UNIVERSIDAD ANDINA DEL CUSCO

FACULTAD DE INGENIERÍA Y ARQUITECTURA

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

TESIS

Clasificación de leucocitos en imágenes microscópicas de frotis sanguíneo usando machine learning y CNN

TESIS PARA OPTAR EL TÍTULO DE

INGENIERO DE SISTEMAS

Presentado por:

Rojas Alvarez, Gonzalo

ASESOR:

Ing. Molero Delgado, Ivan

CUSCO – PERU

2022

Índice

Índice de figuras

Índice de tablas

Agradecimientos

Agradezco a mis padres, hermanos y amigos de la universidad, pues fueron las principales personas que me apoyaron y soportaron durante todo el desarrollo de este trabajo de investigación. Sin su ayuda, es muy probable que esta investigación nunca se hubiera culminado. Gracias por todo su apoyo.

También quiero agradecer a mi asesor Ing. Ivan Molero Delgado, quien me guio durante todo el proceso de esta investigación. También por toda su paciencia y ganas de que esta investigación sea lo más interesante posible.

Finalmente, quero terminar estos agradecimientos con unas palabras de aliento que siempre estuve recordando y repitiendo durante todo este tiempo.

> “ El antiguo código de los Caballeros Radiantes reza: «Viaje antes que destino.» Algunos lo consideran un simple lugar común, pero es mucho más. Un viaje incluirá dolor y fracaso. No son solo los pasos adelante los que debemos aceptar, sino también los traspiés. Las dificultades. El conocimiento de que fracasaremos. De que haremos daño a quienes nos rodean.
>
> Pero si nos detenemos, si aceptamos la persona que somos al caer, el viaje concluye. Ese fracaso pasa a ser nuestro destino.
>
> Amar el viaje implica no aceptar ese final. He descubierto, por medio de dolorosas experiencias, que **el paso más importante que puede dar alguien es siempre el siguiente**.”
>
> Dalinar Kholin – El archivo de las tormentas 3: Juramentada por Brandon Sanderson (2018)

Resumen

El problema principal de las formas actuales de realizar la clasificación de leucocitos es que solo tenemos dos opciones más comunes en el mercado, se utiliza una maquinaria de alto coste para poder hacer el procedimiento de clasificación o un técnico de laboratorio lo tiene que hacer manualmente con una muestra y un microscopio.

El objetivó de esta investigación es poder validar un tercer camino el cual utiliza tecnologías que tuvieron un gran avance recientemente en el mercado como lo son machine learning (más específicamente las Redes Neuronales Convolucionales) para la creación de nuevas soluciones con Inteligencia Artificial.

Con este objetivo en mente se siguieron varios pasos. El primero es la identificación del grado de error de los procesos manuales y automatizados. Para este objetivo, se realiza un análisis bibliográfico y se determinó que el promedio de grado de error de los procesos manuales y automatizados son de 18.33 % y 3.59 % respectivamente. Y para la investigación se decidió utilizar un grado de error mínimo de 5 %. La solución que cumplió con este valor seleccionado tiene garantizado ser una solución mejor que el procedimiento manual y ser comparable con procedimientos automatizados realizados por maquinaria especializada.

Para la obtención del dataset, el cual fue utilizado en los procesos de entrenamiento validación y testeo de los modelos de redes neuronales, se decidió utilizar datos de investigaciones previas realizadas por entidades de diferentes partes del mundo y publicados para su uso en investigación. Se utilizaron 4 fuentes de datos los cuales nos dieron un dataset diverso de 37760 muestras.

Como el dataset obtenido tenía una deficiencia de pocas muestras de un tipo especifico de leucocitos. Se decidió utilizar la técnica de preprocesamiento de datos llamada Data Augmentation; el cual, usando variaciones de rotación, flip, iluminación (valor gamma) y ruido gaussiano permitió aumentar el número de muestras a 59877.

Se procedió a realizar iteraciones para la detección de la estructura del modelo de red neuronal convolucional. Durante estas iteraciones se decidió utilizar técnicas contra el overfitting (el cual es uno de los principales problemas que pueden presentar las redes neuronales). Las técnicas las cuales se utilizaron fueron reducción de complejidad de la estructura de modelo, valores de regularización y capas de DropOut (capas que eliminan

aleatoriamente entradas de una capa oculta de la red neuronal). Todas estas técnicas se vieron efectivas ya que con su ayuda se logró determinar un modelo el cual cumplía con nuestro grado de error mínimo.

El ultimo paso fue un proceso de testeo, en este paso se realizaron las pruebas de predicción de uno a uno, uso de la función de evaluación propio de la framework de keras, generación de 129 dataset de 500 imágenes cada uno para prode verificar resultado obtenido con la función de evaluación previamente utilizado y generación de matriz de confusión para poder ver diferentes resultados de predicción.

Introducción

En la era actual se pudo observar un gran aumento de la capacidad de procesamiento y almacenamiento de las nuevas tecnologías de información, permitiendo el desarrollo y uso de herramientas más sofisticadas para la solución de problemas tanto recientes como antiguos. De esta manera, se desarrollaron nuevas técnicas para la trata de información que permiten el uso y clasificación de grandes cúmulos de datos con mayor eficiencia. Estos 2 hechos han permitido desarrollar algoritmos con la capacidad de buscar características por los cuales se pueda predecir un resultado usando datos previamente procesados. Estos algoritmos pertenecen a un subcampo de la ciencia de la inteligencia artificial llamado Machine Learning.

Machine Learning busca realizar un cambio a la forma de resolver problemas diferente a las soluciones tecnológicas estándar. Normalmente, cuando se quiere que una tecnología solucione un problema, su desarrollador tiende a establecer los pasos, reglas o normas a seguir para poder brindar las salidas requeridas. Por otro lado, Machine Learning busca dar la capacidad a un computador para predecir un resultado según unos datos de entrada ingresados; para que machine learning pueda predecir es necesario utilizar datos parecidos con anterioridad, con los cuales entrenar un modelo matemático el cual, en su proceso de entrenamiento, intenta predecir un resultado el cual es comparado con los datos reales y se procede a realizar un ajuste para poder acercarse más al resultado esperado.

La posibilidad de que un computador pueda encontrar una aproximación a la solución esperada se da gracias a su búsqueda de relaciones en los datos de entrada en el proceso de entrenamiento. Esta capacidad de poder encontrar relaciones en los datos nos permite resolver problemas como los que se puede observar en esta investigación. La clasificación de leucocitos es un proceso realizado para un hemograma o examen hematológico con el fin de brindar la suficiente información con el cual un médico pueda diagnosticar la presencia de alguna enfermedad relacionada con los niveles de leucocitos encontrados en la sangre. Este proceso puede ser realizado de muchas maneras, una de ellas son las maquinas especializadas los cuales usando luz infrarroja pueden realizar un conteo diferenciado de la células de una muestra de sangre y, aunque existen estas máquinas, no todas tienen la opción de realizar un conteo diferencial de todos los tipos de leucocitos; por otro lado, existe el proceso manual el cual es realizado por un técnico de laboratorio.

Este, en un frotis de sangre, tiene que contar y clasificar cada leucocito de manera manual. Esta tesis busca el demostrar la posibilidad de usar machine learning y CNN para poder clasificar leucocitos según imágenes obtenidas con un microscopio en un frotis de sangre como una alternativa a los otros métodos.

En el capítulo I se indica la problemática de la investigación, justificar por qué es importante el problema de clasificar leucocitos, los objetivos de esta investigación y resumir lo que se busca con tal.

En el capítulo II se indica el marco teórico. El marco teórico es información que se ve necesaria para poder entender cómo se podrá realizar la investigación; además, mostrar antecedentes de esta investigación, investigaciones que aportan tanto como experiencia previa o para reforzar la importancia del problema a resolver.

En el capítulo III se hace referencia al apartado de metodología. Aquí se encuentra descrito todo el proceso por el cual la investigación llega al resultado final. Se especifica el tipo de investigación, su diseño inicial y el proceso de recolección y análisis de datos.

En el capítulo IV se puede observar los resultados de la investigación. En este apartado se encuentra la respuesta a los objetivos que nos planteamos y la respuesta del objetivo general.

Capítulo 1. Problema de investigación

1.1. Ámbito de influencia

1.1.1. Ubicación geográfica

El desarrollo del proyecto de investigación se realizó en la ciudad del Cusco, Perú. La obtención de muestras para la investigación se realizó con ayuda de datos de otras investigaciones previas publicadas en internet.

1.1.2. Ámbito de influencia teórica

Área de dominio

La solución que se realizó para la problemática observada pretendia generar una red neuronal CNN (Convolutional Neural Network – Redes neuronales convolucionales), entrenarla y probar si la clasificación de leucocitos en imágenes microscópicas de un frotis sanguíneo cumple con un grado de error aceptable comparado a otras soluciones; por ende, se describió todo el proceso a detalle, desde la toma de imágenes con ayuda de un microscopio, el procesamiento de las imágenes para establecer el formato de los datos de entrada, entrenamiento de la red neuronal y realizacion las pruebas respectivas. Teniendo todo esto en cuenta, se decidió que este proyecto de investigación pertenece al área de tecnologías de información.

Líneas de investigación

El desarrollo de la red neuronal CNN (Convolutional Neural Network), el cual es un algoritmo de machine learning, tiene como objetivo el encontrar un algoritmo que clasifique leucocitos y que cumpla con una tasa de error aceptable según la evaluación de las otras soluciones. Por esta razón, para la línea de investigación se puede mencionar el desarrollo de aplicaciones usando inteligencia artificial; aunque en esta solución no se desarrollara un aplicativo, este algoritmo podría ser utilizado en el desarrollo de una aplicación para los usuarios finales.

1.2. Planteamiento del problema

1.2.1. Descripción de la situación actual

Los análisis hematológicos o hemogramas son utilizados en diagnósticos principalmente para el descarte de enfermedades que tengan que ver con la sangre; estos pueden ser de varios tipos, se puede tratar de revisar una deficiencia de eritrocitos o glóbulos rojos siendo la principal señal de anemia; también existe la deficiencia de leucocitos la cual sería señal de leucemia.

Uno de los exámenes más básicos y con más demanda en cualquier diagnóstico médico es el hemograma. Básicamente un hemograma o examen sanguineo es el análisis de las células de la sangre, este puede ser de 2 tipos; hemograma menor o mayor, en el caso del hemograma mayor se realiza un análisis extra el cual es un estudio diferencial de los glóbulos blancos o leucocitos tratando de distinguir sus tipos y generando un porcentual de los hallados en la muestra según los tipos.

Para poder describir la situación actual de un laboratorio clínico del Cusco, se realizó una visita a un laboratorio clínico. La participación del laboratorio clínico fue solo para tener información del estado de los exámenes hematológicos en el Cusco. Los datos que el laboratorio clínico pudo brindar fueron los siguientes:

> El laboratorio clínico es el área que se encarga de analizar muestras biológicas humanas (sangre, orina, etc.) obtenidas a petición de pacientes con previa revisión de un médico general o especialista para la obtención de datos de la muestra que puedan ser analizados por el especialista para apoyar su diagnóstico con los datos obtenidos.

En el laboratorio clínico, el hemograma es realizado con los siguientes pasos. Primero es necesario la obtención de la muestra, la cual es obtenida de manera presencial con el paciente para evitar la suplantación de identidad y además garantizar la identificación del paciente, esto se realiza en el mismo local en el cual se encuentra el laboratorio. Segundo, la muestra se pone en la máquina de análisis hematología, la cual tiene la capacidad necesaria para realizar un hemograma menor; la maquina tiene como salida los datos de pruebas referentes a la morfología de los eritrocitos (VCM, HCM, CHCM); conteo de glóbulos

blancos, rojos y plaquetas; etc. Tercero y último paso, se debe tener una muestra de la sangre bajo microscopio donde un técnico en laboratorio tiene que realizar un estudio más profundo de los glóbulos blancos de manera manual e ingresarlos a su sistema para poder imprimir los resultados.

Además de los datos brindados por el laboratorio clínico visitado, también se dio una muestra de un resultado de un hemograma. Se puede observar en el anexo 7.1.

1.2.2. Descripción del problema

Dentro de un hemograma existe el conteo diferenciado de leucocitos, normalmente esta clasificación de leucocitos se realiza con una muestra de sangre bajo microscopio (frotis de sangre) y es necesario tener a un técnico de laboratorio clínico que tenga que evaluar y diferenciar todos los tipos de glóbulos blancos que se encuentre para realizar su conteo y calcular el porcentaje de cada tipo de leucocitos por el total de leucocitos analizados; según el manual de prácticas para el laboratorio de hematología de la Universidad Nacional Autónoma de México (Olivera et al., 2017); el error del proceso de recuento manual de leucocitos, eritrocitos y plaquetas es del 10% y; además de los posibles problemas al hacer mal uso de los equipos, errores de dilución ,etc.; un trabajador tiende a ser afectado por factores externos como cansancio, apuro o bajo nivel de interés en sus labores, lo cual permite que la tasa de error varié.

Una mayor tasa de error en un hemograma puede significar varios problemas tanto para el paciente como para el médico, que necesita esos resultados para diagnosticar una enfermedad. En caso de algún error, el paciente tendrá problemas desde el más básico, como que está obligado a realizar otro hemograma en otro lugar lo cual significa un gasto extra para el paciente; o un problema más grave como el mal diagnóstico de una enfermedad al paciente, lo cual puede derivar en una mala toma de decisiones como el uso de medicación errónea y causar problemas médicos al paciente o hasta incluso la muerte.

1.3. Justificación

El uso de machine learning para la automatización de procesos tuvo un gran avance en los últimos 5 años gracias a la liberación de proyectos de código libre como lo son tensorflow y pytorch. Al automatizar un proceso de una de las actividades más realizadas en laboratorios clínicos como lo es el hemograma; se beneficia a los interesados al no solo crear una alternativa, sino también, mostrar su viabilidad frente a otras soluciones como lo pueden ser el proceso manual o maquinas especializadas de alto costo. Y, además, como parte del proceso de crear la red neuronal CNN, se tiene que describir cómo se tendría que realizar todo el proceso, desde la toma de imágenes hasta la descripción de la arquitectura final de la red neuronal el cual tiene como objetivo el de realizar la clasificación de leucocitos. En este proyecto de investigación se tienen en cuenta trabajos de investigación previos que lograron demostrar que el machine learning y CNN si es una alternativa para el procedimiento del conteo diferencial manual de leucocitos, pero una observación que se puede dar en estos proyectos de investigación es el uso de tecnologías en la nube, como el uso de servidores especializados en machine learning para la creación del modelo matemático, en esta investigación se evaluara los requerimientos de hardware para ver si es posible ejecutar el modelo de manera local. Por otro lado, en los proyectos encontrados no se pudo observar el proceso con el cual se desarrolla la solución usando machine learning, nos referimos a proceso a los pasos que se deben de tomar para que sea posible realizar la clasificación de leucocitos en un frotis de sangre usando machine learning, desde la obtención de imágenes, preprocesamiento de estos, creación de los datos de para entrenamiento del modelo de machine learning, etc.

1.4. Formulación del problema

1.4.1. Formulación interrogativa del problema general

¿La clasificación de leucocitos en imágenes microscópicas de frotis sanguíneo realizado con machine learning y CNN (Convolutional Neural Network) puede cumplir un grado de error aceptable que lo valide como posible alternativa?

1.4.2. Formulación interrogativa de los problemas específicos

- ¿Cuál sería el grado de error aceptable que debería cumplir la alternativa de clasificación leucocitos?
- ¿Cómo se tendría que tratar las imágenes microscópicas de frotis sanguíneo para su uso en una red neuronal CNN (Convolutional Neural Network)?
- ¿Cuál sería la arquitectura de la red neuronal CNN (Convolutional Neural Network) que cumpliría con el grado de error aceptable?

1.5. Objetivos

1.5.1. General

Demostrar si la clasificación de leucocitos en imágenes microscópicas de frotis sanguíneo realizado con machine learning y CNN (Convolutional Neural Network) cumple un grado de error aceptable

1.5.2. Específicos

- Realizar una revisión bibliográfica para definir un grado de error aceptable.
- Definir el proceso de preprocesamiento de imágenes microscópicas para usarlos como datos de entrada en una red neuronal CNN (Convolutional Neural Network)
- Determinar la arquitectura de red neuronal CNN (Convolutional Neural Network) que cumple con el grado de error aceptable

1.6. Metodología

1.6.1. Tipo de investigación

En este trabajo de investigación tuvo como objetivo el uso de conocimientos previos para la validación de una parte de una solución a un problema. Por esta razón, el fin de la investigación es la creación de nuevo conocimiento (saber si el uso de machine learning y CNN son una solución óptima).

En esta investigación no se espera realizar la creación de software para un consumidor real, ni creación de prototipos funcionales u otras formas de aplicativos para implementación en producción.

Gracias a lo explicado, podemos determinar que el tipo de investigación de este proyecto es de una investigación básica. (Sampieri et al., 2014)

1.6.2. Nivel de investigación

En este proyecto de investigación se buscaba explicar como el machine learning y CNN puede ser una solución para el problema de la clasificación de leucocitos en imágenes microscópicas de frotis sanguíneo, por ende, se puede deducir que el nivel de investigación es el de correlación ya que se trata de observar el efecto de una variable en otra. También se podría considerar del tipo descriptiva, ya que se espera describir el proceso de demostrar si la solución encontrada es válida. (Martínez-Castro et al., 2014)

1.6.3. Método de investigación

Como se indicó con anterioridad, en este proyecto de investigación usaremos machine learning y CNN (Convolutional Neural Network) para la problemática, en este proceso uno de los objetivos que tendrá el investigador es el de encontrar la mejor arquitectura de CNN para cumplir con la tasa de error aceptable. Por esta razón se puede entender que el investigador será un participe activo en todo el proceso de la investigación. De esta forma podemos entender esta investigación será una investigación de acción. (Raschka, 2020)

1.7. Alcances y limitaciones

El proyecto de investigación buscaba un modelo de machine learning para la clasificación de leucocitos en imágenes microscópicas de un frotis de sangre; se tendrá como punto de inicio el poner la muestra bajo microscopio para la clasificación de leucocitos de manera manual hecha por un trabajador de laboratorio, en este punto se especificó como se tendría que realizar la toma de imágenes que servirán para que el algoritmo de machine learning pueda realizar el proceso y se consideró que el proceso termino cuando el proceso de como salida el porcentaje por clase de leucocito hallado en la muestra puesta bajo el microscopio. Se planteo diferentes arquitecturas de redes neuronales CNN (Convolutional Neural Network) con el objetivo de encontrar un grado de error aceptable al clasificar leucocitos, además de compararlo con el proceso manual. No se buscará crear un software o aplicativo el cual brindar a los clientes finales, ya que este software tendría sus propios objetivos y métricas dentro de una investigación que no toman importancia para el objetivo de la investigación actual.

Los limitantes válidos para esta investigación serán la falta de recursos económicos para la compra de equipos como la compra de microscopios para la toma de imágenes u otros aparatos. No se podrá realizar pruebas reales con muestras de pacientes de laboratorios ya que es un tema delicado, como lo es la salud de la población; pero se espera utilizar bases de datos de imágenes microscópicas previamente clasificadas para utilizarlos como ejemplos de entradas y salidas en el proceso de entrenamiento de una red neuronal.

1.8. Hipótesis

La clasificación de leucocitos en imágenes microscópicas de frotis sanguíneo realizado con machine learning y CNN (Convolutional Neural Network) si cumple con un grado de error aceptable

1.9. Variables e indicadores

1.9.1. Variables dependientes

- Clasificación de leucocitos en imágenes microscópicas

1.9.2. Indicadores de variables dependientes

Los indicadores del proceso de conteo y clasificación de leucocitos son:

- Grado de error en la clasificación de leucocitos
- Grado de error en la clasificación de leucocitos por tipo de leucocito

1.9.3. Variables independientes

- Aplicación de Machine learning y red neuronal CNN

1.9.4. Indicadores de variables independientes

Los indicadores del modelo usando machine learning del proceso serian:

- Numero de neuronas
- Numero de ciclos
- Numero de capaz
- Función de activación
- Tiempo de aprendizaje
- Resolución de imágenes microscópicas

1.10. Cuadro de operalización de variables

Tabla 1
Tabla de operalización de variables

Variables	Definición Conceptual	Dimensiones	Indicadores	Ítems
Clasificación de leucocitos en imágenes microscópicas	Nos referimos únicamente al proceso de clasificar un leucocito según sus características en los cinco tipos que tiene: Neutrófilos, Linfocitos, Monocitos, Eosinófilos y Basófilos. Normalmente este proceso debe de ser hecho por un técnico en laboratorio usando su conocimiento y experiencia	Nivel de eficiencia	Grado de error en la clasificación de leucocitos	Unidad de medida será el porcentaje. Numero de fallos entre número de células evaluadas multiplicado por 100
			Grado de error en la clasificación de leucocitos por tipo de leucocito	Unidad de medida será el porcentaje. Numero de fallos del tipo X entre número de células evaluadas del tipo X multiplicado por 100
Aplicación de Machine learning y red neuronal CNN	Machine learning es el uso de tecnologías para que una computadora pueda aprender a realizar una función con un entrenamiento previo, redes neuronales es	Eficacia	Numero de neuronas	Unidad de media es entero. Se obtuvo del detalle de experimentación
			Numero de ciclos	Unidad de media es entero.

la mejor forma de poder entender este proceso. Redes neuronales CNN son redes neuronales que una de sus capas ocultas en es una capa convolucional, esta capa convolucional permite encontrar patrones en imágenes de una forma más eficiente.			Se obtuvo del detalle de experimentación
		Numero de capaz	Unidad de media es entero. Se obtuvo del detalle de experimentación
		Función de activación	Unidad de media es tipo de Función de activación. Se obtuvo del detalle de experimentación
		Tiempo de aprendizaje	Unidad de media es segundos. Se obtuvo del detalle de experimentación
		Resolución de imágenes microscópicas	Unidad de media es pixeles por pixeles. Se obtuvo del detalle de experimentación
	Eficiencia	TPR	True Positive Rate – Porcentaje de aciertos positivos

				Se obtuvo del análisis “Cofusion matrix”
			TNR	True Negative Rate – Porcentaje de aciertos negativos. Se obtuvo del análisis “Cofusion matrix”
			Accuracy	Accuracy – Porcentaje de aciertos general. Se obtuvo del análisis “Cofusion matrix”

1.11. Matriz de consistencia

Tabla 2
Matriz de consistencia

Matriz de consistencia de la investigación Título: Clasificación de leucocitos en imágenes microscópicas de frotis sanguíneo usando machine learning y CNN				
Problemas General	Objetivos General	Hipótesis	Variables / Indicadores Variable independiente	Metodología
¿La clasificación de leucocitos en imágenes microscópicas de frotis sanguíneo realizado con machine learning y CNN puede cumplir un grado de error aceptable que lo valide como posible alternativa?	Demostrar si la clasificación de leucocitos en imágenes microscópicas de frotis sanguíneo realizado con machine learning y CNN cumple un grado de error aceptable	La clasificación de leucocitos en imágenes microscópicas de frotis sanguíneo realizado con machine learning y CNN si cumple con un grado de error aceptable	Clasificación de leucocitos en imágenes microscópicas • Grado de error en la clasificación de leucocitos • Grado de error en la clasificación de leucocitos por tipo de leucocito	Tipo de investigación - Investigación Básica Cuantitativa Nivel de investigación - Correlación - Descriptiva Método de investigación - Investigación de acción Metodología para el desarrollo de la investigación - Metodología KDD (Pereira et al., 2016)
Específicos	Específicos		Variable dependiente	
¿Cuál sería el grado de error aceptable que debería cumplir la alternativa de clasificación leucocitos?	Realizar una revisión bibliográfica para definir un grado de error aceptable.		Aplicación de Machine learning y red neuronal CNN • Numero de neuronas • Numero de ciclos • Numero de capaz • Función de activación • Tiempo de aprendizaje • Resolución de imágenes microscópicas • TPR, TNR, Accuracy	
¿Cómo se tendría que tratar las imágenes microscópicas de frotis sanguíneo para su uso en una red neuronal CNN?	Definir el proceso de preprocesamiento de imágenes microscópicas para usarlos como datos de entrada en una red neuronal CNN			
¿Cuál sería la arquitectura de la red neuronal CNN que cumpliría con el grado de error aceptable?	Determinar la arquitectura de red neuronal CNN que cumple con el grado de error aceptable			

Capítulo 2. Marco teórico

2.1. Antecedentes

2.1.1. Antecedentes internacionales

Human peripheral blood leukocyte classification method based on convolutional neural network and data augmentation (Wang & Cao, 2020)

Fecha de publicación: 22 de noviembre del 2019

Autores: Yapin Wang, and Yiping Cao

Resumen

El articulo habla del uso de redes neuronales convolucionales para la clasificación de leucocitos. Para empezar, en el mismo artículo se da a entender que la solución del uso de redes neuronales convolucionales para la clasificación de leucocitos ya fue probada en varias investigaciones con anterioridad. El objetivo de la investigación era el mejoramiento de los datasets mediante técnicas de “data augmentation”.

El objetivo de las técnicas de “data augmentation” es el de extender y mejorar los datasets, esto con la finalidad de que la red neuronal tenga más datos con los cuales trabajar en su entrenamiento.

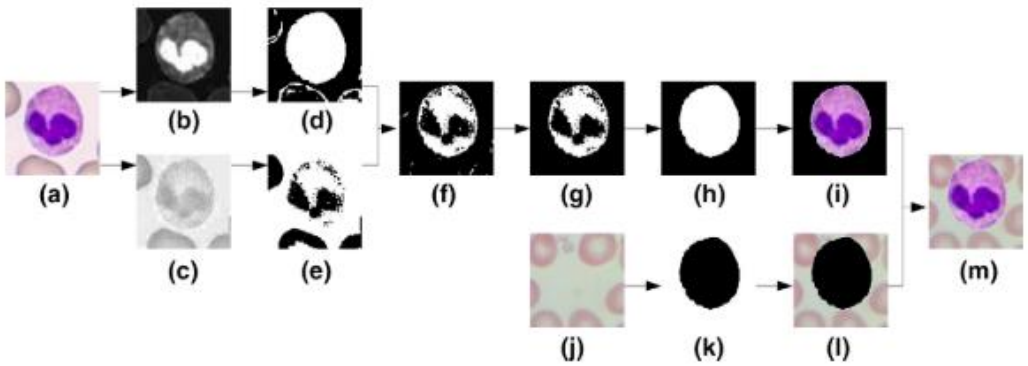
La técnica de “data augmentation” usada es el de generar nuevos datos de entrenamiento de las imágenes obtenías, usando filtros para poder realzar zonas de importancia para la finalidad del algoritmo. En la figura 1 se puede observar un ejemplo de cómo se puede crear nuevas imágenes donde se realza los gránulos de los leucocitos para facilitar a la red neuronal el encontrar cual es el objetivo que se espera del resultado.

Aporte según variables

En esta investigación se realiza un proceso de aumento de disminución de las variables como numero de neuronas por capa y aumento de capas ocultas. También se realiza un proceso de data augmentatio definido por los investigadores. Pero también se utilizan métodos de data augmentation

estándar. LA resolución utilizada para sus imágenes de entrada del modelo de red neuronal convolucional es de 217 x 217 pixeles.

Figura 1
Diagrama representativo de preprocesamiento



Nota: Extraído de (Wang & Cao, 2020)

White Blood Cell Classification and Counting Using Convolutional Neural Network (Macawile et al., 2018)

Fecha de publicación: 2018

Autores: Merl James Macawile, Vonn Vincent Quiñones, Alejandro Ballado Jr., Jennifer Dela Cruz, Meo Vincent Caya

Resumen

El artículo muestra la problemática de como el conteo selectivo de glóbulos blancos o leucocitos es información importante para la detección de enfermedades que afectan al sistema inmunológico como infecciones, anemia, leucemia, cáncer, AIDS (síndrome de inmunodeficiencia). Ellos hablan de proponer un método del uso de redes neuronales convolucionales para el selección y conteo.

Las redes neuronales convoluciones es un tipo de redes neuronales especializadas en el análisis de imágenes, no es un algoritmo de implementación única, este puede variar de cómo se utiliza. En el caso del

paper mencionan que esto lo realizan mediante implementación en 3 arquitecturas de CNN diferentes como lo son AlexNet, GoogleNet y ResNet-101.

Al final del paper muestran los resultados de modelos implementados en las 3 tecnologías y se puede observar que en cada una existen diferencias y que no hay una única solución para una implementación de esta solución.

Dentro de las conclusiones se menciona que el modelo que tuvo más porcentaje de aciertos fue la de GoogleNet con un 97.85%, por ende, la que tendría la tasa de fallos más baja sería el modelo de GoogleNet.

Aporte según variables

Como se puede ver en la tabla 4, en esta investigación realiza los cálculos de los valores TPR, TNR y accuracy al igual que están planteados en la investigación. Estos valores de esta investigación son 89.18%, 97.85% y 96.63% respectivamente.

Tabla 3 *TPR, TNR y accuracy mejores alcanzados*

Classification	Sensitivity (%)	Specificity (%)	Accuracy (%)
Basophil	100	99.43	99.44
Eosinophil	0	100	98.88
Lymphocyte	100	78.38	95.51
Monocyte	50	99.43	98.31
Neutrophil	75.86	99.33	95.51

Nota: Extraído de White Blood Cell Classification and Counting Using Convolutional Neural Network (Macawile et al., 2018)

Desarrollo de un sistema de análisis automático para la segmentación, clasificación y conteo de leucocitos en imágenes digitales de frotis de sangre periférica (Villegas Noguera, 2019)

Fecha de publicación: Julio, 2019

Autor: Ing. Angel L. Villegas N.

La investigación fue realizada para optar para el título de Magister en Ingeniería Eléctrica. El objetivo de la investigación es el de desarrollar un sistema de análisis automático para la segmentación, clasificación y conteo de leucocitos en imágenes digitales de frotis de sangre periférica como herramienta de apoyo al diagnóstico en el área hematológica. En la investigación uno de los objetivos específicos es el de analizar técnicas de segmentaciones imágenes para selección el algoritmo a implementar. Para poder lograr esto, el investigador busca hacer pruebas con los datos de un dataset que fue hecho por el profesor Fabio Scotti de la Universidad de Milán, Italia, este dataset tiene el nombre ALL-IDB (Acute Lymphoblastic Leukemia Image Database). Otros bancos de datos que se nos presenta son WBC Image Dataset (WBC-ID) y LISC: Leukocyte Images for Segmentation and Classification.

Esta investigación brinda un gran aporte al trabajo de investigación ya que nos brinda información de datasets con los cuales se pueden realizar pruebas de los algoritmos de machine learning y también guiarse para la creación de un nuevo dataset con los equipos que brinda la universidad.

Aporte según variables

> Diferencias máximas alcanzados o grados de error máximos alcanzados fueron de entre 2.84% y 3.08% a las clasificaciones realizadas por un experto.

Comprehensive Analysis of Deep Learning Methodology in Classification of Leukocytes and Enhancement Using Swish Activation Units (Harshanand & Sangaiah, 2020)

Fecha de publicación: 2020

Autores: Harshanand, B. A.; Sangaiah, Arun Kumar

Resumen

El principal aporte que realiza la investigación es el de comparar arquitecturas predefinidas con la arquitectura propuesta de su red neuronal CNN. Las arquitecturas predefinidas son Resnet50, VGG16, Exception, InceptionV3. Dentro de sus resultados se debería tener más importancia a los siguientes puntos:

- Las arquitecturas con normalización tipo Batch mostraron más consistencia que las que no lo utilizaban.
- El optimizador que se uso es el SGD (Standard Gradient Descent).
- La función de activación usado es RELU
- La estrategia de aprendizaje usada fue Full Training

Aporte según variables

Función de activación más utilizada en modelos de redes neuronales artificiales es ReLU.

2.1.2. Antecedentes nacionales

Diseño de un algoritmo para la automatización del conteo de células del tejido sanguíneo mediante procesamiento digital de imágenes (Bustamante Alvarez, 2019)

Fecha de publicación: 2019

Autor: Rafael Bustamante Alvarez

Resumen

En esta investigación se plantea una problemática muy similar a la que estamos planteando en este proyecto de investigación, la principal diferencia es que en esta tesis doctoral se busca la creación de un algoritmo para el procedimiento de conteo de células sanguíneas, esto es el conteo de leucocitos, eritrocitos y plaquetas.

La solución que da es el uso de filtros para la identificación de células en específico. Los filtros que se usan hacen que al convertir cada píxel en formato RGB se puede realizar una función numérica el cual solo admite un rango de colores y de esta forma solo identificar los eritrocitos como blanco y lo demás como negro asiendo que para la maquina sea más fácil realizar el conteo. En el caso de los leucocitos se aplica un filtro especial el cual tiñe su núcleo de un color específico para que la maquina pueda identificarlos como leucocito.

El principal aporte que esta investigación le da a mi proyecto es que respalda la importancia de poder crear una forma de automatizar el proceso de conteo y clasificación de leucocitos con una tasa de error aceptable.

Aporte según variables

El grado de error mínimo identificado en esta investigación es de 5%. Esto significa que el grado de error que la solución de esta investigación tiene que superar es del 5%. Este es un aporte a nuestras variables y a nuestro objetivo especifico 1. También en los resultados se define que la solución tiene un grado de error promedio de 2.55% en su conteo de células sanguíneas.

Estimación de área glaciar utilizando redes neuronales convolucionales u-net en imágenes multiespectrales sentinel 2 en el glaciar AUSANGATE, 2019 (Caillahua, 2019)

Fecha de publicación: 2019

Autor: Percy Elbis Colque Caillahua

Resumen

El objetivo de esta tesis de pregrado es probar la arquitectura U-Net en la estimación de área glaciar con imágenes multiespectrales Sentinel 2. U-Net es una arquitectura de una red neuronal convolucional (CNN). Esta arquitectura fue creada con el objetivo de analizar imágenes de origen biológico.

El apoyo a la investigación es mostrar una idea del todo el proceso de la creación de una red neuronal, obtención de datos, entrenamiento de la red neuronal y pruebas. Se tomará en cuente el proceso para el desarrollo de la investigación

Aporte según variables

Para el desarrollo de un modelo de red neuronal convolucional, en esta investigación mencionan que utilizan una arquitectura estándar. Esta arquitectura consta de 1000 neuronas y de 13 capas ocultas. El tiempo de entrenamiento fue de 11 horas aproximadamente. Estos datos nos ayudan a entender que la complejidad del modelo tiende a aumentar los tiempos de entrenamiento y sería bueno probar con modelos mas simple para poder ver si se alcanza nuestros objetivos iniciales.

Mapping vegetation, water bodies and urban areas in PeruSAT-1 satellite imagery (Ramirez et al., 2019)

Fecha de publicación: 2019

Autor: Alejandro Ramirez, Adison Pacheco y Joel Telles

Resumen

En el artículo se menciona que se usa las imágenes obtenidas con el satélite PeruSAT-1 el cual es dirigido por la agencia espacial CONIDA, el uso de estas imágenes es de propiedad nacional con la supervisión de la agencia CONIDA. El objetivo principal de la investigación es el crear un modelo capaz de realizar la clasificación de las áreas de vegetación, extensiones de agua (ríos, lagos, etc.) y áreas urbanas en imagen tomada por satélite, para esto se usa como datos de entrenamiento las imágenes extraídas del satélite PeruSAT-1.

Para este objetivo, se utiliza la siguiente metodología; primero, la obtención de los datos para el entrenamiento y pruebas, las imágenes obtenidas del satélite son de una resolución de 10000x10000 que se debió comprimir en imágenes de 1000x1000 pixeles para no exceder la capacidad computacional que ellos tenían, luego se tiene que crear la capa de salida de manera manual donde se indique como debería de ser el resultado del modelo para la etapa de entrenamiento; luego utilizan el algoritmo de linear support vector machine (SVN) para la creación del modelo.

En los resultados se puede observar que usando 15000 datos/imágenes (12000 de entrenamiento y 3000 para testeo) se logró un nivel de aceptación del 98.76%.

Aporte según variables

Las resoluciones utilizadas en esta investigación por imagen son de 1000 x 1000 pixeles. Estas resoluciones son altas por que la investigación busca clasificar todas las áreas que se encuentran en la imagen. En nuestro caso la resolución puede ser mucho mas baja porque las características que nosotros queremos que se aprenda la red neuronal solo se presentan en el leucocito y no en todo el plasma que le rodea.

2.2. Bases teóricas

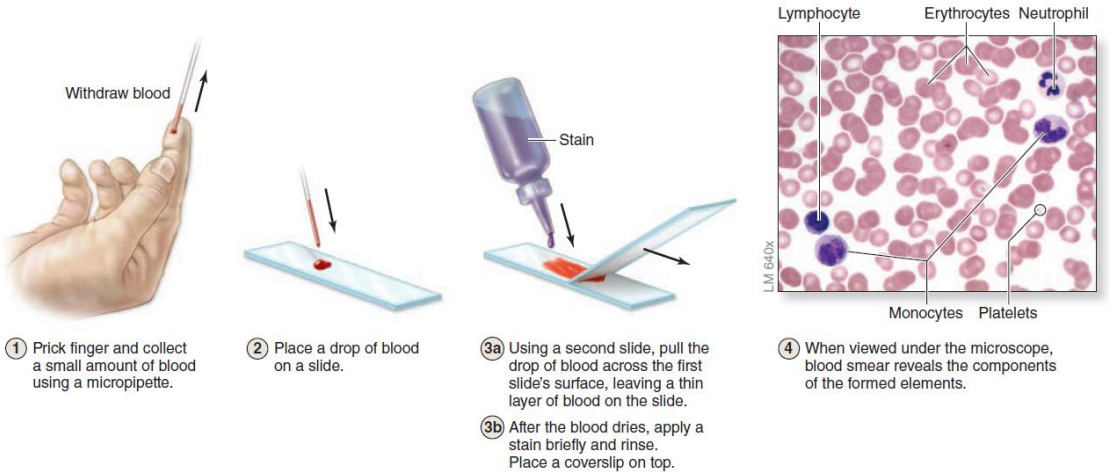
2.2.1. Hematología

2.2.1.1. Frotis o extendido de sangre

Un frotis es el procedimiento de poner gota de sangre bajo el microscopio. Este procedimiento consta de los siguientes pasos:

- Colocar una gota de sangre en una placa de muestra o portaobjetos de vidrio.
- Con otra placa se esparce la muestra de sangre para en el área más grande posible sin cortarlo.
- Posteriormente se debe de aplicar la tinción de Giemsa para preparar la muestra.
- Después ya se puede realizar su observación bajo un microscopio.

Figura 2
Pasos para la realización de un frotis



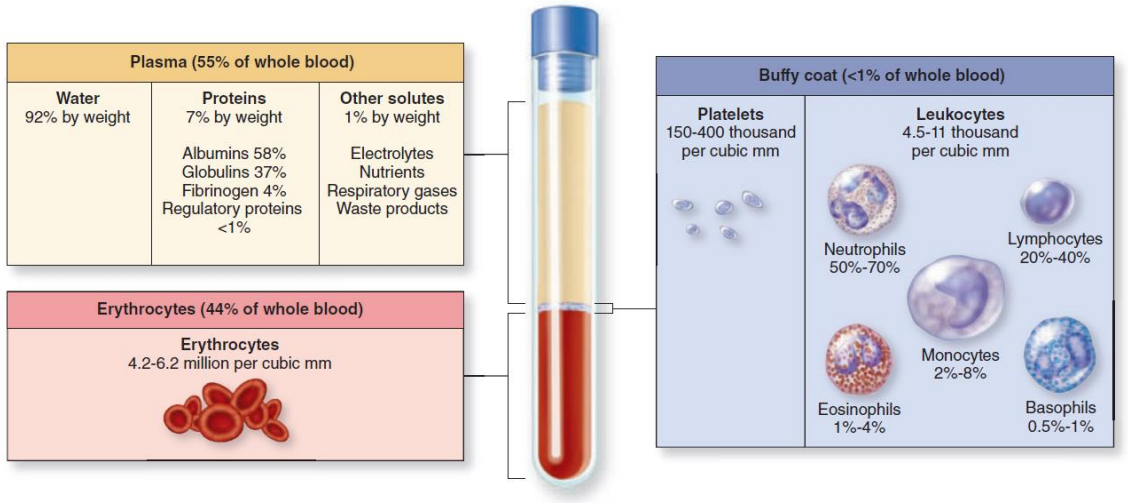
Nota: Extraído de Histología Texto Y Atlas Correlación con Biología Molecular y Celular (Pawlina, 2015)

2.2.1.2. Sangre

La sangre puede considerarse un tejido conectivo fluido, dado que está constituida por células y una "sustancia intercelular" líquida, el plasma sanguíneo. La sangre circula a través del organismo por los vasos sanguíneos.

En la sangre existen los comúnmente llamados “formed elements”, en español elementos formados, o por llamarlos de una forma más entendible serían los compuestos biológicos o células que se puede hallar en la sangre; los formed elements son los eritrocitos (glóbulos rojos), leucocitos (glóbulos blancos) y las plaquetas. La composición en general de la sangre seria de plasma un 55%, eritrocitos un 44% y la capa leucocitaria (compuesto por plaquetas y leucocitos) un 1%. La función principal de la sangre es el de la distribución de compuestos como O_2, CO_2, hormonas, metabolitos, etc. Y otras funciones de la sangre serían las de la autorregulación de la temperatura corporal y el mantenimiento de equilibrio ácido-base y osmótico. (Pawlina, 2015)

Figura 3
Composición de la sangre



Nota: Extraído de Histología Texto Y Atlas Correlación con Biología Molecular y Celular (Pawlina, 2015)

2.2.1.3. Composición de la sangre

- **Plasma**

Aunque el plasma es una gran parte de la sangre está compuesta principalmente por agua, un 90 % del plasma es agua y sirve para disolver las soluciones que se encuentra en el plasma como proteínas, electrolitos, etc. (L. Mescher, 2018). Las principales proteínas encontradas en el plasma son:

- Albúmina

  Se sintetiza en el hígado. Se encarga de ejercer el gradiente de concentración entre la sangre y el líquido tisular extracelular.

- Globulina

  Se comprende por inmunoglobulina (anticuerpos) y globulinas no inmunes las cuales se encargan de mantener la presión osmótica dentro del sistema vascular y también como proteína transportadora.

- Fibrinógeno

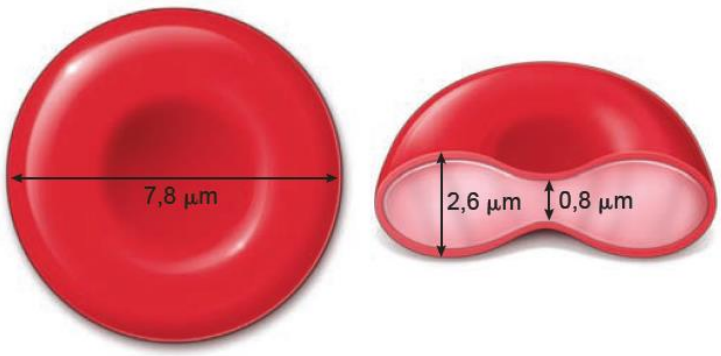
. **Células**

. **Glóbulos rojos o eritrocitos**

Son células con una forma bicóncava, su función principal es el de fijar oxígeno y liberarlos en los tejidos, y también el de rescatar dióxido de carbono de los tejidos para su eliminación del sistema. Su color se obtiene por el pigmento llamado hemoglobina. (L. Mescher, 2018)

En términos de funcionalidad, tiene 2 tipos de proteínas importantes las proteínas integrales de la membrana y las proteínas periféricas de la membrana.

Figura 4
Morfología de eritrocito



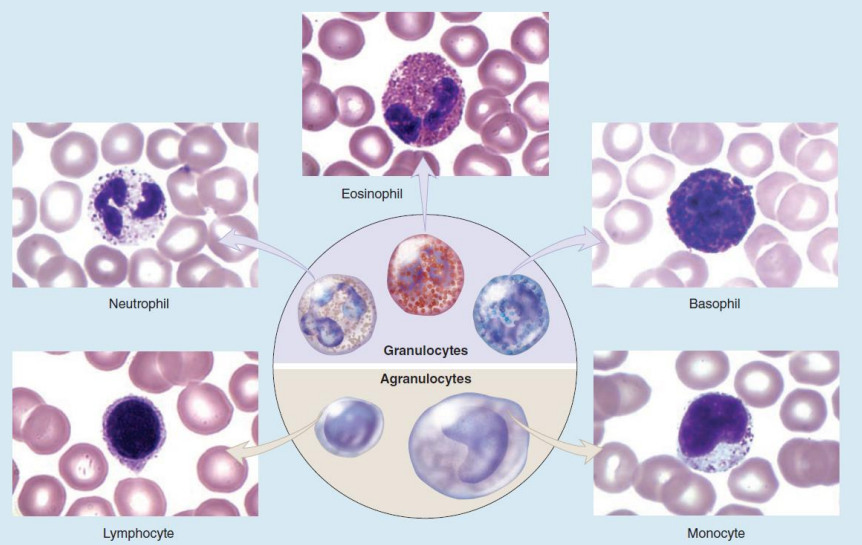
Nota: Extraído de Histología Texto Y Atlas Correlación con Biología Molecular y Celular (Pawlina, 2015)

. **Glóbulos blancos o leucocitos**

Los leucocitos tienen la tarea primordial de encargarse de las defensas contra microorganismos externos extraños en la sangre y ayuda en tejidos lesionados o infectados dejando la microvasculatura. El número de leucocito en la sangre baria según la edad, sexo y condiciones fisiológicas. De normal en el cuerpo de un adulto hay 4500 – 11000 leucocitos por microlitro de sangre.

Los leucocitos se dividen en dos grupos y esto es según la presencia o ausencia de gránulos específicos, estos grupos son los granulocitos y los agranulocitos.

Figura 5
Tipos de leucocitos



Nota: Extraído de Histología Texto Y Atlas Correlación con Biología Molecular y Celular (Pawlina, 2015)

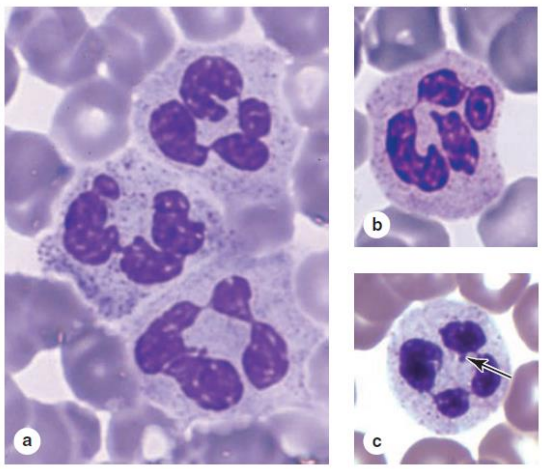
Granulocitos

Los leucocitos granulocitos se caracterizan por la presencia de gránulos específicos, estos granulocitos se forman cuando su único núcleo es segmentado en múltiples lóbulos que adoptan una morfología variable, por esta razón también se les llama leucocitos polimorfonucleares o polimorfos, esta clasificación se subdivide en 3:

- Neutrófilos

  Los neutrófilos es el tipo de leucocito y granulocito más común. Los neutrófilos más antiguos cuentan con 2 a 4 lóbulos unidos por hebras de tejido nuclear. Dentro de lo gránulos existen 3 tipos; gránulos azurófilos, gránulos específicos y gránulos terciarios. Su función es la de fagocitar (absorbe microorganismos y los desase en su interior) y eliminar microorganismos para combatir infecciones. (L. Mescher, 2018)

Figura 6
Neutrófilos en frotis

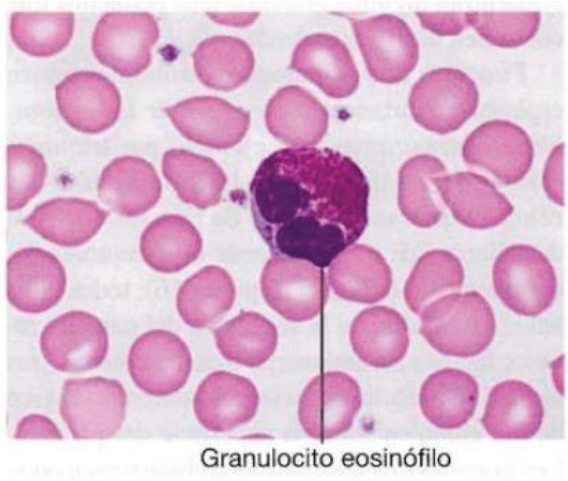


Nota: Extraído de Junqueira's basic histology (L. Mescher, 2018)

- Eosinófilos

  Consta del 1% al 4% de los leucocitos. El núcleo consta con dos lóbulos grandes unida por una fina hebra de cromatina. Su función principal es el combatir las infestaciones parasitarias. (L. Mescher, 2018)

Figura 7
Granulocito en frotis

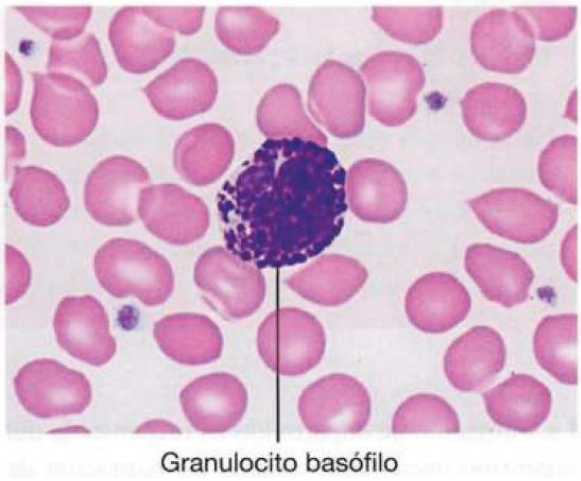


Nota: Extraído de Junqueira's basic histology (L. Mescher, 2018)

- Basófilos

  Se caracteriza por tener un núcleo con 2 o 3 lóbulos. Su cromatina tiene pequeños grumos que en el frotis se tiñen con menos intensidad que otros. Tiene unos gruesos gránulos citoplasmáticos los cuales hacen difícil la visualización del núcleo. No se tiene identificada la función principal de este tipo de leucocito, pero se cree que sirve como respuesta inmunitaria adaptativa. (L. Mescher, 2018)

Figura 8
Granulocitos en frotis



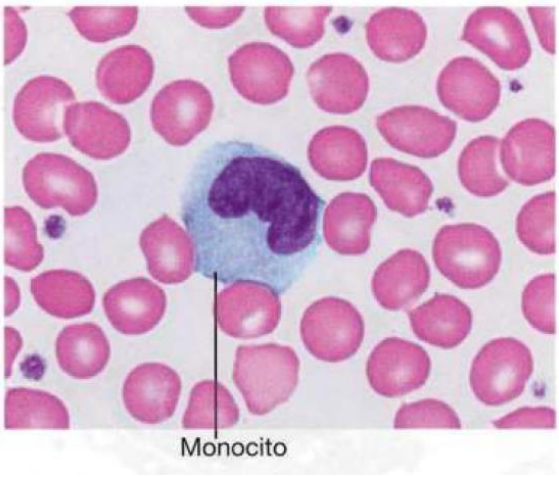
Nota: Extraído de Junqueira's basic histology (L. Mescher, 2018)

Agranulocitos

- Monocitos

  Posee un núcleo normalmente con forma arriñonada (forma de riñón), en el frotis, el abundante citoplasma es de color gris azulado y suele contener vacuolas y gránulos azurófilos dispersos. Cumple una función en sistema fagocito mononuclear junto a los macrófagos. (Brüel et al., 2015)

Figura 9
Monocito en frotis

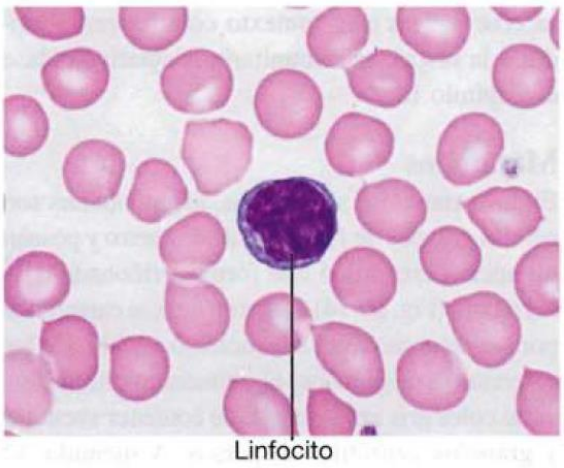


Nota: Extraído de Junqueira's basic histology (L. Mescher, 2018)

- Linfocitos

  Posee un núcleo redondeado con una pequeña entrante en un lado. El núcleo casi ocupa toda la célula y solo se ve un borde azul de citoplasma. Existe una clasificación de linfocitos, linfocitos T y linfocitos B. Se encargan de la defensa inmunológica del organismo. (Salin & Winston, 1992)

Figura 10
Linfocito en frotis



Nota: Extraído de Junqueira's basic histology (L. Mescher, 2018)

Plaquetas

Tienen una vida media de 10 días y cumplen una función principal en la hemostasia, proceso en el cual en caso de un corte o hemorragia las plaquetas se encargan de no permitir que la sangre se siga propagando, creando un tapón plaquetario o placa trombótica. (Salin & Winston, 1992)

Tinción de frotis de sangre

Los frotis sanguíneos se realizan empleando diversos colorantes y técnicas, que pueden agruparse en dos tipos de tinciones: de rutina (o tradicionales) y especiales.

Según e-histología, “La tinción del frotis sanguíneo pueden realizarse con colorantes de tipo Romanovsky que se componen de eosina (colorante ácido) y tiacinas (colorantes básicos); las tiacinas constituyen el azul de metileno y aquellos derivados obtenidos

mediante desmetilación oxidativa (colorantes azules). Las tinciones tipo Romanovsky son las más usadas en hematología.” (A. Villena et al., n.d.)

Los métodos de tinción más utilizados en frotis sanguíneos son:

- Tinción de Wright

  Es la empleada con mayor frecuencia en frotis de sangre periférica o de médula ósea. Está clasificada como tinción policromática, ya que tiñe compuestos básicos y ácidos de las células. La eosina es ácida y tiñe a los componentes básicos de color rosado a rojo (como los gránulos eosinófilos). El metileno es básico y tiñe a los componentes básicos de color azul (como el RNA).
- Tinción de Giemsa

  Es la segunda tinción más utilizada, después de la de Wright en frotis sanguíneos. El colorante empleado se compone de la mezcla en tampón fosfato o en agua de azul de metileno, que tiñe componentes ácidos, eosina, que tiñe los componentes básicos, y distintos azures, que en función del pH del medio y del grado de basófila de las estructuras pueden producir coloraciones metacromáticas.
- Tinción panóptico rápido

  Se caracteriza por producir una coloración rápida y sencilla de las estructuras celulares. Reúne la calidad y policromía de los métodos clásicos de Wright y Giemsa con gran rapidez de actuación. Tiene la característica de tratarse de un método de inmersión en el cual se sumerge el frotis durante un tiempo establecido en una solución colorante. Se producen coloraciones en las células sanguíneas similares a las de las tinciones de Wright y de Giemsa, aunque si se modifica el número de inmersiones se puede alterar la intensidad de los rosas y azules.

2.2.2. Machine learning

Según Sebastian Raschka (2019), machine learning es el uso de algoritmos de autoaprendizaje y data seleccionada (datos de entrada y salida preprocesado) para su preparación (entrenamiento) para que, según las características extraídas de los datos de prueba, pueda generar un pronóstico de datos de entrada nuevos que se le presenten.

Además, según Zsolt Nagy (2018), “Machine Learning es un campo de estudio relacionado con dar a computadoras la capacidad de aprender sin ser programado explícitamente. Se dice que un programa de computadora aprende de la experiencia, E, con respeto a una tarea, T, y una medida de rendimiento, P, si su rendimiento en T, según lo medido por P, mejora con la experiencia E.”

De estas dos definiciones podemos concluir que Machine Learning es cuando una persona (desarrollador) crea un algoritmo, el cual tiene parámetros, pero no reglas concretas, que tiene como objetivo de aprender (encontrar relaciones, referencias etc.) para poder realizar pronósticos de nuevos datos de entrada.

Esta definición de machine learning es simple y fácil de entender, pero le falta algunas explicaciones de profundidad. Primero, los algoritmos de autoaprendizaje son modelos y estos algoritmo o modelos son muy diferentes unos de otros ya que todos los algoritmos son creados con una finalidad diferente. Según el objetivo de cada modelo, los datos de entrada cambian, por ende, el modelo debe de estar preparado para el análisis de datos en un formato especifico. Después, se tiene las capas de análisis, estos pueden variar el aprendizaje de muchas maneras, un ejemplo de esto es que cuando se quiere analizar una imagen, esta no se analiza de una manera en específico (eso no se entendería por aprender) cuanto más complejo sea el modelo los datos de entrada son pasados por diferentes filtros, los cuales no están delimitados por el desarrollador, sino que el mismo algoritmo varia para luego evaluar si el resultado de salida se acerca a los datos de salida esperados, una vez que este ciclo se termine se vuelve a iniciar variando los filtros a utilizarse.

2.2.2.1. Técnicas de machine learning según los tipos de aprendizaje

Dentro de machine learning se puede encontrar 3 líneas de aprendizaje por las cuales se separan los algoritmos, estos son aprendizaje supervisado, aprendizaje no supervisado y aprendizaje por reforzamiento.

- **Aprendizaje supervisado**

  Se dice que la técnica de aprendizaje es supervisada cuando se sabe que se quiere adquirir de la información de entrada. Esto significa que, una vez

analizada cualquier dato nuevo, esta va a ser clasificado en un formato de salida esperado, como lo puedes ser las etiquetas.

Un ejemplo de esto sería el dar como datos de entra una imagen donde se pueda visualizar si hay gatos o perros, y para el entrenamiento solo le das 2 posibles datos de salida, que hay un gato o hay un perro. El algoritmo trabaja para identificar, según sus datos de entrada, cuando nosotros consideramos que el animal en la imagen es un gato o es un perro.

- **Aprendizaje no supervisado**

Se dice que la técnica de aprendizaje es no supervisada cuando solo se tiene los datos de entrada y no los de salida para el entrenamiento. En esta técnica de aprendizaje, nosotros no sabemos de antemano cuales son las relaciones que el algoritmo tiene que encontrar, sino que esta tratara de identificar todas las posibles relaciones que vea en los datos y los considerara como posibles datos de salida.

Un ejemplo de esto es el dar a un algoritmo no supervisado una gran cantidad de datos de compradores de un sitio web (cosas que compraron, ubicación, edad, etc.) y hacer que este identifique posibles relaciones entre los datos. El algoritmo dará como salida todas las posibles relaciones que encuentre, pero esto no significa que esa relación sea real, para validar esto se tendría que usar una nueva data de entrada y ver si se encuentra la misma relación o no.

- **Aprendizaje por reforzamiento**

Se dice que la técnica de aprendizaje es por reforzamiento cuando las iteraciones se dan cuando ocurre un feedback o realimentación desde el ambiente donde se desenvuelve.

Al igual que el aprendizaje no supervisado, no se le da o pide una salida concreta a hallar; y, además, a diferencias de los otros 2 tipos de aprendizaje, el tiempo de entrenamiento es constante ya que se espera que haya un método por el cual el algoritmo puede ser guiado con feedback y así perfilar los datos de salida esperados con cada nueva iteración.

2.2.2.2. Técnicas de machine Learning según el tiempo

Esta forma de separar a los algoritmos de machine learning se basa específicamente en el tiempo implementado para el entrenamiento de los algoritmos, estos se dividen en 2:

- **Aprendizaje estático**

  Se considera que un algoritmo de machine learning es estático si solo es necesario un entrenamiento en todo su tiempo de despliegue. En el caso de estos algoritmos siempre se pueden observar una clara desventaja, el cual es que al no tener un proceso de feedback no se pueden adaptar al presentarle nuevos datos que no esté relacionados con los que vio en su entrenamiento.

- **Aprendizaje dinámico**

  Se considera que un algoritmo de machine learning es dinámico cuando los datos que este usa para entrenar son los mismos que analiza durante su pedido de despliegue, esto no le permite al algoritmo solo entrenar una vez, sino que es necesario que este en un constante entrenamiento y en muchos casos es posible hasta que el modelo del ML cambie también.

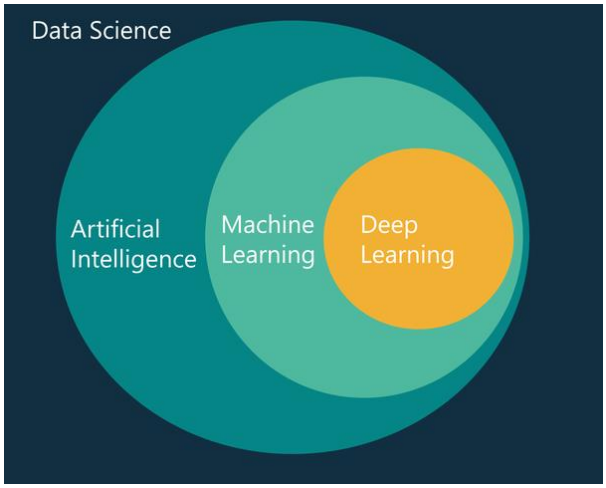
Uno de los objetivos de esta investigación es el escoger un técnica o estrategia de machine learning con la cual poder realizar procesamiento de imágenes, para esto se tomará en cuenta 2 técnicas, esta son el uso de redes neuronales convolucionales (el cual pertenece al tema de Deep learning) y el de K-means.

2.2.2.3. Deep learning

Es una forma más compleja de machine learning; en machine learning se busca que la máquina aprenda pero esto no especifica una forma de hacerlo, por ejemplo se puede hacer que una computadora aprenda solo identificando un patrón de dibujo usando como input de entrenamiento una imagen de baja resolución predefinida (como la imagen de números en casillas de 3x3), y cuando se despliegue este software solo bajar a la resolución de los inputs y lo compara con esta para ver si encuentra alguna imagen que sea igual, esto

funciona para resolver algunos problemas y se puede considerar que la computadora aprendió pero esto no significa que se considere deep learning. Deep learning busca el aprendizaje mediante el uso de redes neuronales, las redes neuronales fueron ideadas con la intención de simular la actividad de una red de neuronas del cerebro humano, más adelante se especificará sobre ese tema. Deep learning al usar redes neuronales se considera que es una forma de realizar aprendizaje profundo, esto significa que no solo es la replicación de datos que se vieron con anterioridad, sino que también se busca el encontrar una relación entre los datos que se dieron previamente y así tratar de realizar una predicción. (Raschka & Mirjalili, 2019)

Figura 11
Grafico representativo del alcance de deep learning



Nota: Extraído de Python Machine Learning (Raschka & Mirjalili, 2019)

. **Redes neuronales**

Las redes neuronales, como se indicó antes, fueron ideadas con la intención de simular la actividad de una red neuronal del cerebro humano. Esto significa que cuando se vio cómo trabaja una neurona esta recibe una entrada y luego de procesarla envía una salida a una o varias neuronas de la siguiente capa. Este procedimiento se realiza hasta que se tenga una salida que satisfaga la necesidad del problema. Una red neuronal trabaja en capas, estas capas se dividen en capa de entrada, capas ocultas y capa de salida. Una red neuronal va variando su forma de predecir los datos después de cada iteración del entrenamiento

. **Redes neuronales convolucionales**

Según (Raschka & Mirjalili, 2019), las redes neuronales convolucionales son redes neuronales que tienen un operación o proceso convolucional. Una convolución, en matemáticas, es la operación entre una función con otra para la obtención de una función con valor real, que tenga valor real significa que la función tiene una base real probada ósea que con anterioridad se haya probado que es cierto mediante algún método como la experimentación. Este concepto es utilizado para la paralización de datos de imágenes tratándolas como matrices. Para entender esto de mejor manera se mostrará un ejemplo del procedimiento de una convolución en una imagen (la imagen tendría que ser reducida a un valor por píxel y puesta en una matriz para representarlo):

- Como primer paso, se tiene que escoger una matriz con el que se va a operar la imagen (matriz-filtro).

Figura 12
Ejemplo canalización 2D - matriz filtro e imagen

1	1	1
0	0	0
-1	-1	-1

143	143	143	143	143	143	143	143	143	143
143	143	143	143	143	143	143	143	143	143
143	143	143	143	143	143	143	143	143	143
143	143	143	143	143	143	143	143	143	78
143	143	143	143	143	143	143	143	78	78
143	143	143	143	143	143	143	78	78	78
143	143	143	78	78	78	78	78	78	78
143	143	78	78	78	78	78	78	78	78
143	78	78	78	78	78	78	78	78	78
143	78	78	78	78	78	78	78	78	78
143	78	78	78	78	78	78	78	78	78

- Luego se tendrá que definir la operación que se realizará, en este caso se tendrá en desplazamiento de la matriz-filtro sobre la matriz-imagen de derecha a izquierda y de arriba abajo, siempre encajando en la matriz-imagen. Primero, como en los bordes no se puede hacer encajar la matriz-filtro, se pone un valor predefinido.

Figura 13
Ejemplo canalización 2D - Bordes de imagen

1	1	1
0	0	0
-1	-1	-1

143	143	143	143	143	143	143	143	143	143
143	143	143	143	143	143	143	143	143	143
143	143	143	143	143	143	143	143	143	143
143	143	143	143	143	143	143	143	143	78
143	143	143	143	143	143	143	143	78	78
143	143	143	143	143	143	143	78	78	78
143	143	143	78	78	78	78	78	78	78
143	143	78	78	78	78	78	78	78	78
143	78	78	78	78	78	78	78	78	78
143	78	78	78	78	78	78	78	78	78
143	78	78	78	78	78	78	78	78	78

255	255	255	255	255	255	255	255	255	255
255									255
255									255
255									255
255									255
255									255
255									255
255									255
255									255
255									255
255	255	255	255	255	255	255	255	255	255

- Ahora la operación a utilizar es la multiplicación de los valores y sumarlos. En el ejemplo seria:

$$143 * 1 + 143 * 1 + 143 * 1 + 143 * 0 + 143 * 0 + 143 + 0 + 143 * -1 + 143 * -1 + 143 * -1$$

Figura 14
Ejemplo canalización 2D - primera operación

1	1	1
0	0	0
-1	-1	-1

143	143	143	143	143	143	143	143	143	143
143	143	143	143	143	143	143	143	143	143
143	143	143	143	143	143	143	143	143	143
143	143	143	143	143	143	143	143	143	78
143	143	143	143	143	143	143	143	78	78
143	143	143	143	143	143	143	78	78	78
143	143	143	78	78	78	78	78	78	78
143	143	78	78	78	78	78	78	78	78
143	78	78	78	78	78	78	78	78	78
143	78	78	78	78	78	78	78	78	78
143	78	78	78	78	78	78	78	78	78

255	255	255	255	255	255	255	255	255	255
255	0								255
255									255
255									255
255									255
255									255
255									255
255									255
255									255
255									255
255	255	255	255	255	255	255	255	255	255

Figura 15
Ejemplo canalización 2D - avance de matriz y operación

1	1	1
0	0	0
-1	-1	-1

143	143	143	143	143	143	143	143	143	143
143	143	143	143	143	143	143	143	143	143
143	143	143	143	143	143	143	143	143	143
143	143	143	143	143	143	143	143	143	78
143	143	143	143	143	143	143	143	78	78
143	143	143	143	143	143	143	78	78	78
143	143	143	78	78	78	78	78	78	78
143	143	78	78	78	78	78	78	78	78
143	78	78	78	78	78	78	78	78	78
143	78	78	78	78	78	78	78	78	78
143	78	78	78	78	78	78	78	78	78

255	255	255	255	255	255	255	255	255	255
255	0	0	0	0	0	0	0	0	255
255	0	0	0	0	0	0	0	65	255
255	0	0	0	0	0	0	65	130	255
255	0	0	0	0	0	65	130	130	255
255	0	65	130						255
255									255
255									255
255									255
255									255
255	255	255	255	255	255	255	255	255	255

- Al finalizar se tendra otra matriz que puede ser reconvertia a una imagen y, en este caso gracias a la matriz-filtro y el modo de operarlo, se podra ver los bordes de la imagen.

Figura 16
Ejemplo canalización 2D - matriz de imagen operada

1	1	1
0	0	0
-1	-1	-1

143	143	143	143	143	143	143	143	143	143
143	143	143	143	143	143	143	143	143	143
143	143	143	143	143	143	143	143	143	143
143	143	143	143	143	143	143	143	143	78
143	143	143	143	143	143	143	143	78	78
143	143	143	143	143	143	143	78	78	78
143	143	143	78	78	78	78	78	78	78
143	143	78	78	78	78	78	78	78	78
143	78	78	78	78	78	78	78	78	78
143	78	78	78	78	78	78	78	78	78
143	78	78	78	78	78	78	78	78	78

255	255	255	255	255	255	255	255	255	255
255	0	0	0	0	0	0	0	0	255
255	0	0	0	0	0	0	0	65	255
255	0	0	0	0	0	0	65	130	255
255	0	0	0	0	0	65	130	130	255
255	0	65	130	195	195	195	130	65	255
255	65	130	195	195	195	130	65	0	255
255	130	130	65	0	0	0	0	0	255
255	65	65	0	0	0	0	0	0	255
255	0	0	0	0	0	0	0	0	255
255	255	255	255	255	255	255	255	255	255

2.2.2.4. K-means

K-means o k-medias es una técnica de machine learning en el cual es un algoritmo no supervisado, esto significa que los datos para entrenar no constan de salidas predefinidas o etiquetas. K-means es un algoritmo que busca agrupar los datos que se les da en k grupos mediante la identificación de medias de cada dato (esta media puede ser hallada de diferentes maneras según el tipo de datos que se quiere evaluar; por ejemplo, en “palabras” puede ser el

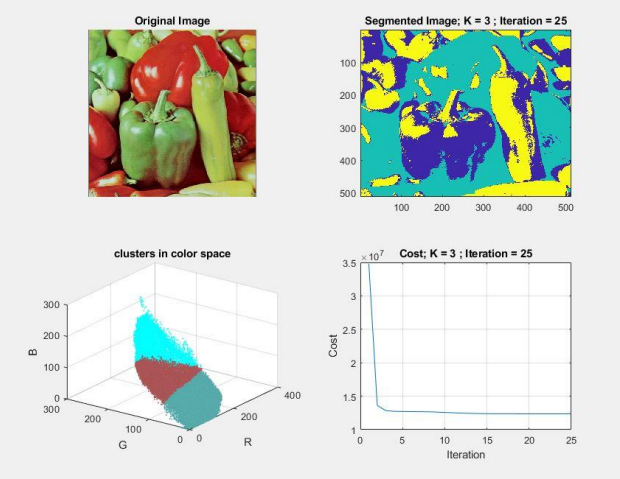
conteo de “palabras” especificas o en un grupo de datos buscar realizar una operación para poder hallar un resultado que un conjunto de datos comparta). Esta técnica permite el procesamiento de imágenes.

Para el procesamiento de imágenes con k-means es necesario la interacción de los siguientes pasos:

- Se deben de escoger el número de grupos que se quiere separar los datos K y darle datos que pertenezcan a esos grupos con alguna estrategia (una estrategia valida seria de manera aleatoria).
- En cada grupo de imágenes se definen los centroides los cuales tomaran algún dato de un sector de imagen. La estructura de los centroides es la misma en todos los grupos.
- Al analizar los datos de los centroides en cada grupo se va actualizando la estructura de estos para poder definir alguna similitud entre los datos de los grupos.

Cada interacción se ve desarrollada por el segundo y tercer paso.

Figura 17
Ejemplo de k-means



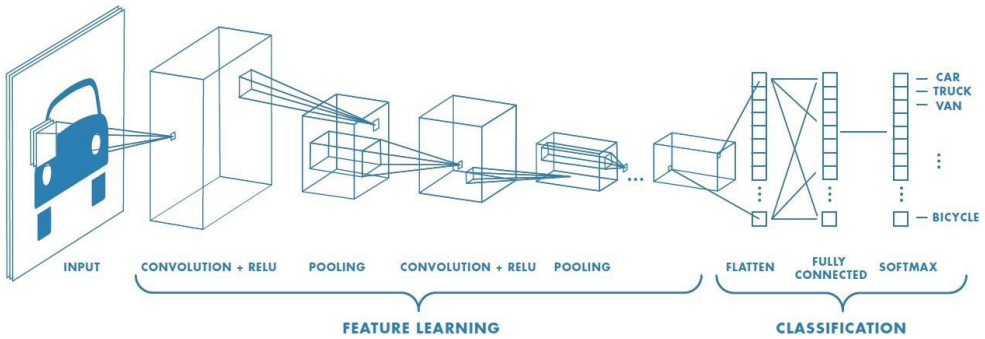
Nota: Extraído de Why is Nearest Neighbor a Lazy Algorithm? (Raschka, 2020)

2.2.2.5. Redes Neuronales Convolucionales

Según IBM cloud education (2020) :

> "Las redes neuronales convolucionales se distinguen de otras redes neuronales por su rendimiento superior con entradas de señales de imagen, habla o audio. Tienen tres tipos principales de capas, que son: capas convolucionales, capas pooling y capas Fullt-connected."(IBM, 2020)

Figura 18 *Estructura estándar de redes neuronales convolucionales*



Nota: Extraído de A Comprehensive Guide to Convolutional Neural Networks (Saha, 2018)

Las redes neuronales convolucionales se caracterizan por tener dos parte: La primera que tiene el objetivo de encontrar las características en las imágenes y la segunda de evaluar los datos de las características previamente para definir patrones o poder realizar la clasificación correcta.

La primera capa de obtención tiene dos capas importantes: capas convolucionales y capas pooling.

Capas convolucionales

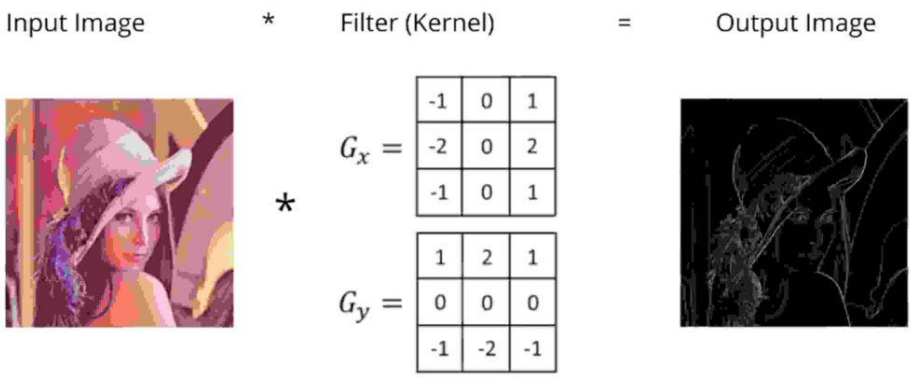
Las capas convoluciones se encargan de aplicar convoluciones a la imagen con el objetivo de crear mapas de características que se interpretarían como una nueva imagen.

Las convoluciones son la aplicación de filtros que permitan obtener información de un área más amplio de una imagen, en vez de solo analizar pixel por pixel. Este proceso se realiza de esta forma porque, en imágenes complejas, las características normalmente dependen de conjuntos de

pixeles y no de pixeles únicos. Un conjunto de pixeles no puede dar más información que cada uno de manera independiente.

Un ejemplo de convolución se puede ver en la figura 19. En este ejemplo se ve que aplicando dos filtros diferentes a la misma imagen y luego uniendo los resultados, podemos obtener un mapa de características el cual serian todos los bordes de la imagen. Esto se logra ya que al aplicar el filtro Gx, obtenemos los bordes verticales; y con el filtro Gy, obtenemos los bordes horizontales.

Figura 19 *Ejemplo de convolución*



Capas Pooling

Las capas de pooling se encargan de reducir los mapas de características su resolución con el objetivo de comprimir el aprendizaje. Esta compresión del aprendizaje se hace siempre entre dos capas convolucionales. Así las entradas de una capa convoluciones siempre son los mapas de características reducidos de resolución a los cuales aplicar nuevos filtros para crear muchos más mapas de características.

El proceso contante de crear mapas de características e ir comprimiendo permite que después de varios ciclos poder tener mapas de características de resolución tan baja que contengan información relevante extraída por las convoluciones.

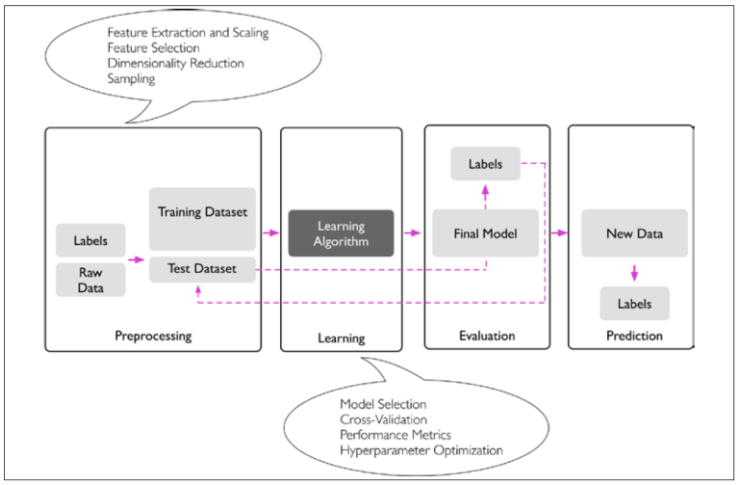
Una vez que se tienen mapas de características con información relevante se pasa a la parte de clasificación.

Primero, se utiliza una capa Flaten para que todos los mapas de características se vuelvan un array simple de datos. Después, este array pasa a ser los datos de entra de una red de capas Full-connected las cuales evalúan todas la información de los mapas de características para poder determinar qué características son relevantes para tomar la decisión de clasificar.

2.2.2.6. Proceso de desarrollo de una aplicación de machine learning

Según (Raschka & Mirjalili, 2019); para la creación de cualquier sistema, aplicación o algoritmo de machine learning es importante considerar los siguientes pasos:

Figura 20
Proceso de creación de aplicación de machine learning



Nota: Extraído de Python Machine Learning (Raschka & Mirjalili, 2019)

- Primero, el preprocesamiento, en esta etapa se trata de adaptar los datos que se tienen para que la aplicación pueda trabajar con ellos. Rara vez se tiene los datos de una manera perfecta tal que una computadora los pueda entender. Algunas formas de preprocesar datos es el de mantener un estándar para los datos; un ejemplo sería los perfiles de personas, según el objetivo de la aplicación algunos datos serán relevantes y otros no y estos deben de ser descartado para que la maquina no tenga problemas al encontrase con datos innecesarios.
- Según, la selección de modelo y entrenamiento, como se pudo observar con anterioridad; la creación de un modelo no solo busca resolver el

problema, sino que se busca que tenga el mejor desempeño a la hora de cumplir su objetivo. Un modelo puede ser el más exacto, pero solo si su entrenamiento demanda excesivo poder de cómputo el cual puede ser un problema en algunos casos. O por otro lado ser tan simple que nunca llegara a satisfacer su objetivo. Siempre se busca entrenar un modelo el cual muestre resultados en un tiempo que se considere justo.

- Evaluación y predicción, en este momento ya se tiene un modelo entrenado que en teoría debería de ser capaz de cumplir con el objetivo planteado, en esta fase se busca dar nuevos datos y asegurarse que de algunos resultados esperados para poder garantizar su fiabilidad y poder ser utilizado sin ninguna supervisión humana en un futuro.

2.2.2.7. Librerías para machine learning

. **Tensorflow**

Tensorflow es una biblioteca de código abierto desarrollada por Google. Su objetivo principal es el desarrollo de software para machine learning, específicamente para las creación y entrenamiento de redes neuronales. Esta biblioteca en un principio fue desarrollada para el lenguaje de programación Pyhon, pero con el tiempo fue creándose bibliotecas para otros lenguajes como Ruby, JavaScript y C++.

. **Pytorch**

Al igual que Tensorflow, es una biblioteca de código abierto desarrollado en un principio por Facebook y alterado para su comunidad. Está presente en los lenguajes de programación Python, c++ y CUDA.

2.2.2.8. Overfitting

Según Jason Brownlee:

> "El overfitting ocurre cuando un modelo aprende los detalles y el ruido en los datos de entrenamiento en la medida en que afecta negativamente el rendimiento del modelo en nuevos datos. Esto significa que el ruido o las fluctuaciones aleatorias en los datos de entrenamiento se recogen y se aprenden como conceptos por el modelo. El problema es que estos conceptos no se aplican a nuevos datos y afectan negativamente las habilidades de los modelos para generalizar." (Jason Brownlee, 2019)

Overfitting hace referencia a un modelo que a sido sobreentrenado. Esto significa que el modelo a logrado encontrar relaciones tan únicas de los datos de entrenamiento que cuando se le de nueva data, por no tener estas características tan únicas lo resolverá de la manera que el investigador no esperaba.

2.2.2.9. Underfitting

Según Jason Brownlee:

> "Underfitting se refiere a un modelo que no puede modelar los datos de entrenamiento ni generalizar a nuevos datos.
>
> Un modelo de aprendizaje de máquina con underfitting no es un modelo adecuado y será obvio, ya que tendrá un desempeño deficiente en los datos de capacitación."(Jason Brownlee, 2019)

Underfitting hace referencia a un modelo el cual no pudo llegar a un nivel de aprendizaje necesario para poder ser útil al fin en el que se piensa utilizar. Esto se puede dar por muchos motivos. El esquema del modelo es mu simple y no permite definir relaciones de la data de entrenamiento, la data de entrenamiento es muy pobre y por lo tanto no se puede definir relaciones, etc.

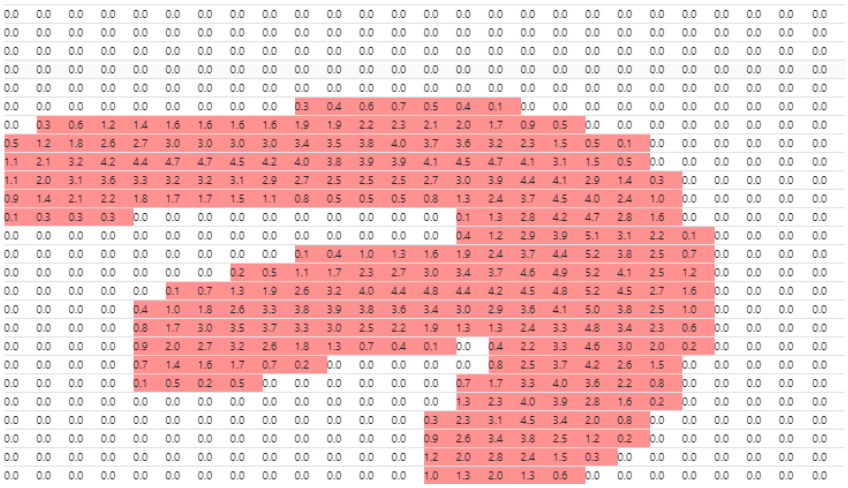
2.2.2.10. Capas

Capas son el conjunto de neuronas a las cuales se les dará una misma data de entrada y de las cuales se enviara datos de salida a las neuronas de las siguiente capa. Algunos tipos de capas son:

- **MaxPooling2D**

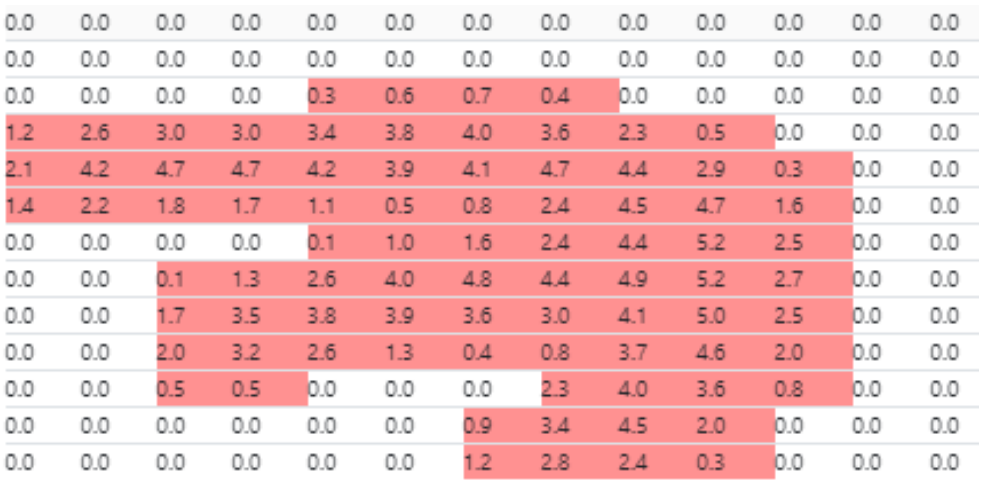
Capa oculta tipo MaxPooling2DLa función de esta capa oculta es la de realizar una operación matemática para reducir el número total de pixeles de la imagen de entrada. Un ejemplo de esto sería:

Figura 21
MaxPolling2D 26x26 ejemplo 1



Nota: Extraído de Max Pooling in Convolutional Neural Networks explained – deeplizard (DEEPLIZARD, 2018)

Figura 22
MaxPolling2D 13x13 ejemplo 2



Nota: Extraído de Max Pooling in Convolutional Neural Networks explained – deeplizard (DEEPLIZARD, 2018)

2.2.2.11. Precisión y perdida de entrenamiento

Los valores que se pueden captar en el proceso de entrenamiento de la red neuronal son: accuracy (precisión), loss (perdida) , val_acurracy (precisión de validación), val_loss (perdida en validación).

2.2.2.12. Cálculo de valores de evaluación de modelos de machine learning

Según (Raschka & Mirjalili, 2019) , la forma correcta de realizar una medición de desempeño de un modelo es con los valores de True Positive Rate (TPR), False Positive rate (FPR) y Accuracy.

> "En el diagnóstico de tumores, por ejemplo, nos preocupa más la detección de tumores malignos para ayudar a un paciente con el tratamiento adecuado. Sin embargo, también es importante reducir el número de tumores benignos clasificados incorrectamente como malignos (FP) para no preocupar innecesariamente a los pacientes. A diferencia del FPR, el TPR proporciona información útil sobre la fracción de ejemplos positivos (o relevantes) que se identificaron correctamente del conjunto total de positivos (P)."(Raschka & Mirjalili, 2019)

Las fórmulas de los valores de True Positive Rate (TPR), False Positive rate (FPR) y Accuracy son:

$$TPR = \frac{TP}{TP + FN}$$

$$TNR = \frac{TN}{TN + FP}$$

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

2.2.3. Análisis bibliográfico de Grado de error

Primero, porque necesitamos un grado de error mínimo. Todos los métodos de conteo diferenciado de leucocitos tienen un grado de error, este es utilizado para poder saber que tan equivocado puede estar el resultado encontrado con lo real.

Esta investigación tiene como objetivo cumplir un grado de error mínimo con el fin de poder demostrar que el uso de la tecnológica de redes neuronales convolucionales se puede utilizar como alternativa a los métodos manuales y automatizados por maquinaria especializada para la clasificación de leucocitos.

Como se describió anteriormente, se tiene que comparar la solución propuesta en esta investigación con el método manual y automatizado por maquinaria especializada para la clasificación de leucocitos. Como primer paso, debemos de conocer el grado de error de ambas soluciones para posteriormente definir un umbral que debemos de superar.

2.2.3.1. Grado de error del método manual

Para poder determinar el grado de error en el método manual se revisaron guías de laboratorio clínicos donde se indican grado de error al momento de contar y clasificar leucocitos.

Según la “GUÍA DE LABORATORIO DE HEMATOLOGÍA” de la Universidad Veracruzana. “Las cuentas visuales son relativamente inexactas, hasta con 20% de error, pero los contadores electrónicos por lo general son precisos dentro de 1 a 2%.” (Dr. Eduardo Rivadeneyra Domínguez, n.d.)

Según el “Manual de Laboratorio de Hematología” de la Universidad Nacional Autónoma De México. “El recuento manual tiene un error del 10%, el cual puede deberse a múltiples causas”(PATRICIA VIDAL MILLÁN & PABLO JUÁREZ DE LOS SANTOS, 2020)

Revisando guías y manuales de laboratorio del Perú, el método que en la gran mayoría se recomienda para el procedimiento del recuento de leucocitos o glóbulos blancos es la Cámara De Neubauer. En el “Manual De Procedimientos De Laboratorio En Técnicas Básicas De Hematología” (ZAMBRANO et al., 2005) y en “Procedimientos de laboratorio : manual : laboratorios locales I : laboratorios locales II” (Macalupú,

2013) ambos manuales publicados por el Ministerio de Salud del Perú. En ambos se indica el uso de Cámara De Neubauer para el conteo de leucocitos.

Según Bastidas el grado de error que se puede presentar al utilizar la cámara de Nubauer es de 20% a 30%. Es sus palabras :

> "Errores de hasta 20% y 30% son comunes con este método de recuento debido al pipeteo, a los errores estadísticos por ser la muestra poco representativa, errores del volumen de muestra realmente introducido en la cámara, etc.
>
> Sin embargo, el hematocitómetro sigue siendo uno de los métodos más ampliamente utilizados en los laboratorios de todo el mundo." (Bastidas, 2016)

2.2.3.2. Grado de error de máquinas especializadas

Para poder determinar el grado de error de máquinas especializadas se revisaron las fichas técnicas de estas máquinas donde se puede observar el porcentaje de error brindado por el fabricante.

Tabla 4
Grados de error de máquinas especializadas obtenidas de fichas técnicas

Modelo de maquina especializada (analizadores hematológicos)	Grado de error
Hematology Analyzer KT-6610 (Genrui Biotech Inc, 2017)	2%
HA3 Hematology Analyzer (BioSystems, 2018)	3%
The QBC AUTOREAD PLUS System (Drucker Diagnostics, 2018)	6.3%
The Automated Hematology Analyzer, the Pentra 80 (Franck SEGUY, n.d., 2018)	5%

DxH 900 High-volume Lab Hematology Analyzer (Beckman Coulter, n.d., 2017)	7%
SK8800 Hematology Analyzer (Sinothinker, 2020)	2%
SK9000 Hematology Analyzer (Sinothinker, 2020)	2%
SK9600 Hematology Analyzer (Sinothinker, 2020)	2%
PHOENIX NCC-5500 (NeoMedica, 2016)	1.5%
PHOENIX NCC-51 (NeoMedica, 2016)	1.5%
PHOENIX NCC-61 (NeoMedica, 2016)	2.5%
PHOENIX NCC-2310 (NeoMedica, 2016)	2.5%
PHOENIX NCC-3300 (NeoMedica, 2016)	2%
GloCyte Automated Cell Counter for CSF - 2016 (CAPTODAY, 2018)	2.7% - 16.3%
pocH-100i - 2004 (CAPTODAY, 2018)	3.5%
Advia 560/560AL Hematology - 2015 (CAPTODAY, 2018)	3.4%
Advia 360 Hematology System - 2015 (CAPTODAY, 2018)	4%

2.2.3.3. Grado de error de otras investigaciones

En la investigación titulada “Evaluación de los parámetros de desempeño de un contador hematológico” (Parés et al., 2015) se indica que:

> “ El requisito de calidad seleccionado para este labora- torio fue variabilidad biológica mínima (5), los valores de error total permitido (ETp%) según el mismo son: leucocitos: 23,17%; eritrocitos: 6,61%; plaquetas: 20,15%; hemoglobina: 6,29%.”(Parés et al., 2015)

Capítulo 3. Metodología

3.1. Tipo de investigación

La presente investigación tiene un enfoque Cuantitativo y pertenece al nivel descriptivo. Según Sampieri:

> "Los estudios descriptivos pretenden especificar las propiedades, características y perfiles de procesos, objetos o cualquier otro fenómeno que se someta a un análisis. Es decir, miden o recolectan datos y reportan información sobre diversos conceptos, variables, aspectos, dimensiones o componentes del fenómeno o problema a investigar. En un estudio descriptivo el investigador selecciona una serie de cuestiones (que, recordemos, denominamos variables) y después recaba información sobre cada una de ellas, para así representar lo que se investiga (describirlo o caracterizarlo)." (Sampieri et al., 2014)

En esta investigación se busca comparar y describir la solución que consta del uso de CNN y machine learning para la clasificación de leucocitos en imágenes microscópicas. Una de las variables que más importancia tiene en el desarrollo de la investigación es el grado de error que se calculará para la solución propuesta, esta variable nos permitirá responder varias preguntas, como: ¿La solución es suficientemente eficiente como para poder ser una opción válida?, ¿Qué tan eficiente es en comparación a otras soluciones?, ¿En relación con su costo, es una solución viable?, Etc.

3.2. Diseño y metodología de la investigación

La investigación se realizara siguiendo los pasos de la metodología KDD (Knowledge Discovery in Databases), pero se altera el procedimiento ya que el objetivo de la investigación no es solo que la tecnología encuentre la características distintivas en nuestra data. Sino también poder validar el proceso para la obtención de datos.

KDD tiene los siguientes pasos:

- Selección.
- Preprocesamiento/limpieza.
- Transformación/reducción.
- Técnicas de procesamiento de datos o Minería de datos (data mining).

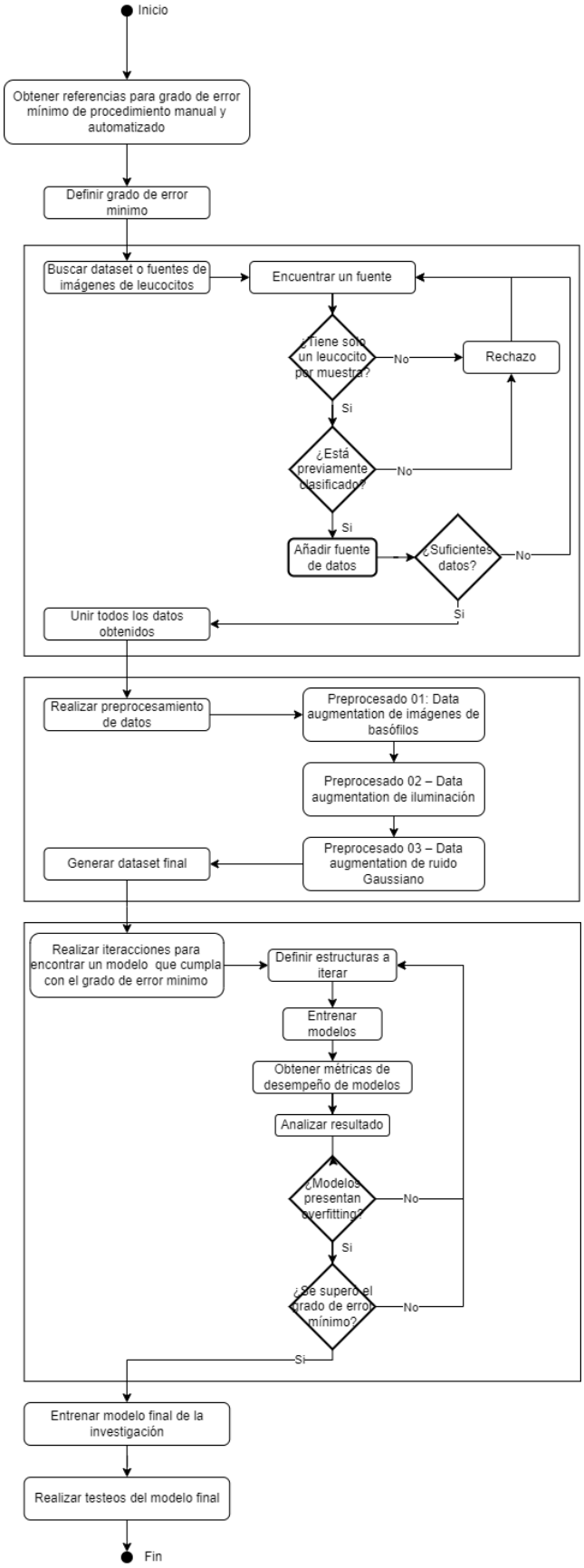
- Interpretación/evaluación

En esta investigación buscamos validar la técnica de procesamiento de datos y no el conocimiento qur podemos sacar de estos. Pero utilizando los mismos pasos y sabiendo cual es el resultado (conocimiento), esta metodología nos permitirá validar la técnica una vez utilizada.

Los pasos que nosotros seguimos en esta investigación son:

- Selección: Búsqueda y selección de todos los datasets y fuentes que cuenten con las características necesarias para nuestra investigación.
- Preprocesamiento/limpieza: Uniremos todos los datasets y modificaremos su formato de compresión para su uso con el framework de tensortflow.
- Transformación/reducción: En este caso, para poder tener un numero de muestras similar por cada tipo de leucocito, se utilizara técnicas de data augmentation para el aumento de muestras.
- Técnicas de procesamiento: La técnica de procesamiento en esta investigación es de Convolutional Neural Networks. En esta investigación, se realizo iteracciones con el objetivo de encontrar un modelo de CNN que cumpla con el grado de error mínimo que se obtuvo por un análisis bibliográfico.
- Interpretación/evaluación: Para la evaluación de nuestra solución, realizamos 4 tipos de evaluaciones con los cuales garantizar que el proceso nos da un grado de precisión que supere el grado de error mínimo aceptable.

Figura 23 Diagrama de flujo del desarrollo de la investigación.



En esta investigación se busca manipular las variables que conforman una red neuronal (Numero de neuronas, Numero de ciclos, Numero de capaz, etc.) para encontrar un modelo del cual se pueda obtener un grado de error con el cual responder a las preguntas de esta investigación.

La investigación se realizará en varias etapas:

3.2.1. Diseño del dataset o base de datos para la red neuronal

Según el objetivo de la investigación los datos serían las imágenes microscópicas de leucocitos tomadas con un microscopio. Este conjunto de datos (imágenes) tienen que ser dividido en 3 sets de datos:

- Set de imágenes de entrenamiento
- Set de imágenes de validación
- Set de imágenes de pruebas .

Las imágenes de entrenamiento y validación deben de estar previamente clasificadas preferentemente por una persona con experiencia o clasificados por una investigación previamente validada.

Este dataset originalmente estaba planeado ser conformado por imágenes obtenidas con equipos del laboratorio clínico de la UAC. Pero, por problemas globales como fue la pandemia de COVID19 del 2020-2021, no se pude tener el acceso físico a los equipos y tampoco se pudo conseguir el contacto con los respectivos encargados de estas áreas dentro de la Universidad Andina del Cusco.

Para poder continuar con la investigación se tomó la decisión de usar datasets de investigaciones previas que tengan características que satisfagan las necesidades de nuestros objetivos. Estas características serian:

- Dataset compuestos por imágenes de leucocitos obtenidas por microscopios.
- El formato del dataset no es problema siempre y cuando se pueda tratar junto con información de otros datasets.
- No se especificará resoluciones o calidad de imagen.
- Debe de poder ser validada su obtención y poder documentar su origen.

Esta decisión de obtener datos de diferentes fuentes trae consigo un nuevo problema. El que nuestro dataset no sería completamente homogéneo. Se consideraría data homogénea cuando cada dato dentro de nuestro tiene el mismo origen y, por lo tanto, mismas características.

Este problema también se podría considerar una ventaja. Al no tener data homogénea no tendremos problemas de un sesgo de obtención de datos. Un sesgo de origen es cuando todos los datos de un dataset fuero obtenidos con las mismas técnicas, en el mismo lugar y del mismo grupo de pruebas. Un sesgo de obtención de datos significaría que el trabajo realizado en esta investigación solo tendría relevancia en la misma ya que no se podría reproducir en otras futuras investigaciones .

Para la documentación de las pruebas se tiene que mostrar los siguientes puntos de relevancia:

- Origen
    - Nombre de investigación o proyecto
    - Autor
- Propósito original
- Numero de imágenes obtenidas total
- Numero de imágenes obtenidas por clase
- Imágenes de ejemplo
- Observaciones (opcional)

3.2.1.1. Unión de datasets

Como se especificó en el punto anterior, el dataset o base de datos que se utilizara para este proyecto estará conformado por otros dataset encontrados en internet que cumplan con las características necesarias.

En las características se especifica que se tendrá en cuenta los dataset siempre y cuando puede ser unido con los otros datasets. Esta característica tiene un

propósito de no traer imágenes con características más allá de las que podría tener una imagen normal de un frotis de sangre.

En caso de que alguna de los datasets presente características dispares a los demás datasets, se tendrá que realizar una evaluación de si se podrá corregir estas características. En caso de que si se pueda se tendrá que explicar el preprocesamiento que se tendrá que llevar a cabo para ese caso en específico.

El esquema de la explicación de ese procesamiento tendrá que ser el siguiente:

- Título: Preprocesamiento de dataset xx
- ¿Por qué se tiene que preprocesar ese dataset?
- Característica que cambiar o eliminar
- Explicación de método de preprocesamiento.
- Código de preprocesamiento
- Ejemplos de los resultados.

Para el proceso de unión de los dataset simplemente se tendrá que juntar las imágenes en una respectiva carpeta según el tipo de leucocito a la que corresponde. Un punto importante es que tendremos que tener todas las imágenes en un mismo formato, pero esta se encargara el preprocesamiento anterior.

3.2.2. Preprocesamiento de datos (imágenes) del dataset extraído

¿Qué consideramos un preprocesamiento de datos?

> Preprocesamiento de datos es todo aquel proceso de alteración del estado de un dataset antes de que se utilizado en el entrenamiento de una red neuronal.

¿Por qué realizar un preprocesamiento de datos?

> Se realizará un preprocesamiento de datos siempre que se quiera mejorar la calidad de un dataset de entrenamiento. Esto permitirá evitar encontrarse con el principal problema del entrenamiento de redes neuronales, el overfitting. Overfitting o sobreajuste es un punto

en el cual cualquier red neuronal puede llegar. Mejorar la calidad de un dataset de entrenamiento permite que el entrenamiento puede ser de más ciclos (epochs – épocas) donde no se presente el overfitting. El overfitting se presenta cuando la red neuronal empieza a encontrar relaciones que solo se encuentran en el dataset original y que no se podrán generalizar para otra data.

Con estas preguntas ya podemos tener en claro la razón del preprocesamiento en el desarrollo de la investigación, ahora tenemos que definir un esquema para poder mostrar el despliegue de un preprocesamiento. Para un preprocesamiento debemos indicar los siguientes puntos:

- Objetivo del preprocesamiento
- Método o técnica utilizada
- Código
- Resultado

3.2.3. Desarrollo y ajuste de la red neuronal CNN para pruebas de clasificación

Para poder definir el diseño de modelo de red neuronal primero es necesario tener resultados varios resultados. Se debe de poder comparar los resultados según las variaciones que se hagan en cada diseño. De esta forma se podrá observar que diseño obtiene menor grado de error.

Las variaciones que se tendrá en cuenta en este proyecto serán las siguientes:

Tabla 5
Variables de esquemas de modelos para la primera iteración

Código de diseño	Numero de neuronas	Numero de ciclos	Numero de capas ocultas	Función de Activación
D001	77	10	8	RELU
D002	61	10	6	RELU

D003	41	10	8	RELU
D004	77	15	8	RELU
D005	245	15	8	RELU
D006	61	15	6	RELU
D007	41	15	8	RELU
D008	77	10	8	SIGMOID
D009	61	10	6	SIGMOID
D010	41	10	8	SIGMOID
D011	77	15	8	SIGMOID
D012	61	15	6	SIGMOID
D013	41	15	8	SIGMOID

3.3. Población y muestra

La población de muestra será el total de imágenes con el cual se entrenará los modelos de redes neuronales. Este se le llamara Dataset. Los Dataset variaran con los preprocesamientos que se les aplique. Pero se espera que cualquier variación que se haga al dataset se documente y se figure su estado en ese punto de la investigación.

3.4. Recolección y análisis de datos – desarrollo de la investigación

3.4.1. Definir el grado de error aceptable

Usando la información recopilada y que se muestra en el marco teórico sobre el "análisis bibliográfico de Grado de Error" (Capitulo 2, punto 2.2.3 Análisis bibliográfico de Grado de error). Podemos calcular los promedios de grados de error que llega en los procesos manuales y con maquinaria especializada.

El promedio del grado de error del proceso manual es:

$$Prom.\,Proceso\,Manual = \frac{20\% + 10\% + \frac{20\% + 30\%}{2}}{3} = 18.33\%$$

El promedio del grado de error de la maquinas especializadas seria:

$$Prom.\,M.\,E. = \frac{2\% + 3\% + 6.3\% + 5\% + 7\% + 2\% + 2\% + 2\% + 1.5\% + 1.5\% + 2.5\% + 2.5\% + 2\% + \frac{2.7\% + 16.3\%}{2} + 3.5\% + 3.4\% + 4\%}{17}$$
$$= 3.59\%$$

Entonces podríamos decir que:

1. El grado de error aceptable para esta investigación tiene que ser mayor a 18.33%.
2. Si se alcanza un grado de error mayor a 3.59% se puede considerar como una solución equivalente o mejor a las maquinarias especializadas del mercado.

Por ende, para esta investigación se decidió definir el grado de error aceptable como el 5%. Este valor cumple con el requisito más importante el cual es ser mayor al grado de error del procedimiento manual. La principal razón de este valor también es que cuando se estaba haciendo las preguntas a personal de laboratorio clínico este indico que 5% es normalmente utilizado como el error estándar que se maneja cuando se hable de investigaciones académicas en el ámbito clínico, por estas razones se decidió utilizar este valor.

Este grado de error será utilizado como una meta que la arquitectura debe de lograr para poder validar que es suficientemente eficiente para el objetivo de esta investigación.

3.4.2. Etapa de obtención de dataset

Como se definió en la metodología. El primer paso de la investigación será la obtención de data para posteriormente juntarlo en un dataset. Los dataset encontrados serán lo siguiente.

3.4.2.1. Dataset 01

Nombre de dataset o proyecto

A Single-cell Morphological Dataset of Leukocytes from AML Patients and Non-malignant Controls (AML-Cytomorphology_LMU)

Autor

Matek, C., Schwarz, S., Marr, C., & Spiekermann, K.

Enlace

https://wiki.cancerimagingarchive.net/pages/viewpage.action?pageId=61080958

Propósito original de la data

El Munich AML Morphology Dataset fue obtenido por el Hospital Universitario de Munich de 100 pacientes diagnosticados con leucemia mieloide aguda entre los años 2014 y 2017. Este dataset fue originalmente creado para la investigación titulada “Human-level recognition of blast cells in acute myeloid leukaemia with convolutional neural networks. Nat Mach Intell”. Una vez finalizada la investigación, el autor público su data set para permitir que otros investigadores puedan realizar sus propios algoritmos de reconocimientos de leucocitos.

Numero de imágenes obtenidas total

14712

Numero de imágenes obtenidas por clase

Tabla 6
Numero de muestras obtenidas del Dataset 01

Clase de imagen	Numero de imágenes
Basófilo	78
Eosinófilo	424
Linfocito	3937
Monocito	1789
Neutrófilo	8484

Cita

"Matek, C., Schwarz, S., Marr, C., & Spiekermann, K. (2019). A Single-cell Morphological Dataset of Leukocytes from AML Patients and Non-malignant Controls [Data set]. The Cancer Imaging Archive. https://doi.org/10.7937/tcia.2019.36f5o9ld"

Observaciones

Como se puede observar en el conteo de imágenes por clases, este dataset muestra un serio desbalance en los basófilos y eosinófilos. Estos tendrán que ser complementados con imágenes de otros datasets o tendrán que ser mejorados con técnicas de preprocesamiento de datos como data augmentation.

Imágenes de ejemplo

Figura 24
Conteo de imágenes obtenidas de dataset 01 realizado con Python

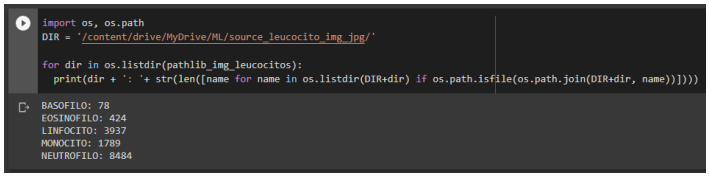


Figura 25
Eosinófilo 00001

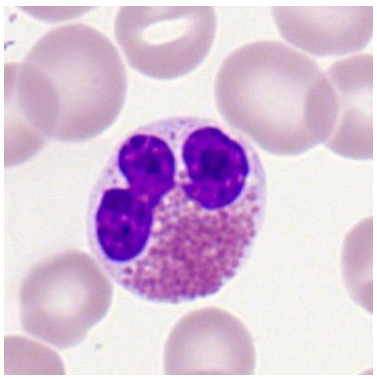
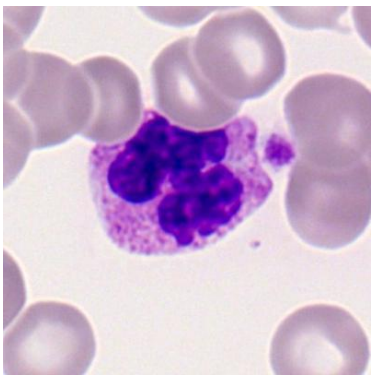


Figura 26
Basófilo 00001



3.4.2.2. Dataset 02

Nombre de dataset o proyecto

Blood Cell Images - BCCD Dataset

Autor

Paul Mooney

Enlace

https://www.kaggle.com/datasets/paultimothymooney/blood-cells

Propósito original de la data

Kaggle es una aplicación web que permite la publicación de datasets con propósitos académicos. Los datasets publicados en Kaggle tiene el objetivo de que sean utilizados para Data Science; en proyectos, practicas, portafolios , etc.

El dataset Blood Cell Images es un dataset de imágenes de las células de la sangre (no específicamente de leucocitos). La razón de la importancia del dataset es explicado en la página de Kaggle, este es:

> "El diagnóstico de enfermedades de origen sanguíneo a menudo implica identificar y caracterizar las muestras de sangre de los pacientes. Los métodos automatizados para detectar y clasificar

subtipos de células sanguíneas tienen importantes aplicaciones médicas."(Mooney, 2018)

Numero de imágenes obtenidas total

12507

Numero de imágenes obtenidas por clase

Tabla 7
Numero de muestras obtenidas del Dataset 02

Clase de imagen	Numero de imágenes
Basófilo	0
Eosinófilo	3133
Linfocito	3108
Monocito	3095
Neutrófilo	3171

Cita

Mooney, P. (2018, January 10). Blood Cell Images | Kaggle. Kaggle. https://www.kaggle.com/paultimothymooney/blood-cells

Observaciones

Las imágenes del dataset no son necesariamente específicas para la clasificación de leucocitos, busca la clasificación de células sanguíneas en general. Pero uno de los sub datasets si tiene clasificación por leucocitos. Este sub dataset tiene imágenes donde los leucocitos no son el objetivo principal de la imagen. Esta característica no es un discriminante, solamente indica una evaluación para saber si es necesario un preprocesamiento para juntarlo con los demás datasets.

Imágenes de ejemplo

Figura 27
Código de conteo de dataset 02

count_data.py

```
import os, os.path
DIR = 'D:/Proyectos/Tesis/datasets/Blood-cells/dataset-extraido/'

for dir in os.listdir(DIR):
  print(dir + ': '+ str(len([name for name in os.listdir(DIR+dir) if os.path.isfile(os.path.join(DIR+dir, name))])))
```

Figura 28
Conteo de dataset 02

```
C:\Users\g_onz\AppData\Local\Programs\Python\Python39\python.exe D:/Proyectos/Tesis/datasets/count_data.py
EOSINOFILO: 3133
LINFOCITO: 3108
MONOCITO: 3095
NEUTROFILO: 3171

Process finished with exit code 0
```

Figura 29
Linfocito

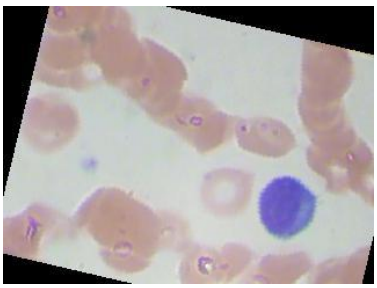
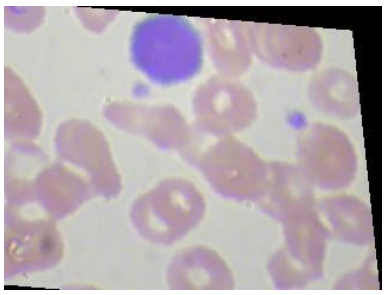


Figura 30
Monocito



3.4.2.3. Dataset 03

Nombre de dataset o proyecto

LISC: Leukocyte Images for Segmentation and Classification

Autor

Paul Mooney

Propósito original de la data

El dataset fue originalmente obtenido para la investigación titulada "Automatic recognition of five types of white blood cells in peripheral blood. Computerized Medical Imaging and Graphics"

Enlace

http://users.cecs.anu.edu.au/~hrezatofighi/Data/Leukocyte%20Data.htm

Numero de imágenes obtenidas total

247

Numero de imágenes obtenidas por clase

Tabla 8
Numero de muestras obtenidas del Dataset 03

Clase de imagen	Numero de imágenes
Basófilo	53
Eosinófilo	39
Linfocito	52
Monocito	48
Neutrófilo	50

Cita

Rezatofighi, S. H., & Soltanian-Zadeh, H. (2011). Leukocyte Data. http://users.cecs.anu.edu.au/~hrezatofighi/Data/Leukocyte Data.htm

Observaciones

Los problemas de este dataset son dos. Primero, la imagen no necesariamente tiene a los leucocitos con enfoque principal, este problema puede ser resuelto con preprocesamiento, acercando la imagen a la zona del leucocito.

El segundo problema es más serio. Hay imágenes donde aparecen más de un leucocito, aunque en estos casos se tendría que trabajar tratando de separar las imágenes o duplicando la imagen y censurando un leucocito en cada una.

Imágenes de ejemplo

Figura 30
Código de conteo de dataset 03

count_data.py

```
import os, os.path
DIR = 'D:/Proyectos/Tesis/datasets/LISC/dataset-extraido-jpg/'
for dir in os.listdir(DIR):
  print(dir + ': ' + str(len([name for name in os.listdir(DIR + dir) if os.path.isfile(os.path.join(DIR + dir, name))])))
```

Figura 31
Conteo de dataset 03

```
C:\Users\g_onz\AppData\Local\Programs\Python\Python39\python.exe D:/Proyectos/Tesis/datasets/count_data.py
BASOFILO: 53
EOSINOFILO: 39
LINFOCITO: 52
MONOCITO: 48
NEUTROFILO: 50
```

Figura 32
Neutrófilo

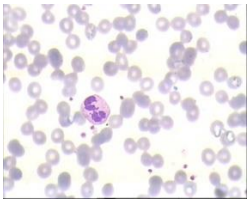
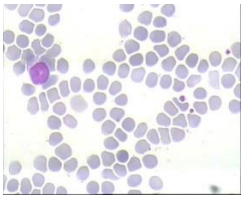


Figura 33
Linfocito



3.4.2.4. Dataset 04

Nombre de dataset o proyecto

A dataset for microscopic peripheral blood cell images for development of automatic recognition systems

Autores

Andrea Acevedo, Anna Merino, Santiago Alferez, Ángel Molina, Laura Boldú, José Rodellar

Enlace

https://www.sciencedirect.com/science/article/pii/S2352340920303681

Propósito original de la data

El dataset fue obtenido en laboratorio central del Hospital Clinic de Barcelona. Las imágenes fueron capturadas usando el analizador CellaVision DM96. Originalmente consta de 17092 imágenes individuales de células con una resolución de 360 x 363.

Este dataset fue desarrollado con el principal propósito de apoyar a cualquier investigación de machine learning y deep learning.

Numero de imágenes obtenidas total

10299

Numero de imágenes obtenidas por clase

Tabla 9
Numero de muestras obtenidas del Dataset 04

Clase de imagen	Numero de imágenes
Basófilo	1218
Eosinófilo	3117
Linfocito	1214

Monocito	1420
Neutrófilo	3330

Cita

Acevedo, A., Merino, A., Alférez, S., Molina, Á., Boldú, L., & Rodellar, J. (2020). A dataset of microscopic peripheral blood cell images for development of automatic recognition systems. Data in Brief, 30, 105474. https://doi.org/10.1016/j.dib.2020.105474

Imágenes de ejemplo

Figura 34
Código de conteo de dataset 04

count_data.py

```
import os, os.path
DIR = 'D:/Proyectos/Tesis/datasets/PBC_dataset/dataset-extraido/'

for dir in os.listdir(DIR):
  print(dir + ': '+ str(len([name for name in os.listdir(DIR+dir) if os.path.isfile(os.path.join(DIR+dir, name))])))
```

Figura 35
Conteo de dataset 04

count_data

```
C:\Users\g_onz\AppData\Local\Programs\Python\Python39\python.exe D:/Proyectos/Tesis/datasets/count_data.py
BASOFILO: 1218
EOSINOFILO: 3117
LINFOCITO: 1214
MONOCITO: 1420
NEUTROFILO: 3330
```

Figura 36
Basófilo 01

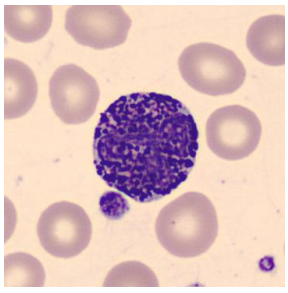
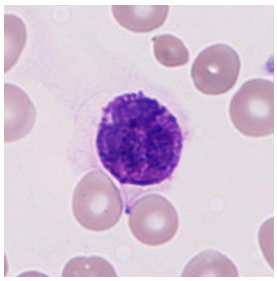


Figura 37
Basófilo 02



3.4.3. Unión de datasets

3.4.3.1. Preprocesamiento de dataset 02

¿Por qué se tiene que preprocesar este dataset?

Las imágenes que componen el dataset son imágenes de frotis sanguíneos con la peculiaridad de estar rotadas pero que contiene la imagen en el centro con las dimensiones originales. Esto genera que el vacío que dejo las rotaciones de la imagen se rellenó con el color negro (en el formato RGB seria (0,0,0)).

Característica que cambiar o eliminar

- Vacíos en las imágenes

Explicación de método de preprocesamiento.

El preprocesamiento que se llevara a cabo consta de rellenar los vacíos negros con un color esquema de colores que se asemejen a el color de fondo de los frotis sanguíneos.

Código de preprocesamiento

```
1.  from PIL import Image
2.  import random as rd
3.  import os, os.path
4.
5.  DIR = '/content/drive/MyDrive/ML/tesis_datasets/Blood-cells/'
6.  res_dir = '/content/Blood-cells-result/'
7.  os.mkdir(res_dir)
8.  for dir in os.listdir(DIR):
9.    inside_dir = DIR+dir+'/'
10.   os.mkdir(res_dir+dir)
11.   for file_name in os.listdir(inside_dir):
12.     img = Image.open(inside_dir+file_name)
13.     # img = img.convert('RGBA')
14.     pixeles = img.load()
15.     largo, ancho = img.size
16.     for h in range(largo):
17.       for w in range(ancho):
18.         # print(pixeles[h,w])
19.         (r,g,b) = pixeles[h,w]
20.         if r <= 40 and g <= 40 and b <= 40:
21.         # if (r,g,b) == (0,0,0):
22.           pixeles[h,w] = (
    rd.randint(210,218),rd.randint(210,218),rd.randint(210,218))
23.     img.save(res_dir+dir+'/'+file_name)
24.
25.
```

Ejemplos de los resultados.

Figura 38
Eosinófilo _0_187 de dataset 02 antes del preprocesamiento

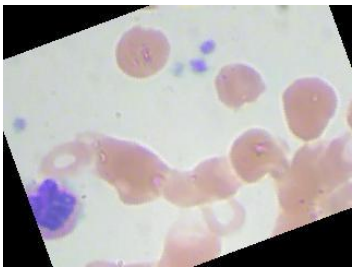
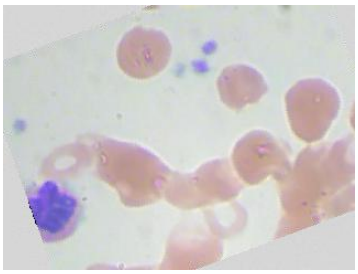


Figura 39
Eosinófilo _0_187 de dataset 02 después de preprocesamiento



3.4.3.2. Estado de Datasets para proceso de unión

Para el proceso de unión de los datasets debemos de tener en cuenta que vamos a trabajar con imágenes que tienen que estar en un mismo formato para evitar problemas con la creación del dataset en tensorflow.

Según la documentación de tensorflow, su método `image_dataset_from_directory(main_directory, labels='inferred')` solo admite imágenes en los formatos jpeg, png, bmp y gif (*Tf.Keras.Preprocessing.Image_dataset_from_directory*, n.d.)

Los formatos en los que se encuentra cada dataset son:

Tabla 10
Formatos de imágenes de los datasets

Dataset	Extensión original	Resolución
Dataset 01	.tiff	400 x 400 pixeles
Dataset 02	.jpeg	320 x 240 pixeles
Dataset 03	.bmp	720 x 576 pixeles
Dataset 04	.jpg	360 x 363 pixeles

Ahora pasaremos a escoger el formato que usaremos. Primero descartaremos gif ya que ninguna de nuestros dataset se presente en gif. BMP (mapa de bits) es un formato más enfocado a la edición de imágenes. BMP es un formato más pesado hablando de tamaño de archivo ya que guarda más bits por cada pixel. Esta característica le permite retratar mejor los colores que para su edición es mucho más importante.

En el caso de proyecto no necesitamos unos pixeles tan detallados de una imagen ya que este causara que se tenga que manejar volúmenes mayores al momento de entrenar la red neuronal. También que durante el preprocesado de imágenes en ningún momento se escalara las imágenes a una resolución mayor del original. Ahora igual viendo los tamaños de los dataset obtenidos, el único que tiene el formato BMP es el más pequeño. En caso de tener que transformar los demás datasets al formato BMP no ganaríamos más información ya que se originaria de información que ya fue previamente comprimida en un formato con pérdidas como los es JPG. Así que teniendo en cuenta los tres puntos anteriores podemos descartaríamos BMP como el formato final.

PNG es un formato de compresión de imágenes sin perdidas. Además, PNG tiene la característica de guardar un dato más por píxel, este es el nivel de transparencia. Aunque parece ser que PNG nos podría servir para el objetivo del proyecto, si tuviéramos imágenes ya en formatos que sean sin perdidas ese sería el caso. Pero como se puede ver en la tabla 10, 2 datasets ya tienen el formato JPG (JPG y JPEG son el mismo formato solo que .jpg proviene de un sistema heredado del nombre de archivos en MS-DOS). También, por el tipo de imágenes que estamos utilizando, la transparencia no añadiría ningún beneficio a la hora de que nuestra red neuronal entrene, así que tampoco sería beneficioso usar PNG.

JPEG es el formato con el que nos quedaríamos. La principal desventaja de JPEG es la compresión con pérdidas. La compresión con pérdidas se refiere que al usar algoritmos que compriman nuestra imagen, ignoren o eliminan una cantidad de información de nuestra imagen con el propósito de generalizar y guardar esa misma información en un espacio más pequeño. Las ventajas que tenemos actualmente con JPEG son, primero, la compresión de archivos es la más baja de todas. Al tener archivos más pequeños no tendremos que necesitar una memoria

RAM exagerada para poder entrenar nuestras redes neuronales. Esta característica es especialmente útil ya que el entrenamiento y preprocesado de las imágenes se realizarán con el servicio de Google COLAB que es un servicio de ejecución de notebooks de Python en la nube. COLAB te da una capacidad de hardware limitada para cada sesión de tus notebooks y al no tener que llenar esta RAM que nos da COLAB permite que la ejecución se pueda realizar sin ningún error (llenar la RAM hace que salte un error y se detenga todo el proceso). También que los formatos originales de 2 de nuestros datasets son JPEG y que estos ya no se tendrán que cambiar.

3.4.3.3. Preprocesado de dataset 01 y 03

¿Por qué se tiene que preprocesar este dataset?

El formato escogido es JPEG. El formato de los dataset 01 y 03 son TIFF y BMP respectivamente. Estos deben de pasar a JPEG.

Característica que cambiar o eliminar

- Formato de imágenes

Explicación de método de preprocesamiento.

Preprocesado de dataset 01

Usando la librería PIL (Pillow) de Python podemos cargar las imágenes que están en formato TIFF, convertir el formato de pixeles a RGB y guardar en formato JPEG. El nivel de compresión utilizado es el default que viene con la función de guardar.

Preprocesado de dataset 02

Usando la librería PIL (Pillow) de Python podemos cargar las imágenes que están en formato BMP (mapa de bits), convertirlo y guardarlo en formato JPEG. El nivel de compresión utilizado es el default que viene con la función de guardar.

Código de preprocesamiento

Preprocesamiento de dataset 01

```
from PIL import Image
def tiff_to_jpg(prefijo,carpeta):
  numero = 1
  for name in data_dir2.glob(carpeta+'/*.tiff'):
    im = Image.open(name)
    im = im.convert('RGB')
    ceros = 5 - len(str(numero))
    texto_num = ''
    for x in range(ceros):
      texto_num= texto_num +'0'
    texto_num=texto_num+str(numero)
    im.save('/content/drive/MyDrive/ML/source_leucocito_img_jpg/'+carpeta+'/'+
prefijo+ texto_num + '.jpg', 'JPEG')
    numero+=1
tiff_to_jpg('EOS_','EOSINOFILO')
tiff_to_jpg('LFC_','LINFOCITO')
tiff_to_jpg('MNO_','MONOCITO')
tiff_to_jpg('NTF_','NEUTROFILO')

```

Preprocesamiento de dataset 03

```
from PIL import Image
import os.path
import pathlib
dataset = 'LISC'
DIR = 'D:/Proyectos/Tesis/datasets/'+dataset+'/dataset-extraido/'
res_dir = 'D:/Proyectos/Tesis/datasets/'+dataset+'/dataset-extraido-jpg/'
try:
    os.mkdir(res_dir)
except:
    print("Carpeta  ya creada")

for type in os.listdir(DIR):
    inside_dir = DIR + type + '/'
    try:
        os.mkdir(res_dir+type+'/')
    except:
        print("Carpeta  ya creada")
    inside_dir = pathlib.Path(inside_dir)
    contador = 1
    for fileName in inside_dir.glob('*.bmp'):
        img = Image.open(fileName)
        ceros = 5 - len(str(contador))
        nombre=''
        for x in range(ceros):
            nombre = nombre + '0'
        nombre = type[:3] + '-' + nombre + str(contador) + '.jpg'
        img.save(res_dir+type+'/'+nombre,'JPEG')
        contador+=14

```

Ejemplos de los resultados

Figura 40
Carpetas de dataset original y convertido

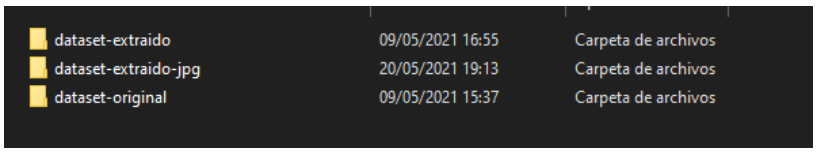


Figura 41
Imágenes originales dataset 03

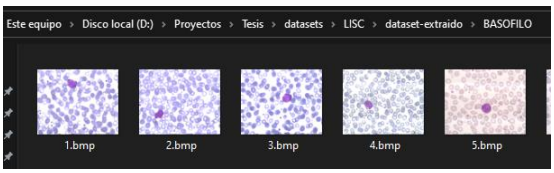


Figura 42
Imágenes en formato JPEG dataset 03

3.4.3.4. Dataset final

Numero de imágenes obtenidas total

37760

Numero de imágenes por dataset por tipo y total

Tabla 11
Numero de muestras del Dataset final

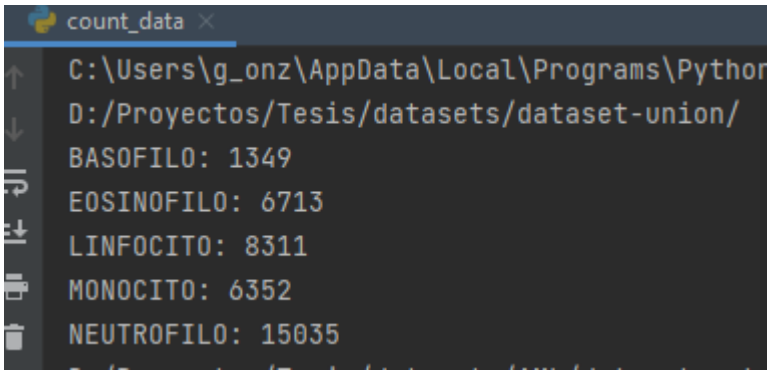
Clase	Dataset 01	Dataset 02	Dataset 03	Dataset 04	Dataset final
Basófilo	78	0	53	1218	1349
Eosinófilo	424	3133	39	3117	6713
Linfocito	3937	3108	52	1214	8311
Monocito	1789	3095	48	1420	6352
Neutrófilo	8484	3171	50	3330	15035
Total	14712	12507	242	10299	37760

Código de conteo

```
1.  import os, os.path
2.  DIR_list = [
3.    'D:/Proyectos/Tesis/datasets/dataset-union/',
4.    'D:/Proyectos/Tesis/datasets/AML/dataset-extraido-jpg/',
5.    'D:/Proyectos/Tesis/datasets/Blood-cells/dataset-extraidoV2.0/',
6.    'D:/Proyectos/Tesis/datasets/LISC/dataset-extraido-jpg/',
7.    'D:/Proyectos/Tesis/datasets/PBC_dataset/dataset-extraido/',
8.  ]
9.
10. for DIR in DIR_list:
11.   print(DIR)
12.   for dir in os.listdir(DIR):
13.     print(dir + ': '+ str(len([name for name in os.listdir(DIR+dir) if
    os.path.isfile(os.path.join(DIR+dir, name))])))
```

Prueba

Figura 43
Conteo de dataset final



3.4.4. Preprocesado de dataset

3.4.4.1. Preprocesado 01 – Data augmentation de imágenes de basófilos

El objetivo de aplicar Data Augmentation a las imágenes de basófilos es el de aumentar el número de muestras que se tiene actualmente. En este punto de la investigación tenemos un total de 1349 muestras de imágenes de basófilos. La muestra de basófilos solo es un 3.57% del total de muestras de nuestro dataset.

Objetivo del preprocesamiento

Aumentar número de muestras de basófilos

Método o técnica utilizada

Se usará el paquete keras de tensorflow, más específicamente la función ImageDataGenerator. Dentro de esa función se debe de proporcionar las variantes que se quiere tomar en cuenta para las imágenes seleccionadas. Como se está trabajando con imágenes donde hay un objetivo que debe de ser identificado podemos descartar el uso de re-escalamiento y de métodos que cambien la imagen del basófilo, ya que podríamos cortar o eliminar características importantes por error. Como las células no tienen una orientación definida, la opción más simple es usar la orientación para generar nuevas muestras.

En este caso se tomará en cuenta rotaciones entre 0 y 360 grados. También flip (invertir imagen) tanto horizontal como vertical.

Código

```
from numpy import expand_dims
from keras.preprocessing.image import load_img
from keras.preprocessing.image import array_to_img
from keras.preprocessing.image import img_to_array
from keras.preprocessing.image import ImageDataGenerator
from matplotlib import pyplot
import tensorflow as tf
import os.path

DIR = '/content/drive/MyDrive/ML/dataset-union/BASOFILO/'
n=0
for img in os.listdir(DIR):
  DIR2 = '/content/drive/MyDrive/ML/dataset-union/BASOFILO/'
  img = DIR2 + img
  try:
    img = load_img(img)
  except:
    print('No es una imagen')
    continue
  img.save('/content/prueba3/bas_'+str(n)+'.jpg')
  n+=1
  data = img_to_array(img)
  samples = expand_dims(data, 0)
  datagen = ImageDataGenerator(rotation_range=360,horizontal_flip=True,vertical_flip=True)
  it = datagen.flow(samples, batch_size=1)
  r=8
  for i in range(r):
    batch = it.next()
    image = batch[0].astype('uint8')
    save = array_to_img(image)
    save.save('/content/prueba3/bas_'+str(n)+'.jpg')
    n+=1
```

Resultado

Figura 44
Preprocesamiento 01 – Basófilo original

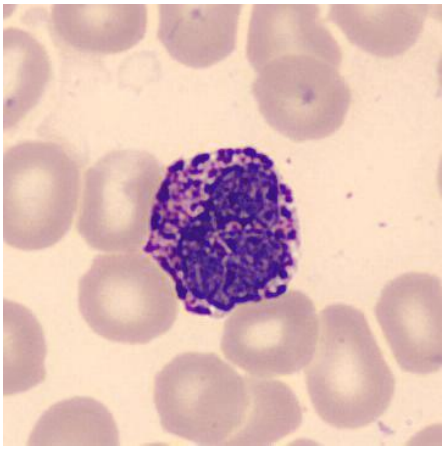


Figura 45
Preprocesamiento 02 – Basófilo generado 01

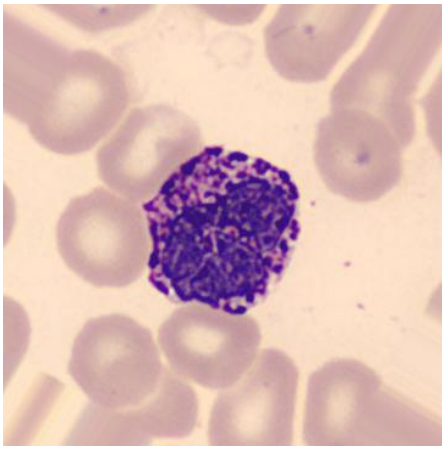


Figura 46
Preprocesamiento 02 – Basófilo generado 02

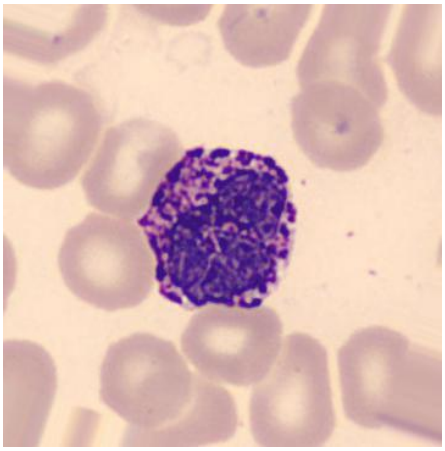


Figura 47
Preprocesamiento 02 – Basófilo generado 03

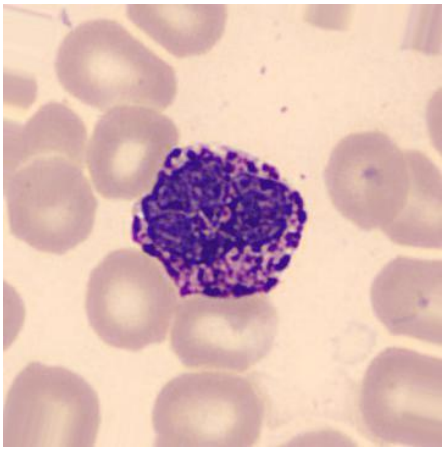


Figura 48
Preprocesamiento 02 – Basófilo generado 04

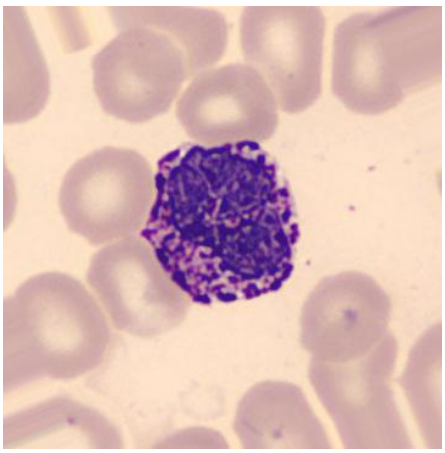


Figura 49 *Preprocesamiento 02 – Basófilo generado 05*

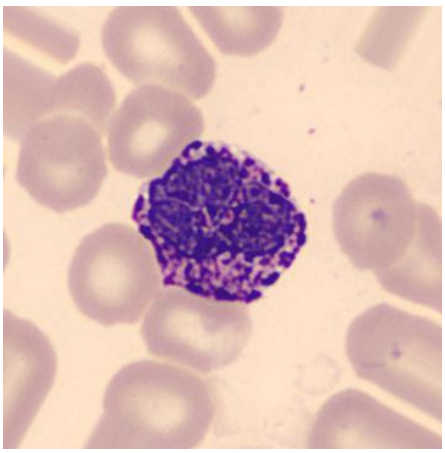


Tabla 12
Numero de muestras del dataset – preprocesamiento 01

Tipo de leucocito	**N. muestras original**	**N. muestras obtenidas**	**N. total de muestras**
Basófilo	1349	12141	12141
Eosinófilo	6713	0	6713
Linfocito	8311	0	8311
Monocito	6352	0	6352
Neutrófilo	15035	0	15035
Total			48552

3.4.4.2. Preprocesado 02 – Data augmentation de iluminación

Las imágenes rescatadas en el dataset son obtenidas del proceso de frotis sanguíneo. En este proceso se realiza un paso llamado tinción. Como se explicó en el marco teórico, hay 3 principales métodos de tinción para frotis sanguíneos. Cada uno está enfocado a resaltar un elemento de la sangre y características. Pero según el Ministerio de Salud de Chile; un frotis sanguíneo, independientemente de la tinción utilizada, los colores que identifican cada elemento de la sangre deberían de siempre los mismo siempre y cuando se realice el método correcto (De Salud, n.d.). Esta verificación nos indica que no debería de haber un cambio de coloración significativa por el método de tinción utilizada en el proceso de obtención de frotis sanguíneo.

Ahora, si bien los métodos y técnicas utilizadas para la obtención del frotis sanguíneo están reguladas como pudimos ver. No existe regulaciones al momento de obtener una imagen del frotis sanguíneo. En el proceso de obtención de imágenes el uso de una fuente de luz es importante. Pero este no se puede especificar siempre su nivel de intensidad. Es preproceso busca el de aumentar las muestras con un nivel de iluminación más alto.

Objetivo del preprocesamiento

Aumentar número de muestras con una iluminación mayor.

Método o técnica utilizada

Se usará el paquete imgaug. Este paquete se describe a sí mismo como “Esta biblioteca de Python le ayuda a aumentar las imágenes para sus proyectos de aprendizaje automático. Convierte un conjunto de imágenes de entrada en un nuevo conjunto mucho más grande de imágenes ligeramente alteradas.” (Aleju, n.d.)

Código

```
import os.path
import random as r

import imageio
import imgaug as ia
import imgaug.augmenters as iaa
from keras.preprocessing.image import array_to_img

# Vaiables
DIR = '/content/drive/MyDrive/ML/dataset-union/'
saveDir = '/content/saveData/'
dataPreprocessPorcent = 0.1;
r.seed( 123 )


#Preprocedure
contrast=iaa.GammaContrast(gamma=0.5)

try:
  os.mkdir(saveDir)
except:
  print('Carpeta ya creada')
  # continue

for subdir in os.listdir(DIR):
  DIR2 = '/content/drive/MyDrive/ML/dataset-union/'
  saveSubDir = saveDir + subdir +'/'
  subdir = DIR2 + subdir
  dir = os.listdir(subdir)
  imgSelected =r.sample(dir,int(len(dir)*dataPreprocessPorc
ent))
  n=0
  try:
    os.mkdir(saveSubDir)
  except:
    print('Carpeta ya creada')
  for img in imgSelected:
```

Resultado

Figura 50
Preprocesamiento 02 – Leucocito original

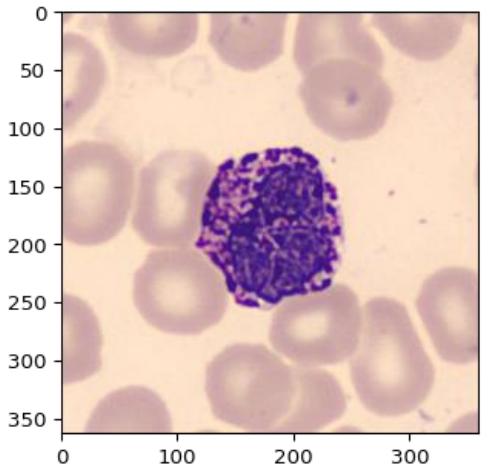


Figura 51
Preprocesamiento 02 – Leucocito post-preprocesamiento

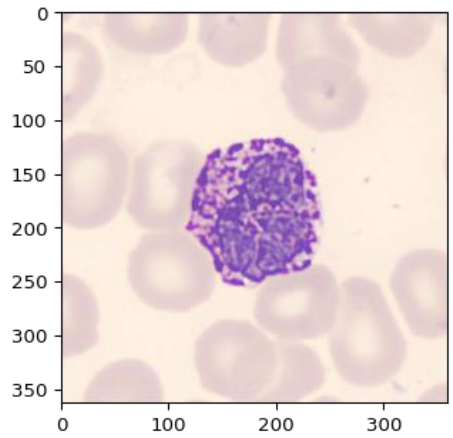


Tabla 13
Numero de muestras del dataset – preprocesamiento 02

Tipo de leucocito	**N. muestras original**	**N. muestras obtenidas**	**N. total de muestras**
Basófilo	12141	135	12276
Eosinófilo	6713	671	7384
Linfocito	8311	831	9142
Monocito	6352	635	6987
Neutrófilo	15035	1503	16538
Total			52327

3.4.4.3. Preprocesado 03 – Data augmentation de ruido Gaussiano

Se utiliza la técnica de ruido gaussioano con el objetivo de generar imágenes que simulen que fueron corrompidas. Este paso de data augmentation permite validar este tipo de errores.

Objetivo del preprocesamiento

Aumentar número de muestras con ruido Gaussiano

Método o técnica utilizada

Se usará el paquete imgaug. Este paquete se describe a sí mismo como “Esta biblioteca de Python le ayuda a aumentar las imágenes para sus proyectos de aprendizaje automático. Convierte un conjunto de imágenes de entrada en un nuevo conjunto mucho más grande de imágenes ligeramente alteradas.” (Aleju, n.d.)

Código

```
import os.path
import random as r

import imageio
import imgaug as ia
import imgaug.augmenters as iaa
from keras.preprocessing.image import array_to_img

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

# Vaiables
DIR = '/content/drive/MyDrive/ML/dataset-union/'
saveDir = '/content/saveData/'
dataPreprocessPorcent = 0.1;
r.seed( 1234 )


#Preprocedure
preproceso_list = [
  iaa.AdditiveGaussianNoise(scale=0.1*255,per_chan
nel=True),
  iaa.AdditiveGaussianNoise(scale=0.1*255,per_chan
nel=True)
]

try:
  os.mkdir(saveDir)
except:
  print('Carpeta ya creada')
  # continue

for subdir in os.listdir(DIR):
  DIR2 = '/content/drive/MyDrive/ML/dataset-
union/'
  saveSubDir = saveDir + subdir +'/'
  subdir = DIR2 + subdir
```

Resultado

Figura 52
Preprocesamiento 03 – Basófilo original

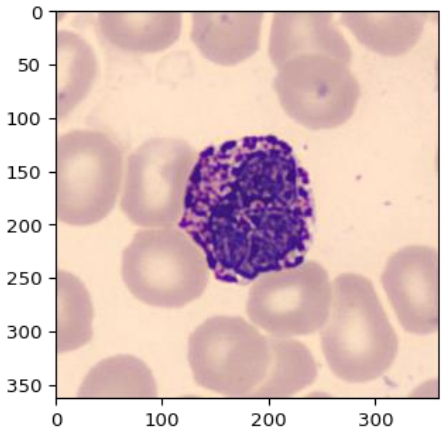


Figura 53
Preprocesamiento 03 – Basófilo preprocesado 01

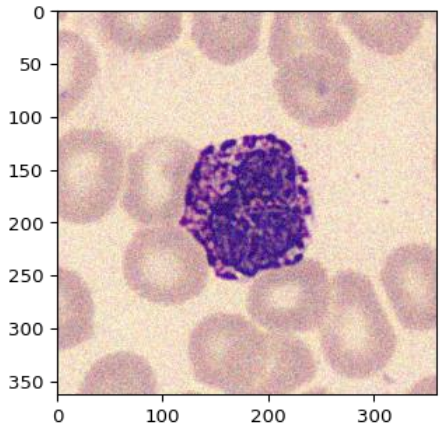


Figura 54
Preprocesamiento 03 – Basófilo preprocesado 02

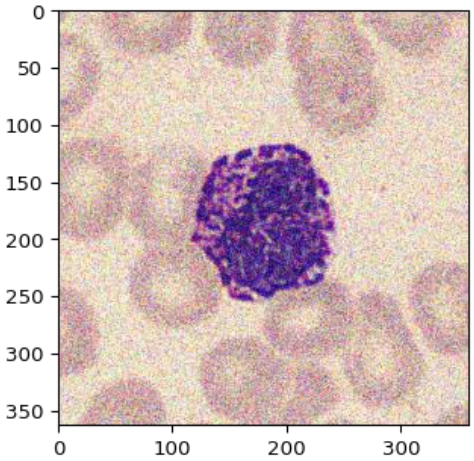
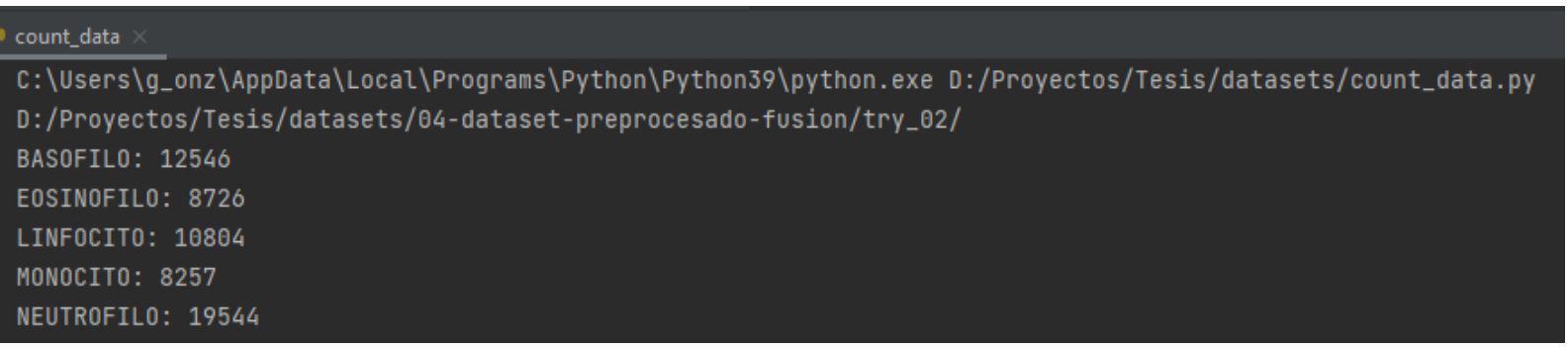


Tabla 14
Numero de muestras del dataset – preprocesamiento 03

Tipo de leucocito	N. muestras original	N. muestras obtenidas	N. total de muestras
Basófilo	12276	270	12546
Eosinófilo	7384	1342	8726
Linfocito	9142	1662	10804
Monocito	6987	1270	8257
Neutrófilo	16538	3006	19544
Total			59877

Figura 55
Conteo de muestra de dataset después de preprocesamiento 03

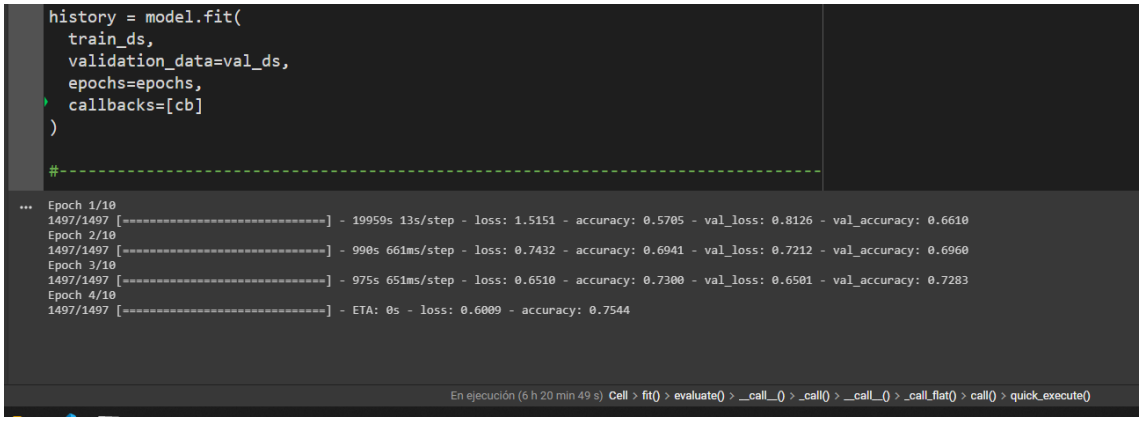


3.4.5. Mejoramiento del espacio de Google Colab

Durante el proceso de entrenamiento de los modelos de redes neuronales con el dataset 02 se encontraron algunos problemas de rendimiento.

El principal problema es la extensa duración de entrenamiento. Durante la primera prueba de entrenamiento del modelo D0001 con el Dataset DS02 se pudo observar una extensa duración la cual no permitía que el investigador pueda tener una hora en la cual esperar que finalice el proceso. Como se puede observar en la Figura 56, la duración de entrenamiento hasta el cuarto ciclo fue de 6 horas y 20 minutos.

Figura 56
Primer intento de entrenamiento de D001 - problema de tiempo



Los problemas de que la duración de entrenamiento sea tan extensa son:

- Google Colab es un servicio en la nube que trabaja con sesiones. Cada sesión tiene un tiempo máximo. La duración máxima por sesión es de 12 horas. Pero, Google Colab debe registrar actividad de parte del usuario constante para llegar a las 12 horas máximas de sesión. Mantenerse en estado de procesamiento sin actividad de acción del usuario se considera inactivo. Colab manda señales para saber si hay un usuario activo. Si no se le da respuesta, Colab cierra sesión y se elimina toda la data de RAM y de disco duro de esa sesión.
- El investigador puede perder el fácilmente la sesión. En estos casos, el investigador se ve obligado a reiniciar todo el proceso de entrenamiento.

3.4.5.1. Mejoras de Google Colab Pro

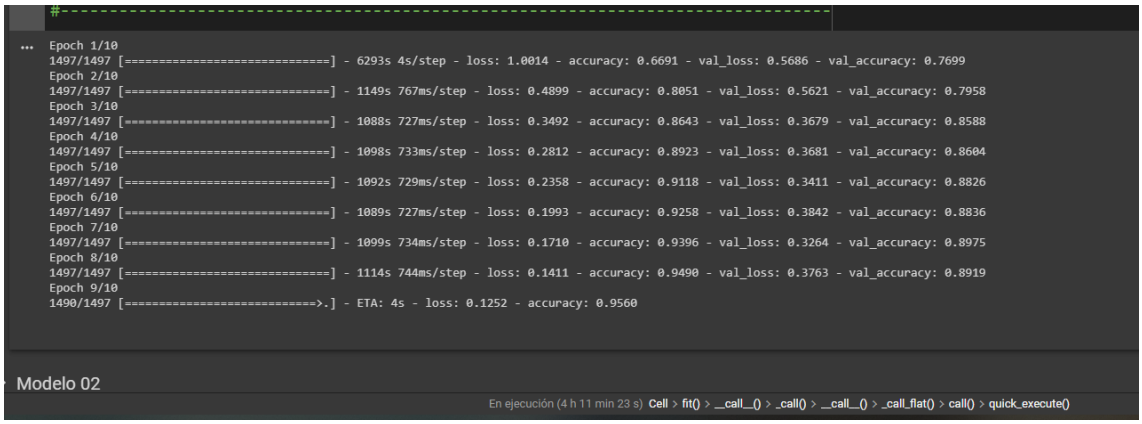
Los siguientes datos fueron obtenidos del documento que Google Colab nos brinda al momento de pasar al servicio de Colab pro.

Tabla 15
Tabla de mejoras de Google Colab a Google Colab Pro

Característica	Descripción
GPU más rápidas	Acceso a GPU T4 o P100. Las sesiones gratis usan GPU K80.
Más memoria	Acceso a "High-RAM" en el menú "Runtime > Change runtime type". Esto significa que las máquinas virtuales con las que tu sesión esté conectada tendrán al menos 20 GB de RAM.
Tiempos de ejecución más extensos	El tiempo máximo de cada sesión es aproximadamente el doble que los límites de los usuarios no suscritos. Aproximadamente 24 horas.

3.4.5.2. Resultado de usar Google Colab Pro

Figura 57
Mejora de tiempo con Google Colab Pro



3.4.6. Iteración 01 - Entrenamiento con Dataset 02

3.4.6.1. Resultados de entrenamiento de DS02-D001

Tabla 16
Resultados de entrenamiento - iteración 01 - DS02-D001

Diseño	D001
Precisión (accuracy) - entrenamiento	0.957
Loss - entrenamiento	0.126
Precisión (accuracy) – validación	0.909
Loss - validación	0.280
Grado de error - entrenamiento	0.043
Grado de error - validación	0.091

3.4.6.2. Resultados de entrenamiento de DS02-D002

Tabla 17
Resultados de entrenamiento - iteración 01 - DS02-D002

Diseño	D002
Precisión (accuracy) - entrenamiento	0.9238
Loss - entrenamiento	0.2029
Precisión (accuracy) – validación	0.8119
Loss - validación	0.5442
Grado de error - entrenamiento	0.0762
Grado de error - validación	0.1881

3.4.6.3. Resultados de entrenamiento de DS02-D003

Tabla 18
Resultados de entrenamiento - iteración 01 - DS02-D003

Diseño	D003
Precisión (accuracy) - entrenamiento	0.8537
Loss - entrenamiento	0.3694
Precisión (accuracy) – validación	0.8326
Loss - validación	0.4426
Grado de error - entrenamiento	0.1463
Grado de error - validación	0.1674

3.4.6.4. Resultados de entrenamiento de DS02-D004

Tabla 19
Resultados de entrenamiento - iteración 01 - DS02-D004

Diseño	D004
Precisión (accuracy) - entrenamiento	0.9676
Loss - entrenamiento	0.0978
Precisión (accuracy) – validación	0.9069
Loss - validación	0.3302
Grado de error - entrenamiento	0.0324
Grado de error - validación	0.0931

3.4.6.5. Resultados de entrenamiento de DS02-D005

Tabla 20
Resultados de entrenamiento - iteración 01 - DS02-D005

Diseño	D005
Precisión (accuracy) - entrenamiento	0.9813
Loss - entrenamiento	0.0631
Precisión (accuracy) – validación	0.9309
Loss - validación	0.2757
Grado de error - entrenamiento	0.0187
Grado de error - validación	0.0691

3.4.6.6. Resultados de entrenamiento de DS02-D006

Tabla 21
Resultados de entrenamiento - iteración 01 - DS02-D006

Diseño	D006
Precisión (accuracy) - entrenamiento	0.9681
Loss - entrenamiento	0.0927
Precisión (accuracy) – validación	0.8730
Loss - validación	0.4253
Grado de error - entrenamiento	0.0319
Grado de error - validación	0.1270

3.4.6.7. Resultados de entrenamiento de DS02-D007

Tabla 22
Resultados de entrenamiento - iteración 01 - DS02-D007

Diseño	D007
Precisión (accuracy) - entrenamiento	0.8897
Loss - entrenamiento	0.2950
Precisión (accuracy) – validación	0.8639
Loss - validación	0.3755
Grado de error - entrenamiento	0.1103
Grado de error - validación	0.1361

3.4.6.8. Resultados de entrenamiento de DS02-D008

Tabla 23
Resultados de entrenamiento - iteración 01 - DS02-D008

Diseño	D008
Precisión (accuracy) - entrenamiento	0.3270
Loss - entrenamiento	1.5566
Precisión (accuracy) – validación	0.3240
Loss - validación	1.5578
Grado de error - entrenamiento	0.6730
Grado de error - validación	0.6760

3.4.6.9. Resultados de entrenamiento de DS02-D009

Tabla 24
Resultados de entrenamiento - iteración 01 - DS02-D009

Diseño	D009
Precisión (accuracy) - entrenamiento	0.3270
Loss - entrenamiento	1.5566
Precisión (accuracy) – validación	0.3240
Loss - validación	1.5578
Grado de error - entrenamiento	0.6730
Grado de error - validación	0.6760

3.4.6.10. Resultados de entrenamiento de DS02-D010

Tabla 25
Resultados de entrenamiento - iteración 01 - DS02-D010

Diseño	D010
Precisión (accuracy) - entrenamiento	0.3309
Loss - entrenamiento	1.5553
Precisión (accuracy) – validación	0.3240
Loss - validación	1.5577
Grado de error - entrenamiento	0.6691
Grado de error - validación	0.6760

3.4.6.11. Resultados de entrenamiento de DS02-D011

Tabla 26
Resultados de entrenamiento - iteración 01 - DS02-D011

Diseño	D011
Precisión (accuracy) - entrenamiento	0.8923
Loss - entrenamiento	0.2980
Precisión (accuracy) – validación	0.8397
Loss - validación	0.4183
Grado de error - entrenamiento	0.1077
Grado de error - validación	0.1603

3.4.6.12. Resultados de entrenamiento de DS02-D012

Tabla 27
Resultados de entrenamiento - iteración 01 - DS02-D012

Diseño	D012
Precisión (accuracy) - entrenamiento	0.3270
Loss - entrenamiento	1.5564
Precisión (accuracy) – validación	0.3240
Loss - validación	1.5576
Grado de error - entrenamiento	0.6730
Grado de error - validación	0.6760

3.4.6.13. Resultados de entrenamiento de DS02-D013

Tabla 28
Resultados de entrenamiento - iteración 01 - DS02-D013

Diseño	D013
Precisión (accuracy) - entrenamiento	0.3270
Loss - entrenamiento	1.5561
Precisión (accuracy) – validación	0.3240
Loss - validación	1.5576
Grado de error - entrenamiento	0.6730
Grado de error - validación	0.6760

3.4.7. Resultados de entrenamiento de Dataset 02 con los modelos D001 – D0013

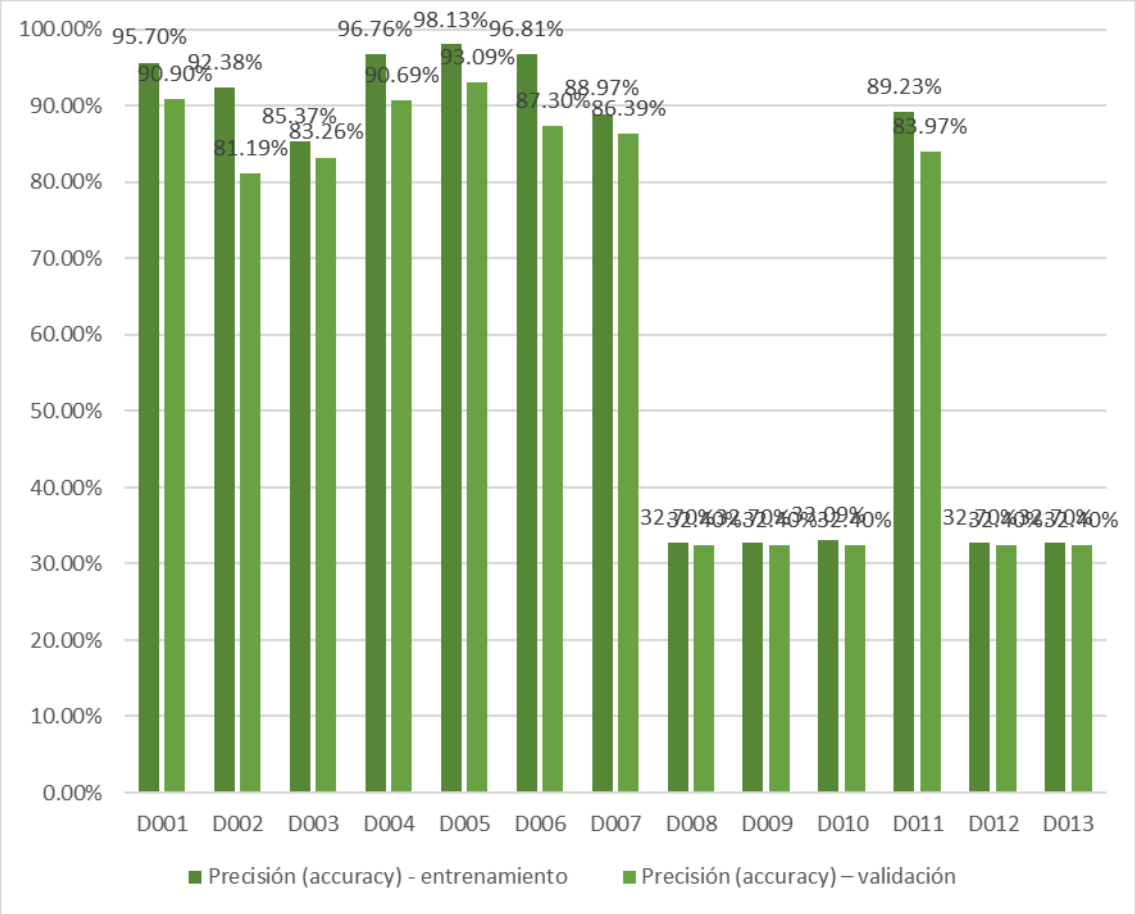
Tabla 29
Resumen resultados de entrenamiento – iteración 01

Diseño	D001	D002	D003	D004	D005	D006	D007	D008	D009	D010	D011	D012	D013
Precisión (accuracy) - entrenamiento	0.9570	0.9238	0.8537	0.9676	0.9813	0.9681	0.8897	0.3270	0.3270	0.3309	0.8923	0.3270	0.3270
Loss - entrenamiento	0.1260	0.2029	0.3694	0.0978	0.0631	0.0927	0.2950	1.5566	1.5566	1.5553	0.2980	1.5564	1.5561
Precisión (accuracy) – validación	0.9090	0.8119	0.8326	0.9069	0.9309	0.8730	0.8639	0.3240	0.3240	0.3240	0.8397	0.3240	0.3240
Loss - validación	0.2800	0.5442	0.4426	0.3302	0.2757	0.4253	0.3755	1.5578	1.5578	1.5577	0.4183	1.5576	1.5576
Grado de error - entrenamiento	0.0430	0.0762	0.1463	0.0324	0.0187	0.0319	0.1103	0.6730	0.6730	0.6691	0.1077	0.6730	0.6730
Grado de error - validación	0.0910	0.1881	0.1674	0.0931	0.0691	0.1270	0.1361	0.6760	0.6760	0.6760	0.1603	0.6760	0.6760

Tabla 30
Resumen estadístico resultados de entrenamiento – iteración 01

Diseño	D001	D002	D003	D004	D005	D006	D007	D008	D009	D010	D011	D012	D013
Precisión (accuracy) - entrenamiento	95.70%	92.38%	85.37%	96.76%	98.13%	96.81%	88.97%	32.70%	32.70%	33.09%	89.23%	32.70%	32.70%
Precisión (accuracy) – validación	90.90%	81.19%	83.26%	90.69%	93.09%	87.30%	86.39%	32.40%	32.40%	32.40%	83.97%	32.40%	32.40%
Grado de error - entrenamiento	4.30%	7.62%	14.63%	3.24%	1.87%	3.19%	11.03%	67.30%	67.30%	66.91%	10.77%	67.30%	67.30%
Grado de error - validación	9.10%	18.81%	16.74%	9.31%	6.91%	12.70%	13.61%	67.60%	67.60%	67.60%	16.03%	67.60%	67.60%

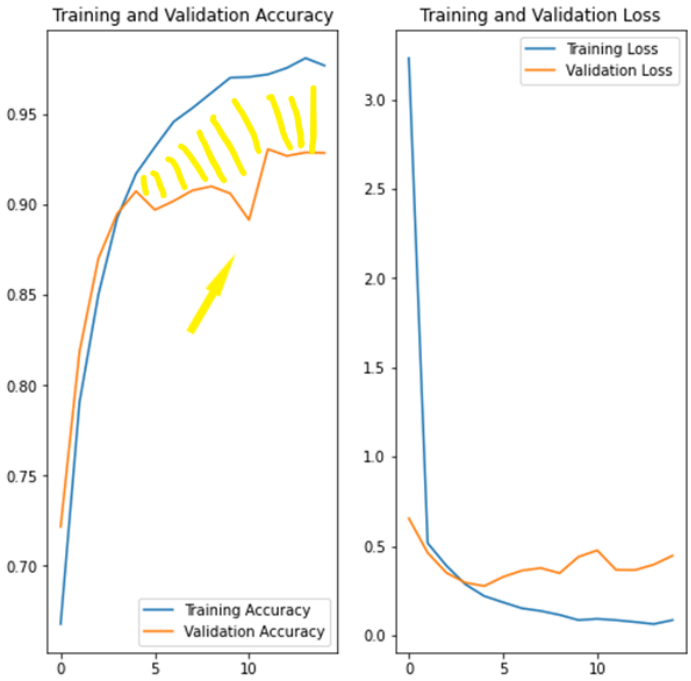
Figura 58
Grafico de barras - accuracy - iteración 01



3.4.7.1. Análisis de Resultados

Como se puede observar en el gráfico de precisión obtenida de los modelos entrenados, el modelo que obtuvo más precisión es el D005. Lo más probable de este resultado es el overfitting. La razón de overfitting más segura puede ser el exceso de neuronas por capa. En el diagrama de comparación de entrenamiento y validación se ve que la diferencia que hay entre la precisión obtenida en el proceso de entrenamiento y la precisión del proceso de validación se va agrandando de manera irregular. Este es un gran indicio de overfitting, ya que muestra cómo se red neuronal se va entrenando para mejorar la clasificación del conjunto de entrenamiento, pero al momento que se le presenta información diferente (conjunto de validación) la precisión no sube a un ritmo regular y hasta empieza a bajar.

Figura 59
Grafico de demostración de overfitting – D005



En el lado contrario. Revisando los diagramas de resultados, las gráficas comparativas que menos muestran overfitting son D003, D007 y D011

Figura 60
Grafico de demostración de no overfitting – D003

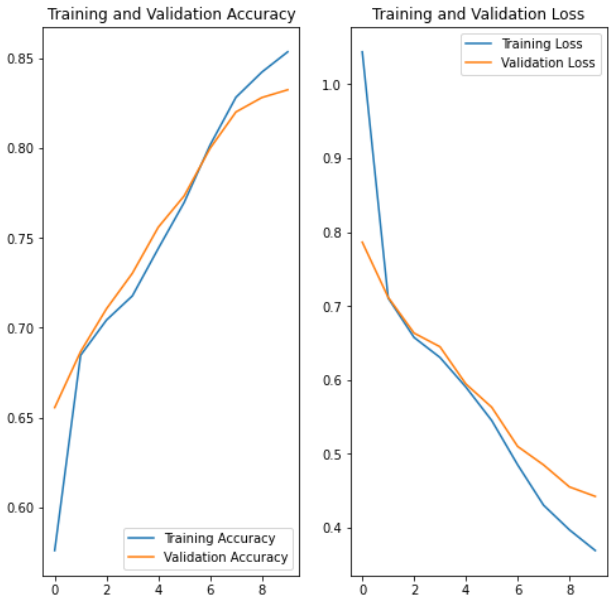


Figura 61
Grafico de demostración de no overfitting – D007

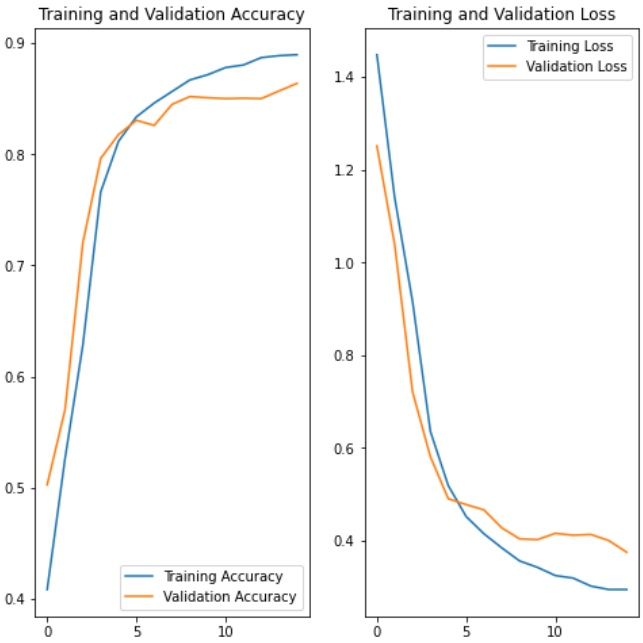
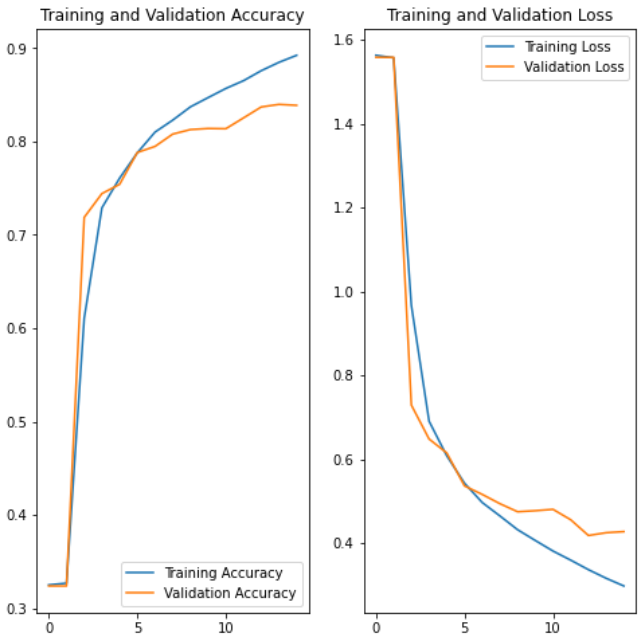


Figura 62
Grafico de demostración de no overfitting – D011



El modelo D003 y D007 son el mismo esquema de modelo, la diferencia es el número de ciclos que cada uno fue entrenado. D003 fue 10 ciclos y D007 fue 15 ciclos. Por esta razón ambos modelos se tratarán como uno.

Otro resultado importante e interesante es el del modelo D011. Este modelo es el único que usando la función de activación SIGMOID puedo entrenar el modelo y también

mostrar que no hay overfitting durante su entrenamiento. Pienso que valdría la pena probar mutaciones de este modelo y obtener resultados.

3.4.7.2. Decisión de toma de acción

Los modelos con los que iteraremos serán (D003 | D007) y D011. La siguiente iteración será la I02. Las variaciones que se manejarán serán:

1. Primero, vamos a revisar los resultados si entrenamos más ciclos los modelos resultantes (20 ciclos cada modelo). I02-D001 = (D003 | D007) y I02-D002 = D011
2. Aumentar un nuevo par de capas ocultas a 15 ciclos.
3. Aumentar 2 pares de capas ocultas a 15 ciclos.

3.4.8. Iteración 02 – Modelos de CNN a probar

Para todos los procesos de entrenamiento se utilizará el dataset D002. Y los modelos que se entrenarán tendrán las siguientes características:

Tabla 31
Resumen de esquema de modelos - iteración 02

Código de diseño	Numero de neuronas	Numero de ciclos	Numero de capas ocultas	Función de Activación
I02-D001	41	20	10	RELU
I02-D002	77	20	8	SIGMOID
I02-D003	49	15	10	RELU
I02-D004	93	15	10	SIGMOID
I02-D005	57	15	12	RELU
I02-D006	109	15	12	SIGMOID

3.4.9. Iteración 02 – entrenamiento y resultados por modelo

3.4.9.1. Resultados de entrenamiento de I02-D001

Tabla 32
Resultados de entrenamiento - iteración 02 - DS02-D001

Diseño	I02-D001
Precisión (accuracy) - entrenamiento	0.9122
Loss - entrenamiento	0.2249
Precisión (accuracy) – validación	0.8545
Loss - validación	0.4291
Grado de error - entrenamiento	0.0878
Grado de error - validación	0.1455

3.4.9.2. Resultados de entrenamiento de I02-D002

Tabla 33
Resultados de entrenamiento - iteración 02 - DS02-D002

Diseño	I02-D002
Precisión (accuracy) - entrenamiento	0.7128
Loss - entrenamiento	0.7971
Precisión (accuracy) – validación	0.7182
Loss - validación	0.7820
Grado de error - entrenamiento	0.2872
Grado de error - validación	0.2818

3.4.9.3. Resultados de entrenamiento de I02-D003

Tabla 34
Resultados de entrenamiento - iteración 02 - DS02-D003

Diseño	I02-D003
Precisión (accuracy) - entrenamiento	0.8790
Loss - entrenamiento	0.3138
Precisión (accuracy) – validación	0.8687
Loss - validación	0.3479
Grado de error - entrenamiento	0.1210
Grado de error - validación	0.1313

3.4.9.4. Resultados de entrenamiento de I02-D004

Tabla 35
Resultados de entrenamiento - iteración 02 - DS02-D004

Diseño	I02-D004
Precisión (accuracy) - entrenamiento	0.8685
Loss - entrenamiento	0.3425
Precisión (accuracy) – validación	0.8575
Loss - validación	0.3648
Grado de error - entrenamiento	0.1315
Grado de error - validación	0.1425

3.4.9.5. Resultados de entrenamiento de I02-D005

Tabla 36
Resultados de entrenamiento - iteración 02 - DS02-D005

Diseño	I02-D005
Precisión (accuracy) - entrenamiento	0.8679
Loss - entrenamiento	0.3419
Precisión (accuracy) – validación	0.8605
Loss - validación	0.3639
Grado de error - entrenamiento	0.1321
Grado de error - validación	0.1395

3.4.9.6. Resultados de entrenamiento de I02-D006

Tabla 37
Resultados de entrenamiento - iteración 02 - DS02-D006

Diseño	I02-D006
Precisión (accuracy) - entrenamiento	0.8764
Loss - entrenamiento	0.3134
Precisión (accuracy) – validación	0.8863
Loss - validación	0.2984
Grado de error - entrenamiento	0.1236
Grado de error - validación	0.1137

3.4.10. Resultados de entrenamiento Iteración 02 con los modelos D001 – D0006

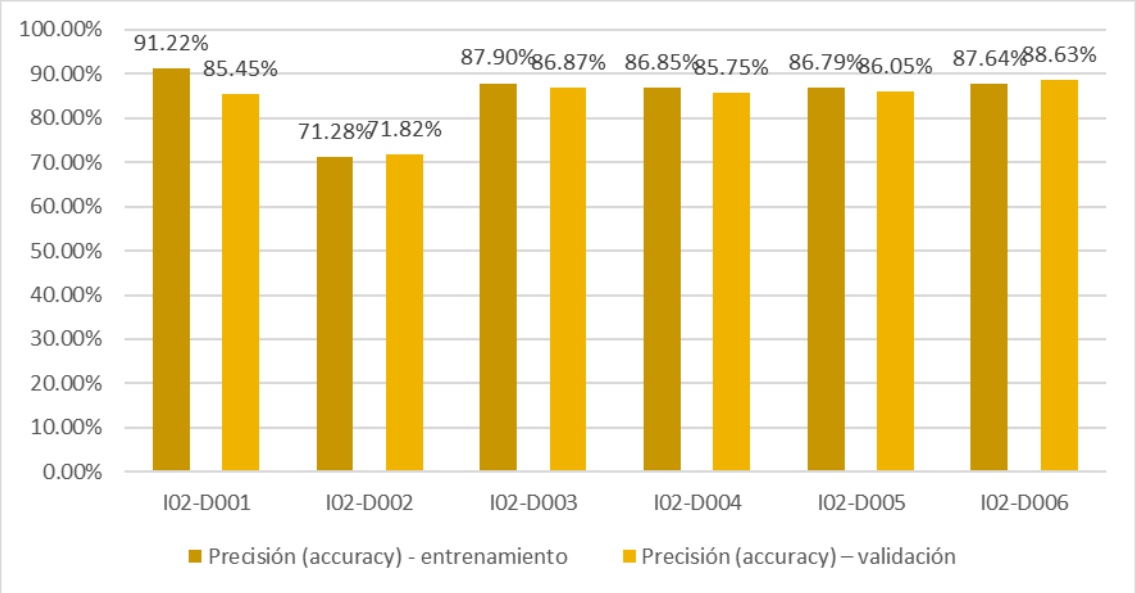
Tabla 38
Resumen resultados de entrenamiento – iteración 02

Diseño	I02-D001	I02-D002	I02-D003	I02-D004	I02-D005	I02-D006
Precisión (accuracy) - entrenamiento	0.9122	0.7128	0.879	0.8685	0.8679	0.8764
Loss - entrenamiento	0.2249	0.7971	0.3138	0.3425	0.3419	0.3134
Precisión (accuracy) – validación	0.8545	0.7182	0.8687	0.8575	0.8605	0.8863
Loss - validación	0.4291	0.782	0.3479	0.3648	0.3639	0.2984
Grado de error - entrenamiento	0.0878	0.2872	0.121	0.1315	0.1321	0.1236
Grado de error - validación	0.1455	0.2818	0.1313	0.1425	0.1395	0.1137

Tabla 39
Resumen estadístico resultados de entrenamiento – iteración 02

Diseño	I02-D001	I02-D002	I02-D003	I02-D004	I02-D005	I02-D006
Precisión (accuracy) - entrenamiento	91.22%	71.28%	87.90%	86.85%	86.79%	87.64%
Precisión (accuracy) – validación	85.45%	71.82%	86.87%	85.75%	86.05%	88.63%
Grado de error - entrenamiento	8.78%	28.72%	12.10%	13.15%	13.21%	12.36%
Grado de error - validación	14.55%	28.18%	13.13%	14.25%	13.95%	11.37%

Figura 63
Grafico de barras - accuracy - iteración 02



3.4.10.1. Análisis de Resultados

En esta iteración se usó la estrategia de reducir el número de neuronas por capa. Viendo los resultados, se observa que el overfitting se redujo. En muchos casos los valores de precisión y loss son más estables y equivalentes. Normalmente, al aumentar capas a un modelo se tiende a hacer que las capas ocultas más profundas, analicen data más específica. También, las nuevas capas más profundas hacen que el modela tienda al overfitting.

En los modelos se puede observar que se puede aumentar las épocas. No hubo un punto claro en el que tienda al overfitting o se estanque en el entrenamiento. Lo mejor es darle más épocas y buscar el punto perfecto.

La siguiente estrategia que se puede utilizar para retrasar el overfitting es el uso de regularización. Esta estrategia se utilizaría en modelos que ya se supo que llegan a un gran nivel de precisión.

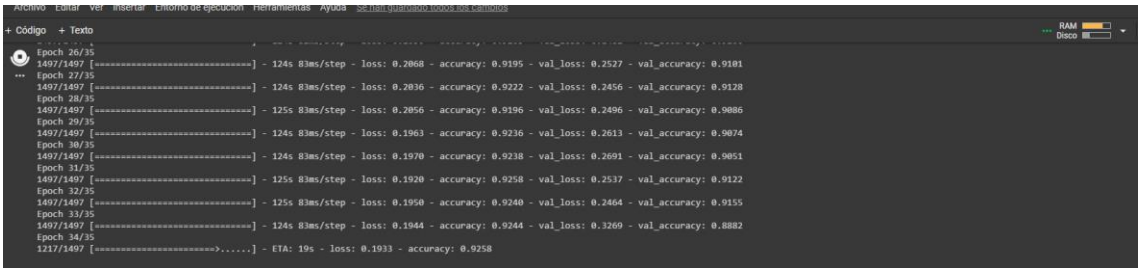
3.4.10.2. Decisión de toma de acción

La siguiente iteración será la I03. Las variaciones que se manejarán serán:

1. Vamos a aumentar los ciclos de I02-D003 a I02-D006 a 30 ciclos (iba a ser 50 ciclos, por tema de memoria RAM se tenía que disminuir). Este tiene el objetivo

de identificar el ciclo en el cual empiezan los indicios de overfitting. Estos serán los modelos I03-D001 a I03-D004.

Figura 64
Ejemplo de sobrecarga de RAM de sesión de Google Cloud



2. Vamos a usar la estrategia de regularización. Para esta opción, vamos a usar los modelos anteriores que tuvieron el mayor índice de precisión. DS02-D004, DS02-D005 y DS02-D006.Aumentaremos a 20 ciclos de entrenamiento también. Estos serán los modelos I03-D005, I03-D006 y I03-D007. Valor de 0.001
3. Vamos a usar la estrategia de regularización. Estos serán los modelos I03-D008, I03-D009 y I03-D010. Valor de 0.0001
4. Vamos a usar la estrategia de regularización. Estos serán los modelos I03-D011, I03-D012 y I03-D013. Valor de 0.01

3.4.11. Iteración 03 – Modelos de CNN a probar

Para todos los procesos de entrenamiento se utilizará el dataset D002. Y los modelos que se entrenarán tendrán las siguientes características:

Tabla 40
Resumen de esquema de modelos - iteración 03

Código de diseño	Numero de neuronas	Numero de ciclos	Numero de capas ocultas	Función de Activación	Valor de Regularización
I03-D001	49	50	10	RELU	-
I03-D002	93	30	10	SIGMOID	-
I03-D003	57	35	12	RELU	-
I03-D004	109	30	12	SIGMOID	-
I03-D005	77	15	8	RELU	0.001

I03-D006	245	15	8	RELU	0.001
I03-D007	61	15	6	RELU	0.001
I03-D008	77	15	8	RELU	0.0001
I03-D009	245	15	8	RELU	0.0001
I03-D010	61	15	6	RELU	0.0001
I03-D011	77	15	8	RELU	0.01
I03-D012	245	15	8	RELU	0.01
I03-D013	61	15	6	RELU	0.01

3.4.12. Iteración 03 – entrenamiento y resultados por modelo

3.4.12.1. Resultados de entrenamiento de I03-D001

Tabla 41
Resultados de entrenamiento - iteración 03 - DS02-D001

Diseño	I03-D001
Precisión (accuracy) - entrenamiento	0.9461
Loss - entrenamiento	0.1393
Precisión (accuracy) – validación	0.9004
Loss - validación	0.3012
Grado de error - entrenamiento	0.0539
Grado de error - validación	0.0996

3.4.12.2. Resultados de entrenamiento de I03-D002

Tabla 42
Resultados de entrenamiento - iteración 03 - DS02-D002

Diseño	I03-D002
Precisión (accuracy) - entrenamiento	0.9232
Loss - entrenamiento	0.2199
Precisión (accuracy) – validación	0.8897
Loss - validación	0.3035
Grado de error - entrenamiento	0.0768
Grado de error - validación	0.1103

3.4.12.3. Resultados de entrenamiento de I03-D003

Tabla 43
Resultados de entrenamiento - iteración 03 - DS02-D003

Diseño	I03-D003
Precisión (accuracy) - entrenamiento	0.9271
Loss - entrenamiento	0.1897
Precisión (accuracy) – validación	0.9179
Loss - validación	0.2380
Grado de error - entrenamiento	0.0729
Grado de error - validación	0.0821

3.4.12.4. Resultados de entrenamiento de I03-D004

Tabla 44
Resultados de entrenamiento - iteración 03 - DS02-D004

Diseño	I03-D004
Precisión (accuracy) - entrenamiento	0.8910
Loss - entrenamiento	0.2920
Precisión (accuracy) – validación	0.8950
Loss - validación	0.2905
Grado de error - entrenamiento	0.1090
Grado de error - validación	0.1050

3.4.12.5. Resultados de entrenamiento de I03-D005

Tabla 45
Resultados de entrenamiento - iteración 03 - DS02-D005

Diseño	I03-D005
Precisión (accuracy) - entrenamiento	0.9603
Loss - entrenamiento	0.2578
Precisión (accuracy) – validación	0.9292
Loss - validación	0.3715
Grado de error - entrenamiento	0.0397
Grado de error - validación	0.0708

3.4.12.6. Resultados de entrenamiento de I03-D006

Tabla 46
Resultados de entrenamiento - iteración 03 - DS02-D006

Diseño	I03-D006
Precisión (accuracy) - entrenamiento	0.9528
Loss - entrenamiento	0.3144
Precisión (accuracy) – validación	0.9004
Loss - validación	0.5192
Grado de error - entrenamiento	0.0472
Grado de error - validación	0.0996

3.4.12.7. Resultados de entrenamiento de I03-D007

Tabla 47
Resultados de entrenamiento - iteración 03 - DS02-D007

Diseño	I03-D007
Precisión (accuracy) - entrenamiento	0.9338
Loss - entrenamiento	0.3400
Precisión (accuracy) – validación	0.8645
Loss - validación	0.5849
Grado de error - entrenamiento	0.0662
Grado de error - validación	0.1355

3.4.12.8. Resultados de entrenamiento de I03-D008

Tabla 48
Resultados de entrenamiento - iteración 03 - DS02 - D008

Diseño	I03-D008
Precisión (accuracy) - entrenamiento	0.9805
Loss - entrenamiento	0.1166
Precisión (accuracy) – validación	0.9228
Loss - validación	0.3590
Grado de error - entrenamiento	0.0195
Grado de error - validación	0.0772

3.4.12.9. Resultados de entrenamiento de I03-D009

Tabla 49
Resultados de entrenamiento - iteración 03 - DS02 - D009

Diseño	I03-D009
Precisión (accuracy) - entrenamiento	0.9860
Loss - entrenamiento	0.1161
Precisión (accuracy) – validación	0.9401
Loss - validación	0.3075
Grado de error - entrenamiento	0.0140
Grado de error - validación	0.0599

3.4.12.10. Resultados de entrenamiento de I03-D010

Tabla 50
Resultados de entrenamiento - iteración 03 - DS02 - D010

Diseño	I03-D010
Precisión (accuracy) - entrenamiento	0.7988
Loss - entrenamiento	0.5025
Precisión (accuracy) – validación	0.7674
Loss - validación	0.6875
Grado de error - entrenamiento	0.2012
Grado de error - validación	0.2326

3.4.12.11.Resultados de entrenamiento de I03-D011

Tabla 51
Resultados de entrenamiento - iteración 03 - DS02 - D011

Diseño	I03-D011
Precisión (accuracy) - entrenamiento	0.9005
Loss - entrenamiento	0.4372
Precisión (accuracy) – validación	0.9084
Loss - validación	0.4269
Grado de error - entrenamiento	0.0995
Grado de error - validación	0.0916

3.4.12.12.Resultados de entrenamiento de I03-D012

Tabla 52
Resultados de entrenamiento - iteración 03 - DS02 - D012

Diseño	I03-D012
Precisión (accuracy) - entrenamiento	0.9197
Loss - entrenamiento	0.4441
Precisión (accuracy) – validación	0.9304
Loss - validación	0.4294
Grado de error - entrenamiento	0.0803
Grado de error - validación	0.0696

3.4.12.13.Resultados de entrenamiento de I03-D013

Tabla 53
Resultados de entrenamiento - iteración 03 - DS02 - D013

Diseño	I03-D013
Precisión (accuracy) - entrenamiento	0.8792
Loss - entrenamiento	0.5937
Precisión (accuracy) – validación	0.8756
Loss - validación	0.6029
Grado de error - entrenamiento	0.1208
Grado de error - validación	0.1244

3.4.13. Resultados de entrenamiento Iteración 03 con los modelos D001 – D0013

Tabla 54
Resumen resultados de entrenamiento – iteración 03

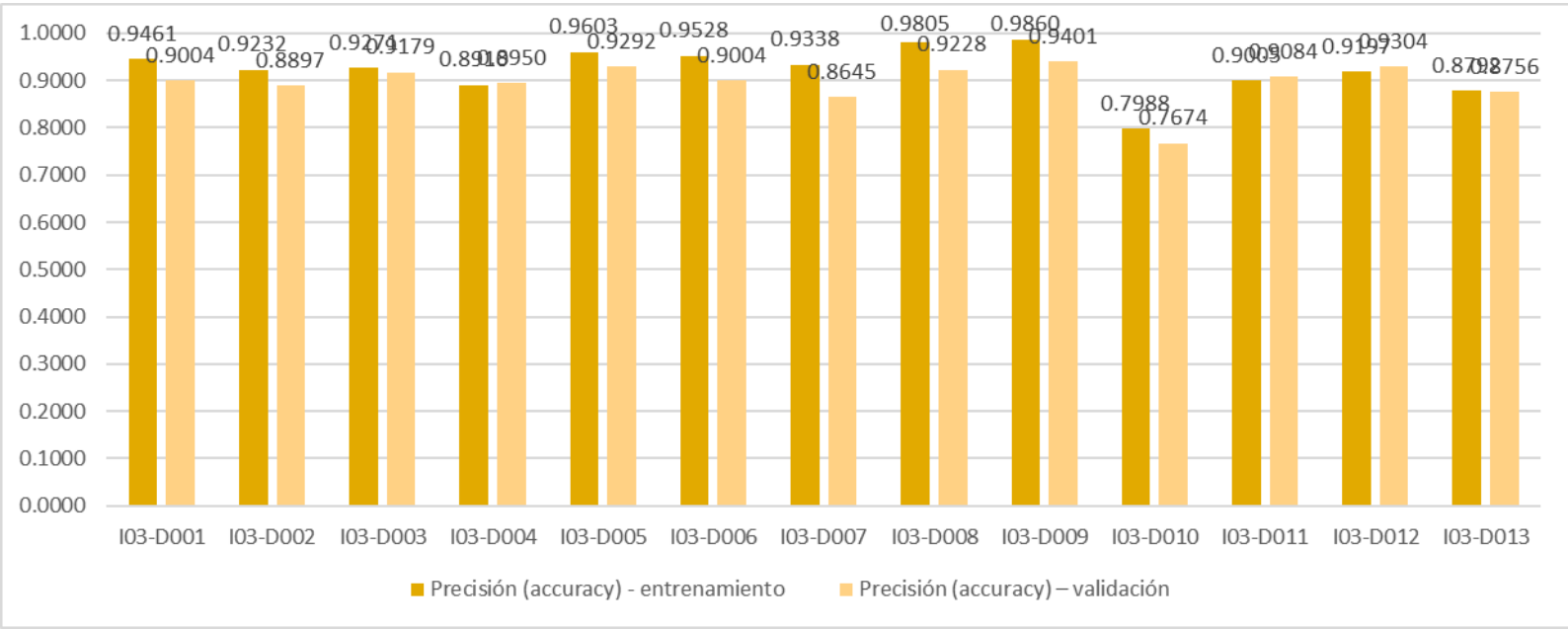
Diseño	Precisión (accuracy) - entrenamiento	Loss - entrenamiento	Precisión (accuracy) – validación	Loss - validación	Grado de error - entrenamiento	Grado de error - validación
I03-D001	0.9461	0.1393	0.9004	0.3012	0.0539	0.0996
I03-D002	0.9232	0.2199	0.8897	0.3035	0.0768	0.1103
I03-D003	0.9271	0.1897	0.9179	0.2380	0.0729	0.0821
I03-D004	0.8910	0.2920	0.8950	0.2905	0.1090	0.1050
I03-D005	0.9603	0.2578	0.9292	0.3715	0.0397	0.0708
I03-D006	0.9528	0.3144	0.9004	0.5192	0.0472	0.0996
I03-D007	0.9338	0.3400	0.8645	0.5849	0.0662	0.1355
I03-D008	0.9805	0.1166	0.9228	0.3590	0.0195	0.0772
I03-D009	0.9860	0.1161	0.9401	0.3075	0.0140	0.0599
I03-D010	0.7988	0.5025	0.7674	0.6875	0.2012	0.2326
I03-D011	0.9005	0.4372	0.9084	0.4269	0.0995	0.0916
I03-D012	0.9197	0.4441	0.9304	0.4294	0.0803	0.0696

I03-D013	0.8792	0.5937	0.8756	0.6029	0.1208	0.1244

Tabla 55
Resumen estadístico resultados de entrenamiento – iteración 03

Diseño	Precisión (accuracy) - entrenamiento	Precisión (accuracy) – validación	Grado de error - entrenamiento	Grado de error - validación
I03-D001	94.61%	90.04%	5.39%	9.96%
I03-D002	92.32%	88.97%	7.68%	11.03%
I03-D003	92.71%	91.79%	7.29%	8.21%
I03-D004	89.10%	89.50%	10.90%	10.50%
I03-D005	96.03%	92.92%	3.97%	7.08%
I03-D006	95.28%	90.04%	4.72%	9.96%
I03-D007	93.38%	86.45%	6.62%	13.55%
I03-D008	98.05%	92.28%	1.95%	7.72%
I03-D009	98.60%	94.01%	1.40%	5.99%
I03-D010	79.88%	76.74%	20.12%	23.26%
I03-D011	90.05%	90.84%	9.95%	9.16%
I03-D012	91.97%	93.04%	8.03%	6.96%
I03-D013	87.92%	87.56%	12.08%	12.44%

Figura 65
Grafico de barras - accuracy - iteración 03



3.4.13.1. Análisis de Resultados

Primero, analizando los modelos D001-D003 donde extendemos los ciclos para poder observar si el overfitting se presenta. Se observa que, en los 3 casos, el overfitting eventualmente se presenta. En los tres casos se puede observar que cuando el modelo ya entra a un 90% de presión en entrenamiento, ya hay overfitting. El caso en que el overfitting se presenta más tarde es en el modelo I03-D003.

Figura 66
Gráfico de Accuracy y Loss – I003 - D001

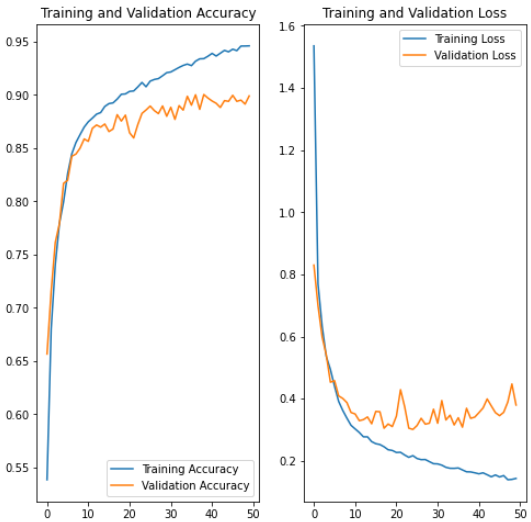


Figura 67
Grafico de Accuracy y Loss - I003 - D002

Figura 68
Grafico de Accuracy y Loss - I003 - D003

Se puede observar que el uso de las regularizaciones es efectivo según el valor que se le da. El caso en el cual se puede observar más efecto es con el valor 0.01 donde se puede observar que estabilizamos el overfitting en los tres modelos. El problema actual es que los modelos no están llegando a niveles de presión atan altos. El mejor sería el modelo I03-D012 el cual llego a una precisión de 93.04% .

3.4.13.2. Decisión de toma de acción

La siguiente iteración será la I04. Las variaciones que se manejarán serán:

1. Vamos a usar la estrategia de modelos con dropout. Para esta opción, vamos a usar los modelos anteriores que tuvieron el mayor índice de precisión. DS02-

D004, DS02-D005 y DS02-D006.Aumentaremos a 20 ciclos de entrenamiento también. Estos serán los modelos I04-D001, I04-D002 y I04-D003. Valor de 0.25.

2. Vamos a usar la estrategia de modelos con dropout. Estos serán los modelos I04-D004, I04-D005 y I04-D006. Valor de 0.50
3. Vamos a usar la estrategia de modelos con dropout. Estos serán los modelos I04-D007, I04-D008 y I04-D009. Valor de 0.75

3.4.14. Iteración 04 – Modelos de CNN a probar

Para todos los procesos de entrenamiento se utilizará el dataset D002. Y los modelos que se entrenarán tendrán las siguientes características:

Tabla 56
Resumen de esquema de modelos - iteración 04

Código de diseño	Numero de neuronas	Numero de ciclos	Numero de capas ocultas	Función de Activación	Dropout
I04-D001	77	15	8	RELU	0.25
I04-D002	245	15	8	RELU	0.25
I04-D003	61	15	6	RELU	0.25
I04-D004	77	15	8	RELU	0.50
I04-D005	245	15	8	RELU	0.50
I04-D006	61	15	6	RELU	0.50
I04-D007	77	15	8	RELU	0.75
I04-D008	245	15	8	RELU	0.75
I04-D009	61	15	6	RELU	0.75

3.4.15. Iteración 04 – entrenamiento y resultados por modelo

3.4.15.1. Resultados de entrenamiento de I04-D001

Tabla 57
Resultados de entrenamiento - iteración 04 - DS02 - D001

Diseño	I04-D001
Precisión (accuracy) - entrenamiento	0.8697
Loss - entrenamiento	0.3431
Precisión (accuracy) – validación	0.8610
Loss - validación	0.3875
Grado de error - entrenamiento	0.1303
Grado de error - validación	0.1390

3.4.15.2. Resultados de entrenamiento de I04-D002

Tabla 58
Resultados de entrenamiento - iteración 04 - DS02 - D002

Diseño	I04-D002
Precisión (accuracy) - entrenamiento	0.9421
Loss - entrenamiento	0.1548
Precisión (accuracy) – validación	0.9263
Loss - validación	0.2012
Grado de error - entrenamiento	0.0579
Grado de error - validación	0.0737

3.4.15.3. Resultados de entrenamiento de I04-D003

Tabla 59
Resultados de entrenamiento - iteración 04 - DS02 - D003

Diseño	I04-D003
Precisión (accuracy) - entrenamiento	0.9203
Loss - entrenamiento	0.2377
Precisión (accuracy) – validación	0.8962
Loss - validación	0.3371
Grado de error - entrenamiento	0.0797
Grado de error - validación	0.1038

3.4.15.4. Resultados de entrenamiento de I04-D004

Tabla 60
Resultados de entrenamiento - iteración 04 - DS02 - D004

Diseño	I04-D004
Precisión (accuracy) - entrenamiento	0.3270
Loss - entrenamiento	1.5555
Precisión (accuracy) – validación	0.3240
Loss - validación	1.5341
Grado de error - entrenamiento	0.6730
Grado de error - validación	0.6760

3.4.15.5. Resultados de entrenamiento de I04-D005

Tabla 61
Resultados de entrenamiento - iteración 04 - DS02 - D005

Diseño	I04-D005
Precisión (accuracy) - entrenamiento	0.5361
Loss - entrenamiento	1.1382
Precisión (accuracy) – validación	0.4747
Loss - validación	1.2793
Grado de error - entrenamiento	0.4639
Grado de error - validación	0.5253

3.4.15.6. Resultados de entrenamiento de I04-D006

Tabla 62
Resultados de entrenamiento - iteración 04 - DS02 - D006

Diseño	I04-D006
Precisión (accuracy) - entrenamiento	0.3270
Loss - entrenamiento	1.5555
Precisión (accuracy) – validación	0.3240
Loss - validación	1.5574
Grado de error - entrenamiento	0.6730
Grado de error - validación	0.6760

3.4.15.7. Resultados de entrenamiento de I04-D007

Tabla 63
Resultados de entrenamiento - iteración 04 - DS02 - D007

Diseño	I04-D007
Precisión (accuracy) - entrenamiento	0.3270
Loss - entrenamiento	1.5555
Precisión (accuracy) – validación	0.3240
Loss - validación	1.5574
Grado de error - entrenamiento	0.6730
Grado de error - validación	0.6760

3.4.15.8. Resultados de entrenamiento de I04-D008

Tabla 64
Resultados de entrenamiento - iteración 04 - DS02 - D008

Diseño	I04-D008
Precisión (accuracy) - entrenamiento	0.7879
Loss - entrenamiento	0.5191
Precisión (accuracy) – validación	0.6377
Loss - validación	1.0580
Grado de error - entrenamiento	0.2121
Grado de error - validación	0.3623

3.4.15.9. Resultados de entrenamiento de I04-D009

Tabla 65
Resultados de entrenamiento - iteración 04 - DS02 - D009

Diseño	I04-D009
Precisión (accuracy) - entrenamiento	0.3270
Loss - entrenamiento	1.5554
Precisión (accuracy) – validación	0.3240
Loss - validación	1.5574
Grado de error - entrenamiento	0.6730
Grado de error - validación	0.6760

3.4.16. Resultados de entrenamiento Iteración 04 con los modelos D001 – D009

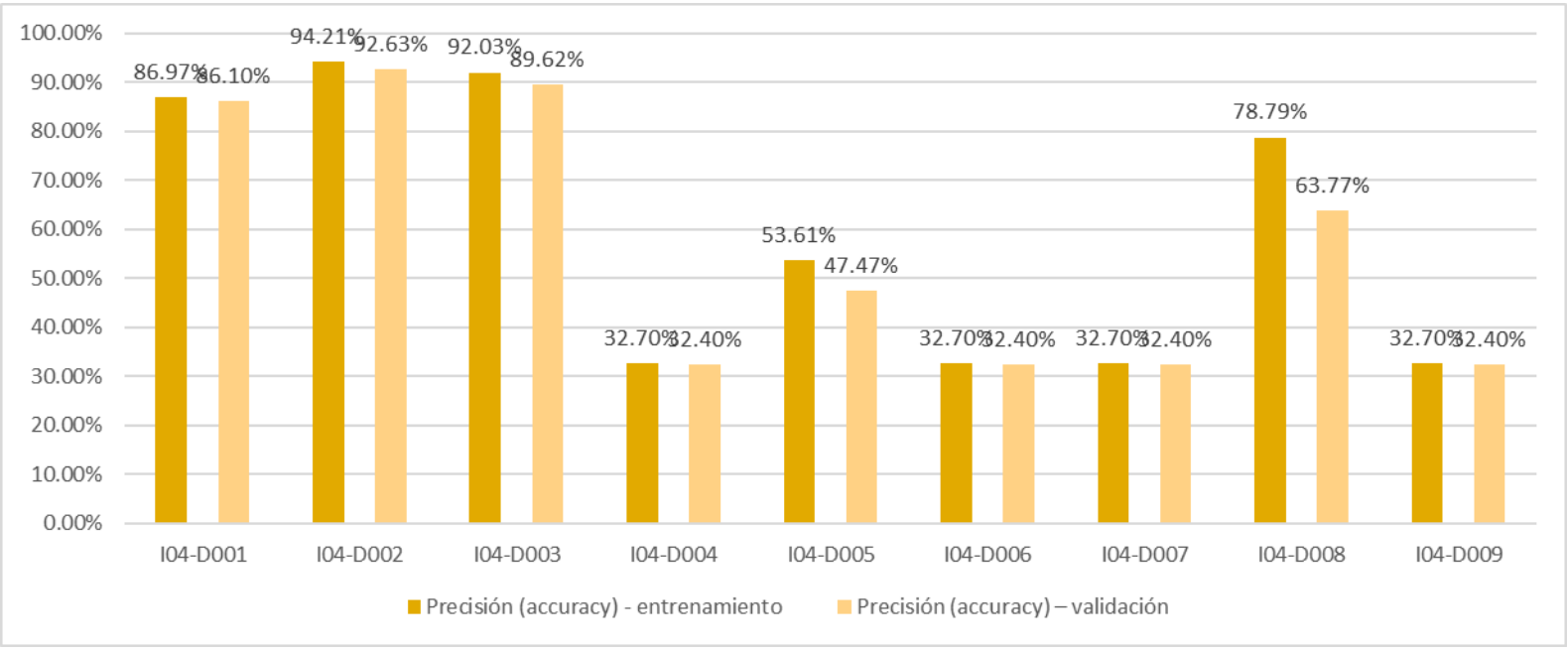
Tabla 66
Resumen resultados de entrenamiento – iteración 04

Diseño	Precisión (accuracy) - entrenamiento	Loss - entrenamiento	Precisión (accuracy) – validación	Loss - validación	Grado de error - entrenamiento	Grado de error - validación
I04-D001	0.8697	0.3431	0.8610	0.3875	0.1303	0.1390
I04-D002	0.9421	0.1548	0.9263	0.2012	0.0579	0.0737
I04-D003	0.9203	0.2377	0.8962	0.3371	0.0797	0.1038
I04-D004	0.3270	1.5555	0.3240	1.5341	0.6730	0.6760
I04-D005	0.5361	1.1382	0.4747	1.2793	0.4639	0.5253
I04-D006	0.3270	1.5555	0.3240	1.5574	0.6730	0.6760
I04-D007	0.3270	1.5555	0.3240	1.5574	0.6730	0.6760
I04-D008	0.7879	0.5191	0.6377	1.0580	0.2121	0.3623
I04-D009	0.3270	1.5554	0.3240	1.5574	0.6730	0.6760

Tabla 67
Resumen estadístico resultados de entrenamiento – iteración 04

Diseño	Precisión (accuracy) - entrenamiento	Precisión (accuracy) – validación	Grado de error - entrenamiento	Grado de error - validación
I04-D001	86.97%	86.10%	13.03%	13.90%
I04-D002	94.21%	92.63%	5.79%	7.37%
I04-D003	92.03%	89.62%	7.97%	10.38%
I04-D004	32.70%	32.40%	67.30%	67.60%
I04-D005	53.61%	47.47%	46.39%	52.53%
I04-D006	32.70%	32.40%	67.30%	67.60%
I04-D007	32.70%	32.40%	67.30%	67.60%
I04-D008	78.79%	63.77%	21.21%	36.23%
I04-D009	32.70%	32.40%	67.30%	67.60%

Figura 69
Grafico de barras - accuracy - iteración 04



3.4.16.1. Análisis de Resultados

Problema; mientras más porcentaje de neuronas desactivemos, menos probabilidad de que nuestra red empiece a converger/aprender. En el caso de usar un dropout de 50% o 75%, las redes son incapaces de resolver nuestro problema con tan pocas neuronas. Cuando una red se estanca, significa que en cada proceso de backpropagation se suben o

disminuyen los pesos en un intento de mejorar la red. No es capaz encontrar de encontrar una pendiente en el error, es por lo que los pesos siempre vuelven al mismo valor.

Problema, usar dropout siempre retrasara el aprendizaje a nivel de ciclos. Se observo que en los casos donde se logró el entrenamiento con niveles mayores de dropout del 25%, se redujo el nivel de precisión obtenida por cada ciclo.

Con un nivel de dropout de 25% se observa una gran mejora en evitar el overfitting. La mejora de overfitting se debe a que, en cada paso de información de entrada a una capa siguiente, se elimina el 25% de las conexiones con la siguiente capa para de esta forma la red no se concentre en una única característica y que este constantemente adaptándose.

El modelo con más neuronas es el modelo con más capacidad de converger a nivel que el overfitting aumenta. Observando los resultados, podemos car la conclusión de que el número de neuronas es importante para el uso de esta estrategia.

3.4.16.2. Decisión de toma de acción

La siguiente iteración será la I04. Las variaciones que se manejarán serán:

1. Los modelos I04-D001, I04-D002 y I04-D003 se volverán a evaluar con un rango de ciclos más grande, en este caso será 35 ciclos. Estos serán los modelos I05-D001, I05-D002 y I05-D003.
2. Probaremos el modelo I04-D002 con un nivel de regularización de 0.001. Este será el modelo I05-D004.
3. Probaremos el modelo I04-D002 con un nivel de regularización de 0.01. Este será el modelo I05-D005.
4. Probaremos el modelo I04-D002 eliminando la capa de dropout previo a la segunda capa convolucional. Este será el modelo I05-D006.
5. Probaremos el modelo I05-D006 con un nivel de regularización de 0.01. Este será el modelo I05-D007.

3.4.17. Iteración 05 – Modelos de CNN a probar

Para todos los procesos de entrenamiento se utilizará el dataset D002. Y los modelos que se entrenarán tendrán las siguientes características:

Tabla 68
Resumen de esquema de modelos - iteración 05

Código de diseño	Numero de neuronas	Numero de ciclos	Numero de capas ocultas	Función de Activación	Dropout	Valor de Regularización
I05-D001	77	35	8	RELU	0.25	-
I05-D002	245	35	8	RELU	0.25	-
I05-D003	61	20	6	RELU	0.25	-
I05-D004	245	35	8	RELU	0.25	0.001
I05-D005	245	15	8	RELU	0.25	0.01
I05-D006	245	15	8	RELU	0.25	-
I05-D007		15	8	RELU	0.25	0.01
I05-D008	245	15	8	RELU	0.25	0.001

3.4.18. Iteración 05 – entrenamiento y resultados por modelo

3.4.18.1. Resultados de entrenamiento de I05-D001

Tabla 69
Resultados de entrenamiento - iteración 05 - DS02 - D001

Diseño	I05-D001
Precisión (accuracy) - entrenamiento	0.9554
Loss - entrenamiento	0.1275
Precisión (accuracy) – validación	0.9101
Loss - validación	0.2472
Grado de error - entrenamiento	0.0446

Grado de error - validación	0.0899

3.4.18.2. Resultados de entrenamiento de I05-D002

Tabla 70
Resultados de entrenamiento - iteración 05 - DS02 - D002

Diseño	I05-D002
Precisión (accuracy) - entrenamiento	0.9740
Loss - entrenamiento	0.0885
Precisión (accuracy) – validación	0.9495
Loss - validación	0.1820
Grado de error - entrenamiento	0.0260
Grado de error - validación	0.0505

3.4.18.3. Resultados de entrenamiento de I05-D003

Tabla 71
Resultados de entrenamiento - iteración 05 - DS02 - D003

Diseño	I05-D003
Precisión (accuracy) - entrenamiento	0.9441
Loss - entrenamiento	0.1693
Precisión (accuracy) – validación	0.8839
Loss - validación	0.3453
Grado de error - entrenamiento	0.0559
Grado de error - validación	0.1161

3.4.18.4. Resultados de entrenamiento de I05-D004

Tabla 72
Resultados de entrenamiento - iteración 05 - DS02 - D004

Diseño	I05-D004
Precisión (accuracy) - entrenamiento	0.9649
Loss - entrenamiento	0.2618
Precisión (accuracy) – validación	0.9496
Loss - validación	0.3139
Grado de error - entrenamiento	0.0351
Grado de error - validación	0.0504

3.4.18.5. Resultados de entrenamiento de I05-D005

Tabla 73
Resultados de entrenamiento - iteración 05 - DS02 - D005

Diseño	I05-D005
Precisión (accuracy) - entrenamiento	0.8853
Loss - entrenamiento	0.5457
Precisión (accuracy) – validación	0.8983
Loss - validación	0.5186
Grado de error - entrenamiento	0.1147
Grado de error - validación	0.1017

3.4.18.6. Resultados de entrenamiento de I05-D006

Tabla 74
Resultados de entrenamiento - iteración 05 - DS02 - D006

Diseño	I05-D006
Precisión (accuracy) - entrenamiento	0.9641
Loss - entrenamiento	0.1092
Precisión (accuracy) – validación	0.9469
Loss - validación	0.1659
Grado de error - entrenamiento	0.0359
Grado de error - validación	0.0531

3.4.18.7. Resultados de entrenamiento de I05-D007

Tabla 75
Resultados de entrenamiento - iteración 05 - DS02 - D007

Diseño	I05-D007
Precisión (accuracy) - entrenamiento	0.8888
Loss - entrenamiento	0.5486
Precisión (accuracy) – validación	0.9096
Loss - validación	0.4953
Grado de error - entrenamiento	0.1112
Grado de error - validación	0.0904

3.4.18.8. Resultados de entrenamiento de I05-D008

Tabla 76
Resultados de entrenamiento - iteración 05 - DS02 - D008

Diseño	I05-D008
Precisión (accuracy) - entrenamiento	0.9369
Loss - entrenamiento	0.3862
Precisión (accuracy) – validación	0.9441
Loss - validación	0.3802
Grado de error - entrenamiento	0.0631
Grado de error - validación	0.0559

3.4.19. Resultados de entrenamiento Iteración 05 con los modelos D001 – D009

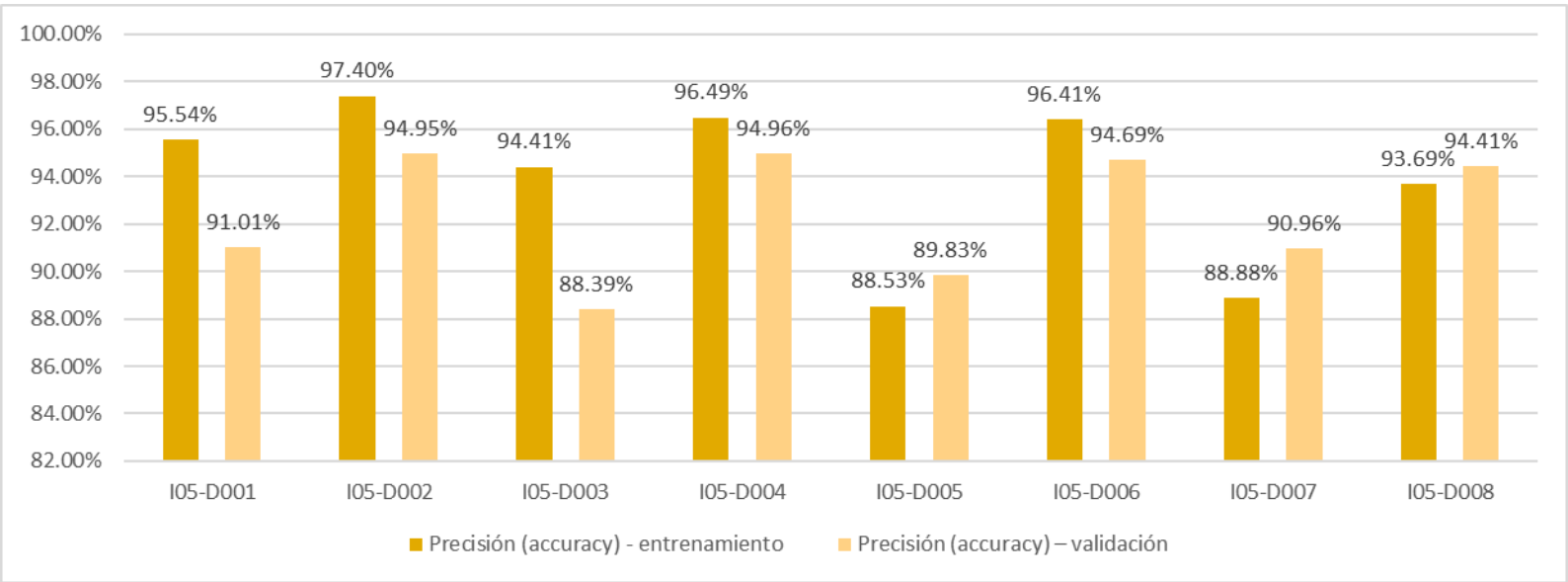
Tabla 77
Resumen resultados de entrenamiento – iteración 05

Diseño	Precisión (accuracy) - entrenamiento	Loss - entrenamiento	Precisión (accuracy) – validación	Loss - validación	Grado de error - entrenamiento	Grado de error - validación
I05-D001	0.9554	0.1275	0.9101	0.2472	0.0446	0.0899
I05-D002	0.9740	0.0885	0.9495	0.1820	0.0260	0.0505
I05-D003	0.9441	0.1693	0.8839	0.3453	0.0559	0.1161
I05-D004	0.9649	0.2618	0.9496	0.3139	0.0351	0.0504
I05-D005	0.8853	0.5457	0.8983	0.5186	0.1147	0.1017
I05-D006	0.9641	0.1092	0.9469	0.1659	0.0359	0.0531
I05-D007	0.8888	0.5486	0.9096	0.4953	0.1112	0.0904
I05-D008	0.9369	0.3862	0.9441	0.3802	0.0631	0.0559

Tabla 78
Resumen estadístico resultados de entrenamiento – iteración 05

Diseño	Precisión (accuracy) - entrenamiento	Precisión (accuracy) – validación	Grado de error - entrenamiento	Grado de error - validación
I05-D001	95.54%	91.01%	4.46%	8.99%
I05-D002	97.40%	94.95%	2.60%	5.05%
I05-D003	94.41%	88.39%	5.59%	11.61%
I05-D004	96.49%	94.96%	3.51%	5.04%
I05-D005	88.53%	89.83%	11.47%	10.17%
I05-D006	96.41%	94.69%	3.59%	5.31%
I05-D007	88.88%	90.96%	11.12%	9.04%
I05-D008	93.69%	94.41%	6.31%	5.59%

Figura 70
Grafico de barras - accuracy - iteración 05



3.4.19.1. Análisis de Resultados

Problema; el uso excesivo de técnicas para tratar el overfitting no permite que nuestros modelos alcancen grandes niveles de precisión. Tenemos que jugar reduciendo las técnicas encontrando un punto aceptable.

Se puede observar que cuando su usan valores de regularización altos, el valor loss no reduce tanto como con valores de regularización más bajos.

3.4.19.2. Decisión de toma de acción

La siguiente iteración será la I06. En la iteración 06 se procederá a evaluar cada modelo y realizar cambios de uno a uno con la intención de seleccionar el modelo final.

3.4.20. Iteración 06 – entrenamiento y resultados por modelo

3.4.20.1. Entrenamiento de I06-D001

- **Razón de modelo**

Como se pudo observar, el modelo con mayor número de neuronas es el modelo con mayor probabilidad de no estancarse por la técnica de dropout. Empezamos con un valor de regularización alto para poder tener un punto de comparación con respecto a que tanto baja ellos.

- **Esquema utilizado**

Tabla 79
Resumen de esquema de modelos - iteración 06 - D001

Código de diseño	Numero de neuronas	Numero de ciclos	Numero de capas ocultas	Función de Activación	Dropout	Valor de Regularización
I06-D001	245	30	8	RELU	0.25	0.01

- **Resultados obtenidos**

Tabla 80
Resultados de entrenamiento - iteración 06 - DS02 - D001

Diseño	I06-D001
Precisión (accuracy) - entrenamiento	0.9106
Loss - entrenamiento	0.4731
Precisión (accuracy) – validación	0.9303
Loss - validación	0.4286
Grado de error - entrenamiento	0.0894
Grado de error - validación	0.0697

3.4.20.2. Entrenamiento de I06-D002

- **Razón de modelo**

Reducimos el valor de regularización a 0.001 para observar si es posible reducir más el valor loss.

- **Esquema utilizado**

Tabla 81
Resumen de esquema de modelos - iteración 06 - D002

Código de diseño	Numero de neuronas	Numero de ciclos	Numero de capas ocultas	Función de Activación	Dropout	Valor de Regularización
I06-D002	245	30	8	RELU	0.25	0.001

- **Resultados obtenidos**

Tabla 82
Resultados de entrenamiento - iteración 06 - DS02 - D002

Diseño	I06-D002
Precisión (accuracy) - entrenamiento	0.9494
Loss - entrenamiento	0.3270
Precisión (accuracy) – validación	0.9522
Loss - validación	0.3168
Grado de error - entrenamiento	0.0506
Grado de error - validación	0.0478

3.4.20.3. Entrenamiento de I06-D003

- **Razón de modelo**

Reducimos el valor de regularización a 0.0001 para observar si es posible reducir más el valor loss.

- **Esquema utilizado**

Tabla 83
Resumen de esquema de modelos - iteración 06 - D003

Código de diseño	Numero de neuronas	Numero de ciclos	Numero de capas ocultas	Función de Activación	Dropout	Valor de Regularización
I06-D003	245	30	8	RELU	0.25	0.0001

- **Resultados obtenidos**

Tabla 84
Resultados de entrenamiento - iteración 06 - DS02 - D003

Diseño	I06-D003
Precisión (accuracy) - entrenamiento	0.9732
Loss - entrenamiento	0.2527
Precisión (accuracy) – validación	0.9599
Loss - validación	0.2993
Grado de error - entrenamiento	0.0268
Grado de error - validación	0.0401

3.4.20.4. Entrenamiento de I06-D004

- **Razón de modelo**

Probemos aumentando el valor de regularización a 0.0005, esto para poder comprar si realmente se puede observar una subida del Loss.

- **Esquema utilizado**

Tabla 85
Resumen de esquema de modelos - iteración 06 - D004

Código de diseño	Numero de neuronas	Numero de ciclos	Numero de capas ocultas	Función de Activación	Dropout	Valor de Regularización
I06-D004	245	30	8	RELU	0.25	0.0005

- **Resultados obtenidos**

Tabla 86
Resultados de entrenamiento - iteración 06 - DS02 - D004

Diseño	I06-D004
Precisión (accuracy) - entrenamiento	0.9562
Loss - entrenamiento	0.3146
Precisión (accuracy) – validación	0.9582
Loss - validación	0.3153
Grado de error - entrenamiento	0.0438
Grado de error - validación	0.0418

3.4.20.5. Entrenamiento de I06-D005

- **Razón de modelo**

Reducimos el valor de regularización a 0.00005 para observar si es posible reducir más el valor loss. En el entrenamiento de I06-D005 se pudo observar una clara subida relacionada a la subida del valor de regularización.

- **Esquema utilizado**

Tabla 87
Resumen de esquema de modelos - iteración 06 - D005

Código de diseño	Numero de neuronas	Numero de ciclos	Numero de capas ocultas	Función de Activación	Dropout	Valor de Regularización
I06-D005	245	30	8	RELU	0.25	0.00005

- **Resultados obtenidos**

Tabla 88
Resultados de entrenamiento - iteración 06 - DS02 - D005

Diseño	I06-D005
Precisión (accuracy) - entrenamiento	0.9708
Loss - entrenamiento	0.2334
Precisión (accuracy) – validación	0.9541
Loss - validación	0.2937
Grado de error - entrenamiento	0.0292
Grado de error - validación	0.0459

3.4.20.6. Entrenamiento de I06-D006

- **Razón de modelo**

En el entrenamiento de I06-D005 se redujo el Loss, pero también se vio un gran nivel de variabilidad en cada ciclo después del ciclo 10 al 15. Vamos a probar reducir el número de ciclos para ajustar el entrenamiento y reducir el nivel de overfitting.

- **Esquema utilizado**

Tabla 89
Resumen de esquema de modelos - iteración 06 - D006

Código de diseño	Numero de neuronas	Numero de ciclos	Numero de capas ocultas	Función de Activación	Dropout	Valor de Regularización
I06-D006	245	12	8	RELU	0.25	0.00005

- **Resultados obtenidos**

Tabla 90
Resultados de entrenamiento - iteración 06 - DS02 - D006

Diseño	I06-D006
Precisión (accuracy) - entrenamiento	0.9578
Loss - entrenamiento	0.2340
Precisión (accuracy) – validación	0.9462
Loss - validación	0.2656
Grado de error - entrenamiento	0.0422
Grado de error - validación	0.0538

3.4.20.7. Entrenamiento de I06-D007

- **Razón de modelo**

Reducimos el valor de regularización a 0.000025 para observar si es posible reducir más el valor loss.

- **Esquema utilizado**

Tabla 91
Resumen de esquema de modelos - iteración 06 - D007

Código de diseño	Numero de neuronas	Numero de ciclos	Numero de capas ocultas	Función de Activación	Dropout	Valor de Regularización
I06-D007	245	20	8	RELU	0.25	0.000025

- **Resultados obtenidos**

Tabla 92
Resultados de entrenamiento - iteración 06 - DS02 - D007

Diseño	I06-D007
Precisión (accuracy) - entrenamiento	0.9738
Loss - entrenamiento	0.1894
Precisión (accuracy) – validación	0.9517
Loss - validación	0.2558
Grado de error - entrenamiento	0.0262
Grado de error - validación	0.0483

3.4.20.8. Entrenamiento de I06-D008

- **Razón de modelo**

Reducimos el valor de regularización a 0.00004 para observar si es posible reducir más el valor loss.

- **Esquema utilizado**

Tabla 93
Resumen de esquema de modelos - iteración 06 - D008

Código de diseño	Numero de neuronas	Numero de ciclos	Numero de capas ocultas	Función de Activación	Dropout	Valor de Regularización
I06-D008	245	20	8	RELU	0.25	0.00004

- **Resultados obtenidos**

Tabla 94
Resultados de entrenamiento - iteración 06 - DS02 - D008

Diseño	I06-D008
Precisión (accuracy) - entrenamiento	0.9716
Loss - entrenamiento	0.2093
Precisión (accuracy) – validación	0.9576
Loss - validación	0.2537
Grado de error - entrenamiento	0.0284
Grado de error - validación	0.0424

3.4.20.9. Entrenamiento de I06-D009

- **Razón de modelo**

Reducimos el valor de regularización a 0.000035 para observar si es posible reducir más el valor loss.

- **Esquema utilizado**

Tabla 95
Resumen de esquema de modelos - iteración 06 - D009

Código de diseño	Numero de neuronas	Numero de ciclos	Numero de capas ocultas	Función de Activación	Dropout	Valor de Regularización
I06-D009	245	20	8	RELU	0.25	0.000035

- **Resultados obtenidos**

Tabla 96
Resultados de entrenamiento - iteración 06 - DS02 - D009

Diseño	I06-D009
Precisión (accuracy) - entrenamiento	0.9704
Loss - entrenamiento	0.2193
Precisión (accuracy) – validación	0.9559
Loss - validación	0.2723
Grado de error - entrenamiento	0.0296
Grado de error - validación	0.0441

3.4.21. Resultados de entrenamiento Iteración 06 con los modelos D001 – D009

Tabla 97
Resumen resultados de entrenamiento – iteración 06

Diseño	Precisión (accuracy) - entrenamiento	Loss - entrenamiento	Precisión (accuracy) – validación	Loss - validación	Grado de error - entrenamiento	Grado de error - validación	Ciclos
I06-D001	0.9106	0.4731	0.9303	0.4286	0.0894	0.0697	30
I06-D002	0.9494	0.3270	0.9522	0.3168	0.0506	0.0478	30
I06-D003	0.9732	0.2527	0.9599	0.2993	0.0268	0.0401	30
I06-D004	0.9562	0.3146	0.9582	0.3153	0.0438	0.0418	30
I06-D005	0.9708	0.2334	0.9541	0.2937	0.0292	0.0459	30
I06-D006	0.9578	0.2340	0.9462	0.2656	0.0422	0.0538	12
I06-D007	0.9738	0.1894	0.9517	0.2558	0.0262	0.0483	20
I06-D008	0.9716	0.2093	0.9576	0.2537	0.0284	0.0424	20
I06-D009	0.9704	0.2193	0.9559	0.2723	0.0296	0.0441	20

Tabla 98
Resumen estadístico resultados de entrenamiento – iteración 06

Diseño	Precisión (accuracy) - entrenamiento	Precisión (accuracy) – validación	Grado de error - entrenamiento	Grado de error - validación
I06-D001	91.06%	93.03%	8.94%	6.97%
I06-D002	94.94%	95.22%	5.06%	4.78%
I06-D003	97.32%	95.99%	2.68%	4.01%
I06-D004	95.62%	95.82%	4.38%	4.18%
I06-D005	97.08%	95.41%	2.92%	4.59%
I06-D006	95.78%	94.62%	4.22%	5.38%
I06-D007	97.38%	95.17%	2.62%	4.83%
I06-D008	97.16%	95.76%	2.84%	4.24%
I06-D009	97.04%	95.59%	2.96%	4.41%

Figura 71
Comparación del mínimo Loss alcanzado según cada modelo de la Iteración 06

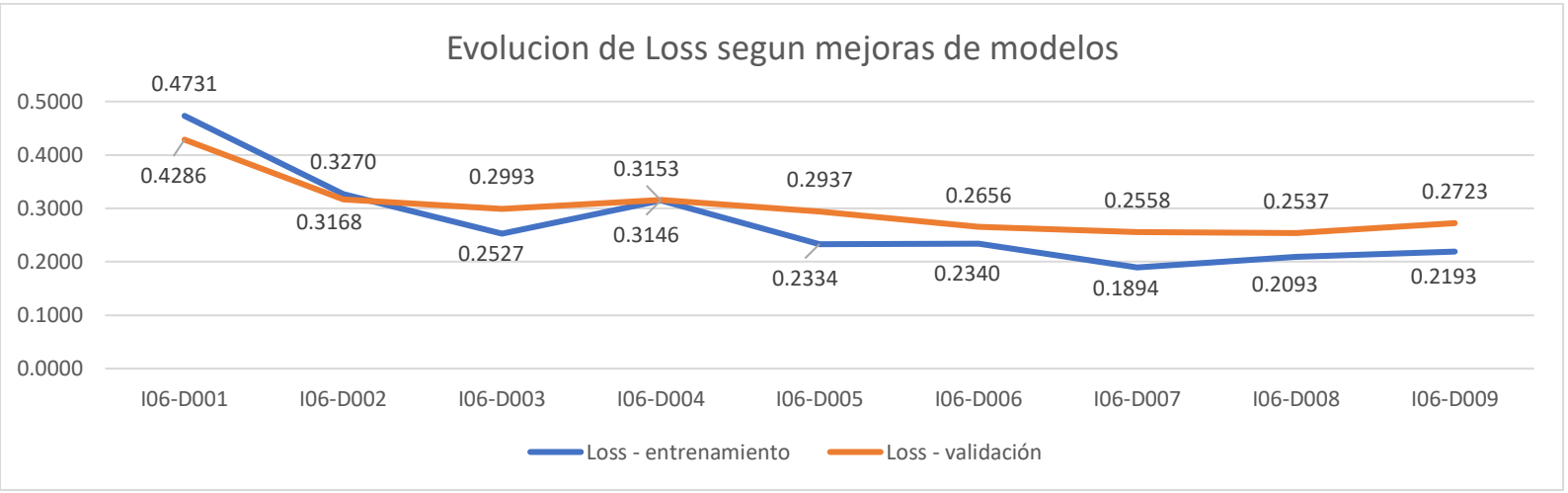
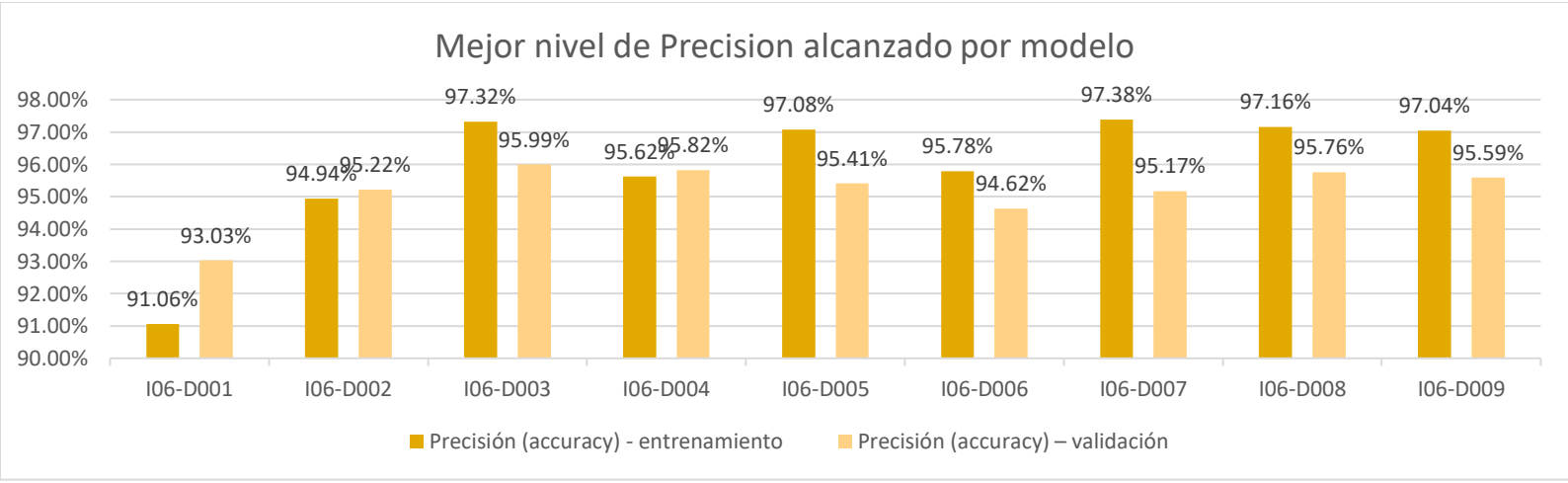


Figura 72
Grafico de barras del máximo valor alcanzado según cada modelo de la Iteración 06



3.4.21.1. Análisis de Resultados

Según lo observado en los gráficos.

- Se puede afirmar que la estrategia de mantener un valor de perdida en 25% y de bajar poco a poco el nivel de regularización si es efectiva con respecto a mejorar la bajada del valor de Loss.
- En el gráfico de Loss, el mínimo valor de Loss de validación alcanzado fue con el modelo I06-D008.
- En el gráfico de Loss, el mínimo valor de Loss de entrenamiento alcanzado fue con el modelo I06-D007.
- En el gráfico de precisión, el modelo I06-D007 tiene 2.21% de diferencia entre su precisión de entrenamiento y de validación.
- En el gráfico de precisión, el modelo I06-D008 tiene 1.4% de diferencia entre su precisión de entrenamiento y de validación.

Figura 73
Precisión y Loss del entrenamiento del modelo I06-D008 – prueba 1

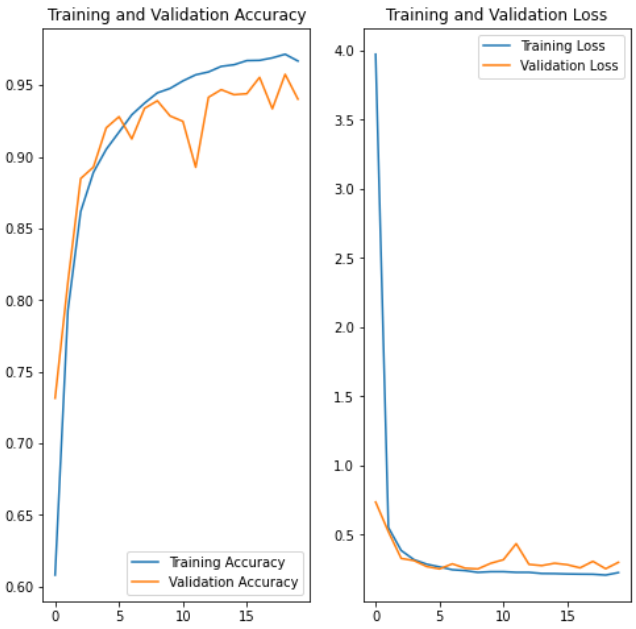
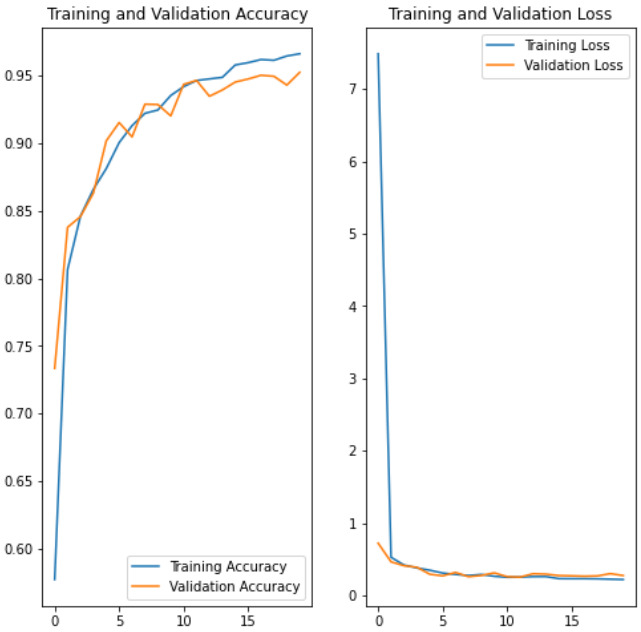


Figura 74
Precisión y Loss del entrenamiento del modelo I06-D008 – prueba 2

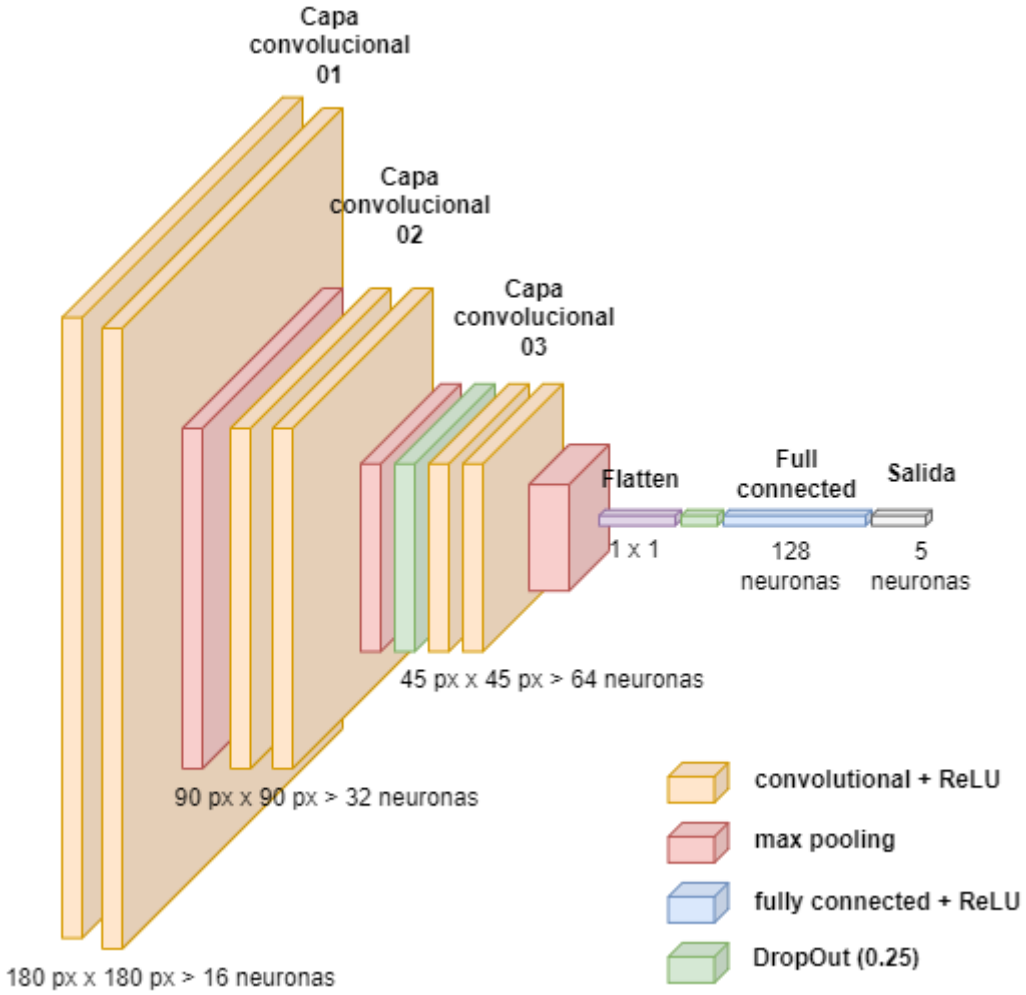


3.4.22. Decisión de modelo final

El modelo final escogido será el modelo I06-D008 por las siguientes razones:

- Cumple con nuestro máximo grado de error necesario el cual es del 5%. O con una precisión mínima del 95%.
- Es el modelo que alcanzo el menor valor de Loss en las pruebas de iteración 06 sin necesidad de un numero de ciclos mayor a 20.
- Es el que tiene la menor diferencia entre la precisión del entrenamiento y de validación. Esta es un gran indicio de no tender al overfitting en un corto número de ciclos.

Figura 75 *Diagrama de modelo final - modelo I06-D008*



3.4.23. Entrenamiento y generación de modelo final

El modelo final esta seleccionado. Ahora tenemos que entrenarlo hasta el punto de que nos interesa y tener un modelo entrenado con el cual podamos realizar predicciones. Este proceso se realizará con los siguientes pasos:

1. Definimos el esquema del modelo.

```
from tensorflow.keras import regularizers
#------------------------------------------------------------------------------
num_classes = 5
regularizer = 0.00004

model_I06_D008_T3 = Sequential([
  layers.Conv2D (16, 3, padding='same', activation='relu',kernel_regularizer=regularizer
s.l2(regularizer)),
  layers.MaxPooling2D(),
  layers.Conv2D(32, 3, padding='same', activation='relu',kernel_regularizer=regularizers
.l2(regularizer)),
  layers.MaxPooling2D(),
  layers.Dropout(0.25),
  layers.Conv2D(64, 3, padding='same', activation='relu',kernel_regularizer=regularizers
.l2(regularizer)),
  layers.MaxPooling2D(),
  layers.Flatten(),
  layers.Dropout(0.25),
  layers.Dense(128, activation='relu',kernel_regularizer=regularizers.l2(regularizer)),
  layers.Dense(num_classes)

])
model_I06_D008_T3.compile(
          optimizer='adam',
          loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
          metrics=['accuracy']
)
```

2. Entrenamos el modelo ciclo por ciclo realizando la validación de parar cuando nuestro modelo entrenado llegue al 95% de precisión en validación.

```
history_I06_D008_T3_array = []
x = True
i=-1
while x:
  i+=1
  history_I06_D008_T3_array.append(model_I06_D008_T3.fit(
    train_ds,
    validation_data=val_ds,
    epochs=1,
  ))
  if history_I06_D008_T3_array[i].history['val_accuracy'][0] >= 0.95:
    x = False
```

3. Juntamos todos los resultados obtenidos. Como estamos entrenado ciclo por ciclo, cada resultado se guarda como un tensor diferente con el resultado de ese único ciclo.

```
acc = []
val_acc = []
loss = []
val_loss = []
for try_history in history_I06_D008_T3_array:
  acc.append(try_history.history['accuracy'][0])
  val_acc.append(try_history.history['val_accuracy'][0])
  loss.append(try_history.history['loss'][0])
  val_loss.append(try_history.history['val_loss'][0])
```

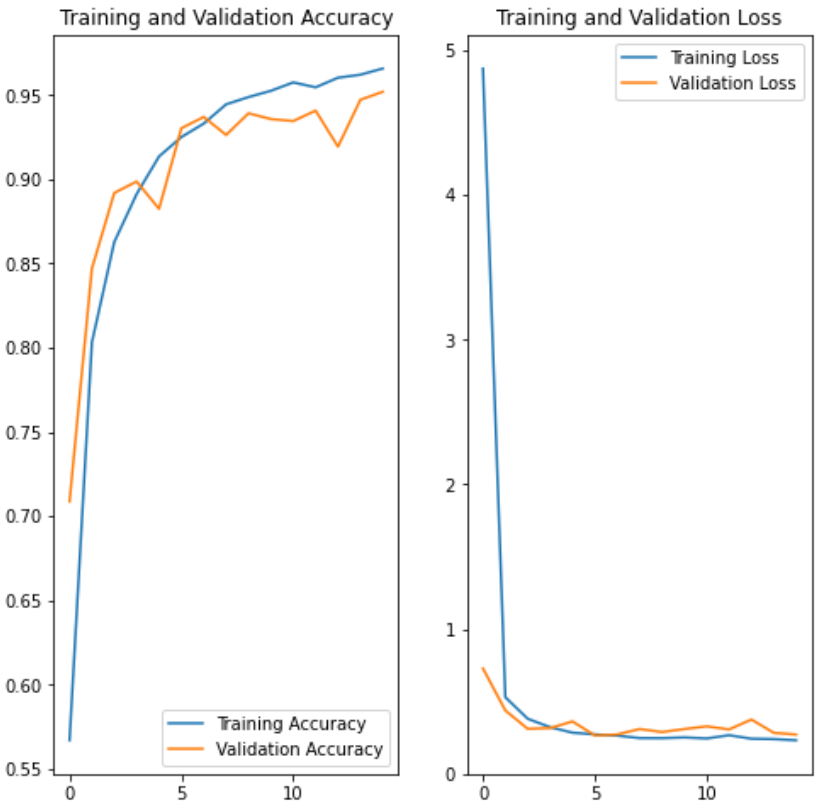
4. Obtenemos la información que nos es relevante y graficamos el diagrama de comparación de precisión y Loss.

```
epochs_range = leng(acc)

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```

Figura 76
Resultado de entrenamiento del modelo final



5. Guardamos el modelo entrenado para usarlo y publicarlo.

```
model_I06_D008_T3.save_weights('/content/drive/MyDrive/ML_tesis/Final_model/pesos/formato_ckpt/pesos.ckpt')
model_I06_D008_T3.save_weights('/content/drive/MyDrive/ML_tesis/Final_model/pesos/formato_hdf5/pesos.h5')

model_I06_D008_T3.save('/content/drive/MyDrive/ML_tesis/Final_model/modelo/formato_default/final_model')
model_I06_D008_T3.save('/content/drive/MyDrive/ML_tesis/Final_model/modelo/formato_hdf5/final_model.h5')
```

La información del modelo final seria.

Figura 77
Resumen de modelo final

```
In [ ]: model_I06_D008_T3.summary()

Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_6 (Conv2D)            (None, 180, 180, 16)      448
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 90, 90, 16)        0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 90, 90, 32)        4640
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 45, 45, 32)        0
_________________________________________________________________
dropout_4 (Dropout)          (None, 45, 45, 32)        0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 45, 45, 64)        18496
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 22, 22, 64)        0
_________________________________________________________________
flatten_2 (Flatten)          (None, 30976)             0
_________________________________________________________________
dropout_5 (Dropout)          (None, 30976)             0
_________________________________________________________________
dense_4 (Dense)              (None, 128)               3965056
_________________________________________________________________
dense_5 (Dense)              (None, 5)                 645
=================================================================
Total params: 3,989,285
Trainable params: 3,989,285
Non-trainable params: 0
_________________________________________________________________
```

3.4.24. Pruebas de predicción con modelo final

3.4.24.1. Preprocesado de imágenes para predicción

Para que el modelo pueda predecir el tipo de leucocito el tipo de leucocito según imágenes. Las imágenes tienen que estar en el mismo estado en el cual se hallaban las imágenes con el que se entrenó el modelo. El preprocesado es el siguiente:

1. Cargar el path donde se encuentra la imagen.
2. Escalar la imagen al mismo tamaño en el cual se manejó el dataset de entrenamiento.
3. Convertir imagen a un array.
4. Lo encapsulamos para simular que se encuentra en un batch para no tener problemas con el formato de entrada de un tensor de predicción.

```
path_img_test = '/content/drive/MyDrive/ML_tesis/TEST/img_test/bas_13.jpg'

img_original = keras.preprocessing.image.load_img(path_img_test)

img_height = 180
img_width = 180
img_reescaled = keras.preprocessing.image.load_img(path_img_test,target_size=(
img_height,img_width))
plt.imshow(img_reescaled)
plt.show()

img_array = keras.preprocessing.image.img_to_array(img_reescaled)

img_batch = np.expand_dims(img_array, axis=0)
```

Figura 78
Eosinófilo original

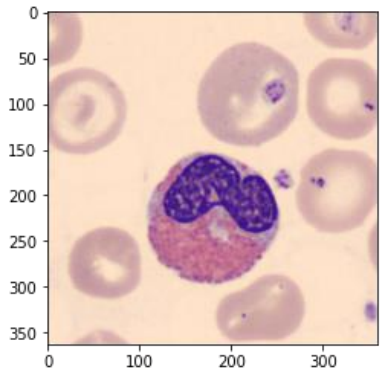


Figura 79
Linfocito original

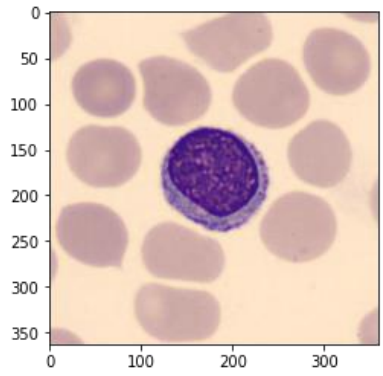


Figura 80
Monocito original

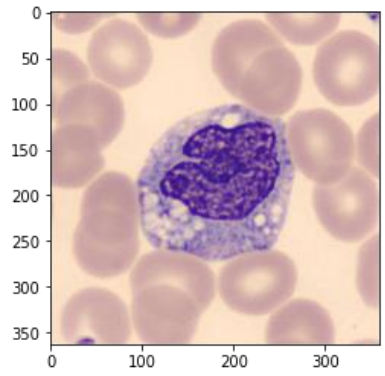


Figura 81
Neutrófilo original

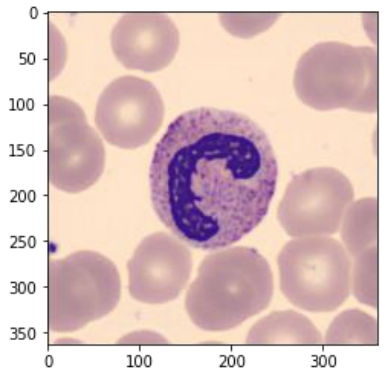


Figura 82
Basófilo original

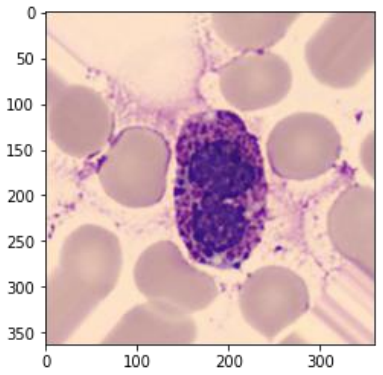


Figura 83
Eosinófilo preprocesado

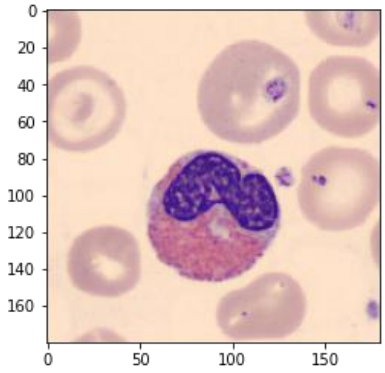


Figura 84
Linfocito preprocesado

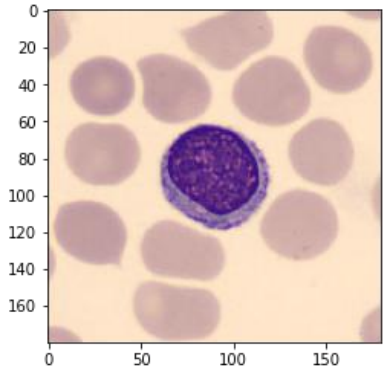


Figura 85
Monocito preprocesado

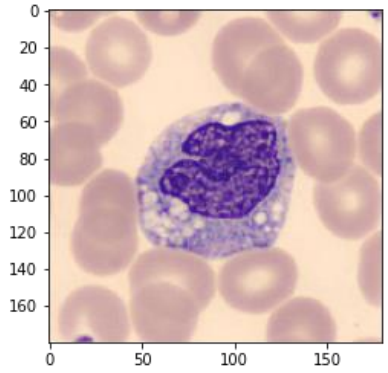


Figura 86
Neutrófilo preprocesado

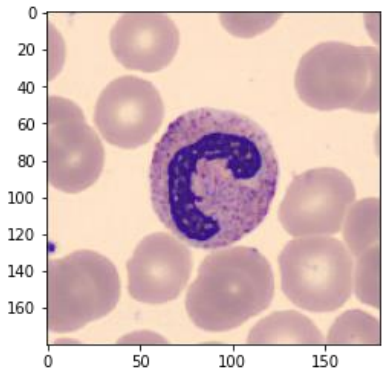
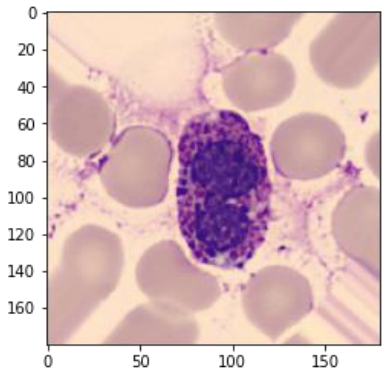


Figura 87
Basófilo preprocesado



3.4.24.2. Resultados de predicción

Una vez realizada la predicción de nuestras imágenes con el modelo. El resultado devuelto es un array de tamaño [1,5]. Ese array contiene número decimales.

El número significa el peso que tiene para ser cada tipo de leucocito. Si el número es negativo, se considera como cero. El que tenga el mayor peso es el que tiene la mayor probabilidad de ser.

Los pesos del array esta distribuidos según el orden alfabético de los tipos.

Tabla 99
Posición de peso de cada tipo de leucocito en array de resultado de predicción

Tipo de leucocito	Posición en array
Basófilo	0
Eosinófilo	1

Linfocito	2
Monocito	3
Neutrófilo	4

Figura 88
Resultados de predicción por imagen

```
# Basofilo
res
```

```
array([[39.970375, 21.090948, 12.707574, 24.327065, 18.685102]],
      dtype=float32)
```

```
# Eosinofilo
res
```

```
array([[ 1.4420302, 28.83935  , -2.7560425, -0.4367484, 11.175965 ]],
      dtype=float32)
```

```
# Lymfosito
res
```

```
array([[12.671475,  8.581212, 26.967503, 14.550474, 14.120594]],
      dtype=float32)
```

```
# Monosito
res
```

```
array([[ -9.8773775, -11.063689 ,  -6.167786 ,   4.3358784,  -3.5612743]],
      dtype=float32)
```

```
# Neutrofilo
res
```

```
array([[-4.835426 ,  2.5423229, -8.10254  , -2.9555664,  7.4468555]],
      dtype=float32)
```

Se observa que los resultados de cada imagen si cumplen con que el peso que le toca es el mayor. Pero hay algunas observaciones interesantes.

Primero, en casos como el basófilo. El resultado muestra pesos positivos en todos los tipos, esto significa que hay una probabilidad de que la imagen sea de cada tipo en una cantidad diferente.

También hay casos como en el monocito. El resultado muestra pesos negativos excepto con la posición del tipo de monocito. Parece ser que al modelo se le hace mucho más fácil identificar monocitos y descartar los demás tipos, que cuando tiene que evaluar otros tipos.

Si usamos un algoritmo para saber exactamente la probabilidad por tipo seria así:

Figura 89
Algoritmo para obtener probabilidad de resultados - Test Basófilo

```
# Basófilo
res = [39.970375, 21.090948, 12.707574, 24.327065, 18.685102]
tipos = ['Basófilo','Eosinófilo','Linfocito','Monocito','Neutrófilo']
tipos_positivos = 5
sum_total= 0
for tipo in res:
  if tipo < 0:
    tipos_positivos-=1
  else:
    sum_total+=tipo

for index in range(5):
  if res[index] > 0:
    print(f'{tipos[index]}: {res[index]*100/sum_total}')
```

```
Basófilo: 34.226760427529584
Eosinófilo: 18.06024647968613
Linfocito: 10.881536410731794
Monocito: 20.831343855541512
Neutrófilo: 16.000112826510986
```

Figura 90
Algoritmo para obtener probabilidad de resultados - Test Neutrófilo

```
# Neutrófilo
res = [-4.835426 ,  2.5423229, -8.10254  , -2.9555664,  7.4468555]
tipos = ['Basófilo','Eosinófilo','Linfocito','Monocito','Neutrófilo']
tipos_positivos = 5
sum_total= 0
for tipo in res:
  if tipo < 0:
    tipos_positivos-=1
  else:
    sum_total+=tipo

for index in range(5):
  if res[index] > 0:
    print(f'{tipos[index]}: {res[index]*100/sum_total}')
```

```
Eosinófilo: 25.450770806135566
Neutrófilo: 74.5492291938443
```

3.4.25. Generación de dataset de test

La generación del dataset de testeo fue realizado en base al dataset 02. El proceso para poder generar imágenes que nuestro modelo nunca vio fue de data augmentation. Se uso las estrategias de usar imágenes aleatorias según semilla, rotación de imágenes e inversión de imágenes. Con estas estrategias nos aseguramos de generar imágenes que nuestro modelo entrenado no uso en su proceso de entrenamiento.

```
# Vaiables
DIR = '/content/drive/MyDrive/ML/dataset-union/'
saveDir = '/content/drive/MyDrive/ML/dataset-test/'
n_muestras_x_tipo = 100

r.seed( 69 )
try:
  os.mkdir(saveDir)
except:
  print('Carpeta ya creada')

for subdir in os.listdir(DIR):
  saveSubDir = saveDir + subdir +'/'
  subdir_list = DIR + subdir
  subdir_list = os.listdir(subdir_list)
  imgSelected =r.sample(subdir_list,int(100))
  n=0
  try:
    os.mkdir(saveSubDir)
  except:
    print('Carpeta ya creada')
  for img in imgSelected:
    imgAllDir = DIR + subdir +'/'+ img
    try:
      img_loaded = load_img(imgAllDir)
    except:
      print('No es una imagen')
    data = img_to_array(img_loaded)
    samples = expand_dims(data, 0)

    datagen = ImageDataGenerator(rotation_range=360,horizon
tal_flip=True,vertical_flip=True)
    it = datagen.flow(samples, batch_size=1)

    batch = it.next()
    image = batch[0].astype('uint8')
    save = array_to_img(image)
    save.save(f'{saveSubDir}DS_test_{subdir}_{str(n)}.jpg')
    n+=1
    if (n%10==0):
      ia.imshow(image)
```

Al momento de crear el dataset, se reconoce las 500 imágenes generadas.

Figura 91
Conteo de muestras de dataset de testeo

```
img_height = 180
img_width = 180
test_ds = tf.keras.preprocessing.image_dataset_from_directory(
  '/content/drive/MyDrive/ML/dataset-test/',
  image_size=(img_height, img_width),
  batch_size=1
)
```

```
Found 500 files belonging to 5 classes.
```

3.4.26. Evaluación con dataset de test

La evaluación del modelo prueba cada imagen del dataset. Y devuelve los datos de Loss y precisión. La precisión obtenida de la evaluación es de 87%.

Figura 92
Uso de función de evaluación de los modelos de Tensorflow

```
eva = model.evaluate(test_ds, return_dict=True)
for name, value in eva.items():
  print(f"{name}: {value:.4f}")
```

```
500/500 [==============================] - 2s 4ms/step - loss: 0.8709 - accuracy: 0.8700
loss: 0.8709
accuracy: 0.8700
```

3.4.27. Evaluación continua por ciclos

3.4.27.1. Descripción de evaluación

Usando código del paso anterior, vamos a desarrollar un bucle con el objetivo inicial de encontrar si con algún dataset generado podemos alcanzar un “True Positive Rate” del 90%. Luego de 129 intentos desarrollados se llegó a la conclusión de que este objetivo no es posible.

Sabiendo su objetivo, ahora describiremos el bucle. Antes del bucle, se definen las variables y constantes para el correcto funcionamiento del bucle. El funcionamiento del bucle es:

1. Generamos las imágenes del dataset. Se utiliza el mismo código usado en la primera generación, pero encapsulada en una función. La función tiene los parámetros de seed (número para la obtención de las imágenes del dataset original), path del dataset original y path destino.

2. Usando la librería de keras, creamos un tensor que almacene las imágenes y su clase.
3. Evaluamos el modelo.
4. Guardamos los datos obtenidos en una hoja de cálculo de Google drive (previamente se tuvo que realizar las conexiones pertinentes con la API de Google cloud).
5. Revisamos si el objetivo de alcanzar el 90% de precisión se logró. Si no, se reinicia el bucle.

```
seguimos_buscando = True
img_height = 180
img_width = 180
worksheet = gc.open('Validacion con DS_test - resultados').sheet
1
worksheet.update_cell(1,1,'N de prueba')
worksheet.update_cell(1,2,'Seed')
worksheet.update_cell(1,3,'Loss')
worksheet.update_cell(1,4,'Accuracy')

i=0
while seguimos_buscando:
  i+=1
  generar_dataset_test(100+i,'/content/drive/MyDrive/ML/dataset-
union/','/content/dataset-test/')
  test_ds = tf.keras.preprocessing.image_dataset_from_directory(
    '/content/dataset-test/',
    image_size=(img_height, img_width),
    batch_size=1
  )
  eva = model.evaluate(test_ds, return_dict=True)
  # Guardamos los datos obtenidos
  try:
    print(worksheet.cell(1,1).value)
  except:
    gc = gspread.authorize(GoogleCredentials.get_application_def
ault())
  worksheet.update_cell(i+1,1,i)
  worksheet.update_cell(i+1,2,100+i)
  worksheet.update_cell(i+1,3,eva['loss'])
  worksheet.update_cell(i+1,4,eva['accuracy'])
  if eva['accuracy'] >= 0.9:
    seguimos_buscando = False
```

3.4.27.2. Resultados

Los resultados obtenidos se podrán observar completos en el aprendice 08.

El número de pruebas realizadas fue de 129. Los datos obtenidos son:

Tabla 100
Resultados de evaluación de modelo final

Promedio de Loss	0,9011705813
Promedio de True Positive Rate	85,79%
True Positive Rate mínimo obtenido	82,00%
True Positive Rate máximo obtenida	89,20%
Loss mínimo obtenido	0,6084449887
Loss máximo obtenida	1,229955077

3.4.28. Evaluación de observación de predicción por tipo

3.4.28.1. Descripción de evaluación por tipo

La evaluación por tipo tiene el objetivo de poder observar los resultados de predicciones por imagen y agrupando los datos por tipos de leucocitos. Al contrario, la evaluación por ciclos nos daba datos de evaluación por el dataset evaluado en su totalidad.

Para el caso de esta evaluación, se generó un dataset de imágenes con las mismas características con el que se generaban los dataset de la anterior evaluación. El dataset este compuesto por 100 imágenes por tipo, 500 muestras en total. Todas las muestras fueron generadas de una muestra del dataset original. También se está utilizando el api de Google para poder guardar todos los datos que obtengamos en una hoja de cálculo de Google.

El algoritmo utilizado realiza lo siguiente:

1. Configura la primera fila de la hoja de cálculo de Google con los encabezados correspondientes.

2. Empieza lo bucles por las carpetas de los tipos y por archivo dentro de cada carpeta.
3. Guarda los datos de numero de prueba y tipo original.
4. Preprocesa la imagen.
5. Realiza la predicción de la imagen.
6. Guarda los valores de salida y genera el porcentaje de probabilidad por cada tipo.

```
dir_imagenes = '/content/drive/MyDrive/ML/dataset-test/'

worksheet.update_cell(1,1,'N de prueba')
worksheet.update_cell(1,2,'Tipo Original')
worksheet.update_cell(1,3,'Basófilo')
worksheet.update_cell(1,4,'Eosinófilo')
worksheet.update_cell(1,5,'Linfocito')
worksheet.update_cell(1,6,'Monocito')
worksheet.update_cell(1,7,'Neutrófilo')
worksheet.update_cell(1,8,'Porcentaje Basófilo')
worksheet.update_cell(1,9,'Porcentaje Eosinófilo')
worksheet.update_cell(1,10,'Porcentaje Linfocito')
worksheet.update_cell(1,11,'Porcentaje Monocito')
worksheet.update_cell(1,12,'Porcentaje Neutrófilo')

contador = 1

for path_dir_imagenes_type in os.listdir(dir_imagenes):
  for path_dir_imagenes_type_img in os.listdir(dir_imagenes+path_dir_i
magenes_type):
    worksheet.update_cell(contador+1,1,contador)
    worksheet.update_cell(contador+1,2,path_dir_imagenes_type)
    path_dir_imagenes_type_img = dir_imagenes+path_dir_imagenes_type+'
/'+path_dir_imagenes_type_img

    img_height = 180
    img_width = 180
    img_reescaled = keras.preprocessing.image.load_img(path_dir_imagen
es_type_img,target_size=(img_height,img_width))

    img_array = keras.preprocessing.image.img_to_array(img_reescaled)
    img_batch = np.expand_dims(img_array, axis=0)

    #------------- Evaluacion-----------------------------
    res = model.predict(img_batch)
    res=res[0]
    #-------------Generando porcentajes-----------------------------
    tipos = ['Basófilo','Eosinófilo','Linfocito','Monocito','Neutrófil
o']

    tipos_positivos = 5
    sum_total= 0
    print(res)
    for tipo in res:
      if tipo < 0:
        tipos_positivos-=1
      else:
        sum_total+=tipo

    for index in range(5):
      if res[index] >= 0:
        worksheet.update_cell(contador+1,3+index,float(res[index]))
        worksheet.update_cell(contador+1,8+index,float(res[index]*100/
sum_total))
      else:
        worksheet.update_cell(contador+1,3+index,float(res[index]))
        worksheet.update_cell(contador+1,8+index,float(0))
    contador+=1
```

Se tuvo que realizar un segundo algoritmo para poder obtener más datos relevantes. En este caso, saber cuál tipo tiene la mayor probabilidad de ser y si esta predicción fue correcta.

El algoritmo sigue los siguientes pasos:

1. Lee los porcentajes desde la hoja de cálculo.
2. Obtiene el porcentaje mayor y escribe el tipo de leucocitos predicho según este porcentaje.
3. Compara si el tipo predicho es igual al tipo original y guarda un 1 si es igual o 0 si es diferente.

```
tipos = ['BASOFILO','EOSINOFILO','LINFOCITO','MONOCITO','NEUTROF
ILO']
worksheet.update_cell(1,13,'Resultado de Prediccion - texto')
worksheet.update_cell(1,14,'Estado de prediccion')

for contador in range(0,500):
  precision_porcentajes = []
  tipo_text = worksheet.cell(contador+2,2).value
  max_porcentaje = 0.00
  prediccion_text = ""
  for tipo in range(5):
    precision_porcentajes.append(float(worksheet.cell(contador+2
,8+tipo).value.replace(',','.')))
    if max_porcentaje < precision_porcentajes[tipo]:
      max_porcentaje = precision_porcentajes[tipo]
      prediccion_text = tipos[tipo]
  worksheet.update_cell(contador+2,13,prediccion_text)
  if prediccion_text == tipo_text:
    worksheet.update_cell(contador+2,14,1)
  else:
    worksheet.update_cell(contador+2,14,0)
```

3.4.28.2. Resultados

Los resultados fueron guardados en una hoja de cálculo que será el apéndice 10. En esta hoja de cálculo, en la primera página se encuentran todos los datos recopilados de los algoritmos descritos previamente. Y en la página 2 se encuentra datos del análisis realizado a los datos de la página 1.

Los datos de análisis de la página 2 son los promedios de aciertos por cada tipo y el promedio del porcentaje obtenido por cada tipo, este promedio tomo en cuenta a los aciertos y a los fallos.

Tipo	True Positive Rate	Probabilidad de ser el tipo correcto promedio obtenido
BASOFILO	98%	36.548%
EOSINOFILO	72%	68.522%
LINFOCITO	90%	43.154%
MONOCITO	73%	52.394%
NEUTROFILO	93%	72.021%

Figura 93
Diagrama de probabilidad de cada predicción - Basófilo

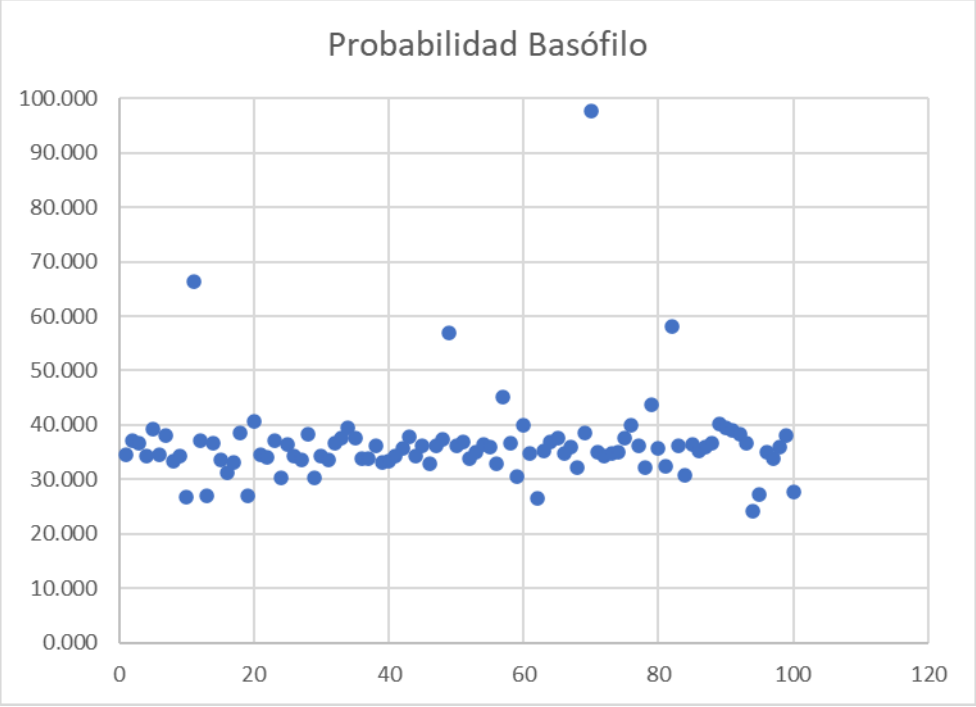


Figura 94
Diagrama de probabilidad de cada predicción - Eosinófilo

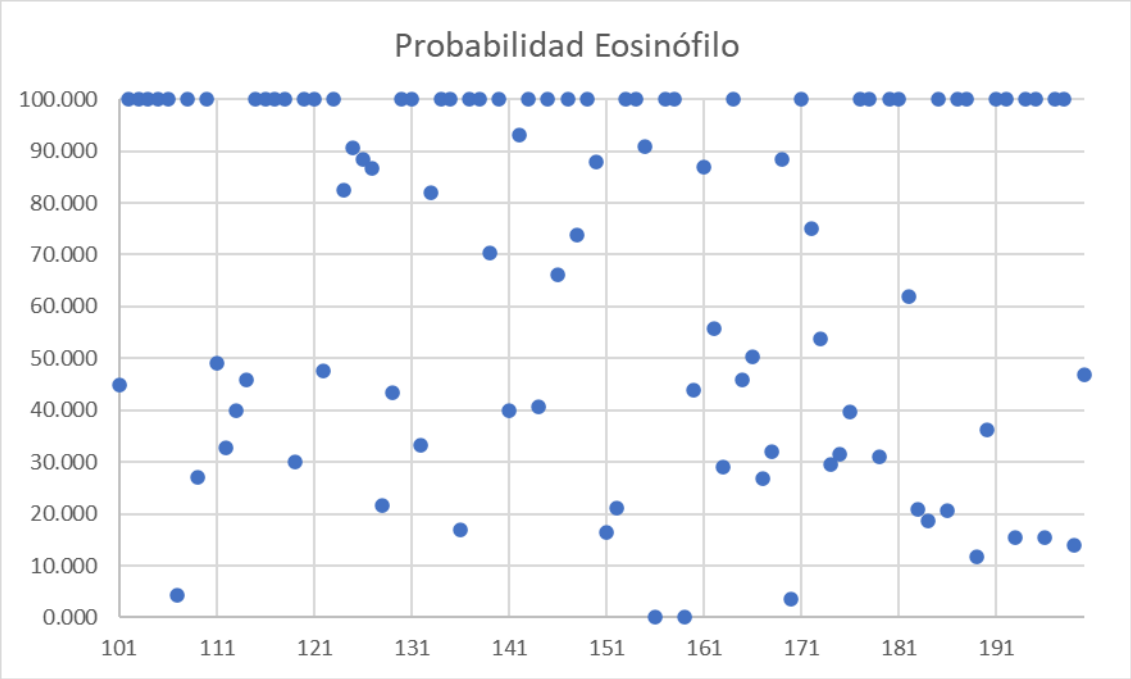


Figura 95
Diagrama de probabilidad de cada predicción – Linfocito

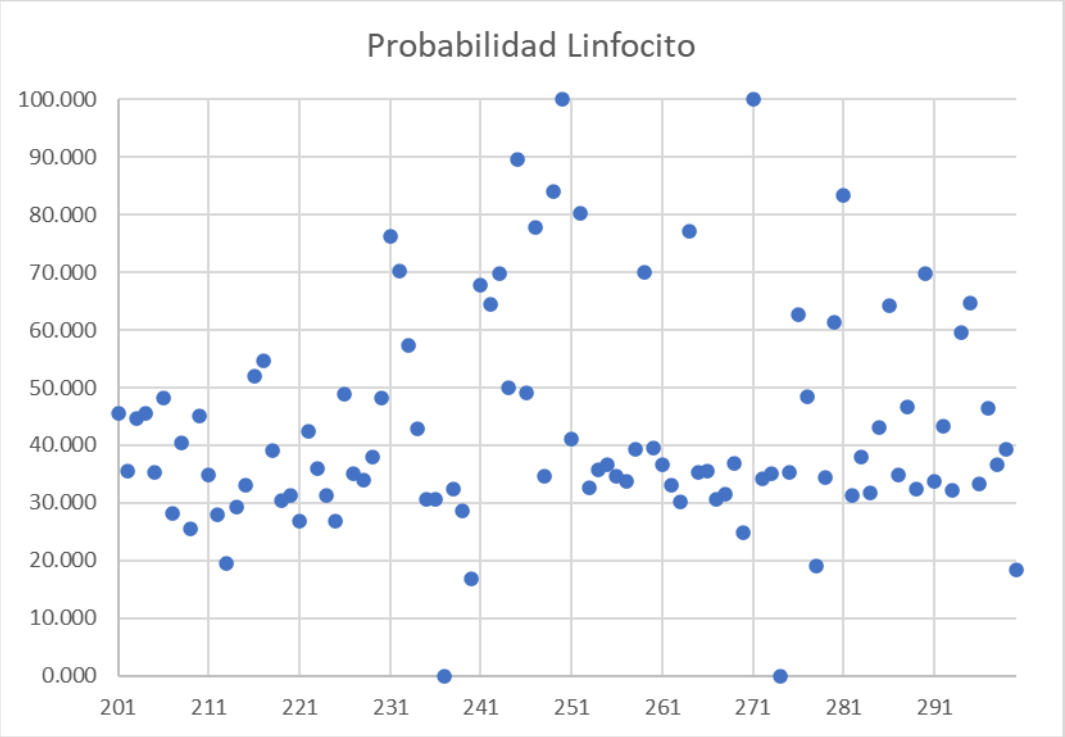


Figura 96
Diagrama de probabilidad de cada predicción - Monocito

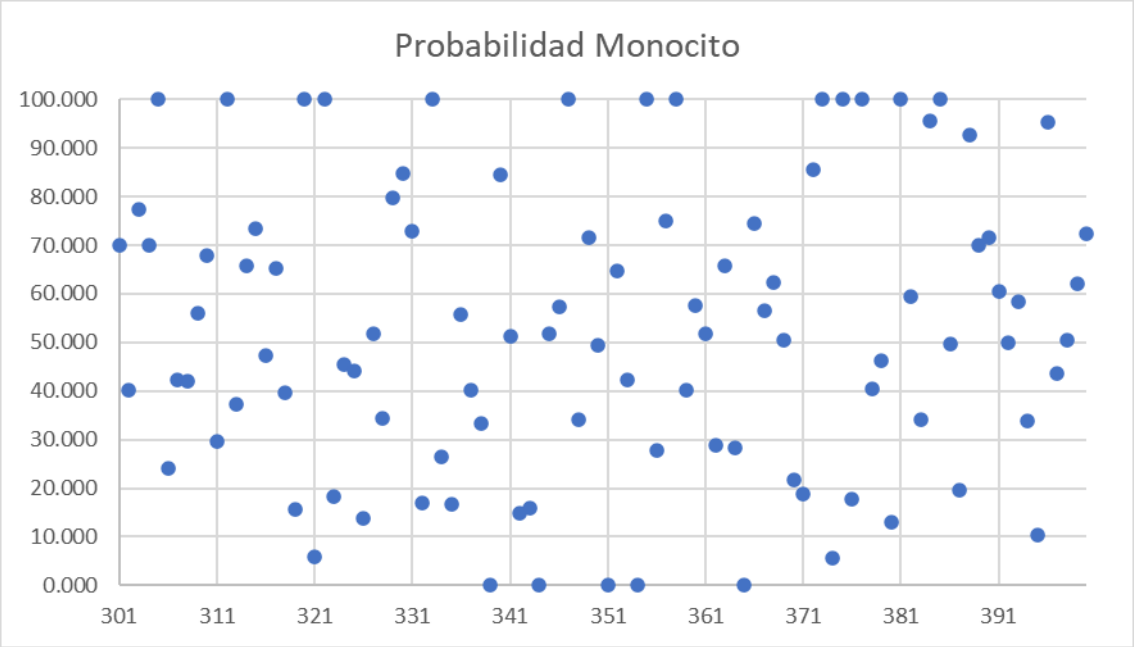
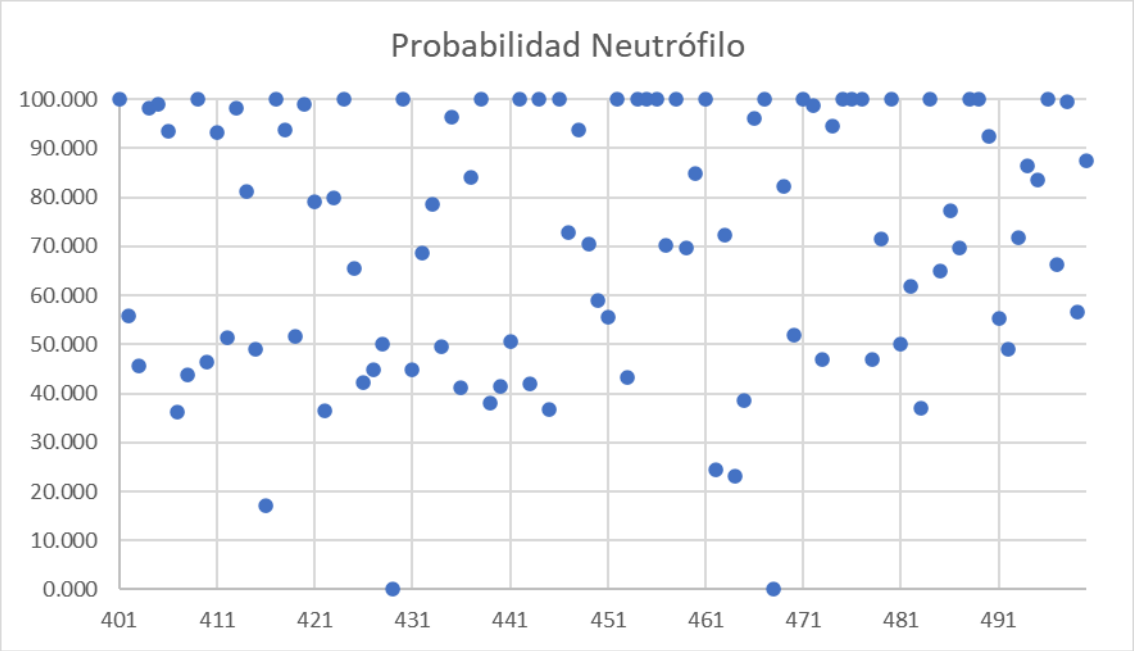


Figura 97
Diagrama de probabilidad de cada predicción - Neutrófilo



También tenemos que realizar un análisis de los valores de True Positive Rate (TPR), False Positive rate (FPR) y Accuracy que podemos obtener de esta evaluación. Para esto vamos a tener que generar una matriz de confusión.

Tabla 101
Matriz de confusión – primera parte

		Predicho						Total
		BASOFILO	EOSINOFILO	LINFOCITO	MONOCITO	NEUTROFILO	NONE	
Real	BASOFILO	98	0	1	0	1	0	100
	EOSINOFILO	2	72	6	1	19	0	100
	LINFOCITO	2	1	90	4	3	0	100
	MONOCITO	3	10	4	73	9	1	100
	NEUTROFILO	4	1	1	1	93	0	100

Ahora podemos calcular las matrices de confusión para cada tipo de leucocito. Recordar que las clasificaciones utilizadas son:

- Positive: Significa que la imagen brindad fue clasificada como el tipo.
- Negative: Significa que la imagen brindad no fue clasificada como el tipo.
- True: Significa que la imagen brindada fue correctamente clasificada.
- False: Significa que la imagen brindada fue erróneamente clasificada.

Tabla 102
Matriz de confusión – basófilo

		TRUE	FALSE
BASOFILO	Positive	98	11
	Negative	389	2

Tabla 103
Matriz de confusión – eosinófilo

		TRUE	FALSE
EOSINOFILO	Positive	72	12
	Negative	388	28

Tabla 104
Matriz de confusión – linfocito

		TRUE	FALSE
LINFOCITO	Positive	90	12
	Negative	388	10

Tabla 105
Matriz de confusión – monocito

		TRUE	FALSE
MONOCITO	Positive	73	6
	Negative	394	27

Tabla 106
Matriz de confusión – neutrófilo

		TRUE	FALSE
NEUTROFILO	Positive	93	32
	Negative	368	7

Usando las fórmulas descritas en el marco teórico. Calculamos los valores de True Positive Rate (TPR), False Positive rate (FPR) y Accuracy por cada tipo. También calculamos el promedio de los valores calculados.

Tabla 107
Mediciones estadísticas

	Sensitivity or True Positive Rate	Specificity or True Negative Rate	Accuracy
BASOFILO	0.98	0.9725	0.974
EOSINOFILO	0.72	0.97	0.92
LINFOCITO	0.9	0.97	0.956
MONOCITO	0.73	0.985	0.934
NEUTROFILO	0.93	0.92	0.922
PROMEDIO	0.852	0.9635	0.9412

Capítulo 4. Resultados

4.1. Cumplimiento de objetivos

Para la finalización de la investigación, se realizará un resumen de todos los resultados obtenidos según los objetivos específicos y al final se indicará el resultado del objetivo general planteado.

4.1.1. Cumplimiento de los objetivos específicos

4.1.1.1. Objetivo Especifico 01: Realizar una revisión bibliográfica para definir un grado de error aceptable.

Como se indicó en la parte de "Recolección y análisis de datos" del Capítulo 04, los dos limites que se manejaron para poder definir un grado de error aceptable fueron del promedio del grado de error del proceso manual y el promedio del grado de error de maquinarias especializadas. Estos datos fueron 18.53% y 3.59%, respectivamente.

Y el grado de error aceptable escogido para poder ser utilizado como meta del proceso de selección de un modelo/esquema de machine learning fue de un 5%. Este grado de error aceptable cumple con que es mejor que un proceso manual y que es un grado de error utilizado en investigaciones clínicas/médicas.

4.1.1.2. Objetivo Especifico 02: Definir el proceso de preprocesamiento de imágenes microscópicas para usarlos como datos de entrada en una red neuronal CNN

Este preprocesamiento de imágenes es la explicación de todo el procedimiento de trata de imágenes de diferentes fuentes para poder ser utilizadas de la mejor manera en nuestra investigación.

A continuación, se dará un resumen de todo el proceso que se siguió desde la obtención de los primeros datos hasta antes del inicio de entrenamiento de modelos.

El primer paso es la obtención de todas las fuentes de imágenes que queremos que conformen nuestro dataset inicial. En este paso las únicas restricciones que pusimos es que el dataset debe de estar previamente clasificado por los tipos de leucocitos que nosotros estamos manejando en esta investigación. Otras características como número de pixeles, nivel de tinción de los frotis o métodos de tinción no fueron excluyentes. Al hacer este proceso se permite obtener datos diversos y posteriormente generar un dataset más heterogéneo. Cada una de las fuentes debe de ser correctamente documentado para poder tener claro conocimiento de su origen.

El siguiente paso es la unión de todas los datasets de fuentes iniciales. Para este paso se realizó una evaluación por si era necesario alguna acción para garantizar la coherencia en el dataset. Se tuvieron que realizar dos correcciones. El primero fue en el dataset 02 , este constaba de imágenes rotadas las cuales para no tener vacíos que pudieran contar como distractores no calculados, se decidió rellenar con una combinación parecida al plasma del fondo de las mismas. La segunda corrección aplicada fue el cambio de formatos de archivos o de formatos de compresión. El dataset 01 y 03 estaban en los formatos TIFF y BMP. Para poder garantizar que no se encuentren problemas al trabajar con diferentes formatos de imágenes, se decidió a cambiarlos a JPEG.

A continuación, se decidió realizar un preprocesado de data augmentation con el objetivo de tener en cantidades parecidas las muestras por cada tipo de leucocito y de aumentar las muestras para poder realizar más ciclos disminuyendo el riesgo del overfitting. Se realizar 3 preprocesamientos. El primero fue de rotaciones aleatorias con el objetivo de aumentar las muestras de basófilos hasta una cantidad parecida a número de muestras de los otros tipos. Después se realizó un preprocesamiento de variación de iluminación con el objetivo de aumentar un 10% el número de muestras. Y por último un preprocesado de ruido gaussiano para aumentar el número de muestras y tener muestras alteradas que permitan representar imágenes con algún fallo.

El dataset final el cual sería utilizado para generar los dataset de entrenamiento, validación y de testeo, estaba compuesto por 59877 muestras. En la documentación extensa del proceso es llamado dataset 02 porque fue el segundo dataset utilizado para pruebas reales. El primer dataset utilizado solo constaba de 14712 muestras y al usarlo en entrenamiento se vio un rápido bifurcación entre el accuracy de entrenamiento y el de validación, esta era una señal clara de overfitting por lo que se decidió crear el dataset 02 de manera más heterogénea .La distribución por tipo de leucocito es:

Tabla 108
Numero de muestras por tipo de dataset final

Tipo de leucocito	**N. total de muestras**
Basófilo	12546
Eosinófilo	8726
Linfocito	10804
Monocito	8257
Neutrófilo	19544
Total	59877

Un proceso extra que se tuvo que realizar fue la subida del dataset final a los servidores de drive.

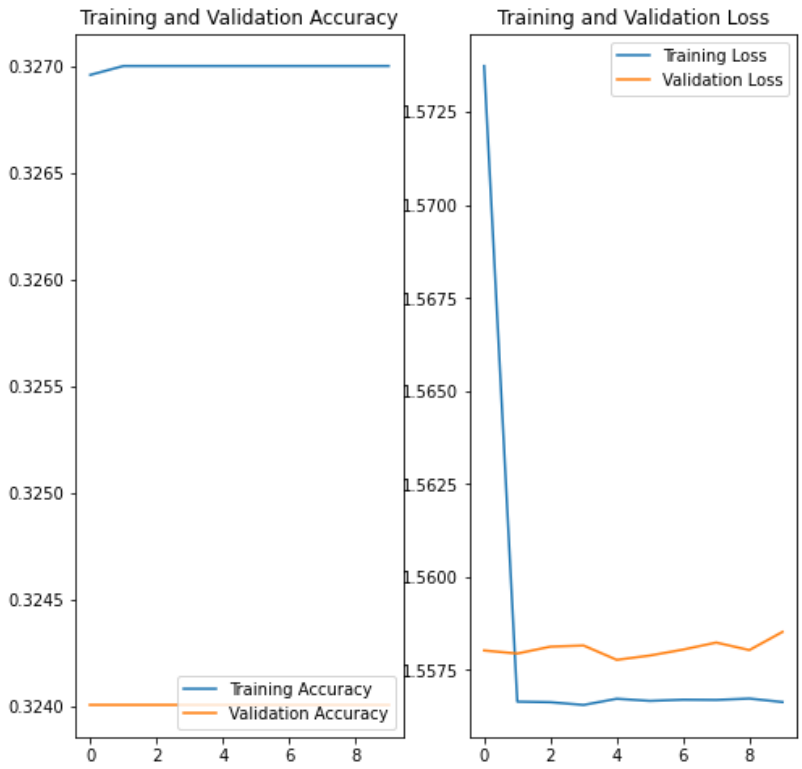
4.1.1.3. Objetivo Especifico 03: Determinar la arquitectura de red neuronal CNN que cumple con el grado de error aceptable

Para empezar, daremos un resumen de todo el proceso para obtener la arquitectura del modelo final. Al definir nuestra metodología se pensó en cuales serían las primeras arquitecturas que se deberían de probar. Se escogió tomar como base la arquitectura base que se da en la documentación de tensorflow para la clasificación de imágenes. Esta arquitectura fue inicialmente probada con diferentes números de ciclos de 10 a 15, alterando el número de capas de 6 a 8 y alterando el número de neuronas. Esta primera variación de las neuronas fue pensada para poder observar en qué punto el overfitting se presenta y que variables son las que son más propensas al overfitting. Ahora que teníamos nuestros primeros modelos, se procedió al entrenamiento.

El proceso de entrenamiento de los modelos tuvo varios problemas. Estos fueron:

1. El primero fue la incapacidad de poder subir nuestro dataset para cada sesión de entrenamiento. La solución a este problema fue fácil, subir el dataset a drive y conectar las sesiones con el drive.
2. La larga duración de cada entrenamiento por modelo. Al inicio se entrenaba un modelo por sesión de Colab. Este proceso duraba de 2 a 3 horas por cada entrenamiento. Esto lo volvía un proceso muy pesado ya que exigía la total atención del investigador para cambiar los modelos. Solución: Se realizaron pruebas y se observó que, si era posible entrenar modelos sin la necesidad de reiniciar cada sesión, sino entrenar todos los modelos de manera consecutiva. Realizando esta mejora se observó que el primer modelo a entrenarse si duraba de 2 a 3 horas. Pero los siguientes modelos bajaban su tiempo de entrenamiento a 20 – 40 minutos. Investigando se concluyó que este comportamiento se daba por la generación da batch del dataset y la carga de su mapeo en memoria.
3. Sobrecargar la memoria RAM que se nos daba por que el dataset se trataba de quedar ahí como canche. Al ser tantas muestras era imposible que se puede tener todo el dataset cargado en memoria. Se tuvo que configurar para que solo se cargue los batch para su uso en el entrenamiento y retirarlo para el siguiente.
4. Modelos que no aprendían. Durante el proceso de entrenamiento y experimentación aparecieron varios modelos los cuales no convergían (no eran capaces de encontrar un mínimo el cual seguir para poder avanzar). Estos modelos no aprendían, se quedaban en una constante búsqueda y el valor de loss de entrenamiento se mantenía constante. Se pudo observar que esta situación se daba cuando un modelo no tenía suficientes neuronas para poder analizar los datos de entrada.

Figura 98
Ejemplo de entrenamiento atascado



5. Obtención del dato “Tiempo de entrenamiento”. El tiempo de entrenamiento no se puede obtener de manera nativa. Para obtenerlo se realizó un callback el cual obtiene el tiempo de inicio y final de cada ciclo. Para luego calcular los segundos que le toma realizar todo el proceso.

Figura 99
Función callback para cálculo de tiempo de entrenamiento

```
class TimingCallback(keras.callbacks.Callback):
    def on_train_begin(self, logs={}):
        self.times = []
    def on_epoch_begin(self, batch, logs={}):
        self.epoch_time_start = time()
    def on_epoch_end(self, batch, logs={}):
        self.times.append(time() - self.epoch_time_start)
```

Se realizo una segunda iteración donde se tuvo como objetivo aumentar más el número de ciclos, capas y neuronas. Las pruebas de la primera iteración mostraron que en muchos casos el número de neuronas no era el suficiente para poder realizar el aprendizaje correctamente. Por esta razón se buscó aumentar todas las variables de los esquemas.

Para la tercera iteración, se vio que aún había posibilidad de entrenar más ciclos sin tener grandes pistas de overfitting y también se ingresó el uso de una técnica para reducir el overfitting, el uso de regularización con valores de 0.01 y 0.0001.

Para la cuarta iteración, se decidió probar usar otra estrategia para reducir el overfitting. El uso de capas de dropout que permitía bloquear de manera aleatoria un número de conexiones de una capa a otra. Esto con el objetivo de reducir el nivel de conspiración que puede tener una capa de la siguiente. Los valores que se probaron de dropout fueron de 25%, 50% y 75%.

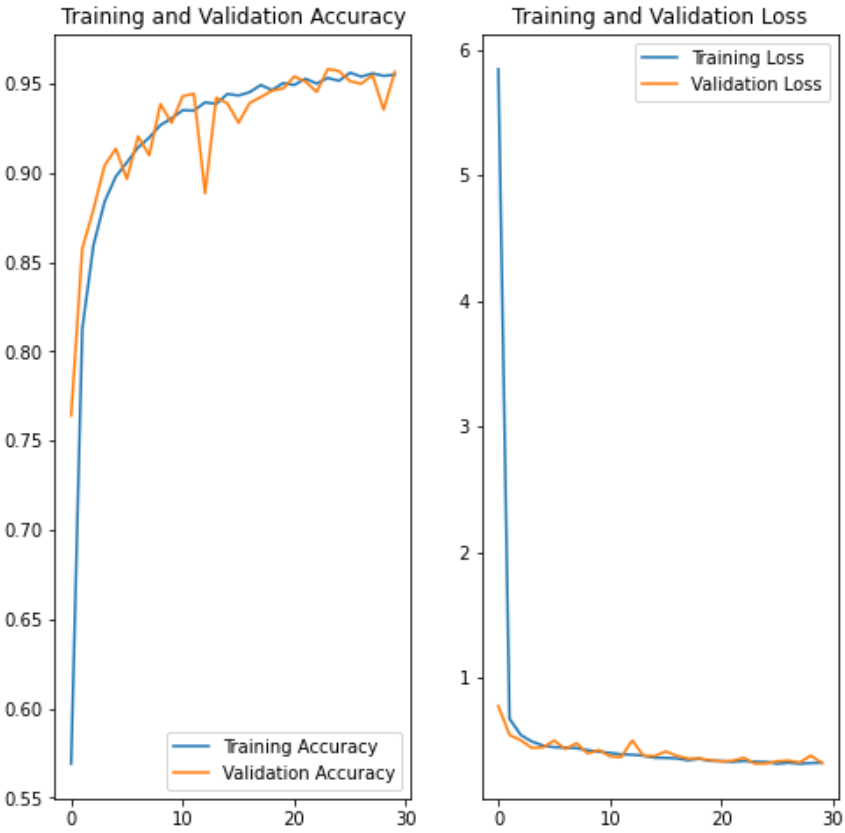
Para la quinta iteración, se observó que el valor de dropout que traía mejores resultados sin afectar de manera tan drástica fue con el valor del 25%. Ahora se buscó llevar al límite los modelos subiendo el número de ciclos a 35. También se decidió probar juntado ambas técnicas contra el overfitting para poder observar los resultados.

Para la sexta iteración, los modelos probados que utilizaban ambas técnicas contra el overfitting permitían un mejor desempeño ya que obliga constantemente a generar nuevas formas de aprender. También se observó que los modelos con más neuronas fueron los que mejor se adaptaron al uso de capas de dropout. Se realiza un cambio a la forma de trabajar en las iteraciones. Hasta la quinta iteración se buscó definir todos los esquemas a probar en la siguiente iteración antes de entrenarlos para poder beneficiarse de trabajar en una única sesión de Colab. Ahora se evaluará coda esquema de manera única con el objetivo de perfilar el esquema final para nuestro modelo final. El valor que más se busca ajustar en este punto es el de regularización. Con las pruebas anteriores se pudo observar que, si afectaba al problema de overfitting, pero a su vez retrasaba mucho el entrenamiento limitando la variación del valor de Loss.

De la sexta iteración, el primer esquema se utilizó el esquema con mayor número de neuronas probando un valor de regularización mayor, de 0.01. Para el segundo esquema se alteró el valor de regularización a 0.001. Para el tercer esquema se alteró el valor de regularización a 0.0001. En este punto se observó que con el valor de 0.0001 ya se empezaban a mostrar indicios de overfitting por lo que se decidió subir el valor. Para el cuarto esquema se alteró el valor de regularización a 0.0005.

Figura 100
Resultado I06-D003

Figura 101
Resultado I06-D004



Observando los resultados, se ve una clara mejora en el overfitting. Pero también se observa que, al usar valores más altos de regularización, no permite la reducción de loss y por la tanto mejorar el accuracy más. Para el quinto se bajó más el valor de regularización, a 0.00005. Y viendo que podemos alcanzar un accuracy alto en los primeros ciclos. Para la sexta interacción se redujo los ciclos a 10. En el octavo ciclos de definió que el valor con el que nos quedaríamos seria de 0.00004. ya que este es el que menor valor de loss generaba. Se realizo otra prueba con un valor de regularización más bajo, pero estos no presentaban mejoras al de 0.00004.

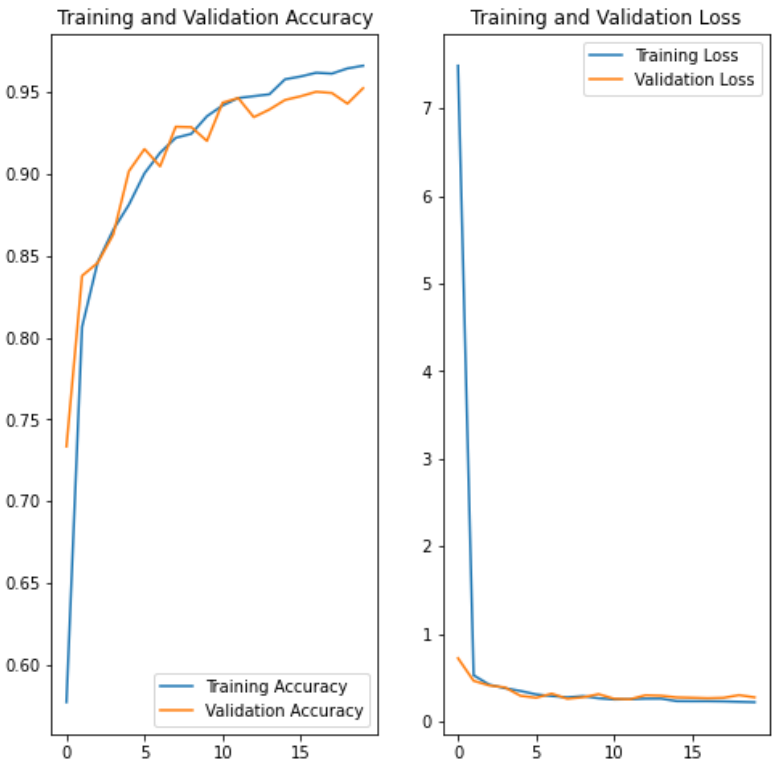
Los datos del esquema final escogido fueron:

Tabla 109
Variables de esquema del modelo final

Código de diseño	Numero de neuronas	Numero de ciclos	Numero de capas ocultas	Función de Activación	Dropout	Valor de Regularización
I06-D008	245	20	8	RELU	0.25	0.00004

Figura 102
Precisión y Loss del entrenamiento del modelo I06-D008 – prueba 1

Figura 103
Precisión y Loss del entrenamiento del modelo I06-D008 – prueba 2



Se observa que en el entrenamiento se llegó a un accuracy en entrenamiento de 97.16% y un 95.76% de accuracy en validación. En la figura 102, se puede observar la comparación de accuracy de todos los modelos de la iteración 06, en esta grafica se observa que el modelo I06-D008 es el segundo con una taza de accuracy mayor. Y en la figura 101, se observa que es el que alcanzo el menor valor de Loss. Por estas razones se decidió usar este modelo como el modelo final.

4.1.2. Cumplimiento del objetivo general: Demostrar si la clasificación de leucocitos en imágenes microscópicas de frotis sanguíneo realizado con machine learning y CNN cumple un grado de error aceptable

En el punto anterior se especificó que los datos de accuracy en entrenamiento y en validación fueron 97.16% y 95.76%. El grado de error de estas variables seria de 2.84 % y 4.24% respectivamente. Ambos valores cumplen con nuestro grado de error aceptable que es del 5%. Pero estos valores son extraídos del proceso de entrenamiento, para poder tener una fiabilidad de que no hay un sesgo por solo comprobar los valores resultantes

del entrenamiento, se decidió generar datasets de testeo desde nuestro dataset original con los cuales evaluar mediante una función que la librería de tensorflow nos facilita.

Se realizo el proceso de entrenamiento del modelo final. Esto con el objetivo de poder entrenar el modelo el numero justo de ciclos necesarios para que el valor de accuracy de entrenamiento supere el 5% de nuestro grado de error aceptable. Se dio la casualidad de que se obtuvo como 15 el número de ciclos necesarios. Se obtuvo el siguiente grafico:

Figura 104
Accuracy y Loss del modelo final

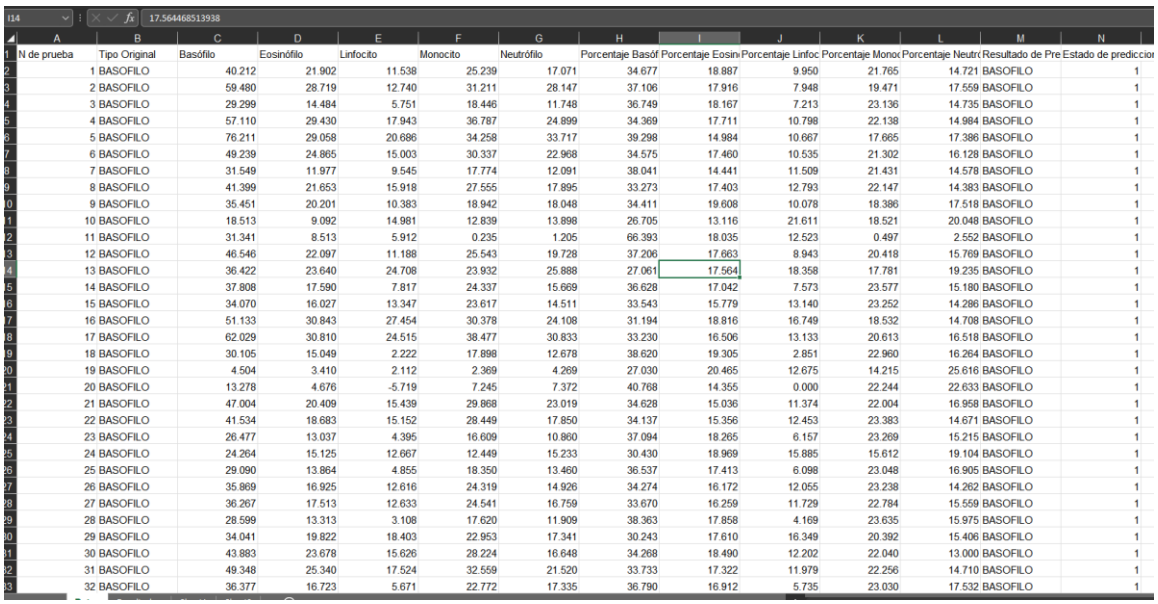
Se realizo un algoritmo en el cual se generaba un nuevo dataset de data aleatoria de nuestro dataset final. Este dataset constaba de un 10% de muestras generadas con data augmentation de la data original. De esta forma se generaron 129 datasets de test y se realizaron 129 evaluaciones. Los resultados de esta prueba fueron :

Tabla 110
Resultado de evaluación de dataset de testeo

Promedio de Loss	0,9011705813
Promedio de True Positive Rate	85,79%
True Positive Rate mínimo obtenido	82,00%
True Positive Rate máximo obtenida	89,20%
Loss mínimo obtenido	0,6084449887
Loss máximo obtenida	1,229955077

El proceso de evaluación fue realizado con la función “evaluate” que es parte del mismo objeto tipo modelo que nos brinda tensorflow. Pera para poder saber realmente si estos valores son los correctos se decidió realizar un algoritmo en el cual, con el ultimo dataset de test generado, realizar predicciones de uno en uno para poder obtener datos y luego analizar esos datos. Esta base de datos se veía así:

Figura 105
Captura de archivo Excel de evaluación de predicción uno a uno



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	N de prueba	Tipo Original	Basófilo	Eosinófilo	Linfocito	Monocito	Neutrófilo	Porcentaje Basóf	Porcentaje Eosin	Porcentaje Linfoc	Porcentaje Monoc	Porcentaje Neutr	Resultado de Pre	Estado de predicci
2	1	BASOFILO	40.212	21.902	11.538	25.239	17.071	34.677	18.887	9.950	21.765	14.721	BASOFILO	1
3	2	BASOFILO	59.480	28.719	12.740	31.211	28.147	37.106	17.916	7.948	19.471	17.559	BASOFILO	1
4	3	BASOFILO	29.299	14.484	5.751	18.446	11.748	36.749	18.167	7.213	23.136	14.735	BASOFILO	1
5	4	BASOFILO	57.110	29.430	17.943	36.787	24.899	34.369	17.711	10.798	22.138	14.984	BASOFILO	1
6	5	BASOFILO	76.211	29.058	20.686	34.258	33.717	39.298	14.984	10.667	17.665	17.386	BASOFILO	1
7	6	BASOFILO	49.239	24.865	15.003	30.337	22.968	34.575	17.460	10.535	21.302	16.128	BASOFILO	1
8	7	BASOFILO	31.549	11.977	9.545	17.774	12.091	38.041	14.441	11.509	21.431	14.578	BASOFILO	1
9	8	BASOFILO	41.399	21.653	15.918	27.555	17.895	33.273	17.403	12.793	22.147	14.383	BASOFILO	1
10	9	BASOFILO	35.451	20.201	10.383	18.942	18.048	34.411	19.608	10.078	18.386	17.518	BASOFILO	1
11	10	BASOFILO	18.513	9.092	14.981	12.839	13.898	26.705	13.116	21.611	18.521	20.048	BASOFILO	1
12	11	BASOFILO	31.341	8.513	5.912	0.235	1.205	66.393	18.035	12.523	0.497	2.552	BASOFILO	1
13	12	BASOFILO	46.546	22.097	11.188	25.543	19.728	37.206	17.663	8.943	20.418	15.769	BASOFILO	1
14	13	BASOFILO	36.422	23.640	24.708	23.932	25.888	27.061	17.564	18.358	17.781	19.235	BASOFILO	1
15	14	BASOFILO	37.808	17.590	7.817	24.337	15.669	36.628	17.042	7.573	23.577	15.180	BASOFILO	1
16	15	BASOFILO	34.070	16.027	13.347	23.617	14.511	33.543	15.779	13.140	23.252	14.286	BASOFILO	1
17	16	BASOFILO	51.133	30.843	27.454	30.378	24.108	31.194	18.816	16.749	18.532	14.708	BASOFILO	1
18	17	BASOFILO	62.029	30.810	24.515	38.477	30.833	33.230	16.506	13.133	20.613	16.518	BASOFILO	1
19	18	BASOFILO	30.105	15.049	2.222	17.898	12.678	38.620	19.305	2.851	22.960	16.264	BASOFILO	1
20	19	BASOFILO	4.504	3.410	2.112	2.369	4.269	27.030	20.465	12.675	14.215	25.616	BASOFILO	1
21	20	BASOFILO	13.278	4.676	-5.719	7.245	7.372	40.768	14.355	0.000	22.244	22.633	BASOFILO	1
22	21	BASOFILO	47.004	20.409	15.439	29.868	23.019	34.628	15.036	11.374	22.004	16.958	BASOFILO	1
23	22	BASOFILO	41.534	18.683	15.152	28.449	17.850	34.137	15.356	12.453	23.383	14.671	BASOFILO	1
24	23	BASOFILO	26.477	13.037	4.395	16.609	10.860	37.094	18.265	6.157	23.269	15.215	BASOFILO	1
25	24	BASOFILO	24.264	15.125	12.667	12.449	15.233	30.430	18.969	15.885	15.612	19.104	BASOFILO	1
26	25	BASOFILO	29.090	13.864	4.855	18.350	13.460	36.537	17.413	6.098	23.048	16.905	BASOFILO	1
27	26	BASOFILO	35.869	16.925	12.616	24.319	14.926	34.274	16.172	12.055	23.238	14.262	BASOFILO	1
28	27	BASOFILO	36.267	17.513	12.633	24.541	16.759	33.670	16.259	11.729	22.784	15.559	BASOFILO	1
29	28	BASOFILO	28.599	13.313	3.108	17.620	11.909	38.363	17.858	4.169	23.635	15.975	BASOFILO	1
30	29	BASOFILO	34.041	19.822	18.403	22.953	17.341	30.243	17.610	16.349	20.392	15.406	BASOFILO	1
31	30	BASOFILO	43.883	23.678	15.626	28.224	16.648	34.268	18.490	12.202	22.040	13.000	BASOFILO	1
32	31	BASOFILO	49.348	25.340	17.524	32.559	21.520	33.733	17.322	11.979	22.256	14.710	BASOFILO	1
33	32	BASOFILO	36.377	16.723	5.671	22.772	17.335	36.790	16.912	5.735	23.030	17.532	BASOFILO	1

Resultados de proceso de testeo final del modelo seleccionado.

Tabla 111
Variables de la evaluación de matriz de confusión

	Sensitivity or True Positive Rate	**Specificity or True Negative Rate**	**Accuracy**
BASOFILO	0.98	0.9725	0.974
EOSINOFILO	0.72	0.97	0.92
LINFOCITO	0.9	0.97	0.956
MONOCITO	0.73	0.985	0.934
NEUTROFILO	0.93	0.92	0.922
PROMEDIO	0.852	0.9635	0.9412

Para finalizar, vamos a explicar cuál es el valor de los datos finales.

El valor de True Positive Rate significaría de solo el tipo de leucocitos que porcentaje se clasifico correctamente. Ejemplo, El 98% de los basófilos fue clasificado correctamente

como basófilos. El 72% de los eosinófilos fue clasificado correctamente como eosinófilos. El promedio total de los TPR fue de 0.852. Entonces se concluye, el 85.2% de los leucocitos evaluados fue correctamente clasificado con su tipo de leucocito.

El valor de TNR hace referencia al porcentaje del total de muestras que fueron correctamente clasificado como que no era de ese tipo. Ejemplo, el 97.25% de las muestras que no eran basófilos fue correctamente clasificado que eran de un tipo diferente al de basófilos. Entonces se concluye, el 96.35% de las muestras que no eran del tipo al que se evalúa en promedio serán correctamente clasificados en otro tipo de leucocito.

El valor de accuracy es un poco más complejo. Este indica el porcentaje de muestras que fueron clasificadas correctamente, tanto como que, si y que no son del tipo evaluado, del total de muestras. Ejemplo, del total de muestras evaluadas, el 97.4% fueron clasificadas correctamente en el grupo de basófilo. Entonces se concluye, el 94.12% de muestras evaluadas será correctamente clasificada con respecto a cada tipo de leucocito.

Capítulo 5. Discusión

El capítulo de discusión será dividido en 2 secciones. Una sección donde poder explicar todos los acontecimientos relevantes que afectaron al desarrolla de la investigación. Y la segunda parte, es la comparación de metodología y resultados con otras investigaciones comparables con esta investigación.

La primera decisión importante fue el de definir el grado de error aceptable en un 5%. Como se explicó antes, el grado de error encontrado en guías técnicas sobre procedimientos hematológicos manuales fue de un 10%. El grado de error encontrado para las maquinarias especializadas "Analistas Hematológicos" fue de un 2%-3%. De esta forma se decidió poner un porcentaje que se encuentre entre estos valores, y al final se decidió usar el 5% como grado de error aceptable.

Se tomo la decisión de usar dataset, bancos de imágenes o bases de datos de imágenes de fuentes externas (otras investigaciones, internet, etc.) en vez de utilizar imágenes recién obtenidas en laboratorios clínicos locales. La razón principal de esta decisión fue por la interrupción de la comunicación con la clínica local por la pandemia de COVID-19. Las clínicas locales tuvieron que seguir reglas más estrictas ya que tuvieron que cubrir todos los casos de análisis de verificación de presencia de COVID-19 en los pacientes. Viendo en la situación que ya no podía utilizar los equipos del laboratorio clínico si tener que ser un riesgo para ellos, evalué la opción de usar otras fuentes. Observando algunos de los antecedentes descritos en la investigación, ya existían datasets de leucocitos previamente clasificados para investigaciones parecidas a esta.

Durante el primer intento de desarrollo de la investigación se tomaron decisiones que afectaron a todo el desarrollo de esta. Estas decisiones fueron:

- Usar el servicio de Google Colab Pro para el uso de procesamiento en la nube. Al inicio se usaría el equipo personal del investigador; pero como esta investigación tomaría mucho tiempo, ya que no era un solo modelo que entrenar sino varios para comparar, se decidió usar un servicio en la nube. Y se optó usar el servicio de paga solo para garantizar su uso por un tiempo más amplio por sesión. Es posible obtener los mismos resultados con la versión gratuita.
- Usar un dataset más homogéneo. En los primeros intentos, se observó que al usar solo un dataset de una sola fuente se llegaba muy rápido al overfitting. Se decidió

utilizar una fusión de datasets de diferentes fuentes y también usar métodos de data augmentation para tener un número similar de muestras por cada tipo de leucocitos.

- Implementación de un método para contar la duración de entrenamiento.
- Utilizar estrategias para poder disminuir el overfitting. Las estrategias utilizadas fueron buscar utilizar de una menor complejidad, usar regularización en las capas ocultadas y agregar capas de Dropout para reducir el nivel de conspiración entre las neuronas.
- El proceso de observación de las gráficas de Accuracy y Loss del proceso de entrenamiento y validación. El objetivo era buscar el mayor nivel de Accuracy y disminuir el valor de Loss, siempre observando que en entrenamiento y validación se observe una continuidad casi paralela. Si en algún momento la continuidad casi paralela se rompe, esto sería una señal de overfitting.

Un problema recurrente para el proceso de entrenamiento fue la duración extensa de entrenamiento de cada modelo. En el primer intento, ya que no es un gran número de imágenes, la duración de entrenamiento de un modelo era entre 15 a 30 minutos. Pero cuando se probó con el Dataset 02 de 59877 imágenes, la duración del entrenamiento paso a ser de 2 a 3 horas. Realizando varias pruebas, el principal problema era que en el primer ciclo del entrenamiento se mapeaba y cargaba todas las imágenes en memoria por primera vez, lo que hacía a este proceso muy lento. Pero después de este primer ciclo, el proceso de entrenamiento era mucho más rápido ya que ya se tenía todo el dataset mapeado. De esta forma se tomó la decisión de trabajar en iteraciones. Para cada iteración se tendría que definir todos sus modelos a probar. Si entrenamos varios modelos de una misma sesión el tiempo de entrenamiento se disminuye significativamente. El primer modelo para entrenar se realizaría en 2 a 3 horas y todos los siguiente durarían de 20 a 50 minutos.

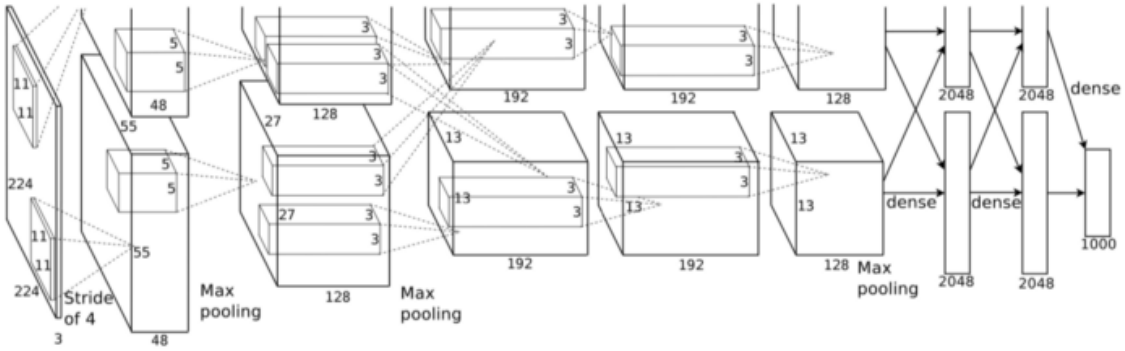
En la siguiente sección se expone la discusión de los resultados logrados y comparados con otros estudios.

Empezando con el antecedente llamado “White blood cell classification and counting using convolutional neural network”(Macawile et al., 2018). Las métricas finales obtenidas en esta investigación son TPR, TNR y Accuracy; los cuales fueron 85.2%, 96.35% y 94.12% respectivamente. Al compararlas con los valores obtenidos por

Macawile, Merl James (Macawile et al., 2018) los cuales fueron de TPR de 89.18% , TNR de 97.85% y Accuracy de 96.63%. Podemos observar que nuestro modelo no pudo alcanzar la precisión que se alcanzó en su proyecto. Pero también se puede observar que ellos no utilizaron imágenes generadas, sino un banco de imágenes de 178 leucocitos lo cual sería un indicio de que no es tan conveniente usar imágenes generadas.

Hablando del modelo utilizado, en la investigación de Macawile, Merl James (Macawile et al., 2018) se busca una comparación entre modelo ya existentes para su aplicación en procesos de clasificación. Mas específicamente, los modelos evaluados por ellos fueron Alexnet, GoogleNet y ResNet-101. El modelo que ellos observaron que dio mejor resultado en la prueba que hicieron fue el de AlexNet. El modelo AlexNet tiene una entrada de imágenes de 224 px x 224 px, utiliza la función de activación ReLU para todas sus capas, está pensada para entrenarse con multi-GPU, tiene 16 capas y 10568 neuronas. Este modelo es gratamente recomendado gracias a que fue la que mejor se desempeñó siendo entrenado con el dataset ILSVRC-2010 en el 2012. Comparándolo con el modelo que se desarrolló en esta investigación se puede afirmar que hay una gran diferencia en términos de dimensiones. Primero, el tamaño de los datos de entrada es más pequeños ya que buscamos tener un el mínimo tiempo de entrenamiento por cada entrenamiento de los modelos a probar, así que nos quedamos con el estándar sugerido por tensorflow para las dimensiones de entrada. La función de activación es la misma usada en esta investigación. Y el número de capas y neuronas es claramente inferior al utilizado en la investigación. Aun pudiendo observar la gran diferencia a nivel de dimensionalidad; los resultados alcanzados en nuestra investigación son inferiores, pero con un máximo de 4.98% . Observando la investigación; es muy interesante que, con un modelo tan complejo, no se logró conseguir una diferencia tan grande en las 3 variables.

Figura 106
Diagrama de modelo Alexnet



Nota: Extraído de ImageNet Classification with Deep Convolutional Neural Networks (Krizhevsky et al., n.d.)

Comparando con la investigación llamada “Leukocytes detection, classification and counting in smears of peripheral blood”(Martínez-Castro et al., 2014). Durante el desarrollo de la investigación de Martínez Castro, no se estaba tan avanzada lo métodos de alto nivel de algoritmos de inteligencia artificial. En esta investigación se utilizó el algoritmo K-means para la predicción de la clasificación del tipo de leucocito y un proceso manual de preprocesamiento de imágenes para ayudar con la clasificación. El preprocesamiento de imágenes de esta investigación era el uso de filtro de colores para la identificación de los leucocitos, más específicamente sus núcleos. Una vez obtenido el filtro correcto, se limpia de impurezas y se aplica a la imagen original para obtener solo el leucocito. Una vez con la imagen recortada se procede a predecir con el algoritmo k-means.

Este proceso fue probado con 45 muestras. En la investigación se muestra que las tazas de probabilidad de falsos positivos solo son visibles para los basófilos , linfocitos y neutrófilos; y estas tasas son del 2%, 3% y 5% respectivamente. Si realizamos un promedio de falso positivo con todos los tipos de esta investigación seria de un 2%. En otras palabras, el TPR es de 98%. En mi investigación se observó un TPR del 89.18%. Esto indicaría que el algoritmo con CNN tiene más probabilidad de dar más falsos positivos a la hora de clasificar que con el algoritmo k-means y su preprocesamiento de imágenes respectivo.

Se puede atribuir la diferencia de TPR de un -8.82% a la falta de un dataset de gran medida en la investigación a comparar. En esta investigación se utilizó un dataset de 59877 muestras (imágenes), comparando con las 45 muestras de evaluación del algoritmo, se puede deducir que los valores obtenidos en esta investigación son más precisos gracias a

la gran variedad de muestras en el dataset. También se puede atribuir el TPR bajo a uso de imágenes generadas en el dataset; ya que no se puede garantizar que todas las imágenes tienen el leucocito 100% integro y sin modificación.

Comparando con la investigación titulada "Diseño de un algoritmo para la automatización del conteo de células del tejido sanguíneo mediante procesamiento digital de imágenes" (Bustamante Alvarez, 2019). La principal diferencia entre esta investigación y la investigación de Bustamante es el objetivo, la investigación de Bustamante busca el conteo de todas las células sanguíneas que se pueden visualizar en un frotis y esta investigación solo se concentra en la clasificación de leucocitos. Se puede decir que esta investigación cubre una parte de la investigación de Bustamante.

Aun así, se puede comparar las partes que guardan relación en ambas investigaciones. Primero, en la investigación de Bustamante se utiliza el uso de redes neuronales para el conteo clasificado de los glóbulos blancos. Se evalúan imágenes amplias (1920 x 1080 pixels) donde se pueden observar varios glóbulos blancos, el algoritmo tiene como objetivo clasificar cada uno, contarlo y devolver un resumen con el conteo. La investigación de Bustamante tiene la hipótesis donde se busca demostrar que el grado de error de su algoritmo propuesto es menor al 5%, esta hipótesis es confirmada ya que se logró obtener un promedio de error del 2.55%. Se tiene que aclarar que este promedio incluye todo su algoritmo, por lo cual se puede usar como tasa de error de conteo selectivo de glóbulos blancos también. Esta investigación logro un grado de error de 5.88% con respecto al acurracy. Esta es la mejor comparación posible, ya que el valor de acurracy cuenta con la integración de falsos positivos y el error de Bustamante cuenta con identificación de células que no existían (falsos positivos) por lo cual la comparación de ambos valores sería lo más justo.

Comparando con la investigación titulada "Desarrollo de un sistema de análisis automático para la segmentación, clasificación y conteo de leucocitos en imágenes digitales de frotis de sangre periférica" (Villegas Noguera, 2019).

El proceso de definir el dataset utilizado para la investigación de Villegas es el más parecido al propuesto en esta investigación. El dataset de Villegas consta de la unión de

6 diferentes dataset con el objetivo de añadir variedad a las muestras que se utilizan. Esta variedad consta de "condiciones de iluminación, tinción, resolución, nitidez, coloración, enfoque y aumento diferentes"(Villegas Noguera, 2019). La principal diferencia radica en el preprocesamiento, Villegas propone separar todo el leucocito del resto de la imagen para poder utilizar solo esta información como muestra para su algoritmo de CNN y de SVG. Este procesamiento esta que hiso mejorar considerablemente la calidad de su dataset, ya que ahora sus muestras no contienen ruido el cual pudiera "distraer" a los algoritmos de IA permitiendo buscar características que no nos interesarían. Para el proceso de definir los modelos de redes neuronales se decidió usar modelos previamente existentes y sus variaciones.

El mejor accuracy que alcanzo Villegas para el banco de datos de validación fue de un 95.86%. Nuestro accuracy alcanzado con nuestro dataset de prueba fue de 94.12%. Se puede ver una diferencia de -1.74%, la cual puede ser explicado por la aplicación de mejores procedimientos para el tratamiento de los datos de entrenamiento. También se debe al uso de modelos de redes neuronales previamente probados y desarrollados lo cuales ya se tiene evidencia que son eficaces para el proceso de clasificación de imágenes.

Comparando con la investigación titulada "Human peripheral blood leukocyte classification method based on convolutional neural network and data augmentation" (Wang & Cao, 2020).

La primera ocupación que podemos realizar es la de los valores resultantes. la investigación de Wang muestra unos valores de TPR, TPN y Acuraccy de 92.5% ,93.4% y 97.6% respectivamente. Los valores obtenidos en esta investigación serian 85.2%, 96.35% y 94.12% respectivamente. Se puede observar que la investigación de Wang tuvo mejores resultados que en esta investigación, Esto se puede deber tanto del modelo de CNN utilizado en su investigación o el dataset y el procesamiento de datos utilizados.

El pensamiento de datos y de la investigación de Juan empieza por la obtención las muestras de los frotis sanguíneos. Las muestras fueron obtenidas directamente desde un microscopio y los datos sólo fueron obtenidos para su investigación. la primera forma de los datos son imágenes amplias donde se puede observar varios leucocitos y otras células sanguíneas, a diferencia que nosotros que ya obtenemos cada muestra por separado según

el tipo de leucocito que fuera. Wang tuvo que pegarme un algoritmo para identificación de los leucocitos; este egoísmo usa el color para identificar posibles puntos donde se encuentra un leucocito cuánto. Una vez identificado un punto, se dibuja a su alrededor un área de 217x217 píxeles. A continuación, se hace un proceso de data augmentation, el cual fue definido por los investigadores. este proceso busca la creación de un filtro el cual permita separar las imágenes de leucocitos con su fondo. una vez que se tiene las muestras de leucocitos y los fondos con otras células sanguíneas, se procede a juntarlas de manera aleatoria generando nuevas muestras para el dataset. en esta investigación nosotros usamos un procedimiento de data augmentation también. Pero la principal diferencia es que nosotros utilizamos una librería externa el cual ya tenía funciones de alteración de imágenes con el cual nosotros pudimos generar nuevas muestras que no eran tan precisas como las que se pudieron generar para la investigación de Wang.

El modelo utilizado en la investigación de Wang fue un modelo más complejo a comparación con utilizado en esta investigación. El modelo de Wang consta de 10 capas ocultas y de 2501 neuronas. El modelo de esta investigación consta de 8 capas ocultas y de 245 neuronas; además de utilizar estrategias de reducción de overfitting como capaz de dropout y valores de regularización.

Comparando con la investigación titulada “Comprehensive Analysis of Deep Learning Methodology in Classification of Leukocytes and Enhancement Using Swish Activation Units” (Harshanand & Sangaiah, 2020).

El TPR y el accuracy obtenido por Harshanand fue de un 97,64% y 97,5%. A comparación con los obtenidos en esta investigación los cuales fueron del 85.2% y 94.12%. La razón de este mejor desempeño en la investigación de Harshanand se debe a las siguientes razones:

- Esta investigación utiliza diferentes arquitecturas o modelos de redes neuronales usadas como estándares, las cuáles son: ResNet50,VGG16, InceptionV3, Exception., Densenet.
- Uso de una nueva función de activación llamada “switch activation function” en vez del clásico RELU.

Observando estas diferencias, podemos indicar que se debería observar si al cambiar la función de activación se puede observar un mejor despeño en el proceso de clasificación.

También se puede hacer la misma observación de otras investigaciones que utilizaron modelos estándares. Nuestro modelo utilizado es mucho más ligero con respecto al número de capas y neuronas utilizadas en comparación a los modelos estándar. Y también que en estos modelos se utilizaron estrategias para reducir el overfitting.

Conclusiones

1. Con esta investigación se afirma que una solución que use Machine learning y CNN para la clasificación de leucocitos en imágenes microscópicas tendría un grado de error que no sería perjudicial en comparación a otras soluciones.
2. Para obtener los grados de errores de procedimientos de laboratorios clínicos existen varias formas. En caso de un procedimiento a mano, los manuales de procedimientos clínicos publicados por universidades o entidades públicas normalmente mencionan este parámetro en las descripciones de estos procedimientos. Para procedimientos realizados por maquinas especializadas; estas máquinas siempre tienen fichas técnicas que el fabricante tiene que brindar al público por ley. Estas fichas técnicas siempre contienen parámetros de error para cada procedimiento que realizan las maquinas.
3. Durante el desarrollo del preprocesamiento de datasets se presentó un problema que era la diferencia del número de muestras entre un tipo de leucocito y los demás tipos. La solución empleada fue utilizar procesos de data augmentation para aumentar el número de muestras de ese tipo en específicamente. Esta solución fue tan útil, que se decidió utilizarla también en el total del dataset para tener una mayor diversidad de muestras.
4. La metodología utilizada para determinar la arquitectura de la red neuronal artificial fue de utilizar una arquitectura base recomendad por TensortFlow para la clasificación de imágenes e ir variando las variables de esta arquitectura para buscar los mejores resultados posibles. Esta metodología fue exitosa ya que se vio mejoras en las variables accuracy y loss en cada nueva iteración. También se utiliza técnicas para tratar el overfitting (valores de regularización y capas dropout) que fueron utilizadas en los últimos ciclos. Estas técnicas redujeron exitosamente el overfitting durante el proceso de entrenamiento.
5. Las redes neuronales es una opción tecnológica aplicable en soluciones donde se busque predecir un resultado. Pero no es cien por cien confiables, dependen de la estrategia que el desarrollador siga. Tienen muchas variables que afectan a su desempeño; como numero de datos, numero de capas, tipos de capas, arquitectura del modelo, valides de datasets, etc. Todas estas variables se reflejan en los dos principales problemas que se presentan en estas soluciones; el underfitting y el overfitting.

6. El uso de Colab como herramienta de cómputo en la nube fue de gran ayuda, ya que permitió que el proceso de entrenamiento se reduzca de 5 – 6 horas por modelo a 45 minutos aproximadamente por modelo.
7. Al momento de comparar con otras investigaciones similares se observó que el uso de esquemas de redes neuronales artificiales planteados por otras entidades investigativas; como Google, Amazon u otros; es la solución más común utilizada al momento de resolver problemas de clasificación. Esta observación apoya a la investigación; puesto que se vio que utilizando estas arquitecturas se puede llegar a grados de error incluso más bajos de los encontrados en esta investigación.

Recomendaciones

- Esta investigación es de naturales científica, se buscaba validar si una solución que utilice CNN seria valida, y se concluyó que si es válido su uso. Por lo tanto, se recomienda investigar sobre la creación de un software que utilice redes neuronales artificiales y otros equipos (microscopios, cámaras, etc.) para la captura, clasificación y conteo de leucocitos para un entorno practico. Y también comparar más características de las diferentes soluciones como tiempo, uso de recursos, etc.
- En la investigación se utilizaron dataset de diferentes fuentes para el entrenamiento, validación y testeo de los modelos. Se recomienda hacer una investigación donde se tenga apoyo de personal de laboratorios clínicos para la generación de un dataset más homogéneo y uniforme; con el cual poder verificar si con esta mejora se puede encontrar un grado de error más bajo.
- CNN es una gran herramienta para trabajar con machine learning e imágenes. Se recomienda el uso de CNN para investigaciones de clasificación de imágenes para nuevas soluciones en diferentes ámbitos (clasificación de frutos, bacterias, imágenes industriales, etc.).
- Durante la investigación, se observó que las redes neuronales permiten la creación de modelos de predicción sin la necesidad de una ecuación previamente definida. Por esta razón, se recomienda el uso de redes neuronales para la validación de investigaciones previas de predicción, modelos predictivos u otras investigaciones donde se utilizaron metodologías matemáticas para la generación de ecuaciones de predicción. Estas investigaciones podrían utilizar los datos de las investigaciones pasadas para el entrenamiento, validación y testeo de sus modelos de redes neuronales.
- Se recomienda hacer investigaciones para el análisis con machine learning de imágenes microscópicas utilizadas en otros procedimientos como el análisis de orina para la clasificación de residuos bajo microscopio.
- Se recomienda investigar sobre la creación y clasificación de los métodos de data augmentation para su uso practico. Hay muchos métodos de data augmetation pero su uso no tiene ninguna clasificación aparte de la forma de alteración de la imagen.

Referencias

A. Villena, P. López-Fierro, B. Razquin, & V. Fernández. (n.d.). *Inicio de e-Histología*. Universidad de León. Retrieved June 21, 2021, from http://www.e-histologia.unileon.es/1inicio/home/Inicio800x600.html

Aleju. (n.d.). *aleju/imgaug: Image augmentation for machine learning experiments.* Retrieved June 24, 2021, from https://github.com/aleju/imgaug

Bastidas, O. (2016). *Technical Note-Neubauer Chamber Cell Counting*. www.celeromics.com

Beckman Coulter. (n.d.). *DxH 900 High-volume Lab Hematology Analyzer | Beckman Coulter*. Beckman Coulter. Retrieved October 28, 2021, from https://www.beckmancoulter.com/products/hematology/dxh-900#/especificaciones

BioSystems, S. A. (2018). *HA3 Hematology Analyzer*. www.biosystems.es

Brüel, A., Christensen, E. I., Tranum-Jensen, J., Qvortrup, K., & Geneser, F. (2015). *Geneser Histologia 4ta edicion* (E. M. PANAMERICANA, Ed.; 4ta edicio). FOItatterne og Munksgaard , Kóbenhavn.

Bustamante Alvarez, R. (2019). Diseño de un algoritmo para la automatización del conteo de células del tejido sanguíneo mediante procesamiento digital de imágenes. In *Universidad Nacional Mayor de San Marcos*.

Caillahua, P. E. C. (2019). *Estimación de área glaciar utilizando redes neuronales convolucionales u-net en imágenes multiespectrales sentinel 2 en el glaciar AUSANGATE, 2019*.

CAPTODAY. (2018). *HEMATOLOGY ANALYZERS*. https://www.captodayonline.com/2018/ProductGuides/11-18_CAPTODAY_HematologyAnalyzers.pdf

De Salud, M. (n.d.). *Instituto de Salud Pública RECOMENDACIONES PARA LA TINCIÓN DE FROTIS SANGUÍNEOS PARA LA LECTURA DEL HEMOGRAMA*.

DEEPLIZARD. (2018). *Max Pooling in Convolutional Neural Networks explained - deeplizard*. https://deeplizard.com/learn/video/ZjM_XQa5s6s

Dr. Eduardo Rivadeneyra Domínguez. (n.d.). *GUÍA DE LABORATORIO DE HEMATOLOGÍA* (UNIVERSIDAD VERACRUZANA & Facultad de Química Farmacéutica Biológica, Eds.). Retrieved October 20, 2021, from https://www.uv.mx/qfb/files/2020/09/Guia-de-Hematologia-Laboratorio.pdf

Drucker Diagnostics. (2018). *The QBC ® AUTOREAD™ PLUS System*. www.druckerdiagnostics.com

Duan, Y., Wang, J., Hu, M., Zhou, M., Li, Q., Sun, L., Qiu, S., & Wang, Y. (2019). Leukocyte classification based on spatial and spectral features of microscopic hyperspectral images. *Optics and Laser Technology*, *112*(November 2018), 530–538. https://doi.org/10.1016/j.optlastec.2018.11.057

Franck SEGUY. (n.d.). *The Automated Hematology Analyzer, the Pentra 80*.

Genrui Biotech Inc. (2017). *Technical Specifications KT-6610*. www.genrui-bio.com

Harshanand, B. A., & Sangaiah, A. K. (2020). Comprehensive Analysis of Deep Learning Methodology in Classification of Leukocytes and Enhancement Using Swish Activation Units. *Mobile Networks and Applications*. https://doi.org/10.1007/s11036-020-01614-3

Hegde, R. B., Prasad, K., Hebbar, H., & Singh, B. M. K. (2019). Image Processing Approach for Detection of Leukocytes in Peripheral Blood Smears. *Journal of Medical Systems*, *43*(5). https://doi.org/10.1007/s10916-019-1219-3

Jason Brownlee. (2019, August 12). *Overfitting and Underfitting With Machine Learning Algorithms*. https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (n.d.). *ImageNet Classification with Deep Convolutional Neural Networks*. Retrieved October 26, 2021, from http://code.google.com/p/cuda-convnet/

L. Mescher, A. (2018). Junqueira's basic histology. In *Journal of Chemical Information and Modeling* (Vol. 15). https://doi.org/10.1017/CBO9781107415324.004

Macalupú, S. Z. (2013). *Procedimientos de laboratorio : manual : laboratorios locales I : laboratorios locales II* (Ministerio de Salud & Instituto Nacional de Salud, Eds.).

Macawile, M. J., Quiñones, V. V., Ballado, A., Cruz, J. dela, & Caya, M. V. (2018). White blood cell classification and counting using convolutional neural network. *2018 3rd International Conference on Control and Robotics Engineering, ICCRE 2018*, 259–263. https://doi.org/10.1109/ICCRE.2018.8376476

Martínez-Castro, J., Reyes-Cadena, S., & Felipe-Riverón, E. (2014). Leukocytes detection, classification and counting in smears of peripheral blood. *Revista Mexicana de Ingenieria Biomedica*, *35*(1), 41–51.

Mooney, P. (2018, January 10). *Blood Cell Images | Kaggle*. Kaggle. https://www.kaggle.com/paultimothymooney/blood-cells

NeoMedica. (2016). *5-Diff Automated Hematology Analyzers*. https://neomedica.rs/wp-content/uploads/2020/01/Hematology-catalogue.pdf

Olivera, G. E. R., Millán, P. V., & Santos, P. J. D. L. (2017). *Manual de prácticas para el laboratorio de hematología*.

Parés, J., Borda, N., Santiago, S. D., Benito, C., & Aranda, C. (2015). Evaluación de los parámetros de desempeño de un contador hematológico. *Acta Bioquím Clín Latinoam*, *49*(4), 399–407.

PATRICIA VIDAL MILLÁN, & PABLO JUÁREZ DE LOS SANTOS. (2020). *Manual de Laboratorio de Hematología* (1st ed.). UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO.

Pawlina, W. (2015). Histología Texto Y Atlas Correlación con Biología Molecular y Celular. In *Ross Histología texto y atlas*.

Ramirez, A., Pacheco, A., & Telles, J. (2019). Mapping vegetation, water bodies and urban areas in PeruSAT-1 satellite imagery. *2019 22nd Symposium on Image, Signal Processing and Artificial Vision, STSIVA 2019 - Conference Proceedings*, 1–4. https://doi.org/10.1109/STSIVA.2019.8730269

Raschka, S. (2020). *Why is Nearest Neighbor a Lazy Algorithm?* https://sebastianraschka.com/faq/docs/lazy-knn.html

Raschka, S., & Mirjalili, V. (2019). *Python Machine Learning* (Third Edit, Issue January 2010). Packt Publishing.

Salin, E. D., & Winston, P. H. (1992). Machine Learning and Artificial Intelligence an Introduction. In S. N. Switzerland (Ed.), *Analytical Chemistry* (1ra edicio, Vol. 64, Issue 1). Springer, Cham. https://doi.org/10.1021/ac00025a742

Sampieri, R. H., Collado, C. F., & Lucio, M. del P. B. (2014). *Metodología de la investigación Sexta edición* (I. EDITORES, Ed.; 6ta Edicio). McGRAW-HILL.

Sinothinker. (2020). *Sinothinker Catalogue 2020*. Sinothinker.

tf.keras.preprocessing.image_dataset_from_directory. (n.d.). Retrieved May 19, 2021, from https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image_dataset_from_directory

Villegas Noguera, Á. L. (2019). *Desarrollo de un sistema de análisis automático para la segmentación, clasificación y conteo de leucocitos en imágenes digitales de frotis de sangre periférica*. http://mriuc.bc.uc.edu.ve/handle/123456789/8271?show=full

Wang, Y., & Cao, Y. (2020). Human peripheral blood leukocyte classification method based on convolutional neural network and data augmentation. *Medical Physics*, *47*(1), 142–151. https://doi.org/10.1002/mp.13904

ZAMBRANO, M. M., CORTIJO, C. M., & 1. (2005). *MANUAL DE PROCEDIMIENTOS DE LABORATORIO EN TÉCNICAS BÁSICAS DE HEMATOLOGÍA*. Ministerio de Salud, Instituto Nacional de Salud.

Saha, S. (2018). A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way | by Sumit Saha | Towards Data Science. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

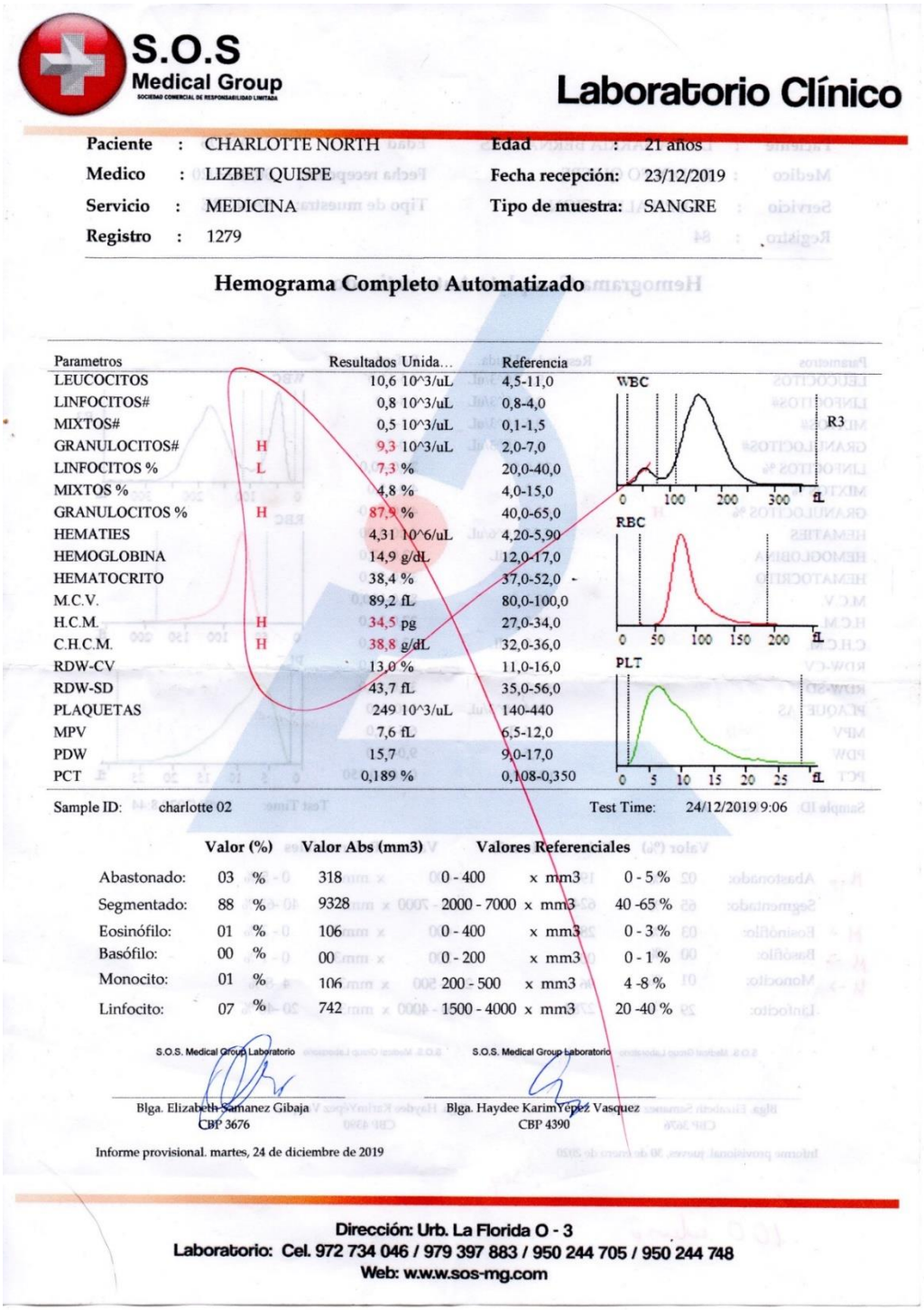
Pereira, S. R. T., Arteaga, I. H., Zambrano, S. J. C., Troya, A. H., & Pérez, J. C. A. (2016). Descubrimiento de patrones de desempeño académico con árboles de decisión en las competencias genéricas de la formación profesional. In Ediciones Universidad Cooperativa de Colombia (pp. 63–86). Ediciones Universidad Cooperativa de Colombia. https://doi.org/10.16925/9789587600490

IBM. (2020). What are Convolutional Neural Networks? | IBM. IBM Cloud Education. https://www.ibm.com/cloud/learn/convolutional-neural-networks

Capítulo 6. Anexos

6.1. Muestra de ficha de datos de un hemograma

Figura 107
Muestra de ficha de datos de un hemograma 1



S.O.S Medical Group
SOCIEDAD COMERCIAL DE RESPONSABILIDAD LIMITADA

Laboratorio Clínico

Paciente	:	CHARLOTTE NORTH	**Edad**	:	21 años
Medico	:	LIZBET QUISPE	**Fecha recepción:**		23/12/2019
Servicio	:	MEDICINA	**Tipo de muestra:**		SANGRE
Registro	:	1279			

Hemograma Completo Automatizado

Parametros		Resultados	Unida...	Referencia
LEUCOCITOS		10,6	10^3/uL	4,5-11,0
LINFOCITOS#		0,8	10^3/uL	0,8-4,0
MIXTOS#		0,5	10^3/uL	0,1-1,5
GRANULOCITOS#	H	9,3	10^3/uL	2,0-7,0
LINFOCITOS %	L	7,3	%	20,0-40,0
MIXTOS %		4,8	%	4,0-15,0
GRANULOCITOS %	H	87,9	%	40,0-65,0
HEMATIES		4,31	10^6/uL	4,20-5,90
HEMOGLOBINA		14,9	g/dL	12,0-17,0
HEMATOCRITO		38,4	%	37,0-52,0
M.C.V.		89,2	fL	80,0-100,0
H.C.M.	H	34,5	pg	27,0-34,0
C.H.C.M.	H	38,8	g/dL	32,0-36,0
RDW-CV		13,0	%	11,0-16,0
RDW-SD		43,7	fL	35,0-56,0
PLAQUETAS		249	10^3/uL	140-440
MPV		7,6	fL	6,5-12,0
PDW		15,7		9,0-17,0
PCT		0,189	%	0,108-0,350

WBC R3 0 100 200 300 fL
RBC 0 50 100 150 200 fL
PLT 0 5 10 15 20 25 fL

Sample ID: charlotte 02 Test Time: 24/12/2019 9:06

	Valor (%)		Valor Abs (mm3)	Valores Referenciales		
Abastonado:	03	%	318	0 - 400	x mm3	0 - 5 %
Segmentado:	88	%	9328	2000 - 7000	x mm3	40 -65 %
Eosinófilo:	01	%	106	0 - 400	x mm3	0 - 3 %
Basófilo:	00	%	00	0 - 200	x mm3	0 - 1 %
Monocito:	01	%	106	200 - 500	x mm3	4 -8 %
Linfocito:	07	%	742	1500 - 4000	x mm3	20 -40 %

S.O.S. Medical Group Laboratorio
Blga. Elizabeth Samanez Gibaja
CBP 3676

S.O.S. Medical Group Laboratorio
Blga. Haydee KarimYepez Vasquez
CBP 4390

Informe provisional. martes, 24 de diciembre de 2019

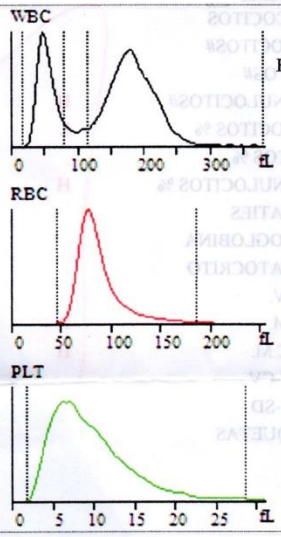
Dirección: Urb. La Florida O - 3
Laboratorio: Cel. 972 734 046 / 979 397 883 / 950 244 705 / 950 244 748
Web: w.w.w.sos-mg.com

Figura 108
Muestra de ficha de datos de un hemograma 2

Paciente	:	LUIZA SARRIA BERNARDES	**Edad**	:	21 AÑOS
Medico	:	RODOLFO QUISPE	**Fecha recepción:**		30/01/2020
Servicio	:	HOSPITALIZACION	**Tipo de muestra:**		SANGRE
Registro	:	84			

Hemograma Completo Automatizado

Parametros		Resultados	Unida...	Referencia
LEUCOCITOS		9,6	10^3/uL	4,5-11,0
LINFOCITOS#		2,8	10^3/uL	0,8-4,0
MIXTOS#		0,5	10^3/uL	0,1-1,5
GRANULOCITOS#		6,3	10^3/uL	2,0-7,0
LINFOCITOS %		29,0	%	20,0-40,0
MIXTOS %		5,2	%	4,0-15,0
GRANULOCITOS %	H	65,8	%	40,0-65,0
HEMATIES		5,35	10^6/uL	4,20-5,90
HEMOGLOBINA		15,7	g/dL	12,0-17,0
HEMATOCRITO		47,6	%	37,0-52,0
M.C.V.		89,1	fL	80,0-100,0
H.C.M.		29,3	pg	27,0-34,0
C.H.C.M.		32,9	g/dL	32,0-36,0
RDW-CV		13,8	%	11,0-16,0
RDW-SD		41,1	fL	35,0-56,0
PLAQUETAS		294	10^3/uL	140-440
MPV		7,6	fL	6,5-12,0
PDW		15,5		9,0-17,0
PCT		0,223	%	0,108-0,350



Sample ID: LUIZA 03 Test Time: 30/01/2020 8:44

		Valor (%)	Valor Abs (mm3)	Valores Referenciales		
M ->	Abastonado:	02 %	192	0 - 400	x mm3	0 - 5 %
	Segmentado:	65 %	6240	2000 - 7000	x mm3	40 -65 %
M S	Eosinófilo:	03 %	288	0 - 400	x mm3	0 - 3 %
M ->	Basófilo:	00 %	00	0 - 200	x mm3	0 - 1 %
M ->	Monocito:	01 %	96	200 - 500	x mm3	4 -8 %
	Linfocito:	29 %	2784	1500 - 4000	x mm3	20 -40 %

S.O.S. Medical Group Laboratorio — Blga. Elizabeth Samanez Gibaja, CBP 3676

S.O.S. Medical Group Laboratorio — Blga. Haydee KarimYépez Vasquez, CBP 4390

Informe provisional. jueves, 30 de enero de 2020

Seg.

100 edema

6.2. Sobrecarga de RAM de sesión de Google Colab

Figura 109
Sobrecarga de RAM de sesión de Google Colab