

[24] L. Theory, «Tinción de Giemsa,» dirección: https://theory.labster.com/giemsa-es/.

[25] J. B. Christopher Layton, «Bancroft's theory and practice of histological techniques E-Book,» *Elsevier health sciences*, vol. 2422, págs. 47-63, 2018, ISSN: 19406029.

[26] UNCLESAMULUS, «Blood Cells Image Dataset | Kaggle,» dirección: https://www.kaggle.com/datasets/unclesamulus/blood-cells-image-dataset.

[27] R. C. Gonzalez, R. E. Woods y P. P. Hall, *Digital Image Processing Third Edition Pearson International Edition prepared by Pearson Education*.

[28] T. Boutell y J. Marti, «PNG (Portable Network Graphics) Specification Version 1.0 Status of this document,» 1997, págs. 1-81. dirección: http://www.w3.org/.

[29] A. S. Incorporated, «Licenses and Trademarks,» 1992. dirección: http://www.adobe.com/Support/TechNotes.htmlandftp://ftp.adobe.com/pub/adobe/DeveloperSupport/TechNotes/PDFfiles.

[30] G. K. Wallace, «The JPEG still picture compression standard,» 1992, págs. xviii-xxxiv. DOI: 10.1109/30.125072.

[31] R. C. Gonzalez y R. E. (E. Woods, *Digital image processing*, pág. 1168, ISBN: 9780133356724.

[32] B. Al, *H A N D B O O K O F IMAGE A m VIDEO PROCESSING*, págs. 1-974.

[33] W. K. Pratt, *Digital image processing : PIKS inside*. Wiley, 2001, pág. 735, ISBN: 0471374075.

[34] W. Burger y M. J. Burge, «Digital Image Processing.» dirección: http://www.springer.com/series/3191.

[35] L. A. Fernández, D. Diaz y R. Depaoli, «Optimización de la ecualización del histograma en el procesamiento de imágenes digitales,» *VII Workshop de Investigadores en Ciencias de la Computación*, págs. 480-483, August 2005. dirección: http://sedici.unlp.edu.ar/handle/10915/21082.

[36] C. Solomon y T. Breckon, «Fundamentals of Digital Image Processing A Practical Approach with Examples in Matlab.»

[37] H. Simon, «Neural Networks - A Comprehensive Foundation - Simon Haykin,»

[38] Ó. Reinoso, L. Jiménez y L. Paya, *Ejemplos Prácticos de Redes Neuronales Mediante MATLAB y PYTHON*. 2022. dirección: https://bibliotecas.ups.edu.ec:2708/lib/upsal/detail.action?docID=30293464&query=redes+neuronales#.

[39] C.-S. 0. de Dezembro de, «APLICANDO DEEP LEARNING PARA DETECCAO DE CELULAS EM AMOSTRAS DE SANGUE,» Accessed: 2023-9-11.

[40] A. Rehman y T. Saba, «Neural networks for document image preprocessing: State of the art,» *Artificial Intelligence Review*, vol. 42, págs. 253-273, 2 abr. de 2014, ISSN: 02692821. DOI: 10.1007/S10462-012-9337-Z/METRICS. dirección: https://link.springer.com/article/10.1007/s10462-012-9337-z.

[41] Y. Lecun, Y. Bengio y G. Hinton, «Deep learning,» *Nature*, vol. 521, págs. 436-444, 7553 mayo de 2015, ISSN: 14764687. DOI: 10.1038/nature14539.

[42] J. Redmon, S. Divvala, R. Girshick y A. Farhadi, «You only look once: Unified, real-time object detection,» *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-December, págs. 779-788, dic. de 2016, ISSN: 10636919. DOI: 10.1109/CVPR.2016.91.

[43] «PLAQUETAS EN SANGRE – Valores Normales, Altas, Bajas, y Tipos.,» dirección: https://www.analiticadesangre.net/plaquetas/.

[44] Á. Artola Moreno, «Clasificación de imágenes usando redes neuronales convolucionales en Python,» 2019.

[45] P. Mooney, «Blood Cell Images | Kaggle,» dirección: https://www.kaggle.com/datasets/paultimothymooney/blood-cells.