Universidad Nacional de Loja

Facultad de la Energía, las Industrias y los Recursos Naturales No Renovables

Carrera de Computación

Creación de un modelo para la clasificación de imágenes de la mancha bacteriana en la hoja de tomate utilizando el modelo VGG16.

Creation of a model for image classification of bacterial leaf spot on tomato using the VGG16 model.

Trabajo de Integración Curricular previa a la obtención del título de Ingeniera en Ciencias de la Computación

AUTORA:

Jennifer Elizabeth Quizhpe Hurtado

DIRECTOR:

Ing. Oscar Miguel Cumbicus Pineda, Mg. Sc

Loja – Ecuador

2025

Certificación

Loja, 31 de enero del 2025

Ing. Oscar Miguel Cumbicus Pineda, Mg.Sc
DIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR

CERTIFICO:

Que he revisado y orientado todo el proceso de la elaboración del Trabajo de Integración Curricular denominado: **Creación de un modelo para la clasificación de imágenes de la mancha bacteriana en la hoja de tomate utilizando el modelo VGG16**¸ previo a la obtención del título de **Ingeniera en Ciencias de la Computación**, de autoría de la estudiante: **Jennifer Elizabeth Quizhpe Hurtado**, con cédula de identidad Nro. **1105653172**, una vez que el trabajo cumple con todos los requisitos exigidos por la Universidad Nacional de Loja, para el efecto, autorizo la presentación del mismo para su respectiva sustentación y defensa.

Firmado electrónicamente por:
OSCAR MIGUEL
CUMBICUS PINEDA

Ing. Oscar Miguel Cumbicus Pineda, Mg. Sc
Director del Trabajo de Integración Curricular

Autoría

Yo, **Jennifer Elizabeth Quizhpe Hurtado**, declaro ser autora del presente Trabajo de Integración Curricular y eximo expresamente a la Universidad Nacional de Loja y a sus representantes jurídicos, de posibles reclamos y acciones legales, por el contenido del mismo. Adicionalmente acepto y autorizo a la Universidad Nacional de Loja la publicación de mi Trabajo de Integración Curricular, en el Repositorio Digital Institucional – Biblioteca Virtual.

Firma: 

Cédula de identidad: 1105653172

Fecha: 31 de enero del 2025

Correo electrónico: jennifer.quizhpe@unl.edu.ec

Teléfono: 0995913572

Carta de autorización por parte de la autora, para consulta, reproducción parcial o total y/o publicación electrónica del texto completo, del Trabajo de Integración Curricular

Yo, **Jennifer Elizabeth Quizhpe Hurtado**, declaro ser el autora del Trabajo de Integración Curricular denominado: **Creación de un modelo para la clasificación de imágenes de la mancha bacteriana en la hoja de tomate utilizando el modelo VGG16**, autorizo al sistema Bibliotecario de la Universidad Nacional de Loja para que, con fines académicos, muestre la producción intelectual de la Universidad, a través de la visibilidad de su contenido en el Repositorio Institucional.

Los usuarios pueden consultar el contenido de este trabajo en el Repositorio Institucional, en las redes de información del país y del exterior con las cuales tenga convenio la Universidad.

La Universidad Nacional de Loja, no se responsabiliza por el plagio o copia del Trabajo de Integración Curricular o de Titulación que realice un tercero.

Para constancia de esta autorización, suscribo, en la ciudad de Loja, a los treinta y un días del mes de enero de dos mil veinticinco.

Firma:

Autora: Jennifer Elizabeth Quizhpe Hurtado
Cédula de identidad: 1105653172
Dirección: Loja
Correo electrónico: jennifer.quizhpe@unl.edu.ec
Teléfono: 0995913572

DATOS COMPLEMENTARIOS:

Director del Trabajo de Integración Curricular: Ing. Oscar Miguel Cumbicus P., Mg.Sc.

Dedicatoria

Quiero dedicar de manera especial este trabajo de integración curricular a mi abuelita Hortencia quien desde el cielo guía mis pasos y me inspira a seguir adelante con fuerza y determinación. Su apoyo, amor y sabiduría fueron pilares fundamentales para hoy tener este logro.

A mi pequeño Matías, quien han sido mi motor, inspiración y compañero a lo largo de este viaje dándome las fuerzas para seguir adelante y no rendirme. Este logro también es para ti, deseando que siempre veas en mí un ejemplo de esfuerzo y superación.

A mis padres Jorge y Luz por su apoyo incondicional, paciencia y sacrificios. Ustedes han sido mi mayor ejemplo de dedicación y perseverancia, todo lo que he logrado en mi vida ha sido gracias a los valores y enseñanzas que me han inculcado.

A mis hermanas Majo y Melanie por ser mi refugio, apoyo y compañeras incondicionales en los momentos difíciles, su presencia hizo que todo lo que he vivido sea más llevadero.

Con todo mi amor y gratitud dedico este trabajo a todos ustedes que han sido mi mayor fortaleza, inspiración y razón de ser.

Jennifer Elizabeth Quizhpe Hurtado

Agradecimiento

Agradezco a Dios por concederme salud, fortaleza y entendimiento necesarios para enfrentar con determinación cada desafío que se me presentó a lo largo de este proceso, su guía espiritual me permitió mantener la perseverancia y claridad para culminar con éxito esta investigación.

A mis padres y hermanas les estoy enteramente agradecida por ser mi principal fuente de apoyo emocional y motivación constante, su amor incondicional, palabras de aliento y la confianza depositada en mis capacidades fueron esenciales para superarme. Ustedes han sido un pilar invaluable que me ha inspirado a esforzarme y dar lo mejor de mí en cada etapa de esta travesía académica.

A mi pareja, agradezco de corazón su compañía, comprensión y palabras de aliento llenas de optimismo que me brindaron la fuerza necesaria para mantenerme enfocada y motivada durante este proceso.

A mi director el Ing. Oscar Miguel Cumbicus Pineda extiendo mi más sincero reconocimiento y gratitud por su dedicación, paciencia y compromiso durante el desarrollo de esta investigación, su experiencia y consejos fueron fundamentales para culminar y superar los retos científicos y metodológicos que se me presentaron. Su guía no solo enriqueció los resultados alcanzados, sino que también me permitió fortalecer mis competencias investigativas.

A mis amigos y compañeros quienes con su apoyo y colaboración contribuyeron a enriquecer este trabajo, les agradezco por cada intercambio de ideas, motivación constante y sobre todo por ser parte de esta etapa de mi vida, su compañía hizo de este proceso una vivencia más significativa.

Finalmente, agradezco a la Universidad Nacional de Loja especialmente a la Facultad de la Energía, las Industrias y los Recursos Naturales no Renovables por brindarme una formación académica integral y de alta calidad, su compromiso con la excelencia académica y el desarrollo científico fue determinante para la culminación de esta investigación.

A todos agradezco por ser parte de este logro que representa no solo el cierre de una etapa importante sino marca el inicio de nuevos desafíos y metas, los cuales emprenderé con la misma dedicación y compromiso que he demostrados hasta ahora.

Jennifer Elizabeth Quizhpe Hurtado

Índice de contenidos

Índice de tablas:

Índice de figuras:

Índice de ecuaciones:

Índice de anexos:

1. Título

Creación de un modelo para la clasificación de imágenes de la mancha bacteriana en la hoja de tomate utilizando el modelo VGG16.

Creation of a model for image classification of bacterial leaf spot on tomato using the VGG16 model.

2. Resumen

Los avances de la inteligencia artificial han revolucionado sectores como la agricultura, destacando su capacidad de optimizar procesos mediante el análisis automatizado de datos con ello, las redes neuronales convolucionales han demostrado ser herramientas fundamentales para el procesamiento y clasificaciones de imágenes, permitiendo identificar patrones complejos de manera eficiente. Estas tecnologías han sido aplicadas en la detección temprana de enfermedades, monitoreo del crecimiento de cultivos y evaluación de la calidad de productos agrícolas, proporcionando soluciones innovadoras. En este contexto, el trabajo de integración curricular se centró en el diseño, ajuste y evaluación de un modelo de aprendizaje profundo basado en la arquitectura VGG16 para la clasificación automatizada de imágenes de hojas de tomate afectadas por la mancha bacteriana. La metodología utilizada fue CRISP-ML(Q) estructurada en tres fases principales: ingeniería de datos, donde se identificaron y recolectaron imágenes del repositorio Kaggle, las cuales fueron procesadas mediante técnicas de limpieza, balanceo y aumento de datos, dando como resultado un conjunto final de 3816 imágenes, lo que incrementó la diversidad y calidad del conjunto de datos. En la fase de ingeniería de modelos, se implementó la arquitectura VGG16 preentrenada en ImageNet, optimizando hiperparámetros y ajustando sus capas de clasificación para adaptarse a la tarea requerida. El entrenamiento se realizó en la plataforma Google Colab, obteniendo como resultado un modelo con desempeño óptimo, el cual se evaluó como parte de la fase de evaluación de modelos, alcanzando un 99.73% de precisión global y una pérdida mínima de 0.78%, superando estudios previos con configuraciones similares. Además, se aplicó la técnica de Zero-Shot Learning para evaluar la capacidad de generalización del modelo con datos recolectados en entornos reales, dando como exactitud el 75%. El modelo desarrollado optimiza el uso de pesticidas, mejora la productividad y fomenta la sostenibilidad agrícola, por lo que se recomienda ampliar el conjunto de datos con imágenes obtenidas en entornos reales y explorar su integración con tecnologías móviles.

Palabras clave: VGG16, precisión global, redes neuronales convolucionales, clasificación de imágenes, CRISP-ML(Q), mancha bacteriana.

Abstract

Advancements in artificial intelligence have revolutionized sectors such as agriculture, highlighting its ability to optimize processes through automated data analysis. In this context, convolutional neural networks have proven to be fundamental tools for image processing and classification, enabling the efficient identification of complex patterns. These technologies have been applied in early disease detection, crop growth monitoring, and quality assessment of agricultural products, providing innovative solutions. This curricular integration project focused on designing, tuning, and evaluating a deep learning model based on the VGG16 architecture for the automated classification of tomato leaf images affected by bacterial spots. The methodology used was CRISP-ML(Q), structured into three main phases:Data engineering, where images were identified and collected from the Kaggle repository. These images were processed using cleaning, balancing, and data augmentation techniques, resulting in a final dataset of 3,816 images, increasing both diversity and quality.Model engineering, where the pre-trained VGG16 architecture on ImageNet was implemented, optimizing hyperparameters and adjusting classification layers to fit the required task.Model evaluation, where training was conducted on the Google Colab platform, yielding a model with optimal performance. The final evaluation showed an overall accuracy of 99.73% and a minimal loss of 0.78%, surpassing previous studies with similar configurations. Additionally, the zero-shot learning technique was applied to assess the model's generalization ability using data collected from real-world environments, achieving an accuracy of 75%. The developed model optimizes pesticide use, enhances productivity, and promotes agricultural sustainability. Therefore, it is recommended to expand the dataset with images obtained in real-world environments and explore its integration with mobile technologies.

Keywords: VGG16, accuracy, convolutional neural networks, image classification, CRISP-ML(Q), bacterial spot.

3. Introducción

La inteligencia artificial (IA) ha transformado significativamente diversas áreas del conocimiento siendo la visión por computadora uno de los campos con mayor impacto en los últimos años, por lo que las redes neuronales convolucionales (CNN) se han consolidado como una herramienta clave para la clasificación y análisis de imágenes esto debido a su gran capacidad para identificar patrones complejos de forma automatizada y eficiente. Actualmente son algunos modelos existentes que permiten realizar este tipo de tarea por lo que en la presente investigación se centra específicamente en la arquitectura VGG16 aplicada a la clasificación de imágenes de hojas de tomate afectadas por la mancha bacteriana, enfermedad causada por *Xanthomonas spp.* [1] la cual representa una amenaza para la calidad y producción agrícola. Este modelo es reconocido por su capacidad de aprendizaje profundo y su eficacia en tareas de clasificación lo que lo convierte en una opción adecuada para abordar este tipo de problemas [24].

El presente trabajo se fundamenta en investigaciones previas las cuales han explorado el uso de redes neuronales convolucionales para clasificar imágenes agrícolas. Estudios recientes como [3], [4], [5] y [6] demostraron la eficacia de arquitecturas como AlexNet, ResNet y VGG en tareas similares, aunque presentaron desafíos relacionados con la complejidad computacional y la adaptabilidad frente a diferentes conjuntos de datos. Por lo que la selección de la arquitectura VGG16 como base para esta investigación se centró en los estudios [7], [8], [9] y [10] los cuales destacan el equilibrio entre simplicidad y rendimiento del arquitectura lo que la convierte en una opción adecuada para abordar problemas como el mencionado anteriormente.

El modelo propuesto busca adaptarse a las necesidades del sector agrícola mediante el desarrollo de una solución tecnológica que permita optimizar la detección temprana de la mancha bacteriana permitiendo a los agricultores tomar decisiones informadas sobre el manejo de sus cultivos. Esto generará una reducción del uso de pesticidas, un aumento en la eficiencia de la producción y una mejora en la sostenibilidad de las prácticas agrícolas promoviendo la innovación en el sector e impulsando su competitividad en un mercado globalizado. Por lo que surge como pregunta de investigación: ¿Qué porcentaje global de acierto se obtendrá al clasificar imágenes de la mancha bacteriana en la hoja del tomate de riñón ajustando el modelo preentrenado VGG16?

Para dar respuesta a la pregunta en cuestión se establece como objetivo principal determinar el porcentaje global de acierto que se obtendrá al clasificar imágenes de la mancha bacteriana ajustando el modelo preentrenado para ello se adaptaron fases de la metodología

CRISP-ML(Q) con el fin de permitir el desarrollo de las tareas necesarias para ajustar y evaluar el desempeño del modelo VGG16 mediante métricas de rendimiento relevantes como exactitud, sensibilidad y F-1Score partiendo de la implementación de técnicas de preprocesamiento como limpieza, balanceo y aumento las cuales combinadas con ajustes específicos de los hiperparámetros permiten mejorar significativamente la capacidad del modelo de clasificar imágenes en condiciones controladas.

El desarrollo del trabajo de integración curricular (TIC) se enfocó en cumplir con todas las fases y tareas definidas por el marco de trabajo para el ajuste, entrenamiento, evaluación y análisis de un modelo de aprendizaje automático ofreciendo una perspectiva innovadora sobre el uso de redes neuronales convolucionales para la detección de enfermedades en cultivos, específicamente la mancha bacteriana en la hoja de tomate. Su relevancia radica en la aplicación práctica de tecnologías avanzadas para resolver problemas críticos dentro del sector agrícola contribuyendo al desarrollo de soluciones sostenibles y eficientes que beneficien tanto a los agricultores como al medio ambiente, esto no solo responde a las necesidades actuales del sector, sino que también abre nuevas posibilidades para el uso de la inteligencia artificial en la agricultura moderna.

Los beneficios destacados del presente trabajo de integración curricular incluyen el desarrollo de un conjunto de imágenes de hojas de tomate afectadas por la mancha bacteriana estandarizado y enriquecido mediante técnicas de balanceo y aumento de datos, el cual puede ser utilizado en futuras investigaciones facilitando la replicación y entrenamiento de nuevos modelos sin la necesidad de ajustar manualmente estas técnicas. Así mismo, se destaca la obtención de un modelo empaquetado completamente funcional basado en la arquitectura VGG16 y diseñado específicamente para la clasificación de imágenes de la mancha bacteriana en la hoja de tomate.

4. Marco teórico

4.1. Antecedentes

Actualmente las redes neuronales convolucionales han sido una de las tecnologías con mayor uso para la clasificación de imágenes, estas son consideradas como una de las técnicas más favorables de la visión por computadora [2]. Hoy en día son algunos los modelos existentes que permiten realizar este tipo de tarea, siendo el caso de VGG16 propuesto por K. Simonyan y A. Zisserman de la Universidad de Oxford, el cual consta de 16 capas, 13 capas CNN y 3 capas totalmente conectadas, fue entrenado con el conjunto de datos ImageNet, el mismo que cuenta con 1,4 millones de imágenes pertenecientes a 1000 clases, según el artículo "Very Deep Convolutional Networks for Large-Scale Image Recognition" este modelo logra alcanzar el 92.7% de precisión [3].

4.1.1 Mancha bacteriana

La mancha bacteriana o "*Bacterial spot*", es una patología causada por algunas especies de la bacteria *Xanthomonas spp* [1], esta resulta difícil de controlar cuando las condiciones climáticas permiten su multiplicación. En la **Figura 1** se puede observar la característica principal de esta enfermedad, misma que consiste en la aparición de pequeñas manchas oscuras en hojas, tallos o frutos las cuales pueden ocasionar deformaciones reduciendo la calidad del producto [4]. Este tipo de patógeno puede sobrevivir acompañado de malas hierbas, ingresa en las plantas sanas a través de estomas o heridas, debido a condiciones húmedas la infección logra desarrollarse con mayor facilidad [5].

Figura 1 Hoja de tomate afectada por la mancha bacteriana *[75]*

Algunos de los síntomas comunes son [4]:

- Lesiones de color verde pálido o amarillo con márgenes irregulares en la superficie superior de los foliolos.

- Crecimiento de hongos debajo de las lesiones.

Actualmente esta enfermedad se ha convertido en una amenaza constante para los agricultores, puesto que afecta la calidad del producto reduciendo así la capacidad de producción [6]. Es fundamental el constante monitoreo, ya que al detectarla en etapa temprana puede ser tratada con la menor cantidad de químicos, prevaleciendo la calidad del producto. Sin embargo, debido a la gran cantidad de productos sembrados es difícil realizar un monitoreo continuo y equitativo ya sea por falta de personal o tiempo, dando como resultado sembríos mal diagnosticados [7].

4.2. Fundamentación Teórica

4.2.1. Inteligencia artificial

El concepto de inteligencia artificial aborda múltiples tecnologías las cuales permiten a las máquinas sentir, actuar y aprender [8]. Uno de los conceptos iniciales se obtuvo durante la prueba de Turing, en la cual se definió a la IA como la capacidad de comunicación entre humano y máquina donde el criterio de juicio es binario [9]. En la actualidad, esta se considera como la ciencia e ingeniería de crear máquinas inteligentes, relacionada con la tarea de utilizar ordenadores para comprender la inteligencia humana, teniendo en cuenta que la IA no se limite a métodos biológicamente observables [10].

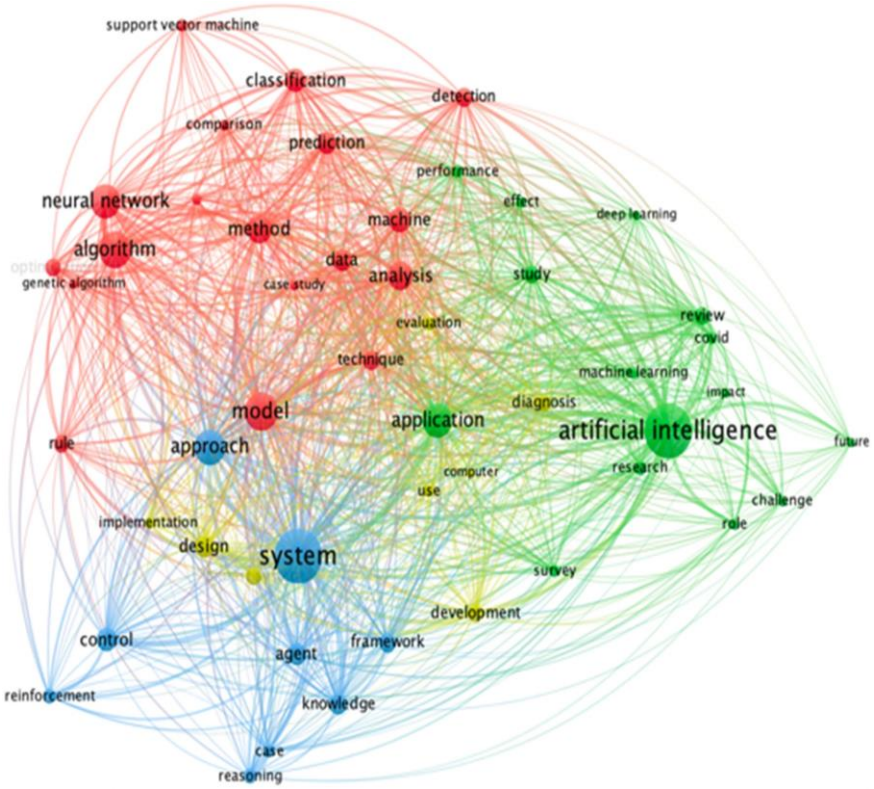

Figura 2 Red semántica IA *[79]*

Este concepto ha logrado tener un impacto mayor, tal como sucedió con las computadoras en la era de la información, convirtiendo a la IA en un pilar fundamental de la tecnología en la era contemporánea [9]. En la **Figura 2** se observa cómo se conectan y agrupan algunos de los conceptos importantes de la inteligencia artificial.

4.2.1.1 Machine Learning

Es una rama de la inteligencia artificial, la cual funciona a través de algoritmos que dotan a los ordenadores la capacidad de aprender sin ser programados, permiten analizar grandes cantidades de datos, extraer patrones y elaborar predicciones [11]. Esta disciplina fue propuesta en 1959 con un enfoque de visión por computadora, implica un proceso de aprendizaje basado en la experiencia, donde el rendimiento del modelo se mide a través de métricas que permiten determinar la precisión y comportamiento del mismo, ya sea en un ambiente real o controlado [12].

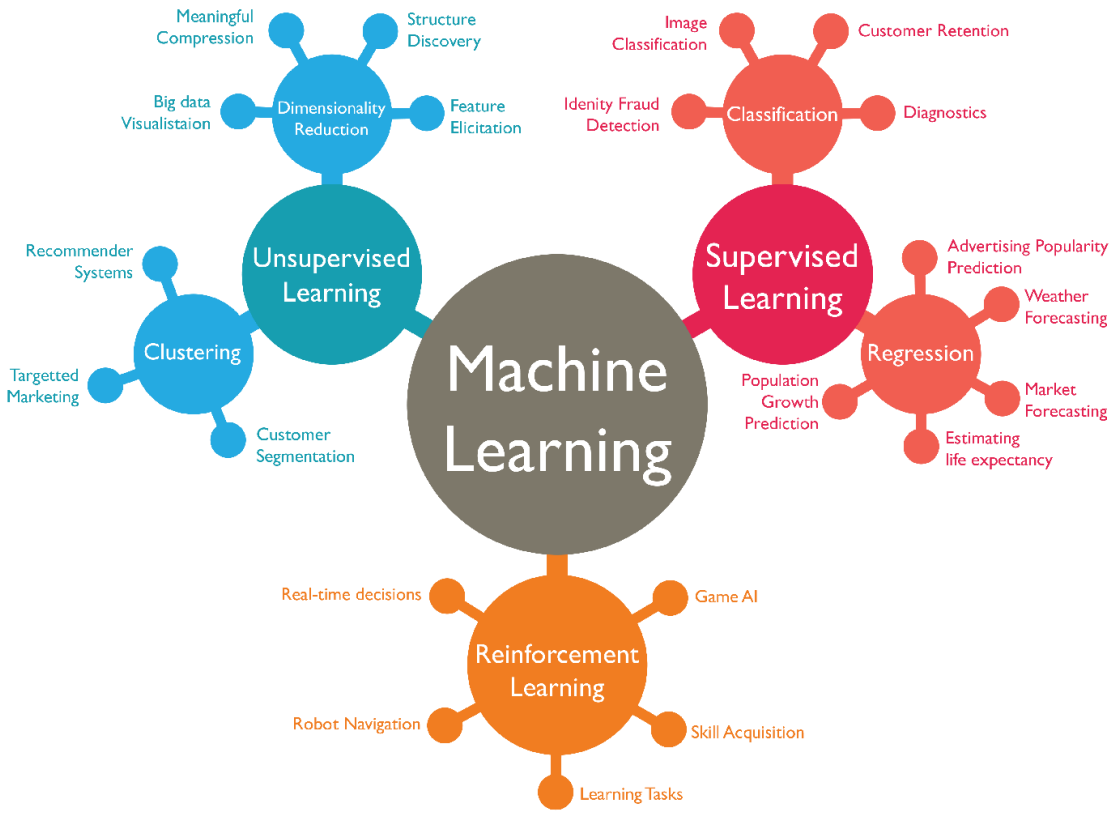

Figura 3 Enfoques de Machine Learning *[80]*

En la **Figura 3** se observa los 3 enfoques de machine learning, aprendizaje supervisado el cual utiliza algoritmos que requieren la división de los datos en conjuntos de entrenamiento y prueba [11]. El no supervisado se enfoca en identificar características ocultas en los datos de entrada [13]. Finalmente, el aprendizaje por refuerzo el cual se basa en la toma de decisiones y mejora continua de las acciones a través de prueba y error [14].

4.2.1.2 Transfer Learning

Esta técnica tiene su origen en la investigación cognitiva, la cual sostiene que el conocimiento puede transferirse entre tareas relacionadas con el fin de mejorar el desempeño de una nueva tarea [15]. Dentro de las ciencias de la computación el Transfer Learning o "Aprendizaje por transferencia" se ha convertido en una de las técnicas más utilizadas por los expertos, puesto que permite trasladar el conocimiento adquirido en un dominio fuente a un dominio objetivo, asumiendo que los dominios de los datos de entrenamiento y prueba pueden no ser los mismos [16]. Esta técnica se compone de un conjunto de métodos, los cuales se basan en utilizar pesos de modelos previamente entrenados para realizar una tarea especifica, los cuales sirven como base para futuras adaptaciones y desarrollo de modelos que realicen tareas nuevas [17].

4.2.1.3 Deep Learning

El aprendizaje profundo utiliza múltiples capas de neuronas que constan de estructuras complejas con el fin de encontrar características de alto nivel, de manera que el modelo pueda transformarlas en la salida esperada [13]. Este término fue introducido por primera vez en 1986 en el aprendizaje automático por Dechter y en el 2000 por Aizenberg en las redes neuronales artificiales [18]. Este tipo de aprendizaje se basa en la extracción manual de características lo que facilita las tareas de clasificación lineal, mostrando un rendimiento eficiente en el reconocimiento de imágenes, voz y comprensión del lenguaje natural [12].

Para la clasificación de imágenes, este tipo de aprendizaje procesa la imagen de entrada para inicialmente obtener características de bajo nivel, sin embargo, en niveles superiores se obtienen características más complejas con el fin de generar la salida del modelo. Dentro de un modelo enfocado en deep learning, las etapas que se realizan son extracción y transformación de características [19].

4.2.1.4 Visión por computador

Denominada también "visión artificial", implica una serie de fases para proporcionar al ordenador la capacidad de percibir y comprender una imagen, imitando el proceso que los seres humanos realizan [20]. Dentro de sus principales funciones se encuentran los algoritmos de análisis, procesamiento, además de comprensión de imágenes y videos [21]. Aunque la arquitectura y etapas de estos sistemas son dependientes de su aplicación, existen algunas etapas generales que se pueden percibir en la mayoría de aplicaciones, estas son [22]:

- Adquisición de imagen
- Preprocesamiento
- Segmentación

- Representación y descripción
- Reconocimiento e interpretación

En la **Figura 4** se muestra las áreas relacionadas con la visión por computador. Actualmente estos son aplicados en diversos campos, tales como medicina, robótica e industria, esto a través de medición, corrección, reconocimiento o detección de fallas [20].

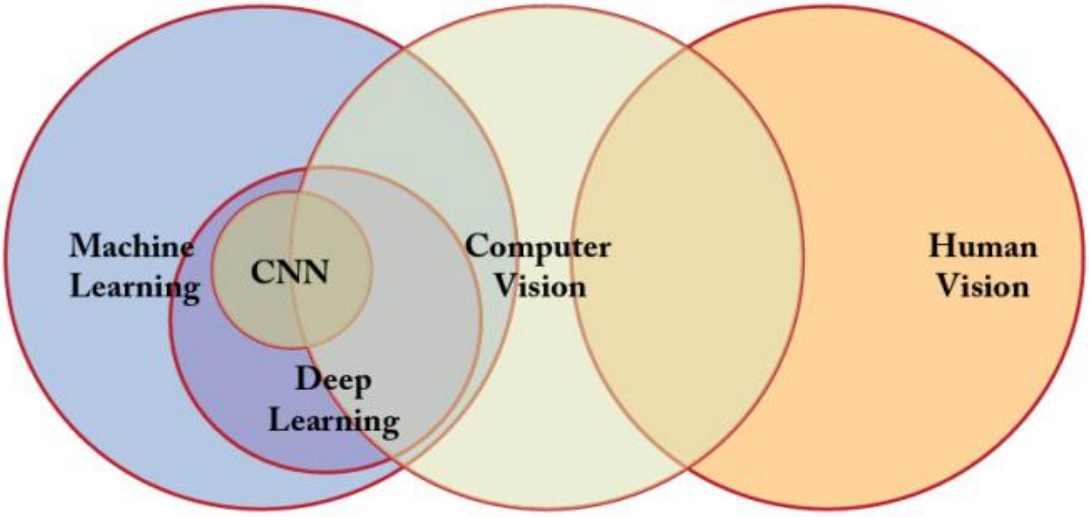

Figura 4 Áreas relacionadas con visión por computador *[82]*

4.2.1.5 Redes neuronales convolucionales

Los investigadores Hubel y Wiesel en la década de 1950 modelaron la corteza visual animal, dando inicio al desarrollo de este modelo [23]. Las redes neuronales convolucionales son consideradas una herramienta óptima para el procesamiento de imágenes, vídeo, voz y audio [2].

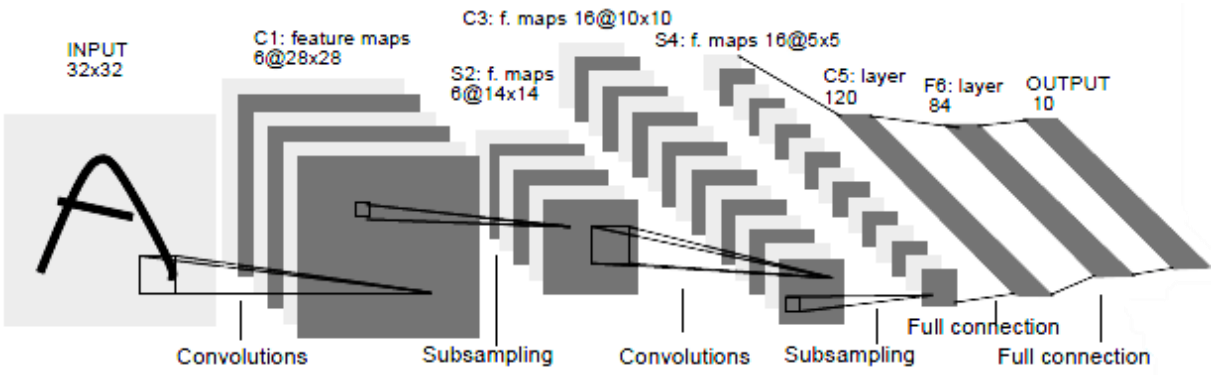

Figura 5 Arquitectura de una red neuronal convolucional *[25]*

En la **Figura 5** se presenta la arquitectura de una red neuronal convolucional con su respectiva jerarquía entre capas, las cuales permiten identificar características en las entradas otorgadas, las primeras capas detectan líneas y curvas mientras que las capas más profundas reconocen formas más complejas, estas pueden ser rostros, siluetas de animales u objetos [23].

Este tipo de red toma como entrada los píxeles de una imagen, si esta se encuentra en escala de grises es considerada de un canal, mientras que una imagen a color necesita de 3 canales [24]. Es importante tomar en cuenta el ancho y alto de los píxeles obtenidos como entrada para el cálculo de las neuronas [25].

- **Modelo VGG16**

Se caracteriza por ser una red de alto rendimiento conocida por su simplicidad y complejidad computacional menor en comparación con arquitecturas de aprendizaje profundo como ResNet e Inception [26]. Esta consta de 16 capas divididas en 5 bloques y una sección de capas completamente conectadas, contiene 2 capas convolucionales en el bloque 1 y 2, 3 capas convolucionales en el bloque 3, 4 y 5, el tamaño para el núcleo de convolución es de 3*3 y ReLu es la función de activación que utiliza después de cada convolución [3]. Al final del bloque cada uno tiene una agrupación máxima, cuyo tamaño del núcleo es de 2*2, donde el resultado final se obtiene utilizando la función softmax [22].

Fue desarrollada por el Grupo de Visualización y Geometría de la Universidad de Oxford [26], se utilizó un conjunto de datos de ImageNet, el cual cuenta con 1,4 millones de imágenes pertenecientes a 1000 clases, debido a la gran cantidad de datos el entrenamiento fue durante semanas y se utilizó GPUs NVIDIA Titan Black [27].

En la **Figura 6** se presenta los componentes que forman la arquitectura del modelo VGG16.

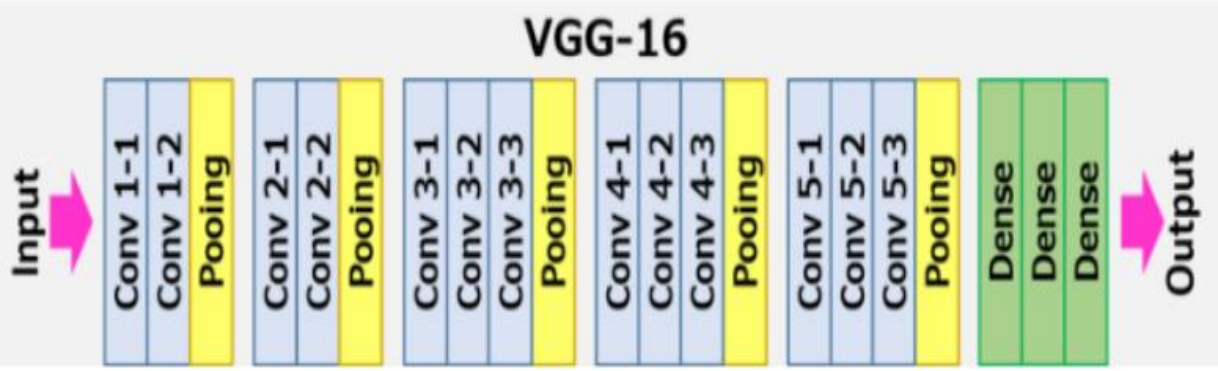

Figura 6 Arquitectura VGG16 *[24]*

VGG16 se especializa en la detectar patrones y características locales en imágenes utilizando capas convolucionales profundas, estas están diseñadas para aprender y extraer características que van desde niveles básicos hasta avanzados [26]. Las capas iniciales se encargan de identificar elementos simples como bordes y texturas utilizando filtros de convolución de 3*3, mientras que las capas convolucionales posteriores combinan las

características simples para reconocer formas y patrones más complejos aprovechando la profundidad y estructura jerárquica del modelo [28].

4.2.1.6 Métricas

Matriz de confusión

Es una métrica sencilla de emplear que permite resumir el rendimiento del algoritmo de clasificación, indica en las columnas el número de predicciones realizadas para cada clase, mientras que en sus filas presenta el número de instancias reales de cada una [29]. Esta requiere interpretación manual para conocer el rendimiento de los algoritmos o modelos, visualizando los aciertos y errores que el modelo está teniendo a la hora de realizar el proceso de aprendizaje [5].

En la **Figura 7** se presenta los componentes de la matriz de confusión, mismos que se describen a continuación [29]:

- ***Verdaderos positivos (True positives VP):*** Datos positivos correctamente reconocidos por el sistema.
- ***Verdaderos negativos (True negatives VN):*** Datos negativos que fueron correctamente reconocidos.
- ***Falsos positivos (False positives FP):*** Datos negativos que son reconocidos por el sistema como positivos.
- ***Falsos negativos (False negatives FN):*** Datos positivos reconocidos por el sistema como negativos.

		Predicción	
		Positivos	**Negativos**
Observación	**Positivos**	Verdaderos Positivos (VP)	Falsos Negativos (FN)
	Negativos	Falsos Positivos (FP)	Verdaderos Negativos (VN)

Figura 7 Matriz de confusión *[31]*

Exactitud

Es una de las métricas más utilizadas por investigadores debido a su facilidad de cálculo y comprensión para evaluar el desempeño del modelo con respecto a una clase específica [30], matemáticamente es la proporción entre el número total de instancias y las instancias

clasificadas correctamente [24]. En la **Ecuación 1** se presenta la fórmula para calcular la exactitud:

Ecuación 1 Fórmula para el cálculo de la exactitud

$$Accuracy = \frac{VP + VN}{VP + FP + FN + VN}$$

Precisión

Mide el número de datos que fueron reconocidos correctamente sobre el total de predicciones (verdaderos o falsos) [29], proporcionado el porcentaje de casos positivos que fueron correctamente identificados [30]. En la **Ecuación 2** se presenta la fórmula para calcular la precisión:

Ecuación 2 Fórmula para el cálculo de la precisión

$$Precisión = \frac{VP}{VP + FP}$$

Sensibilidad

Medida que permite conocer el desempeño de un clasificador con respecto a falsos negativos [31], es decir la proporción de casos positivos que fueron clasificados correctamente [30]. Para obtener matemáticamente dicho valor se toma los verdaderos positivos respecto a la suma entre dicho valor con los falsos negativos [29]. En la **Ecuación 3** se presenta la fórmula para calcular la sensibilidad:

Ecuación 3 Fórmula para el cálculo de la sensibilidad

$$recall = \frac{VP}{VP + FN}$$

4.2.1.7 Conjunto de datos PlantVillage

Para la presente investigación se utilizó el repositorio Kaggle, el cual se enfoca en deep learning y cuenta con una variedad de conjuntos de datos [32], tal como "PlanVillage" mismo que puede ser adquirido a través del siguiente enlace https://www.kaggle.com/datasets/mohitsingh1804/plantvillage.

Este dataset es un conjunto de datos público de 54.305 imágenes de hojas de plantas enfermas y sanas recogidas en condiciones controladas, las imágenes abarcan 14 especies de cultivos, entre ellas: manzana, arándano, cereza, uva, naranja, melocotón, pimiento, patata, frambuesa, soja, calabaza, fresa y tomate [32]. Este dataset contiene imágenes de 17 enfermedades básicas, 4 enfermedades bacterianas, 2 enfermedades causadas por mohos, 2 enfermedades víricas y 1 enfermedad causada por un ácaro, de las cuales 12 especies de cultivos tienen imágenes de hojas sanas que no están visiblemente afectadas por enfermedades [33].

En la **Tabla 1** se presenta la cantidad de imágenes adquiridas para esta investigación, destacando las afectadas por la mancha bacteriana, mismas que son determinantes para el entrenamiento del modelo.

Tabla 1 Cantidad imágenes obtenidas del dataset [33]

	Healthy	**Bacterial Spot**
Imágenes hojas de tomate	1273	1702

4.2.1.8 Balanceo de datos

Submuestreo (Undersampling)

En la **Figura 8** se muestra gráficamente la técnica de submuestreo la cual consiste en equilibrar los datos a través de la eliminación de instancias de la clase mayoritaria dicho proceso se puede realizar de forma aleatoria dejando a cada clase con una cantidad de datos similar [34].

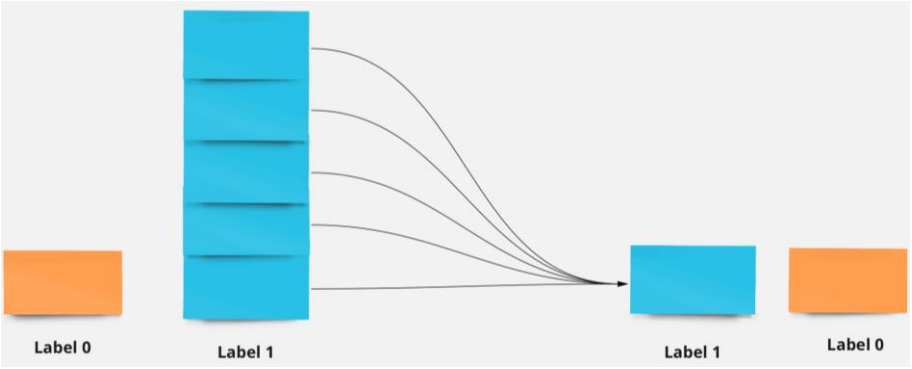

Figura 8 Submuestreo para el balanceo de datos *[79]*

Sin embargo, por la pérdida de datos que se genera esta adquiere un bajo rendimiento dando origen al submuestreo informativo el cual permite eliminar únicamente los patrones insignificantes, manteniendo así los datos con mayor importancia [2].

4.2.1.9 Técnicas de aumento de datos

Estas consisten en aplicar transformaciones a las imágenes, manipulando la posición de las mismas o los valores de intensidad [35], por lo que a continuación se describe las técnicas de aumento de datos utilizadas en la presente investigación.

Transformaciones fotométricas

- **Ajuste de brillo, contraste y color**

Este tipo de técnicas consiste en modificar el contenido de píxeles de las imágenes conservando su estructura espacial de acuerdo a funciones específicas se cambia la composición de los canales RGB [36], esto se realiza ajustando el porcentaje de brillo, contraste o color de la imagen dando origen a datos con características nuevas. Sin embargo, en casos cuya

característica principal es una colorimétrica, emplear esta técnica podría no resultar óptima puesto que se perdería información de la imagen [37].

En la **Figura 9** se muestra un ejemplo de la aplicación de esta técnica.

Figura 9 Ajuste de brillo, contraste y color [39]

Transformaciones geométricas

Las técnicas más utilizadas para aumento de datos, son aquellas basadas en operaciones geométricas básicas, ya que permiten incrementar el tamaño del conjunto de datos mejorando su diversidad [37].

- **Voltear (Flipping)**

Consiste en invertir la imagen a lo largo de uno de sus ejes sea vertical u horizontal, siendo este último una de las variaciones más comunes [35]. En la **Figura 10** se presenta la aplicación de volteo horizontal, este resulta sencillo de aplicar y ha demostrado ser útil en datasets como CIFAR-10 e ImageNet [38], permitiendo mejorar la capacidad de los modelos para reconocer diferentes patrones en imágenes.

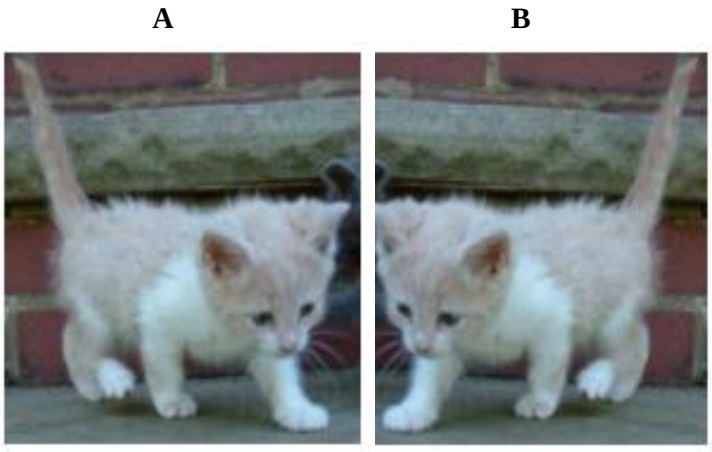

Figura 10 Aplicación de la técnica de volteo. La imagen (A) presenta la imagen original y la imagen (B) muestra la imagen una vez aplicada la técnica *[84]*

- **Recorte (Cropping)**

Esta técnica selecciona al azar una parte aleatoria o específica de una imagen eliminando aquellas partes innecesarias e incrementando el conjunto de datos con imágenes nuevas [38]. Además de equilibrar el conjunto de datos esta técnica permite mejorar la generalización del modelo y reducir el sobreajuste del mismo, obteniendo resultados óptimos durante el entrenamiento de los modelos [36]. En la **Figura 11** se presenta un ejemplo de la aplicación de esta técnica indicando la imagen original, así como el resultado.

Figura 11 Aplicación de la técnica de recorte. La imagen (A) presenta la imagen original y la imagen (B) muestra la imagen una vez aplicada la técnica *[37]*

- **Rotación (Rotation)**

Esta consiste en girar las imágenes sin afectar su contenido, para ello se establece un ángulo específico lo que genera nuevas imágenes [36]. Es importante tener en cuenta que la seguridad de esta técnica se determina por el parámetro del grado de rotación, puesto que a medida que aumenta la rotación las etiquetas de los datos cambia [39]. En la **Figura 12** se presenta ejemplos de imágenes a las cuales se les aplico la técnica de rotación.

Figura 12 Aplicación técnica de rotación *[37]*

4.2.1.10 Hiperparámetros

Optimizador Adam

Este optimizador es uno de los más utilizados dentro del aprendizaje automático puesto que muestra mayor robustez y una estabilidad óptima durante el proceso de entrenamiento [40]. Utiliza dos momentos principales para ajustar los parámetros del modelo, el primer momento se encarga de la velocidad de actualización de los parámetros, mientras que el segundo contiene información acerca de la varianza de las gradientes [41].

Tasa de aprendizaje (Learning rate)

Valor que varía entre 0,1 y 0,0001, el cual para la actualización de los parámetros se multiplica por el valor de la gradiente [42]. Este controla el tamaño de los pasos del modelo en cada iteración, es decir define la velocidad con la que aprende durante el entrenamiento [43].

Decaimiento del peso (Weight decay)

Técnica de regularización la cual permite reducir el sobreajuste de los modelos [44], se obtiene multiplicando cada peso por un factor λ ($0 < \lambda < 1$), esto durante el descenso de gradiente en cada época [45].

Programador de tasa de aprendizaje (Scheduler Learning rate)

Técnica que permite ajustar la tasa de aprendizaje durante el proceso de entrenamiento del modelo [45], es decir controla cuánto se ajustan los pesos del modelo [42].

Épocas (Epoch)

Consiste en el número de iteraciones que el modelo realiza sobre los datos [46], teniendo en cuenta que entre más veces se entrene el modelo este tiene mayor aprendizaje. Sin embargo, si se define un valor elevado de épocas el modelo puede presentar un sobreajuste [42].

4.2.2 Metodología CRISP-ML(Q)

CRISP-ML(Q) es un marco de trabajo para el desarrollo de modelos de machine learning [47], propuesto para ayudar a abordar proyectos de manera eficaz. En la **Figura 13** se presentan las fases con las que esta metodología cuenta, las cuales van desde la comprensión de los datos hasta el monitoreo y mantenimiento [48].

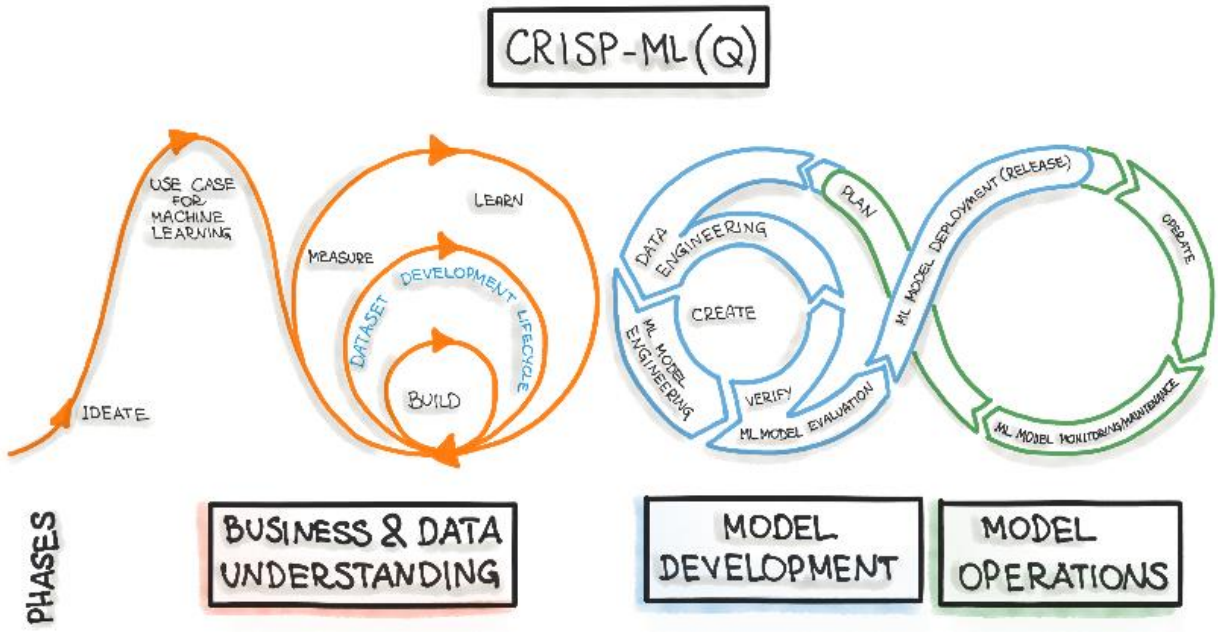

Figura 13 Proceso del ciclo de vida machine learning *[47]*

4.2.2.1 Fases

En la **Tabla 2** se describe cada una de las fases involucradas en el desarrollo e implementación de modelos machine learning, además de los aspectos clave que garantizan la calidad y rendimiento del modelo durante su ciclo de vida.

Tabla 2 Fases de la metodología CRISP-ML(Q) [47]

FASE	*DESCRIPCIÓN*
Comprensión de datos y negocio	Identifica el alcance y los criterios de éxito que tendrá la aplicación, además de verificar la calidad de los datos, asegurando así la viabilidad del proyecto [49]. Es importante que la documentación obtenida sea revisada con el fin de clasificar y definir las tareas de la siguiente etapa [50].
Ingeniería de los datos	El objetivo principal es preparar los datos para la siguiente fase, realizando tareas de selección, estandarización y limpieza de datos [47]. La identificación de características se realiza mediante el uso de filtros sean envolventes o integrados, descartando aquellas muestras que no satisfacen los requisitos de calidad de los datos [50]. Durante la limpieza de datos se detecta y corrige los errores para la fase de estandarización, la cual unifica los datos de entrada para evitar riesgos de datos erróneos, creando canales de transformación para el preprocesamiento de datos [51].
Ingeniería del modelo	Esta fase define los modelos de aprendizaje automático que se implementarán en producción, incluye tareas como la selección, especialización y entrenamiento del modelo [50]. Dependiendo de la tarea a realizar se pueden utilizar modelos previamente entrenados, aplicando métodos de aprendizaje para obtener un modelo final con la capacidad de realizar una tarea específica [49].

FASE	*DESCRIPCIÓN*
Evaluación de modelos de aprendizaje automático	Se evalúa el rendimiento del modelo entrenado haciendo uso de un dataset de prueba [69]. Además, permite asegurar la robustez y reproducibilidad del modelo, teniendo en cuenta los criterios de éxito obtenidos [51].
Despliegue	Una vez alcanzados los criterios de éxito, el modelo puede implementarse en un entorno de preproducción [8]. Para ello se define el hardware de inferencia, se evalúa el modelo en un entorno de pruebas en línea como las A/B y se desarrolla pruebas de aceptación y usabilidad del usuario [50]. Además, se puede generar un plan alternativo para interrupciones del modelo y la implementación de estrategias para la ejecución gradual de nuevas versiones [51].
Monitoreo y mantenimiento	Esta fase asegura la efectividad del modelo una vez que este ha sido puesto en producción, aplicando cambios necesarios o procesos de actualización [49]. Además, evalúa el rendimiento que este tiene frente a los cambios realizados y el entorno en el que se encuentra [51].

4.2.3 Tecnologías y herramientas

Para el desarrollo de la presente investigación el uso de herramientas y tecnologías es esencial para alcanzar los objetivos planteados, ya que optimizan los procesos y mejoran la precisión de resultados.

Google Colaboratory

También conocido como "Colab" es un proyecto cuyo objetivo se centra en la investigación del aprendizaje automático [52]. Este es un servicio alojado en la plataforma Google Cloud el cual se basa en Jupyter Notebooks y proporciona un tiempo de ejecución configurado para deep learning e inteligencia artificial como TensorFlow, Matplotlib y Keras además de un tiempo de ejecución acelerado por GPU [53].

Opencv

Open Source Computer Vision es una biblioteca de código abierto introducida por Intel, diseñada para resolver problemas de visión por computadora [54]. Se caracteriza por sus funciones escritas en lenguaje C, lo que permite que sea utilizada de manera libre tanto para aplicaciones comerciales como privadas [55]. Esta hace uso de la librería "Numpy", la cual posee una sintaxis similar a MATLAB convirtiéndola en una herramienta eficaz para realizar operaciones numéricas, al transformar en matrices de Numpy todas las estructuras matriciales de OpenCV permite una integración óptima con bibliotecas como SciPy y Matplotlib [56]. OpenCV se centra en aplicaciones de visión por computadora a través de soluciones innovadoras, aplicada en campos como la interacción hombre-computador, reconocimiento de objetos y gestos, seguimiento de movimiento y la robótica móvil [57].

Pytorch

Es una biblioteca de código abierto desarrollada por el laboratorio AI Research de Facebook la cual se basa en "Torch" [58]. Es una herramienta óptima para los desarrolladores e investigadores de machine learning, ya que proporciona una plataforma flexible, eficaz y fácil de usar para la creación y entrenamiento de modelos de inteligencia artificial, utilizando gráficos de cálculo dinámico y facilitando cálculos tensoriales eficientes en GPU [59]. Esta biblioteca es una de las más utilizadas dentro del ámbito académico e industrial debido a la sintaxis intuitiva que maneja, así como la capacidad de combinar arquitecturas complejas (DNN, CNN y RNN) con aplicaciones avanzadas como el reconocimiento de voz [60].

Lenguaje de programación Python

Python fue diseñado en 1991 por Guido Van Rossum a partir de su experiencia con otros lenguajes experimentales [24], este es un lenguaje de programación de alto nivel, cuya jerarquía permite la escritura de programas donde la lectura es más fácil respecto a otros lenguajes [61]. A través de su entorno interactivo la realización de pruebas es eficaz, puesto que permite detectar errores de programación que no son percibidos a primera vista, proporcionando la respectiva información para corregirlos [31]

4.3 Trabajos Relacionados

La detección temprana de enfermedades en cultivos resulta esencial para el tratamiento y producción efectiva de los mismos. Dentro de este contexto la implementación de sistemas de visión por computadora y métodos de aprendizaje profundo para la detección de enfermedades sigue en aumento, siendo las redes neuronales convolucionales (CNN) las más utilizadas.

En la **Tabla 3** se presentan propuestas indagadas las cuales abordan aspectos como la selección de la arquitectura, técnicas transfer learning, segmentación de imágenes y evaluación de modelos preentrenados como parte de desarrollo de estos sistemas. Destacando aquel proceso que presenta el valor de precisión más alto durante la predicción realizada y referenciando cual puede ser el mejor modelo para realizar este tipo de tarea

Tabla 3 Trabajos relacionados. TR: Código distintivo

Código	*Título*	*Resumen*
TR01	Sistema de visión artificial para la detección de plantas enfermas mediante aprendizaje profundo	Este trabajo consiste en la implementación de un modelo predictivo para el cual se emplea el enfoque de aprendizaje profundo, este modelo es capaz de clasificar anomalías causadas por los agentes patógenos a partir de fotografías de hojas de las plantas de maíz y jitomate. Además, analiza diferentes redes neuronales convolucionales, sin embargo, selecciona e implementa la red MobilNetV2, la cual genera un modelo ligero y demuestra tener un gran desempeño al realizar la clasificación de

Código	*Título*	*Resumen*
		imágenes. Para el respectivo entrenamiento del modelo se utilizó un conjunto de datos que cuenta con un total de 15.711 imágenes y logra obtener una precisión del 99.65% [62].
TR02	A VGG-19 Model with Transfer Learning and Image Segmentation for Classification of Tomato Leaf Disease	El presente artículo propone un modelo de clasificación de imágenes segmentadas, el cual utiliza un conjunto de datos obtenido de la base de datos PlantVillage, el mismo que cuenta con un total de 16.010 imágenes de hojas de tomate, estas se segmentan mediante el espacio de color HSV para extraer regiones de hojas y fondos negros, dicha segmentación sirve para sincronizar el fondo de las imágenes obtenidas. Para el reentrenamiento en capas posteriores del modelo se aplica transferencia de aprendizaje con el modelo VGG19 lo que permite reducir el tiempo de entrenamiento, el mismo que logra alcanzar el 99.72% de precisión demostrando la eficacia que llega a tener el modelo propuesto. Este enfoque puede ser aplicado a la clasificación de otros conjuntos de datos, destacando su potencial para conjuntos más complejos [63].
TR03	Detection of disease in tomato plant using Deep Learning Techniques	Este trabajo aborda la revisión de la eficiencia de las arquitecturas de aprendizaje profundo para la detección de plagas y enfermedades en tierras de cultivo, se utiliza un conjunto de datos denominado "PlantVillage" de 4506 imágenes las cuales son obtenidas de distintos conjuntos de datos. Además, 1090 imágenes de hojas de tomates infectadas por tizón temprano, enrollamiento de la hoja, septoriosis de la mancha foliar y mancha bacteriana son capturadas en tiempo real. Para identificar las plagas y enfermedades mediante detectores profundos se emplean diferentes arquitecturas de aprendizaje profundo, según el estudio la arquitectura de aprendizaje profundo Faster R-CNN combinada con ResNet proporcionó un mejor rendimiento en comparación con R-FCN y SSD. Por lo que propone una detección automatizada de enfermedades: Faster R-CNN con ResNet como una solución viable para que los productores detecten en etapa temprana cada una de las enfermedades antes mencionadas, obteniendo valores de precisión del 92.25%, 88.57%, 81.41%, 78.44% y 64.09% respectivamente para cada clase predicha [64].
TR04	Redes neuronales convolucionales para la detección y clasificación de enfermedades de plantas basadas en imágenes digítales	El presente estudio emplea la arquitectura AlexNet para clasificar enfermedades en hojas, utiliza el dataset PlantVillage compuesto por 40000 imágenes de hojas sanas y enfermas, las cuales son utilizadas para el respectivo entrenamiento del modelo. Se logra identificar 9 especies de plantas y 24 enfermedades alcanzando una exactitud del 98.90%, siendo este el valor más significativo respecto a las demás arquitecturas comparadas. Además, demuestra que al utilizar las capas de activación de la arquitectura AlexNet se puede realizar una detección y segmentación de lesiones presentes en la hoja de la planta de manera eficaz y rápida, sin tener la necesidad de recurrir a procesos monótonos de entrenamiento [65].
TR05	Comparación de arquitecturas de redes neuronales	El presente trabajo se centra en el ajuste de modelos comparando distintas arquitecturas CNN, tales como AlexNet, GoogleNet, InceptionV3, ResNet18 y ResNet50. El conjunto de datos que se utiliza

Código	*Título*	*Resumen*
	convolucionales para la clasificación de enfermedades en el tomate	se obtiene del dataset PlanVillage, este cuenta con un total de 18,160 imágenes, compuesto por 9 clases con enfermedades y una clase sana. Durante el proceso de entrenamiento y evaluación de cada modelo, se realiza un análisis estadístico multiclase, este se basa en la exactitud, precisión, sensibilidad, especificidad, F-score, área bajo la curva y la curva de características operativas del receptor, obteniendo valores de precisión significativos. Para AlexNet se logra el 98.93%, el 98.65% para Inception V3, 99.06% para ResNet18, 99.2% para ResNet50 y para el modelo de GoogleNet se alcanzó el 99.72% de AUC y 99% de sensibilidad, demostrando que este modelo es el más eficaz para la clasificación de enfermedades en el tomate [66].
TR06	Detección de enfermedades foliares con arquitecturas de redes neuronales convolucionales	El presente estudio trata sobre la evaluación de modelos preentrenados para predecir enfermedades en plantas de diferentes clases del conjunto de datos PlantVillage, de los cuales dos modelos mostraron resultados similares en las métricas de precisión: Xception lideró con un 92%, seguido por ***VGG16*** con un ***91%***. Por otro lado, ResNet50V2 obtuvo un 85%, MobileNetV2 alcanzó un 72%, y el rendimiento más bajo fue para InceptionV3 con un 71%. Sin embargo, VGG16 superó a Xception en términos de eficiencia de procesamiento, ya que demostró el mejor desempeño en términos de tiempo, lo que sugiere que, a pesar de tener menos profundidad, es más eficiente para esta tarea específica [67].
TR07	Plant Disease Detection using AI based VGG-16 Model	Este estudio presenta el modelo de red neuronal convolucional VGG-16 para detectar enfermedades en las plantas, con el fin de permitir a los agricultores tomar medidas oportunas con respecto al tratamiento que esta requieren. Para llevar a cabo este propósito, se seleccionaron 19 clases distintas de enfermedades de plantas, recopilando 15.915 imágenes del conjunto de datos Plant Village para el respectivo entrenamiento y desarrollo de pruebas. Según los resultados experimentales, el modelo propuesto es capaz de alcanzar una precisión del 95,2%, con una pérdida en las pruebas de sólo 0,4418, lo que indica que este modelo establece una dirección clara hacia la detección de enfermedades en plantas basada en aprendizaje profundo [68].
TR08	Development of efficient CNN model for tomato crop disease identification	Este artículo presenta un modelo simplificado de red neuronal convolucional que consta de 8 capas ocultas y utiliza el conjunto de datos PlantVillage, el cual es públicamente accesible. El modelo propuesto supera a los enfoques convencionales de aprendizaje automático y a los modelos preentrenados, logrando una precisión del 98,4%. El conjunto de datos utilizados contiene 39 clases de diversos cultivos, tales como manzanas, patatas, maíz, uvas, entre otros. Además, incluye 10 clases que representan enfermedades presentes en el cultivo del tomate de riñón. Aunque los métodos tradicionales de aprendizaje automático alcanzan una precisión superior del 94,9% con k-NN, la máxima precisión del ***93,5%*** se logra con ***VGG16*** siendo este un modelo preentrenado. Para mejorar el rendimiento de la CNN propuesta, se implementó un preprocesamiento de imágenes el cual permite ajustar el

Código	*Título*	*Resumen*
		brillo mediante un valor aleatorio dentro del ancho de la imagen después de realizar aumentos a la misma, donde el modelo propuesto muestra un rendimiento destacado en conjuntos de datos distintos a PlantVillage, obteniendo una precisión del 98,7% [69].
TR09	Plant leaf damage detection using convolution neural network models	La presente investigación se centra en descubrir enfermedades de las plantas de tomate y papa en una etapa temprana, esto a través de la implementación de redes neuronales convolucionales como AlexNet y VGG16. Para el respectivo entrenamiento utiliza el conjunto de datos PlantVillage, compuesto por 3580 imágenes de hojas de plantas de tomate y 2150 imágenes de hojas de plantas de papa, durante el proceso de entrenamiento y validación se logra obtener una precisión del 84.7% para el modelo AlexNet, mientras que para el modelo ***VGG16*** se logra alcanzar ***96.5%*** de precisión, demostrando que este presenta mayor eficacia al momento de realizar la predicción [70].
TR10	Transfer Learning VGG16 Model for Classification of Tomato Plant Leaf Diseases: A Novel Approach for Multi-Level Dimensional Reduction	El presente artículo propone un modelo que predice enfermedades en hojas de tomate, 9 clases enfermas y 1 clase sana. Este utiliza imágenes del conjunto de datos PlanVillage, a su vez emplea aprendizaje por transferencias con VGG16 con el fin de extraer características por cada imagen. Para evitar el sobreajuste debido a la dimensionalidad, se aplican técnicas de reducción como el Análisis de componentes principales (APC) y la selección de características Boruta, esto permite alcanzar una precisión del 95.68% con MLP y ***95.79%*** con ***VGG16*** [71].
TR11	A Deep Learning-Based Approach in Classification and Validation of Tomato Leaf Disease	Los modelos de aprendizaje profundo han tenido un gran aporte dentro de la clasificación de enfermedades en plantas, por lo que en este estudio se aplican modelos preentrenados como AlexNet, VGG16, GoogleNet, MobileNetv2 y SqyeezeNet. Para el entrenamiento y obtención de las predicciones requeridas, se utilizó el conjunto de datos PlanVillage, alcanzado una precisión del **99.17%** con **VGG16**, mismo que se validó con imágenes capturadas en KVKN, India [72].
TR12	Detection and Categorization of Tomato Leaf Diseases Using Deep Learning	Los métodos manuales para detección de enfermedades en su gran mayoría son inconsistentes por lo que el rendimiento del cultivo de tomate se ve afectado, el uso de redes neuronales convolucionales para este tipo de actividades resulta eficaz. El presente estudio implementa VGG16, ResNet50 y MobileNetV2 como modelos para clasificar 10 categorías de enfermedades en las hojas de tomate, estos al ser evaluados lograron alcanzar precisiones del 68.98% para ResNet50, 90.84% para MobileNetV2 y ***90.19%*** para ***VGG16***. Sin embargo, este último modelo presentó el mejor rendimiento en una proporción de 80:20, demostrando la viabilidad de utilizar modelos superficiales para clasificar enfermedades en hojas de tomate [7].

4.3.1 Conclusión sobre los trabajos relacionados

La revisión bibliográfica presentada en la **Tabla 3** aborda enfoques que han contribuido al avance en la clasificación de enfermedades en plantas, específicamente en cultivos de tomate.

En lo estudios TR04, TR06 y TR07 se resalta el uso de aprendizaje por transferencia, esta técnica aprovecha las características preentrenadas de modelos complejos reduciendo los tiempos de entrenamiento y mejorando la precisión de los modelos. Esta técnica ha sido fundamental para adaptar modelos como VGG16 a tareas específicas, demostrando su eficacia y versatilidad en varios escenarios.

Por otro lado, los trabajos TR01, TR08, TR11 y TR12 resaltan la importancia de las técnicas de aumento de datos y ajuste de hiperparámetros de modelos ya entrenados, ya que estos permiten ampliar y diversificar los datasets originales, mejorando el rendimiento y capacidad de generalizar de cada modelo con el fin de obtener óptimos resultados. Para definir el conjunto de datos adecuado para el proceso de entrenamiento, los estudios TR04, TR06 y TR07 resaltan el uso del dataset PlantVillage, el cual ha sido uno de los más utilizados para entrenar y evaluar modelos de aprendizaje profundo, proporcionando una base sólida y aceptada para la clasificación de imágenes de hojas enfermas.

Finalmente, los estudios TR05, TR09 y TR10 utilizan la matriz de confusión para la obtención de la precisión del modelo a través de una evaluación detallada del rendimiento, lo que permite medir su efectividad en situaciones reales.

5. Metodología

En la presente sección se detallan las tareas y métodos utilizados para el desarrollo del trabajo de integración curricular (TIC). La sección **5.1** aborda el área de estudio empleada para el desarrollo de la investigación, la sección **5.2** detalla el procedimiento requerido para dar alcance a los objetivos planteados y, finalmente en la sección **5.3** se presenta los recursos científicos y técnicos utilizados en el desarrollo del TIC.

5.1. Área de estudio

En la **Figura 14** se presenta el área geográfica donde se desarrolló el trabajo de integración curricular titulado "Creación de un modelo para la clasificación de imágenes de la mancha bacteriana en la hoja de tomate utilizando el modelo VGG16" realizado durante el periodo académico octubre 2024 – febrero 2025. Este se llevó a cabo en la Universidad Nacional de Loja, Facultad de la Energía, las Industrias y los Recursos Naturales no Renovables en la carrera de Ingeniería en Computación, ubicada en la Av. Pío Jaramillo Alvarado y Reinaldo Espinosa Loja, Ecuador con coordenadas **latitud:** -4.030184, y **longitud:** 79.199314.

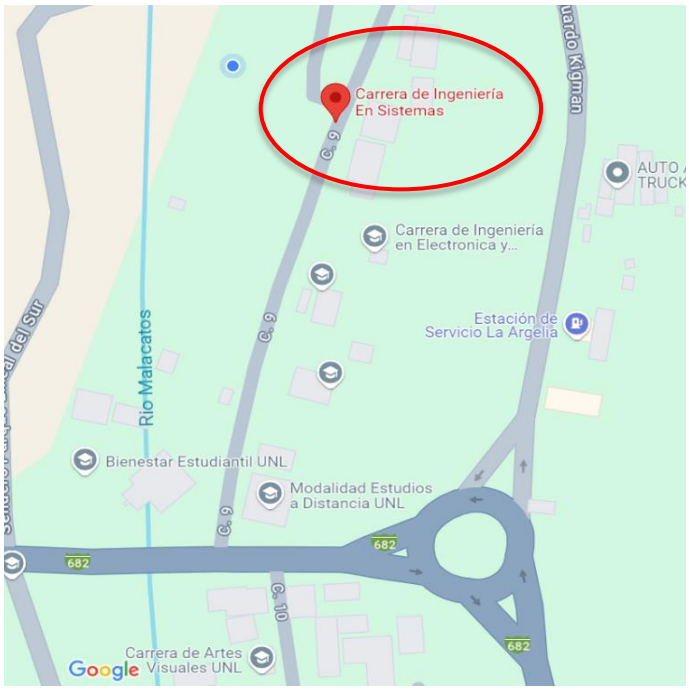

Figura 14 Ubicación de la carrera de Ingeniería en Computación

5.2. Procedimiento

Para el desarrollo de los objetivos planteados en el TIC se utilizó como marco de trabajo CRISP-ML(Q) el cual se estructuró en tres fases, Ingeniería de los datos, Ingeniería de modelos de aprendizaje automático y Evaluación de modelos. En la primera fase se obtuvo el conjunto de datos aplicando técnicas de limpieza, balanceo y aumento. En la fase de ingeniería de

modelos de aprendizaje automático se ajustó el modelo preentrenado VGG16, adaptando hiperparámetros y cambiando el optimizador de aprendizaje. Finalmente, en la evaluación de modelos se obtuvo el porcentaje global de acierto a través de la validación del modelo obtenido dando alcance al objetivo general planteado.

5.2.1 Objetivo 1: Ajustar el modelo VGG16 para la clasificación de imágenes de la mancha bacteriana en la hoja de tomate de riñón.

Para el alcance del presente objetivo se adaptaron las fases de ingeniería de datos e ingeniera de los modelos de aprendizaje automático, las cuales se enfocan en la preparación de datos para posteriormente trabajar con los modelos de aprendizaje automático.

Fase 1: Ingeniería de datos

Tarea 1: Identificar y recolectar imágenes de la hoja de tomate en repositorios

Para la obtención de los conjuntos de entrenamiento y test se realizó una búsqueda en el repositorio Kaggle[1], para la selección del dataset se consideró la revisión bibliográfica mencionada en la sección **4.3** del marco teórico mismo que fue utilizado en estudios relacionados. Una vez seleccionado el dataset de estudio *PlantVillage* se identifican las carpetas que contienen imágenes de hojas sanas como afectadas por la mancha bacteriana, manteniendo un conjunto inicial para entrenamiento con un total de 2.975 imágenes y 743 imágenes para test, como se detalla en la **Tabla 5** del apartado **6.1** de los resultados. La obtención de estos conjuntos de datos permite que el modelo aprenda a identificar patrones distintivos asociados con la mancha bacteriana en la hoja de tomate lo que permite evaluar su desempeño de manera objetiva en un conjunto de validación independiente.

Tarea 2: Procesamiento de los datos

Para optimizar el proceso de entrenamiento del modelo se realizó limpieza, balanceo y aumento de datos, cuyos resultados se presentan en la subsección **6.1.2**. Dentro de la limpieza del conjunto de datos se eliminó aquellas imágenes de baja resolución o pixeladas, lo que redujo el conjunto de datos inicial a un total de 2958 imágenes distribuidas en 1272 para la clase sano y 1686 para hojas infectadas tal como se detalla en la **Tabla 6** de la subsección **6.1.2.1**. A continuación, se aplicó la técnica de *undersampling* para equilibrar las clases al reducir el número de imágenes en la clase mayoritaria generando un nuevo conjunto de datos cuya información se encuentra detallada en la **Tabla 7** de la subsección **6.1.2.2** esto con el fin de evitar sesgo durante el entrenamiento del modelo.

[1] Enlace repositorio Kaggle https://www.kaggle.com/

Finalmente, se utilizaron pautas de los trabajos relacionados [73], [69], [72], [7] para definir las técnicas de aumento adecuadas para incrementar la diversidad del conjunto de datos. Se aplicó dichas técnicas en el 50% de las imágenes balanceadas, estas incluyeron recorte, rotación aleatoria de 15° para generar variabilidad en la orientación, ajuste de brillo y contraste para resaltar detalles específicos y el volteo horizontal para crear imágenes reflejadas, dando como resultado un conjunto de datos con 3816 imágenes conforme se detalla en la **Tabla 8** de la subsección **6.1.2.3**. Una vez finalizado el procesamiento de los datos la calidad y diversidad del conjunto de datos incrementó, lo que permitió que el modelo tenga mayor generalización.

Tarea 3: División del dataset

El conjunto de datos obtenido en la tarea 2 se dividió en conjuntos de entrenamiento y validación en una proporción de 80/20, tal como se muestra en la **Figura 30** del apartado **6.1.3** de la sección de resultados. Para ello se utilizó una selección aleatoria en Pytorch definiendo una semilla con el fin de permitir la reproducibilidad del proceso asegurando que el modelo tuviera la cantidad de datos necesaria para aprender y generalizar.

Fase 2: Ingeniería de modelo

Esta fase se detalla en los apartados **6.1.4** y **6.1.5** de la sección de resultados implementando VGG16 el cual es una arquitectura de red neuronal convolucional profunda preentrenada en ImageNet, cuyo objetivo es reutilizar sus capas para extraer características de las imágenes y ajustar las capas de clasificación para que realicen una nueva tarea, en este caso clasificar las imágenes de hojas de tomate afectadas por la mancha bacteriana.

Tarea 4: Ajustar hiperparámetros de modelo VGG16

El uso de un modelo preentrenado como VGG16 facilitó la extracción de características complejas para la clasificación de imágenes de hojas de tomate sanas como afectadas por la mancha bacteria. Para el proceso de entrenamiento se ajustó ciertos hiperparámetros como la tasa de aprendizaje, el uso de betas para controlar la estabilidad y el valor del decaimiento de los pesos para la regularización del modelo, empleando *CrossEntropyLoss* como función de pérdida y el algoritmo Adam como optimizador. Además, con la finalidad de optimizar recursos el modelo fue diseñado para ejecutarse en GPU o CPU.

En la **Tabla 11** de la subsección **6.1.4** de los resultados se detallan los valores de los hiperparámetros utilizados en los distintos experimentos que se realizó.

Tarea 5: Entrenar el modelo

Para el entrenamiento del modelo se configuraron hiperparámetros los cuales se detallan en la tarea 4 de la sección **6.1.4** de los resultados, esto con el fin de controlar la clasificación de las clases y la actualización de los pesos del modelo definiendo posteriormente el número de

iteraciones para el proceso de aprendizaje. En el apartado **6.1.5** se detallan los resultados obtenidos al entrenar los modelos para ello se utilizó el entorno de Google Colab mismo que proporciona una interfaz óptima y adecuada.

5.2.2 Objetivo 2: Evaluar el modelo entrenado a través de la precisión global de acierto al clasificar imágenes de la mancha bacteriana en hojas de tomate de riñón.

Para el desarrollo del presente objetivo se adaptó la fase de Evaluación de modelos de aprendizaje automático con el fin de obtener el porcentaje de la precisión global del modelo al clasificar imágenes de la mancha bacteriana en la hoja de tomate, los resultados obtenidos se detallan en la sección **6.2**.

Fase 3: Evaluación del modelo

En esta fase se utilizan métricas de rendimiento como la sensibilidad, precisión, F1-Score y exactitud esto con la finalidad de identificar cuál de los modelos resulta el mejor para la tarea de clasificación de imágenes de la mancha bacteriana en hojas de tomate.

Tarea 6: Evaluar el modelo entrenado en el conjunto de entrenamiento

Durante el entrenamiento de los modelos se realizó la evaluación de cada uno utilizando el conjunto de datos generado para la validación, mismo que se detalla en las gráficas de la sección **6.1.5** de los resultados**.** En esta etapa se obtuvo las respectivas curvas de aprendizaje las cuales permitieron visualizar como los modelos aprenden progresivamente a lo largo de cada iteración, donde por cada experimento realizado se obtuvieron valores de pérdida cercanos a 0 y precisiones superiores al 90% (ver sección **6.2.1**).

Tarea 7: Seleccionar del mejor modelo

En base a los resultados que se detallan en la **Tabla 15** del apartado **6.2.1** se seleccionaron los modelos con las mejores métricas obtenidas durante la validación, lo que dio como resultado que los mejores modelos obtenidos por experimento sean elegidos para evaluar su rendimiento en un conjunto de prueba.

Tarea 8: Evaluar el rendimiento del modelo utilizando el conjunto de prueba

Se utilizó el conjunto de prueba adquirido del repositorio digital Kaggle el cual se detalla en el apartado **6.1.1** el cual contiene imágenes no vistas previamente por los modelos. En la subsección **6.2.3** se detalla los resultados obtenidos en esta etapa, en la cual para determinar el modelo más óptimo se utilizó la matriz de confusión para analizar el rendimiento en términos de aciertos y errores.

Tarea 9: Medir la precisión global de acierto durante la clasificación de imágenes de la mancha bacteriana en la hoja de tomate

Calcular métricas como la precisión global de acierto (*exactitud*), sensibilidad (*sensitivity*), puntuación F1 (*F1-Score*) y precisión permitió determinar cuál de los modelos obtenidos es el más eficaz para la clasificación de imágenes de la mancha bacteriana en la hoja de tomate. Para obtener estas métricas se tomó en cuenta los valores de cada matriz de confusión cuyo resultado se detalla en el apartado **6.2.4**, estos valores proporcionan una evaluación integral del rendimiento del modelo.

Tarea 10: Seleccionar modelo final

Basándose en los resultados del apartado **6.2.4** se seleccionó como modelo final aquel con mejor balance entre las métricas de rendimiento, siendo el experimento 2 el que alcanzó un alto desempeño al presentar un equilibrio óptimo entre precisión y sensibilidad.

Tarea 11: Aplicación de la técnica Zero-Shot Learning al modelo definitivo

Para evaluar la capacidad del modelo final de clasificar imágenes obtenidas en un entorno real se implementó la técnica Zero-Shot Learning, este enfoque se utilizó para comprobar la generalización del modelo con imágenes que no formaron parte del conjunto de datos inicial. Como se presenta en el apartado **6.2.6** la recolección de estas imágenes dio como resultado un conjunto de datos con un total de 32 imágenes las cuales fueron utilizadas para la validación del modelo, a su vez se obtuvo el porcentaje de la precisión global de acierto mediante el uso de la matriz de confusión.

Tarea 12: Documentación de resultados

Para realizar la documentación respectiva de los modelos y selección del más óptimo para la clasificación de imágenes de la mancha bacteriana en la hoja de tomate se tomó en cuenta los resultados obtenidos durante el entrenamiento (ver sección **6.1.5**) y evaluación de cada uno utilizando el conjunto de prueba (ver sección **6.2.3**). Además, utilizando la matriz de confusión se analizaron las métricas de rendimiento con el fin de determinar el modelo final.

5.3 Recursos

5.3.1 Científicos

- **Método experimental**

Este método permitió adaptar las fases de la metodología CRISPL-ML(Q) propuesta para el desarrollo de la presente investigación, estas permitieron el tratamiento de los datos para la creación del dataset final, el entrenamiento y evaluación de los modelos obtenidos.

- **Entrevista**

Realizada al Ing. Oscar Cumbicus misma que se encuentra en el **Anexo 1** esta permitió comprender el uso de las redes neuronales convolucionales profundas, así como su importancia y aplicación dentro del campo agrícola justificando el problema técnico.

- **Técnicas de aumento de datos**

La implementación de técnicas de aumento de datos como rotación, recorte y ajustes de brillo y contraste a las imágenes obtenidas como conjunto de datos lo que permitió incrementar la cantidad de datos lo que proporcionó mayor generalización del modelo. Estas técnicas se pueden visualizar en el apartado **6.1.2.3** de la sección de resultados.

- **Ajuste Fino (Fine Tuning)**

Esta técnica permitió el ajuste y adaptación del modelo VGG16 para la clasificación de imágenes de la mancha bacteriana en la hoja de tomate, realizando distintas configuraciones de hiperparámetros con el fin de obtener resultados óptimos del modelo en la clasificación de las clases.

- **Precisión global de acierto**

Esta métrica permitió evaluar el modelo ajustado proporcionando una medida directa de la eficacia con la que modelo está clasificando los datos, es decir que proporción de las predicciones realizadas son correctas.

5.3.2 Hardware

En la **Tabla 4** se detallan las especificaciones del dispositivo electrónico utilizado para el desarrollo del trabajo de integración curricular.

Tabla 4 Especificaciones de dispositivo utilizado para el desarrollo del TIC

Componente	**Detalle**
Marca	Dell
Modelo	Inspiron 14 5000
Procesador	Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz 1.80 GHz
Tarjeta gráfica	NVIDIA GeForce MX150
Memoria	8 GB
Sistema operativo	Windows 10 Home
Disco duro	500 GB

5.3.3 Técnicos

- **Google Colab**

Debido a su capacidad computacional y facilidad de uso fue utilizado como herramienta para el procesamiento de los datos, la definición de la arquitectura, entrenamiento y validación del modelo.

- **Google Drive**

Se utilizó Google Drive para organizar y gestionar los datos para el entrenamiento y test del modelo, así como almacenamiento de los modelos generados durante la fase de entrenamiento.

- **Kaggle**

Repositorio digital de datos el cual proporcionó el conjunto de datos para el entrenamiento y test del modelo, tal como se presenta en la sección **6.1.1** de los resultados.

- **Pytorch**

Al ser una biblioteca de código abierto la cual permite definir y modificar redes neuronales de manera intuitiva esta fue utilizada para el desarrollo del código de entrenamiento del modelo para la clasificación de imágenes de la mancha bacteriana en la hoja de tomate.

5.4 Participantes

Para el desarrollo del trabajo de integración curricular fue necesaria la participación de las siguientes personas:

- Jennifer Elizabeth Quizhpe Hurtado como autora para el desarrollo del Trabajo de Integración Curricular conforme a los objetivos planteados en los apartados **5.2.1** y **5.2.2**.
- Ing. Oscar Miguel Cumbicus Pineda como director del Trabajo de Integración Curricular, el cual realizó las correcciones y supervisión del desarrollo de las tareas planteadas para la culminación del mismo.

6. Resultados

En la siguiente sección se presentan y detallan los resultados obtenidos durante la ejecución de los objetivos específicos, las actividades desarrolladas se establecieron en la **sección 5 de metodología** las cuales se centraron en la preparación de datos, desarrollo, entrenamiento y evaluación del modelo.

6.1 Objetivo 1: Ajustar el modelo VGG16 para la clasificación de imágenes de la mancha bacteriana en la hoja de tomate de riñón.

Para la ejecución de este objetivo se adaptaron las fases de Ingeniería de datos e Ingeniería del modelo de la metodología CRIPS-ML(Q), por lo que a continuación se detallan los resultados obtenidos:

Fase 1: Ingeniería de datos

6.1.1 Tarea 1: Identificar y recolectar imágenes de la hoja de tomate en repositorios como Kaggle, estas deben ser sanas o infectadas por la mancha bacteriana.

Para el proceso de entrenamiento y validación del modelo se obtuvo el conjunto de datos PlantVillage[2]" el cual se adquirió del repositorio digital Kaggle. En la **Figura 15** se presenta la divisóin de los 2 directorios "train" y "val", siendo el segundo el que se utilizó para el test del mejor modelo obtenido durante el proceso de entrenamiento. Este conjunto ha sido utilizado en estudios previos sobre clasificación de imágenes, debido a la cantidad de datos que contiene los cuales pertenecen a diferentes cultivos con sus respectivas enfermedades.

[2] Enlace dataset https://www.kaggle.com/datasets/mohitsingh1804/plantvillage

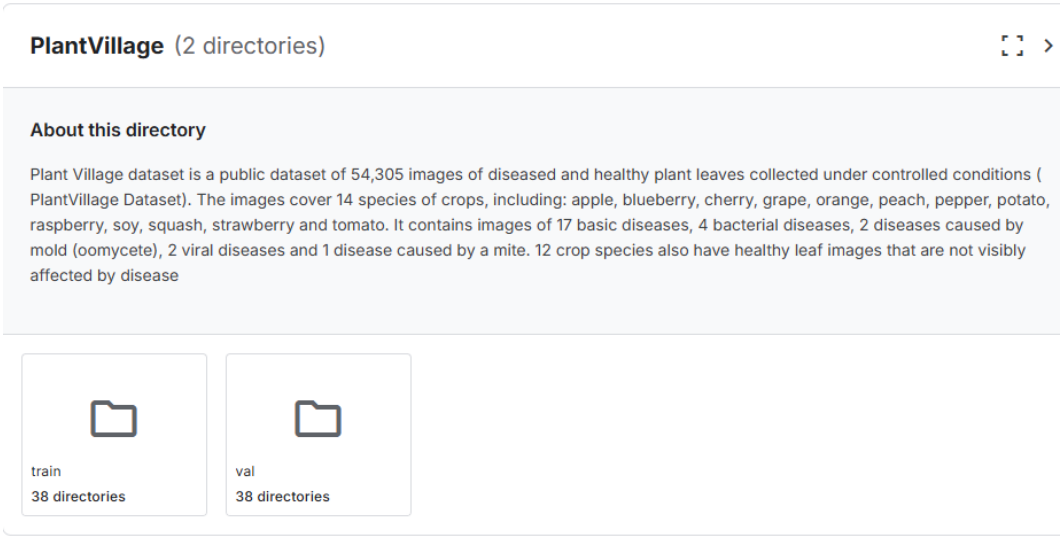

Figura 15 Directorios del dataset "PlantVillage"

Una vez adquirido el conjunto de datos general, se realizó una exploración de datos permitiendo identificar las imágenes pertenecientes a las clases de interés para la investigación. En la **Figura 16** se presenta las clases seleccionadas como conjunto de datos, los cuales pertenecen al cultivo de tomate siendo hojas sanas como afectadas por la mancha bacteriana.

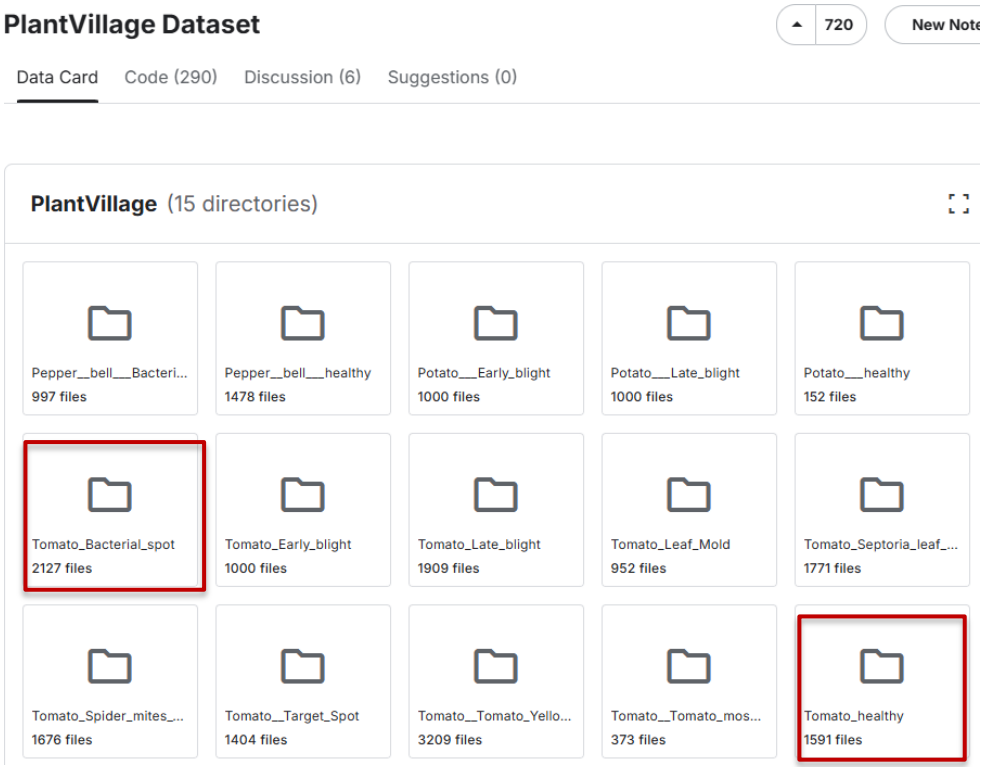

Figura 16 Clases seleccionada del conjunto de datos "PlantVillage" utilizadas para entrenamiento y validación del modelo

Obtenido el dataset general se seleccionó la clase “Tomato_Bacterial_Spot” y “Tomato_healthy” obteniendo un total de 2975 para el entrenamiento y 743 imágenes para validación del modelo. En la **Tabla 5** se detalla la cantidad específica de imágenes por cada clase.

Tabla 5 Cantidad de imágenes recolectadas para entrenamiento y validación

Clase	Nro de imágenes para entrenamiento	Nro de imágenes para validación
Healthy	1273	318
BacterialSpot	1702	425
	2075	743

En la **Figura 17** se presenta una muestra de las imágenes obtenidas de los conjuntos de datos, están son de hojas de tomate sanas como de las afectadas por la mancha bacteriana.

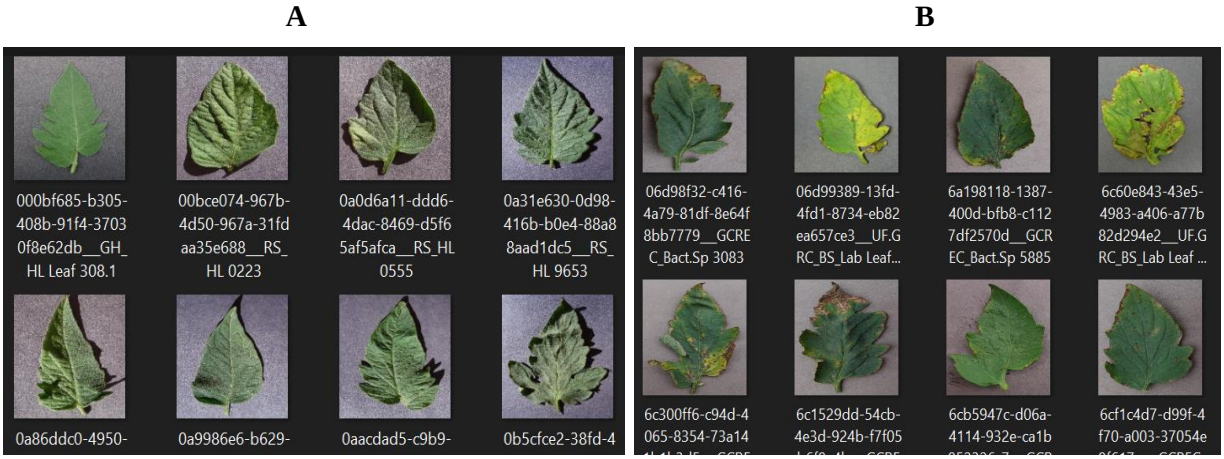

Figura 17 Imágenes de hojas de tomate del conjunto de datos "PlantVillage" donde la imagen (A) pertenece a la clase Sano mientras que la imagen (B) a la clase mancha Bacteriana

6.1.2 Tarea 2: Procesamiento de los datos aplicando limpieza, balanceo y técnicas de aumento.

6.1.2.1 Limpieza de datos

Con el fin de obtener resultados óptimos durante el entrenamiento del modelo se eliminaron aquellas imágenes pixeladas o de baja resolución para ello se aplicó la librería OpenCV. En la **Figura 18** se presenta el código utilizado para definir si una imagen es borrosa, el cual convierte a escalas de grises las imágenes del dataset calculando la varianza de cada imagen con el fin de medir los cambios de intensidad de píxeles para posteriormente comparar con valor del umbral ingresado en la función, determinando así si una imagen es borrosa o no.

Sin embargo, para definir si existe alguna imagen pixelada se obtiene las dimensiones (ancho y alto) de cada una comparándolas con el valor del umbral ingresado, en caso de ser menor esta es considerada de baja resolución. En la **Figura 18** se presenta el código utilizado para identificar si una imagen esta pixelada o no.

```
def es_borrosa(image, threshold=150.0):
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    variance = cv2.Laplacian(gray_image, cv2.CV_64F).var()
    return variance < threshold

def es_pixelada(image, threshold=150):
    height, width = image.shape[:2]
    return height < threshold or width < threshold
```

Figura 18 Código para limpieza de los datos

En la **Figura 19** se presentan los resultados obtenidos al aplicar los filtros mencionados, se eliminaron aquellas imágenes que se consideraron de baja calidad generando un nuevo conjunto de datos.

```
Imagen borrosa eliminada: /content/drive/MyDrive/Colab Notebooks/TESIS/dataset/train/TomateManchaBacteriana/7688c6b6-910a-4f6f-854e-e00bf11ce9b0___UF.GRC_BS_Lab Leaf 904
Imagen borrosa eliminada: /content/drive/MyDrive/Colab Notebooks/TESIS/dataset/train/TomateManchaBacteriana/830824a7-c51d-477e-b987-19ae4d388f48___UF.GRC_BS_Lab Leaf 883
Imagen borrosa eliminada: /content/drive/MyDrive/Colab Notebooks/TESIS/dataset/train/TomateManchaBacteriana/8c780548-b142-4d08-b98d-22492ef5cc4f___UF.GRC_BS_Lab Leaf 897
Imagen borrosa eliminada: /content/drive/MyDrive/Colab Notebooks/TESIS/dataset/train/TomateManchaBacteriana/94532bf7-7d74-455a-8526-938c93da0d63___UF.GRC_BS_Lab Leaf 922
Imagen borrosa eliminada: /content/drive/MyDrive/Colab Notebooks/TESIS/dataset/train/TomateManchaBacteriana/9b1443e6-25d0-4dbb-a3f4-b586d7cce49f___UF.GRC_BS_Lab Leaf 876
Imagen borrosa eliminada: /content/drive/MyDrive/Colab Notebooks/TESIS/dataset/train/TomateManchaBacteriana/a58a9c80-0f76-42ac-8f85-a734ea417286___GCREC_Bact.Sp 3017.JPG
Imagen borrosa eliminada: /content/drive/MyDrive/Colab Notebooks/TESIS/dataset/train/TomateManchaBacteriana/bc54f201-4a3b-446f-af60-ea26285893c8___UF.GRC_BS_Lab Leaf 895
Imagen borrosa eliminada: /content/drive/MyDrive/Colab Notebooks/TESIS/dataset/train/TomateManchaBacteriana/c742e379-a56a-44b5-96d0-1918dd2cb2c9___GCREC_Bact.Sp 3182.JPG
Imagen borrosa eliminada: /content/drive/MyDrive/Colab Notebooks/TESIS/dataset/train/TomateManchaBacteriana/c9c0a326-c094-4cea-94cb-bc240bb3be23___GCREC_Bact.Sp 3652.JPG
Imagen borrosa eliminada: /content/drive/MyDrive/Colab Notebooks/TESIS/dataset/train/TomateManchaBacteriana/d36baaa6-29eb-4bc6-8f05-252fdceee6fc___UF.GRC_BS_Lab Leaf 035
Imagen borrosa eliminada: /content/drive/MyDrive/Colab Notebooks/TESIS/dataset/train/TomateManchaBacteriana/0d56df83-84fb-4189-a5d2-3a6da18a224d___UF.GRC_BS_Lab Leaf 902
Imagen borrosa eliminada: /content/drive/MyDrive/Colab Notebooks/TESIS/dataset/train/TomateManchaBacteriana/1a50d84b-c015-45ea-a041-500d5adcc339___UF.GRC_BS_Lab Leaf 877
Imagen borrosa eliminada: /content/drive/MyDrive/Colab Notebooks/TESIS/dataset/train/TomateManchaBacteriana/1d83362f-5ea5-40f4-aa21-dfff0e99aff7___UF.GRC_BS_Lab Leaf 056
Imagen borrosa eliminada: /content/drive/MyDrive/Colab Notebooks/TESIS/dataset/train/TomateManchaBacteriana/3aa670b8-1a95-41cb-8b98-bf498970f33e___UF.GRC_BS_Lab Leaf 877
Imagen borrosa eliminada: /content/drive/MyDrive/Colab Notebooks/TESIS/dataset/train/TomateManchaBacteriana/542e3014-3c3c-40c2-bf7e-73287f5302e4___GCREC_Bact.Sp 3157.JPG
Imagen borrosa eliminada: /content/drive/MyDrive/Colab Notebooks/TESIS/dataset/train/TomateManchaBacteriana/55b8f3a6-d613-4437-860b-dc50459dd07c___UF.GRC_BS_Lab Leaf 922
```

Figura 19 Resultado de la limpieza de datos

En la **Tabla 6** se detalla la cantidad de imágenes que fueron descartadas al aplicar los filtros.

Tabla 6 Cantidad de imágenes descartadas al aplicar limpieza de datos

Clase	Nro de imágenes descartadas
Healthy	1
BacterialSpot	16
	17

Inicialmente el conjunto de datos contaba con un total de 2975 imágenes, 1273 de la clase *healthy* y 1702 de la clase *bacterialSpot*. En la **Tabla 7** se detalla la cantidad de imágenes

obtenidas al realizar la limpieza de datos generando un total de 2958 imágenes para el entrenamiento del modelo.

Tabla 7 Cantidad de imágenes obtenidas al aplicar limpieza de datos

Clase	Nro de imágenes limpias
Healthy	1272
BacterialSpot	1686
	2958

6.1.2.2 Balanceo de datos

El conjunto de imágenes para el entrenamiento del modelo se encuentra desbalanceado, por lo que se aplica la técnica *undersampling* para equilibrar la cantidad de imágenes en cada clase. Esta técnica consiste en eliminar instancias de la clase mayoritaria, por lo que es necesario conocer la cantidad de imágenes existentes en cada una, donde aquella con menor cantidad se convierte en referencia para reducir la otra clase al mismo tamaño. En la **Figura 20** se presenta el código utilizado para balancear la cantidad de imágenes en cada clase.

```
def undersample_dataset(dataset):
    class_counts = Counter([sample[1] for sample in dataset])
    min_class = min(class_counts, key=class_counts.get)
    min_count = class_counts[min_class]

    class_indices = {class_label: [] for class_label in class_counts.keys()}

    for idx, (data, label) in enumerate(dataset):
        class_indices[label].append(idx)

    undersampled_indices = []
    for label, indices in class_indices.items():
        if len(indices) > min_count:
            undersampled_indices.extend(np.random.choice(indices, size=min_count, replace=False))
        else:
            undersampled_indices.extend(indices)

    return Subset(dataset, undersampled_indices)
```

Figura 20 Código para balanceo de datos

Una vez eliminadas las instancias de la clase mayoritaria se obtiene un conjunto de datos equilibrado, este proceso garantiza que el modelo no tenga sesgo al limitar el número de imágenes a la cantidad disponible en la clase minoritaria permitiendo así que el modelo aprenda con la misma cantidad de instancias. En la **Tabla 8** se detalla la cantidad de imágenes generadas al balancear cada clase.

Tabla 8 Cantidad de imágenes obtenidas al aplicar balanceo de datos

Clase	Nro de imágenes balanceadas
Healthy	1272
BacterialSpot	1272
	2544

6.1.2.3 Aumento de datos

Una vez balanceados los datos se obtiene aleatoriamente el 50% de las imágenes con el fin de aplicar las técnicas de aumento mencionadas en la **sección 5** de metodología**.** Para ello se utilizó la biblioteca Pytorch lo que generó nuevas imágenes por cada una de las originales, incrementando considerablemente el dataset. El 50% de datos a los cuales no se les realiza transformaciones son almacenados junto a las nuevas imágenes generadas, dando origen al dataset final con el cual se realizó el proceso de entrenamiento el modelo. En la **Tabla 9** se detalla la cantidad de imágenes seleccionadas para aplicar las técnicas de aumento que se mencionan a continuación.

Tabla 9 Cantidad imágenes seleccionadas para aplicar técnicas de aumento

Conjunto	Cantidad imágenes
Imágenes seleccionadas	1272

- **Recorte**

En la **Figura 21** se presenta el código utilizado para aplicar recorte en las imágenes, esta técnica consiste en recortar una porción aleatoria de la imagen redimensionándola a un tamaño específico como el de 224*224 píxeles.

```
transforms.RandomResizedCrop(224, scale=(0.8, 1.0)),
```

Figura 21 Código para aplicación de recorte

En la **Figura 22** se presenta un ejemplo de las imágenes resultantes al aplicar esta técnica destacando las características más relevantes de la imagen.

A B

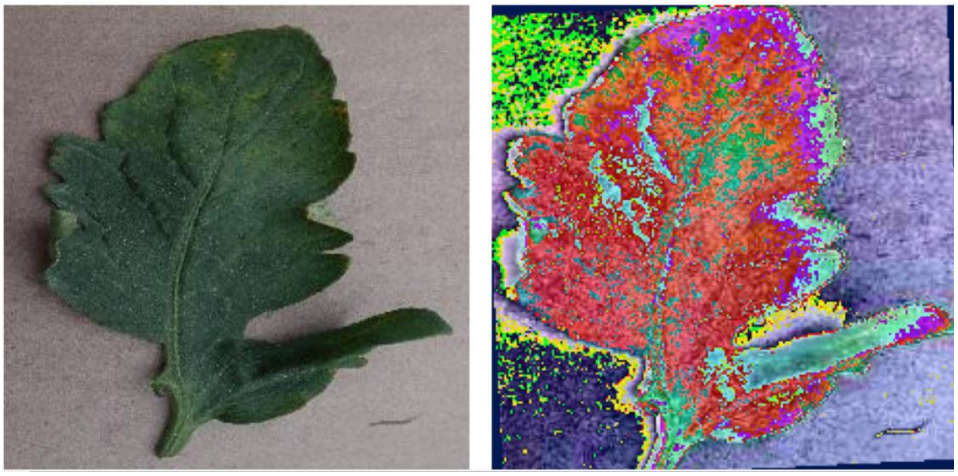

Figura 22 Aplicación de la técnica de recorte. En la imagen (A) se presenta la imagen original mientras que la imagen (B) muestra el resultado obtenido al recortar la imagen con una proporción entre el 80% y 100%.

- **Rotación**

Esta técnica consiste en establecer un giro aleatorio obteniendo como resultado una nueva imagen con grado de rotación en este caso de 15° en cualquier dirección, generando mayor diversificación en el conjunto de datos. En la **Figura 23** se muestra el resultado obtenido al aplicar la técnica mencionada.

A B

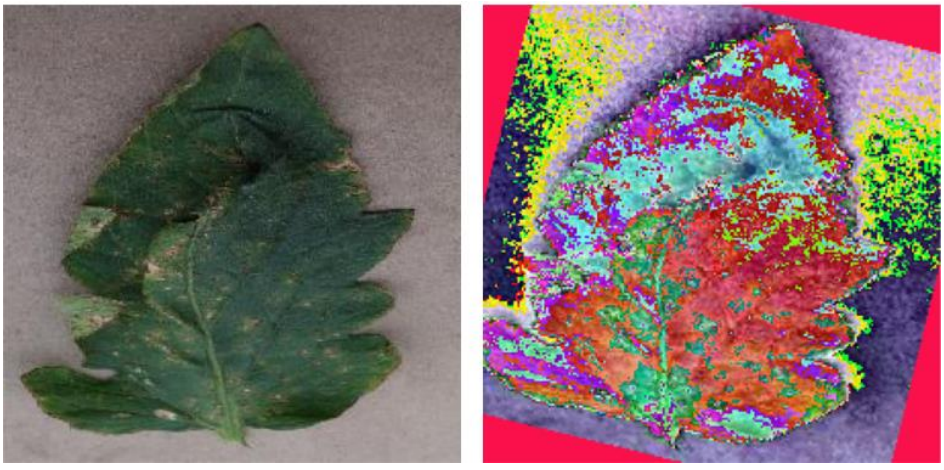

Figura 23 Aplicación de la técnica de rotación. En la imagen (A) se presenta la imagen original mientras que la imagen (B) muestra el resultado obtenido al girar la imagen

En la **Figura 24** se presenta el código empleado para la aplicar esta técnica en el cual se utilizó de la librería Pytorch, esta permite la lectura de las imágenes de cada clase del conjunto de datos para posteriormente aplicar el giro de la imagen original.

```
transforms.RandomRotation(15),
```

Figura 24 Código para aplicación de rotación aleatoria de 15°

- **Brillo, contraste y saturación**

Esta técnica se centró en la aplicación de *ColorJitter* el cual permite ajustar aleatoriamente el brillo, contraste, saturación y matiz de las imágenes. Para el contraste se estableció un valor de 0.1 como multiplicador, lo que significa que los valores de intensidad de los pixeles escalan en un 10 resaltando las diferencias de intensidad. Para el brillo se asigna un valor de 0.1 el cual se suma a cada píxel desplazando la intensidad general de la imagen. En la en la **Figura 25** se presenta el código utilizado para ajustar los valores de brillo, contraste y saturación en las imágenes.

```
transforms.ColorJitter(brightness=0.1, contrast=0.1, saturation=0.1, hue=0.02),
```

Figura 25 Código de ajuste del brillo y contraste en las imágenes

Esta combinación de ajustes permite crear variaciones en las imágenes, simulando distintas condiciones de iluminación lo que mejora la robustez del modelo. En la **Figura 26** se presentan un ejemplo del resultado obtenido al modificar la intensidad de brillo y contraste en las imágenes del conjunto de datos.

A **B**

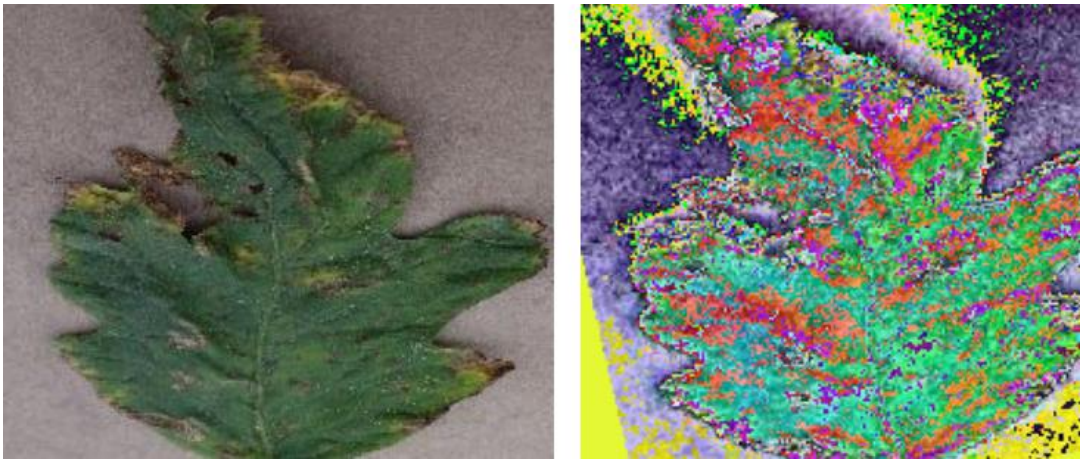

Figura 26 Aplicación de ajustes de brillo y contraste. En la imagen (A) se presenta la imagen original mientras que la imagen (B) muestra el resultado obtenido al aplicar los ajustes mencionados.

- **Volteo**

Para aplicar esta técnica se utilizó la biblioteca *torchvision.transform* la cual permite realizar un giro horizontal en las imágenes invirtiendo la imagen a los largo del eje Y de manera aleatoria con una probabilidad del 50%. Esta función asegura que cada imagen tenga una versión reflejada lo que aumenta la diversidad del conjunto de datos y mejora la capacidad del modelo para generalizar. En la **Figura 27** se presenta el código utilizado para realizar el volteo horizontal de las imágenes.

```
for file in selected_files:
    image_path = os.path.join(root, file)
    image = cv2.imread(image_path)

    flipped_img = cv2.flip(image, 1)  # Volteo horizontal
    flipped_img_path = os.path.join(class_dir, f"{os.path.splitext(file)[0]}_volteo.jpg")
    cv2.imwrite(flipped_img_path, flipped_img)
    total_imagenes += 1
```

Figura 27 Código para aplicar técnica de volteo horizontal

En la **Figura 28** se presenta el resultado de la transformación realizada, donde la imagen **A** es aquella a la cual se aplicó la técnica de volteo.

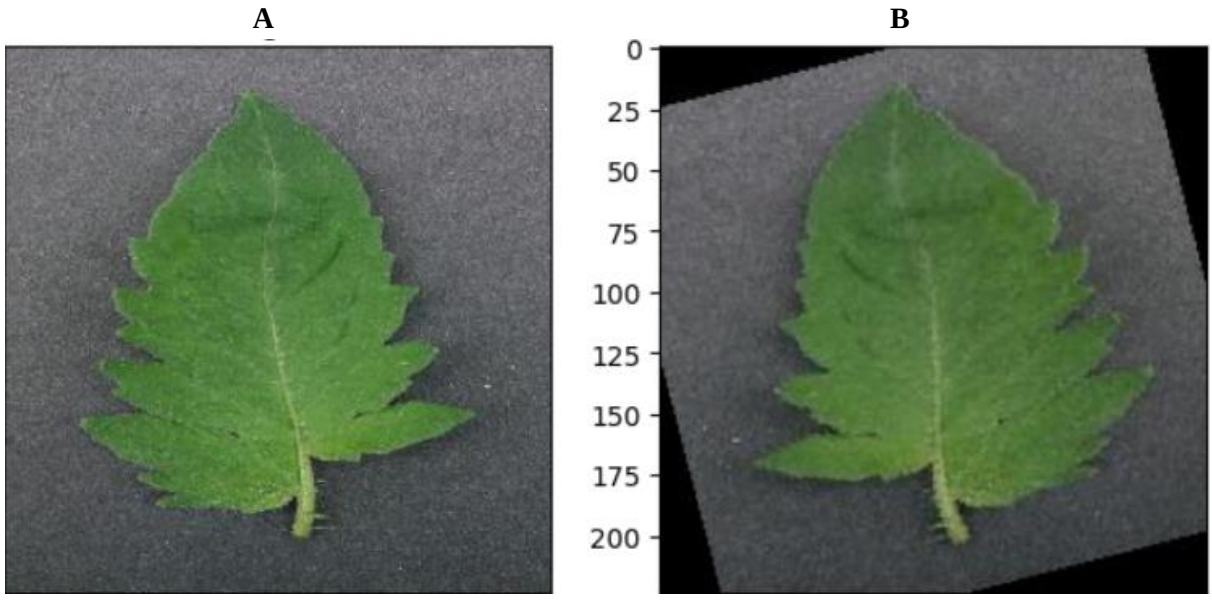

Figura 28 Aplicación de la técnica de volteo. En la imagen (A) se presenta la imagen original mientras que la imagen (B) muestra el resultado obtenido al voltear la imagen.

La aplicación de estas técnicas de aumento en el 50% del conjunto de datos de entrenamiento dio como resultado la obtención total de 3816 imágenes para el entrenamiento

del modelo generando así un dataset[3] con mayor cantidad de datos lo que permite mayor generalización. En la **Tabla 10** se detalla la cantidad exacta de imágenes pertenecientes a cada clase.

Tabla 10 Cantidad de imágenes totales después de aplicar técnicas de aumento de datos.

Conjunto	Cantidad de imágenes
Imágenes seleccionadas	1272
Imágenes aumentadas	2544
	3816

A su vez estas técnicas se aplicaron al conjunto "val" generando un conjunto de datos con el 50% de imágenes aumentadas y aquellas que no se utilizaron para dicho proceso, obteniendo un total de 743 imágenes para el test siendo el valor del conjunto original. En la **Figura 29** se presenta el código utilizado para este proceso.

```python
def apply_augmentation(image):
    """Applies data augmentation with a 15% probability."""
    if random.random() < 0.5:
        transform = transforms.Compose([
            transforms.RandomResizedCrop(224, scale=(0.8, 1.0)),
            transforms.RandomHorizontalFlip(p=0.5),
            transforms.RandomRotation(15),
            transforms.ColorJitter(brightness=0.1, contrast=0.1, saturation=0.1, hue=0.02),
        ])
        image = transform(image)
    image = transforms.Resize((224, 224))(image)
    return image


class AugmentedImageFolder(torchvision.datasets.ImageFolder):
    def __getitem__(self, index):
        path, target = self.imgs[index]
        image = self.loader(path)
        image = apply_augmentation(image)
        if self.transform is not None:
            image = self.transform(image)
        return image, target
```

Figura 29 Código para aplicar técnicas de aumento de datos en el conjunto "val"

6.1.3 Tareas 3: Dividir el dataset obtenido en conjuntos de entrenamiento, validación y prueba.

El conjunto de datos final se dividió para entrenamiento y validación en 80/20, donde el porcentaje de imágenes asignado para cada carpeta. En Pytorch la selección de las imágenes

[3] Enlace al conjunto de datos generado para el entrenamiento del modelo https://drive.google.com/drive/folders/1RUm3ewCXH2bGB0O5X27YazA6nclir1VM?usp=sharing

para la carpeta de entrenamiento y validación se realizó aleatoriamente según el tamaño especificado, previa definición de una semilla la cual permite que el proceso sea reproducible. En la **Figura 30** se presenta la respectiva gráfica indicando los valores absolutos de cada categoría junto con sus respectivos porcentajes, las secciones se encuentran diferenciadas por colores y líneas divisoras.

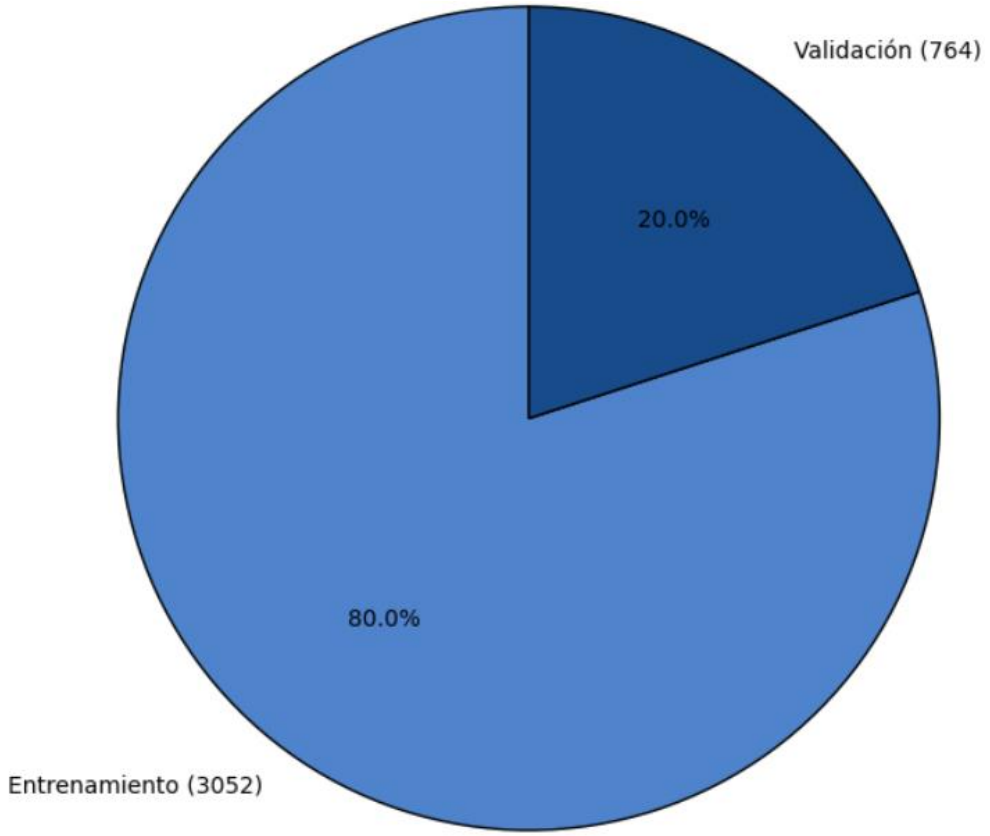

Figura 30 Distribución del conjunto de datos. Categoría "Entrenamiento" representa el 80% con un total de 3052 imágenes y el 20% representa la categoría "Validación" con 764 imágenes.

Fase 2: Ingeniería del modelo

6.1.4 Tarea 4: Ajustar los hiperparámetros del modelo VGG16.

El uso de modelos preentrenados permite aprovechar las características aprendidas en millones de imágenes obteniendo modelos que realizan una tarea específica. Para el desarrollo de un modelo que clasifique si una imagen de hoja de tomate esta sana o afectada por la mancha bacteriana se utilizó una arquitectura VGG16 preentrenada y ajustada para que realice una tarea binaria, lo que permitió aprovechar las capacidades de extracción de características avanzadas de redes convolucionales profundas. En la **Figura 31** se muestra que el modelo se define para ser entrenado con GPU o CPU asegurando una óptima ejecución en base a los recursos disponibles.

```
num_classes = 2
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = VGG16FineTune(num_classes).to(device)
```

Figura 31 Código para definir recursos según la disponibilidad de hardware

Para el proceso de entrenamiento del modelo fue necesario realizar ajuste en hiperparámetros de la arquitectura original estableciendo como función de pérdida *CrossEntropyLoss* siendo esta una de las más óptimas para problemas de clasificación multiclase y, como optimizador del modelo se aplicó el algoritmo *Adam* el cual incluye regularización por descomposición de pesos y mejora la estabilidad de las actualizaciones del mismo. Además, se utilizó un programador de tasa de aprendizaje, el cual resulta útil para evitar el sobreajuste y mejorar la capacidad del modelo para generalizar. En la **Figura 32** se presenta el código utilizado para definir los valores de los hiperparámetros a utilizar.

```
criterion = nn.CrossEntropyLoss()
optimizer = optim.AdamW(model.parameters(), lr=1e-4, betas=(0.9, 0.999), weight_decay=1e-2)
scheduler = optim.lr_scheduler.StepLR(optimizer, step_size=7, gamma=0.1)
```

Figura 32 Código para definir valores de los hiperparámetros

En la **Tabla 11** se detallan los valores de los hiperparámetros utilizados en los experimentos realizados con el fin de optimizar el rendimiento del modelo, cada experimento se diseñó para evaluar el impacto de los hiperparámetros en la efectividad del modelo durante el entrenamiento.

Tabla 11 Valores de los hiperparámetros ajustados

Hiperarámetros		**Experimentos**				
		1	**2**	**3**	**4**	**5**
Learning rate		1e-3	1e-4	1e-5	1e-5	1e-6
Betas	β_1	0.8	0.9	0.9	0.8	0.9
	B_2	0.888	0.999	0.999	0.888	0.999
Weight decay		1e-3	1e-2	1e-5	1e-6	1e-4
Dropout		0.2	0.3	0.4	0.6	0.5
Batch size		32	32	64	32	64
Función de activación		Relu	Relu	Relu	Relu	Relu

Hiperarámetros	Experimentos				
	1	2	3	4	5
Algoritmo de optimización	Adam	AdamW	Adam	AdamW	Adam
Capas descongeladas	Ninguna	Últimas 5	Últimas 3	Ninguna	Último bloque (10 capas)

En esta tabla se puede observar cómo los diferentes valores de *learning rate* permiten evaluar su efecto en la convergencia del modelo, donde las combinaciones del parámetro beta influye en la adaptación dinámica de las tasas de aprendizaje afectando la estabilidad y rapidez con que el modelo converge. Así mismo con el fin de prevenir el sobreajuste establecieron valores de decaimiento del peso, esto regula el modelo y mejora la generalización frete a datos desconocidos.

6.1.5 Tarea 5: Entrenar el modelo con el dataset generado

Para la fase de entrenamiento del modelo es necesario configurar los parámetros iniciales donde el número de épocas se estableció en un valor de 40 iteraciones. Además de las listas que permitirán el registro de pérdidas y precisiones durante el entrenamiento como la validación del modelo, esto con el fin de visualizar a través de gráficas los resultados obtenidos en cada iteración. En la **Figura 33** se presentan los parámetros necesarios para inicializar el proceso de entrenamiento del modelo.

```
num_epochs = 40
train_losses, val_losses = [], []
train_accuracies, val_accuracies = [], []
best_acc = 0.0
best_model_path = ''
```

Figura 33 Parámetros iniciales para entrenamiento del modelo

Establecidos los hiperparámetros se logró completar el proceso de entrenamiento del modelo el cual se ejecutó dentro del entorno de Google Colab. En donde, el modelo se establece en modo entrenamiento a través de la función *model.train()* obteniendo los datos generados en *train_loader* y calculando las predicciones en cada época. La pérdida generada en cada iteración se calcula utilizando la función *criterion* retropropagando las gradientes y actualizando los parámetros del modelo a través del optimizador. En la **Figura 34** se presenta el código utilizado para el entrenamiento de los modelos.

```
for epoch in range(num_epochs):
    model.train()
    running_loss = 0.0
    correct = 0
    total = 0
    for images, labels in train_loader:
        images, labels = images.to(device), labels.to(device)
        optimizer.zero_grad()
        outputs = model(images)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()

        running_loss += loss.item() * images.size(0)
        _, predicted = torch.max(outputs.data, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()

    train_losses.append(running_loss / total)
    train_accuracies.append(correct / total)
    print(f'Epoch [{epoch + 1}/{num_epochs}], Train Loss: {train_losses[-1]:.4f}, Train Accuracy: {train_accuracies[-1]:.4f}')
```

Figura 34 Código utilizado para entrenamiento del modelo

Finalizado el proceso de entrenamiento por cada uno de los experimentos definidos se obtuvo sus valores de precisión y pérdida demostrando el comportamiento de aprendizaje durante las 40 épocas. En la **Tabla 12** se presenta un resumen de los resultados obtenidos durante la ejecución de distintas épocas detallando que los modelos lograron alcanzar valores entre el 99% y 100% de precisión durante esta fase. A continuación, se detalla los valores obtenidos durante el entrenamiento del modelo por cada experimento realizado donde a medida que avanzan las épocas los valores *loss* van disminuyendo lo que indica que el modelo está aprendiendo y ajustando mejor los datos de entrenamiento. Mientras que la precisión (*acc*) aumenta con el número de épocas alcanzando valores cercanos o iguales a 1.

Tabla 12 Resultados de pérdida y precisión obtenidos por época durante la ejecución del entrenamiento del modelo

Épocas	Experimentos									
	1		2		3		4		5	
	Loss	**Acc**	**Loss**	**Acc**	**Loss**	**Acc**	**Loss**	**Acc**	**Loss**	**Acc**
1/40	0.1369	0.9590	0.0722	0.9721	0.1889	0.9335	0.2539	0.9037	0.6013	0.6710
2/40	0.0733	0.9731	0.0089	0.9974	0.0416	0.9885	0.0652	0.9830	0.3650	0.8748
…	…	…	…	…	…	…	…	…	…	…
6/40	0.0466	0.9846	0.0059	0.9987	**0.0067**	**1.0000**	0.0126	0.9957	0.0858	0.9833
…	…	…	…	…	…	…	…	…	…	…
9/40	0.0109	0.9971	**0.0009**	**1.0000**	0.0039	1.0000	0.0070	0.9984	0.0508	0.9898
…	…	…	…	…	…	…	…	…	…	…
13/40	0.0051	0.9977	0.0003	1.0000	0.0035	1.0000	**0.0028**	**1.0000**	0.0568	0.9875
…	…	…	…	…	…	…	…	…	…	…
27/40	**0.0012**	**1.0000**	0.0002	1.0000	0.0026	1.0000	0.0037	0.9987	**0.0489**	**0.9934**
28/40	0.0025	0.9990	0.0004	1.0000	0.0035	1.0000	0.0044	0.9990	0.0543	0.9895
29/40	0.014	0.9997	0.0001	1.0000	0.0028	1.0000	0.0026	0.9997	0.0527	0.9915
…	…	…	…	…	…	…	…	…	…	…
40/40	0.0007	1.0000	0.0003	1.0000	0.0028	1.0000	0.0037	0.9990	0.0519	0.9895

Finalmente el mejor modelo obtenido por experimento se empaquetó en formato ".pth" para posteriormente ser utilizado en la fase de evaluación. En la **Figura 35** se muestra la función con la que se almacenó el mejor modelo obtenido.

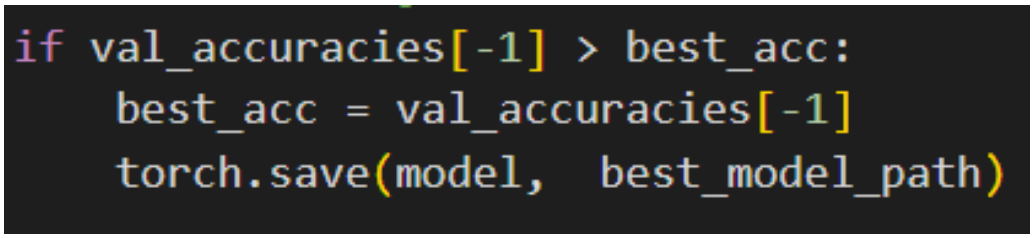

Figura 35 Función para guardar el mejor modelo

En la **Tabla 13** se detallan los valores óptimos con respecto a la precisión y pérdida durante el entrenamiento de los modelos.

Tabla 13 Precisión y pérdida obtenidos durante el entrenamiento de cada modelo

	Experimentos				
	1	**2**	**3**	**4**	**5**
Precisión	1.0000	1.0000	1.0000	1.0000	0.9934
Pérdida	0.0012	0.0009	0.0067	0.0028	0.0489

6.2 Objetivo 2: Evaluar el modelo entrenado a través de la precisión global de acierto al clasificar imágenes de la mancha bacteriana en hojas de tomate de riñón.

Para el desarrollo del presente objetivo se adaptó la fase Evaluación de modelos de aprendizaje automático de la metodología CRISP-ML(Q), misma que se describe a continuación:

Fase 3: Evaluación del modelo

6.2.1 Tarea 6: Evaluación del modelo en el conjunto del entrenamiento

En la **Figura 36** se puede visualizar la curva de aprendizaje que se logró obtener durante el entrenamiento del *experimento 1*[4] en el cual se obtuvo como pérdida aproximadamente 0.2616 y precisión de 0.9228 lo que indica que el modelo con cada iteración aprende eficazmente.

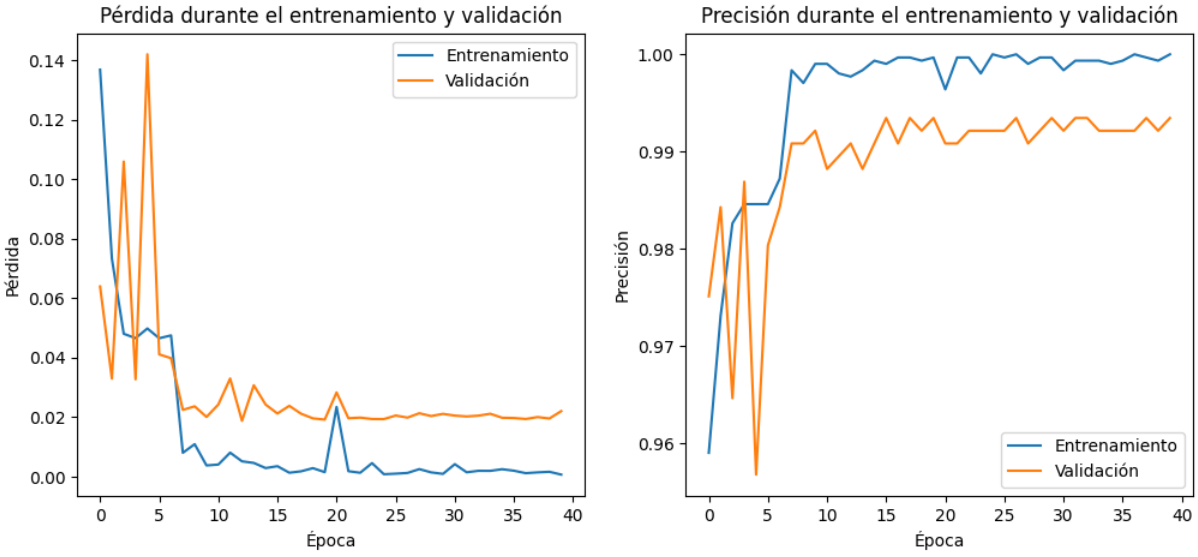

Figura 36 Gráficas de pérdida y precisión durante el entrenamiento y validación del experimento 1

Sin embargo, durante el entrenamiento de los modelos generados en los experimentos restantes el comportamiento de aprendizaje se mantiene constante con valores de pérdida

[4] Entrenamiento del modelo del experimento 1: https://drive.google.com/file/d/17wJMjHbBz1C7xej8wugXBoNRi3bSSeUV/view?usp=sharing

cercanos a 0 y precisión cercana a 1. En la **Figura 37** se presenta la curva de aprendizaje correspondiente al *experimento 2*[5] donde la línea azul corresponde al comportamiento durante el entrenamiento del modelo y la línea naranja a la validación del mismo, demostrando que el modelo tiene un buen desempeño al generar una pérdida baja y precisión alta en ambas fases.

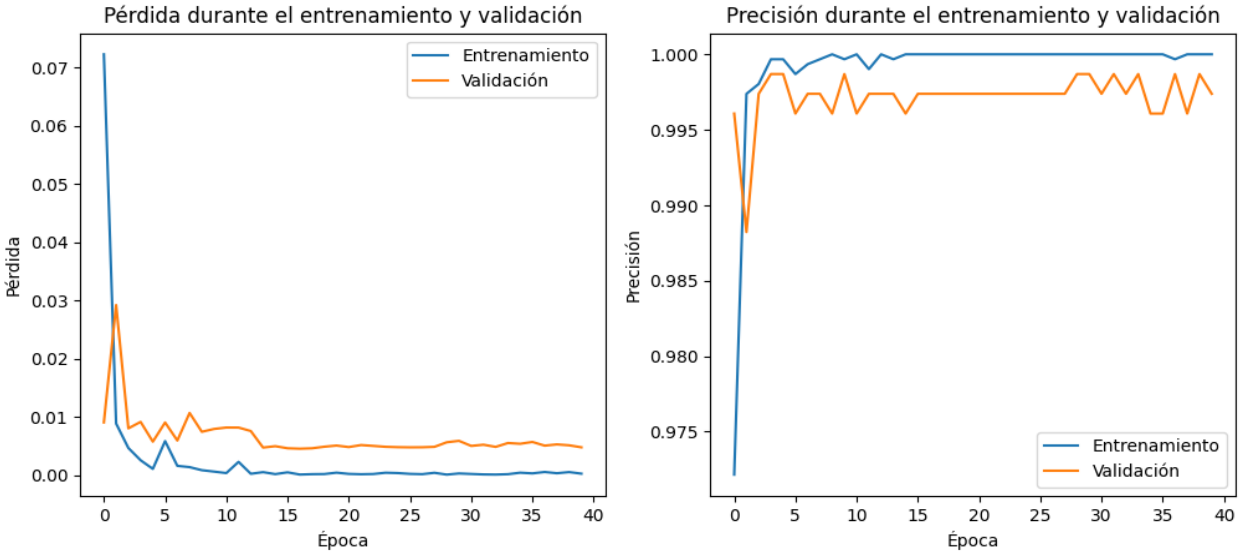

Figura 37 Gráficas de pérdida y precisión durante el entrenamiento y validación del experimento 2

En la **Figura 38** se muestra el comportamiento que el modelo obtuvo durante la fase de entrenamiento, esto como parte del *experimento 3*[6].

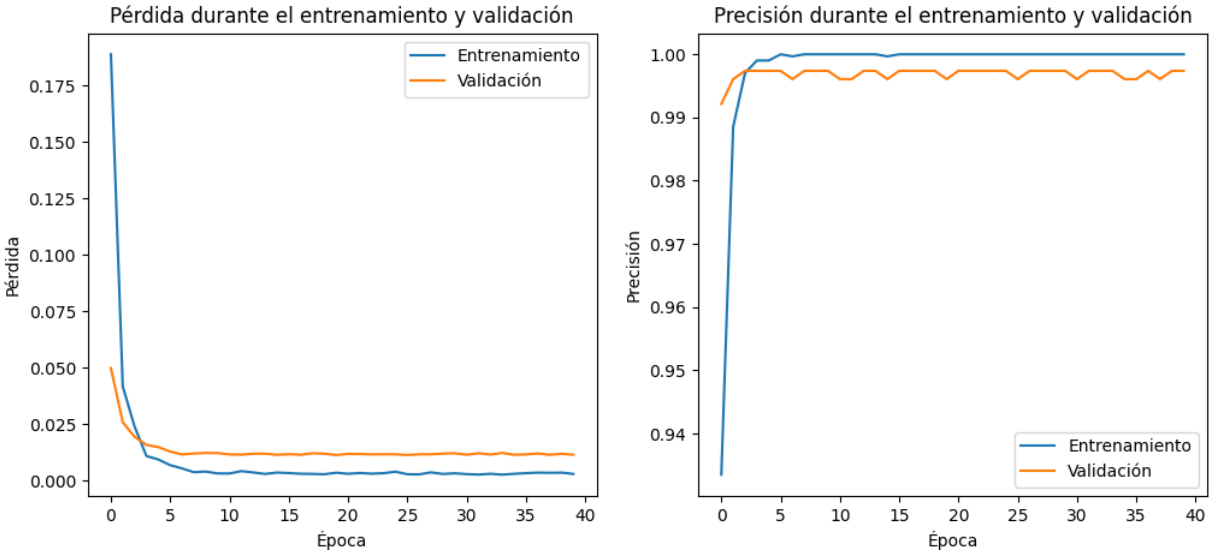

Figura 38 Gráficas de pérdida y precisión durante el entrenamiento y validación del experimento 3

[5] Entrenamiento del modelo del experimento 2:
https://colab.research.google.com/drive/1YmahXnh2xEwf8t3TS9TkoQ8ujOLQB9p9?usp=sharing
[6] Entrenamiento del modelo del experimento 3:
https://drive.google.com/file/d/1W0TDk5QzftOZqPtN_zNMXGc65OjBRgrk/view?usp=sharing

En la **Figura 39** se presenta el comportamiento del modelo durante el *experimento 4*[7], demostrando que la curva de aprendizaje obtenida es óptima teniendo como precisión valores superiores al 90% y de pérdida cercanos a 0.

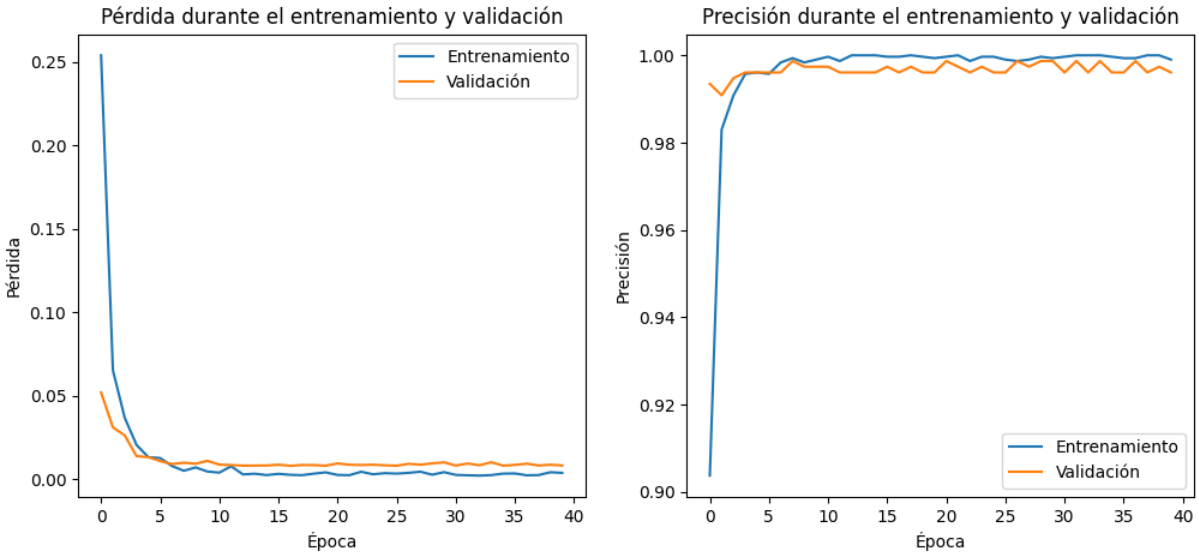

Figura 39 Gráficas de pérdida y precisión durante el entrenamiento y validación del experimento 4

En la **Figura 40** se presenta las gráficas de precisión y pérdida obtenidas durante la fase de entrenamiento del modelo generado en el *experimento 5*[8].

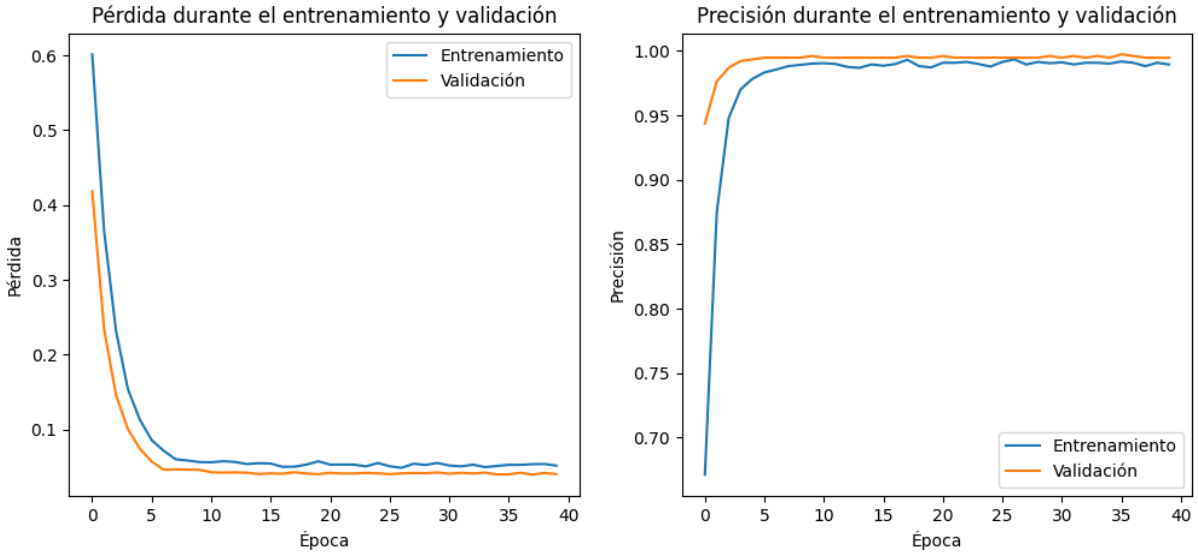

Figura 40 Gráficas de pérdida y precisión durante el entrenamiento y validación del experimento 5

[7] Entrenamiento del modelo del experimento 4: https://drive.google.com/file/d/1MlDygxst_ZOxjAS8KiVcoa8d6QIoPY5q/view?usp=sharing

[8] Entrenamiento del modelo del experimento 5: https://drive.google.com/file/d/1rJeLEOAphlvVSH44lSF92xhNVZt4b5gU/view?usp=sharing

Cada uno de los experimentos que se realizó para obtener la arquitectura e hiperparámetros óptimos se ejecutaron durante el número de épocas definidas con anterioridad obteniendo valores de precisión del 99% así como valores mínimos de pérdida. En la **Tabla 14** se detallan los resultados obtenidos en la validación de los modelos durante la fase de entrenamiento en la cual se visualiza que los valores de *loss* iniciales son menores en la mayoría de los experimentos en comparación a la **Tabla 12** lo que sugiere que el modelo comenzó con un mejor ajuste.

Tabla 14 Resultados de pérdida y precisión obtenidos por época en la validación del modelo durante la fase de entrenamiento

	Experimentos									
Épocas	**1**		**2**		**3**		**4**		**5**	
	Loss	**Acc**	**Loss**	**Acc**	**Loss**	**Acc**	**Loss**	**Acc**	**Loss**	**Acc**
1/40	0.0640	0.9751	0.0091	0.9961	0.0498	0.9921	0.0519	0.9935	0.4183	0.9437
2/40	0.0330	0.9843	0.0292	0.9882	0.0257	0.9961	0.0310	0.9908	0.2339	0.9764
3/40	0.1060	0.9647	0.0081	0.9974	0.0193	0.9974	0.0262	0.9948	0.1462	0.9869
4/40	0.0327	0.9869	**0.0092**	**0.9987**	**0.0157**	**0.9974**	0.0137	0.9961	0.1006	0.9921
…	…	…	…	…	…	…	…	…	…	…
8/40	0.0225	0.9908	0.0107	0.9974	0.0119	0.9974	**0.0098**	**0.9987**	0.0468	0.9948
9/40	0.0236	0.9908	0.0075	0.9961	0.0121	0.9974	0.0092	0.9974	0.0464	0.9948
	…	…	…	…	…	…	…	…	…	…
15/40	0.0242	0.9908	0.0050	0.9961	0.0113	0.9961	0.0082	0.9961	0.0406	0.9948
16/40	**0.0212**	**0.9935**	0.0046	0.9974	0.0116	0.9974	0.0086	09974	0.0415	0.9948
…	…	…	…	…	…	…	…	…	…	…
36/40	0.0197	0.9921	0.0057	0.9961	0.0115	0.9961	0.0085	0.9961	**0.0402**	**0.9974**
37/40	0.0194	0.9921	0.0051	0.9987	0.0119	0.9974	0.0092	0.9987	0.0422	0.9961
…	…	…	…	…	…	…	…	…	…	…
40/40	0.0220	0.9935	0.0048	0.9974	0.0113	0.9974	0.0081	0.9961	0.0405	0.9948

En la **Tabla 15** se detalla los valores de precisión y pérdida obtenidos durante el proceso de validación de los modelos, resaltando que estos fueron los más cercanos a 1.

Tabla 15 Valor de precisión y pérdida obtenidos durante la validación del modelo

	Experimentos				
Precisión	**1**	**2**	**3**	**4**	**5**
Pérdida	0.9935	**0.9987**	**0.9974**	**0.9987**	**0.9974**
Tiempo de ejecución	0.0212	0.0092	0.0157	0.0098	0.0402
Precisión	74m 29s	28m 16s	44m 59s	27m 30s	47m 37s

Nota: Experimentos 2 y 4 con precisión mayor. Experimentos 3 y 5 precisión intermedia con diferencia mínima. Experimento 1 con precisión más baja con respecto a los demás.

6.2.2 Tarea 7: Seleccionar el mejor modelo

En base a los resultados presentados en la tarea anterior se seleccionaron los 5 modelos obtenidos como los mejores con el fin de realizar pruebas que permitan obtener la precisión global de acierto, estos son elegidos debido a la diferencia mínima de precisión que existe entre ellos cuyo valor fue obtenido durante la validación del entrenamiento.

6.2.3 Tarea 8: Evaluar el rendimiento del modelo utilizando el conjunto de prueba

Como se menciona anteriormente se evaluaron los 5 modelos generados para lo cual se utilizó el conjunto de prueba obtenido del dataset PlantVillage denominado "val", este conjunto de datos contiene imágenes que no se emplearon en la fase de entrenamiento ni validación del modelo. Con el fin de obtener una evaluación objetiva de la capacidad del modelo para generalizar se aplicaron las técnicas de aumento mencionadas en la sección **4.2.1.9** obteniendo un conjunto de datos[9] variado y robusto.

Este proceso consistió en calcular métricas de desempeño claves como la precisión global o *accuracy* la cual mide la proporción de predicciones correctas que el modelo obtuvo. Además, se emplearon métricas como *F1-Score*, *sensivity* y *precision* con el fin de identificar el mejor modelo para la clasificación de imágenes de la mancha bacteriana en la hoja de tomate.

Finalizada la evaluación de cada modelo se obtuvo la matriz de confusión la cual permite visualizar el rendimiento del modelo en términos de aciertos y errores, además de generar los valores respectivos para el cálculo de cada una de las métricas mencionadas con anterioridad. En la **Tabla 16** se detallan los valores generados en la matriz de confusión de cada experimento que se realizó.

[9] Enlace al conjunto de datos generado para test del modelo
https://drive.google.com/drive/folders/1OVFm4VhyWCJgDuivRkgiL9HEKAETscQL?usp=sharing

Tabla 16 Detalle de los valores generados en la matriz de confusión de cada experimento

Experimentos	Valores				Total imágenes test
	VP	VN	FP	FN	
1	425	284	0	34	743
2	425	316	0	2	743
3	425	304	0	14	743
4	424	311	1	7	743
5	424	314	1	4	743

En la **Figura 41** se muestra el código utilizado para obtener los porcentajes de precisión y pérdida del modelo en la fase del test.

```
with torch.no_grad():
    for images, labels in test_loader:
        images, labels = images.to(device), labels.to(device)
        outputs = model(images)
        loss = criterion(outputs, labels)
        running_loss += loss.item() * images.size(0)
        _, predicted = torch.max(outputs.data, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()
        all_labels.extend(labels.cpu().numpy())
        all_predictions.extend(predicted.cpu().numpy())

test_loss = running_loss / total
test_accuracy = correct / total

#print(f'Final Loss: {test_loss:.4f}, Final Accuracy: {test_accuracy:.4f}')
print(f'Precisión en el conjunto de prueba: {test_accuracy* 100:.4f}%\n')
print(f'Pérdida en el conjunto de prueba: {test_loss* 100:.4f}%')
```

Figura 41 Código para obtener la precisión y pérdida del modelo durante el test

En la **Tabla 17** se detallan los resultados obtenidos en el test[10] realizado por cada experimento, mientras que en la **Figura 42** se muestra que el modelo obtuvo un desempeño óptimo en todos los experimentos con exactitudes superiores al 95% y pérdidas relativamente bajas. Sin embargo, se resalta que el experimento 2 resultó el mejor para la clasificación de imágenes de la mancha bacteriana en la hoja de tomate, obteniendo como precisión global de acierto el **99.73%** y **0.78%** de pérdida.

[10] Repositorio test realizados:
https://drive.google.com/drive/folders/114xm_eRLjmwmggWy0W_prXMKIKYRPuyW?usp=sharing

Tabla 17 Valores obtenidos para precisión y pérdida del mejor modelo en cada experimento

	Experimentos				
	1	**2**	**3**	**4**	**5**
Exactitud	95.42%	**99.73%**	98.12%	98.92%	99.33%
Pérdida	13.72%	**0.78%**	5.56%	3.71%	5.5%

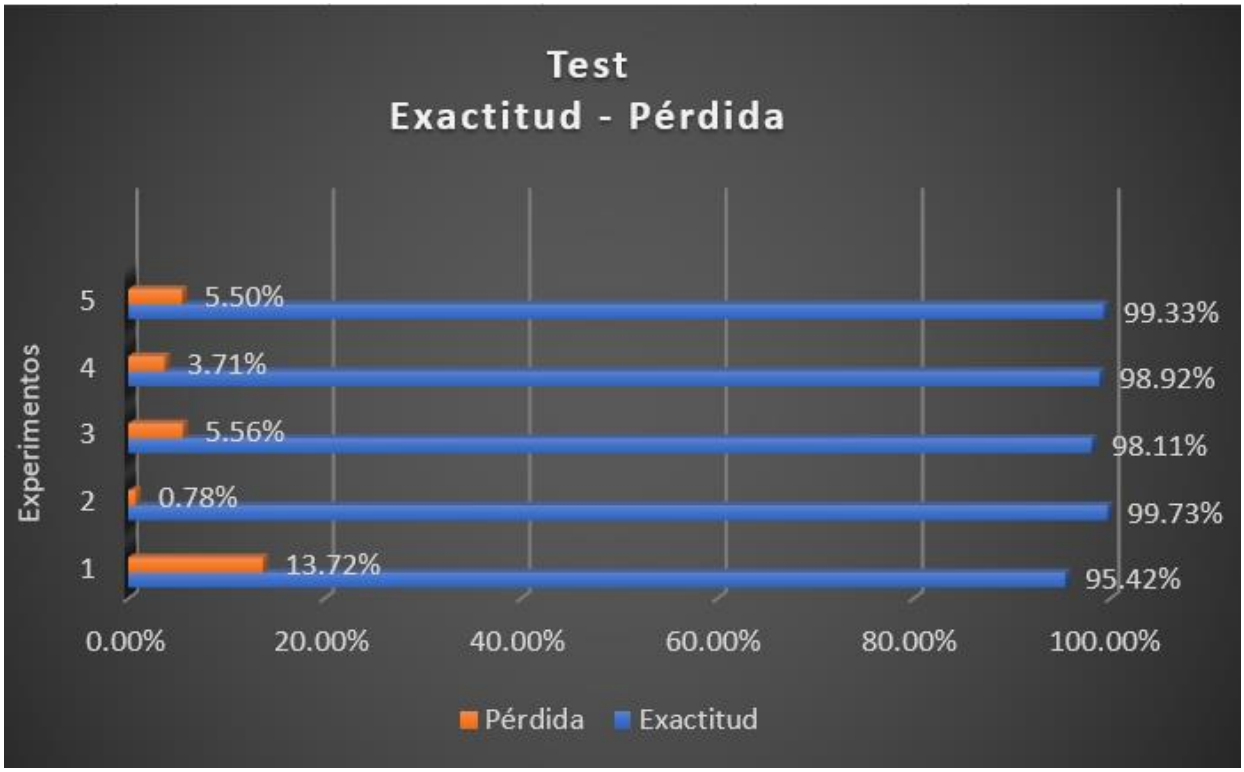

Figura 42 Precisión y pérdida obtenidas en test de los modelos

6.2.4 Tarea 9: Medir la precisión global de acierto durante la clasificación de imágenes de la mancha bacteriana en la hoja de tomate

Para determinar cuál de los modelos generados es el óptimo para la clasificación de imágenes de la mancha bacteriana en la hoja de tomate se calcularon las métricas de rendimiento para las cuales se tomó en cuenta los valores obtenidos en la matriz de confusión (ver **Anexo *2***) generada por cada experimento. En la **Tabla 18** se detalla el procedimiento realizado para obtener los porcentajes correspondientes a la sensibilidad, Puntaje F1, precisión y exactitud de los modelos demostrando que obtuvieron un rendimiento elevado destacando al experimento 2 como el mejor en términos generales.

Tabla 18 Cálculo de métricas a partir de las matrices de confusión obtenidas por experimento

Experimentos	Métricas			
	Sensitivity	**Precision**	**Accuracy**	**F1-Score**
	$\boldsymbol{\frac{VP}{VP+FN}}$	$\boldsymbol{\frac{VP}{VP+FP}}$	$\boldsymbol{\frac{VP+VN}{VP+FP+FN+VN}}$	$\boldsymbol{2*\frac{precision*sensitivity}{precision+sensitivity}}$
1	$=\frac{425}{425+34}$	$=\frac{425}{425+0}$	$=\frac{425+284}{425+0+34+284}$	$=2*\frac{1*0.9259}{1+0.9259}$
	$=0.9259$	$=1$	$=0.9542$	$=0.9926$
	$=\mathbf{92.59}\%$	$=\mathbf{100}\%$	$=\mathbf{95.42}\%$	$=\mathbf{96.15}\%$
2	$=\frac{425}{425+2}$	$=\frac{425}{425+0}$	$=\frac{425+316}{425+0+2+316}$	$=2*\frac{1*0.9953}{1+0.9953}$
	$=0.9953$	$=1$	$=0.9973$	$=0.9976$
	$=\mathbf{99.53}\%$	$=\mathbf{100}\%$	$=\mathbf{99.73}\%$	$=\mathbf{99.76}\%$
3	$=\frac{425}{425+14}$	$=\frac{425}{425+0}$	$=\frac{425+304}{425+0+14+304}$	$=2*\frac{1*0.9681}{1+0.9681}$
	$=0.9681$	$=1$	$=0.9811$	$=0.9838$
	$=\mathbf{96.81}\%$	$=\mathbf{100}\%$	$=\mathbf{98.11}\%$	$=\mathbf{98.38}\%$
4	$=\frac{424}{424+7}$	$=\frac{424}{424+1}$	$=\frac{424+311}{424+1+7+311}$	$=2*\frac{1*0.9836}{1+0.9836}$
	$=0.9836$	$=0.9976$	$=0.9892$	$=0.9917$
	$=\mathbf{98.36}\%$	$=\mathbf{99.76}\%$	$=\mathbf{98.92}\%$	$=\mathbf{99.17}\%$
5	$=\frac{424}{424+4}$	$=\frac{424}{424+1}$	$=\frac{424+314}{424+4+1+314}$	$=2*\frac{0.9976*0.9906}{0.9976+0.9906}$
	$=0.9906$	$=0.9976$	$=0.9933$	$=0.9941$
	$=\mathbf{99.06}\%$	$=\mathbf{99.76}\%$	$=\mathbf{99.33}\%$	$=\mathbf{99.41}\%$

En la **Figura 43** se presenta gráficamente los valores de cada una de las métricas de rendimiento en la cual se visualiza que el modelo del experimento 2 resultó ser el mejor en comparación con los otros experimentos realizados obteniendo el porcentaje más alto en el puntaje F1-Score lo que indica un equilibrio óptimo entre precisión y sensibilidad.

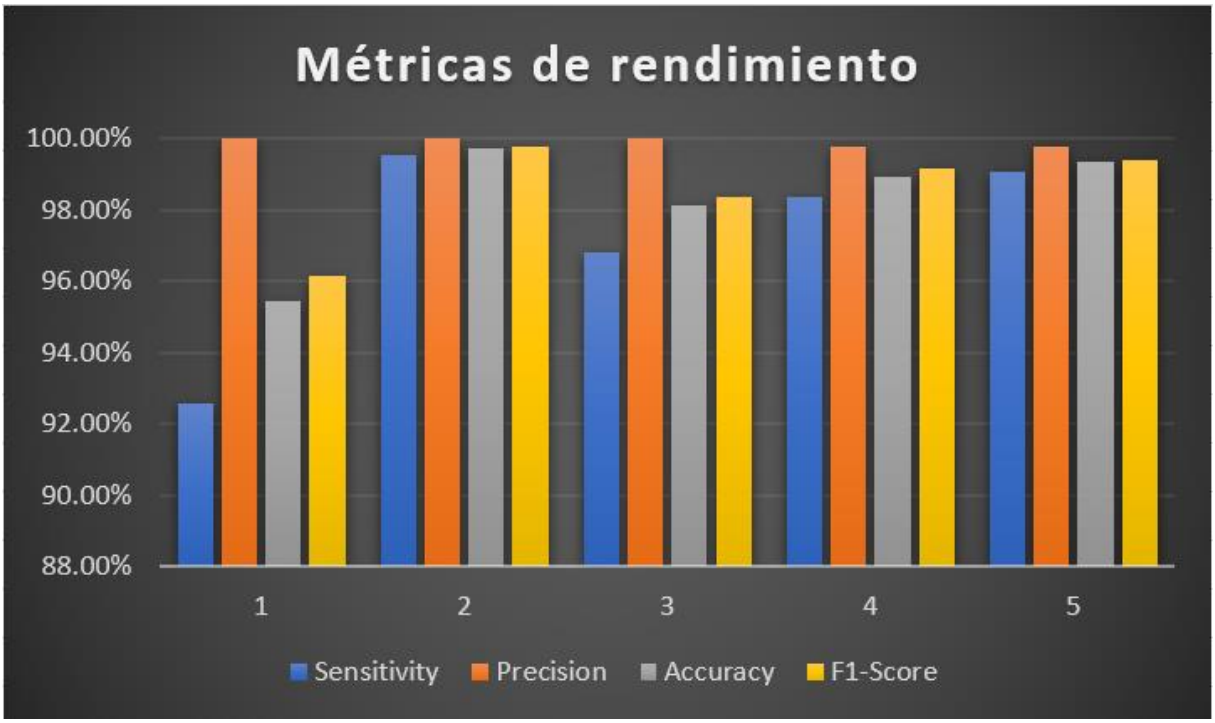

Figura 43 Métricas de rendimiento de los modelos

6.2.5 Tarea 10: Seleccionar modelo final

En base a los resultados presentados en el ***experimento 2*** fue seleccionado como el modelo final para clasificar imágenes de la mancha bacteriana en la hoja de tomate. Este obtuvo un desempeño óptimo en cada una de las métricas evaluadas alcanzando una sensibilidad del 99.53% lo que indica alta capacidad para detectar casos positivos, 99.76% en el puntaje F1-Score y **99.73%** como precisión global de acierto o exactitud lo que refleja una alta proporción de clasificación correctas.

6.2.6 Tarea 11: Aplicación de la técnica Zero-Shot Learning al modelo final

Se aplicó la técnica Zero-Shot Learning al modelo final con el fin de evaluar su capacidad para reconocer y clasificar imágenes de la mancha bacteriana en hojas de tomate con datos obtenidos en un entorno real lo que permitió comprobar la generalización del modelo y su desempeño al identificar adecuadamente las etiquetas reales. En la **Tabla 19** se detalla el

conjunto de datos[11] generado para realizar esta técnica logrando obtener un total de 32 imágenes.

Tabla 19 Cantidad de imágenes obtenidas como conjunto Zero-Shot

Clase	Cantidad de imágenes
Healthy	16
Bacterial Spot	16
	32

En la **Tabla 20** se detalla las especificaciones del recurso utilizado para recolectar las imágenes de un entorno real.

Tabla 20 Recurso utilizado para la recolección de datos (imágenes) para Zero-Shot Learning

	Recurso		
	Apertura de cámara	**Tamaño (Megapíxeles)**	**Procesador móvil**
Cámara de teléfono móvil Xiaomi Poco X4 Pro	f/1.8	48	Mediatek Helio G36

En la **Tabla 21** se presentan los resultados obtenidos al aplicar la técnica Zero-Shot Learning[12] los cuales permitieron determinar que el modelo predice correctamente aquellas imágenes pertenecientes a hojas de tomate afectadas por la mancha bacteriana acertando consistentemente en todas las predicciones lo que refleja un desempeño óptimo en la identificación de esta clase. Sin embargo, al analizar las imágenes etiquetadas como hojas sanas se observa que el modelo predijo incorrectamente esta clase lo que podría estar relacionado con características de las imágenes, siendo el caso del fondo o calidad de las mismas.

[11] Repositorio datos utilizandos para Zero-Shot Learning: https://drive.google.com/drive/folders/1AyET3R4sfFtcWs0jOF9LhOqZ5QjPt7C4?usp=sharing

[12] Código Zero-Shot Learning: https://drive.google.com/file/d/10T4f9hwKEyMV07jBUL9a-zMGZiQsXKjn/view?usp=sharing

Tabla 21 Diagnóstico del modelo en un entorno real

Identificador de la imagen	Imagen	Etiqueta real	Predicción
img_1		Mancha Bacteriana	Mancha Bacteriana
img_2		Mancha Bacteriana	Mancha Bacteriana
img_3		Mancha Bacteriana	Mancha Bacteriana
img_4		Mancha Bacteriana	Mancha Bacteriana

Identificador de la imagen	Imagen	Etiqueta real	Predicción
img_5		Mancha Bacteriana	Mancha Bacteriana
img_6		Mancha Bacteriana	Mancha Bacteriana
img_7		Mancha Bacteriana	Mancha Bacteriana
img_8		Mancha Bacteriana	Mancha Bacteriana
…	…	…	…

Identificador de la imagen	Imagen	Etiqueta real	Predicción
img_25		Hoja Sana	Hoja Sana
img_26		Hoja Sana	Hoja Sana
img_27		**Hoja Sana**	**Mancha Bacteriana**
img_28		**Hoja Sana**	**Mancha Bacteriana**

Identificador de la imagen	Imagen	Etiqueta real	Predicción
img_29		**Hoja Sana**	**Mancha Bacteriana**
img_30		Hoja Sana	Hoja Sana
img_31		Hoja Sana	Hoja Sana
img_32		Hoja Sana	Hoja Sana

En la **Figura 44** se muestra la matriz de confusión generada al evaluar las 32 imágenes del conjunto de datos proporcionado, 16 correspondían a hojas afectadas por la mancha bacteriana de las cuales todas fueran clasificadas correctamente por el modelo. Sin embargo, de las 16 imágenes correspondientes a hojas sanas únicamente 8 fueron clasificadas correctamente.

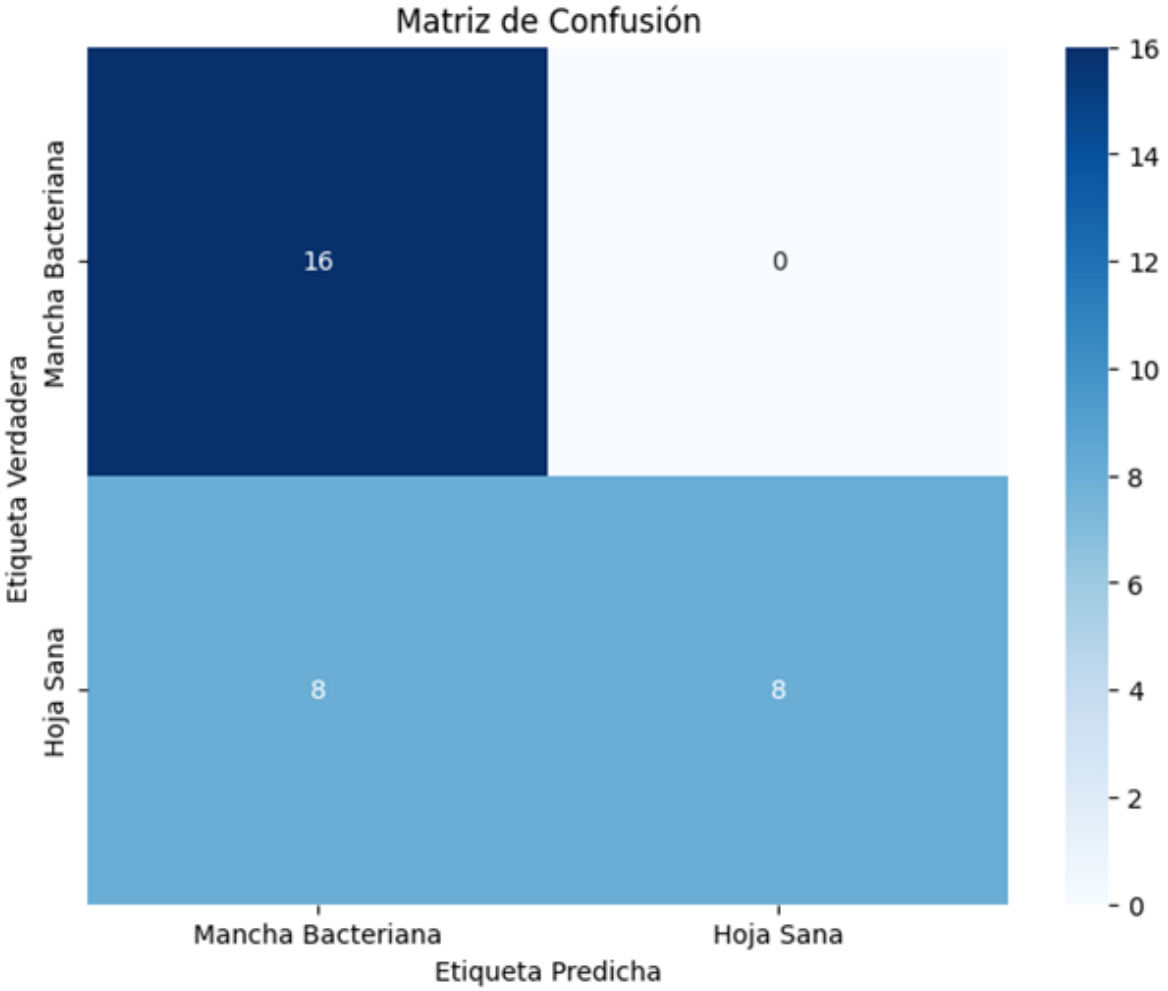

Figura 44 Matriz de confusión Zero-Shot Learning

Para obtener el porcentaje total de clasificaciones correctas se aplicó la **Ecuación 1** obteniendo un **75%** de exactitud lo que demuestra que el modelo es confiable para su uso en la clasificación de la mancha bacteriana en la hoja de tomate, sugiriendo que el uso de este modelo puede ser una herramienta útil en un entorno de producción agrícola. En la **Tabla 22** se detalla el cálculo realizado para obtener la exactitud del modelo al ser evaluado con imágenes de un entorno real.

Tabla 22 Resultados de la fase de prueba en un entorno real

Clase	Total	Predicciones correctas	Predicciones incorrectas	Porcentaje de clasificación correcta
Healthy	16	16	0	$\frac{VP + VN}{VP + FP + FN + VN}$
Bacterial Spot	16	8	8	$\frac{16 + 8}{16 + 0 + 8 + 8}$
Total	32	24	8	$= \mathbf{0.75}$

7. Discusión

Para el desarrollo del trabajo de integración curricular (TIC) se establecieron dos objetivos específicos cuyos resultados permitieron dar alcance al objetivo principal, mismos que se discuten en la siguiente sección:

7.1 Primer objetivo: Ajustar el modelo VGG16 para la clasificación de imágenes de la mancha bacteriana en la hoja de tomate de riñón

El ajuste del modelo VGG16 para la clasificación de imágenes de la mancha bacteriana en la hoja de tomate se realizó adaptando fases de la metodología CRISP-ML(Q) misma que demostró ser eficaz y flexible, el desarrollo de las tareas establecidas garantizó la calidad del conjunto de datos, robustez del modelo y rendimiento óptimo durante la fase de entrenamiento.

La selección del conjunto de datos *PlantVillage* reconocido en investigaciones como [65], [67], [68], proporcionó una base confiable para el desarrollo del modelo. Sin embargo, debido a la variabilidad en la calidad de las imágenes fue necesario aplicar técnicas de limpieza y balanceo lo que resultó en el descarte de 17 imágenes de baja calidad generando un conjunto de datos de 2958 imágenes equilibradas entre cada clase. Según lo que se menciona en investigaciones como [7], [62], [69], [72] se aplicó técnicas de aumento de datos con la finalidad de obtener un conjunto de datos variado (**Tabla 10**), lo que incrementó el tamaño a 3816 imágenes mejorando así la capacidad del modelo para generalizar con datos no vistos.

La optimización del modelo se realizó con cinco configuraciones experimentales de hiperparámetros explorando ajustes de tasa de aprendizaje, betas, regularización de peso y dropout. Estas modificaciones permitieron prevenir el sobreajuste y asegurar una convergencia estable del modelo cuyos resultados reflejan precisiones de 100% en varios casos, lo que evidencia la efectividad de los ajustes realizados. Además, la implementación de un scheduler de tasa de aprendizaje y el uso de optimizadores avanzados como Adam y AdamW permitió adaptar dinámicamente las actualizaciones del modelo.

Durante el entrenamiento de los modelos se monitorizaron la pérdida y precisión de cada iteración demostrando en los experimentos más exitosos que la pérdida se redujo a valores mínimos de 0.0003 y alcanzó una precisión del 100% en las últimas épocas. Por lo que, la implementación de estrategias como congelar capas específicas y la optimización de parámetros como el tamaño de lote contribuyeron significativamente estos resultados, demostrando la eficacia del ajuste fino de modelos preentrenados en problemas específicos.

Finalmente, los resultados obtenidos reflejan el alcance adquirido del presente objetivo logrando obtener precisión alta y pérdida mínima, lo que establece una base sólida para el desarrollo de la evaluación de los modelos.

7.2 Segundo objetivo: Evaluar el modelo entrenado a través de la precisión global de acierto al clasificar imágenes de la mancha bacteriana en hojas de tomate de riñón.

Durante las pruebas realizadas (ver **Tabla 14**) el modelo presentó una curva de aprendizaje óptima reduciendo progresivamente la pérdida a valores cercanos a cero y alcanzado una precisión estable desde las primeras épocas, lo que refleja una alta capacidad de aprendizaje y adaptación a las características del conjunto de datos respaldado por el balance y calidad de las imágenes utilizadas. El proceso de evaluación del modelo desarrollado para clasificar imágenes de la mancha bacteriana en hojas de tomate de riñón permitió verificar la eficacia del enfoque propuesto al alcanzar valores de precisión y desempeño superiores al 99% tanto en las fases de validación (ver **Tabla 15**) como en prueba (ver **Tabla 17**), destacando la solidez del modelo obtenido en el experimento 2 el cual logró alcanzar una precisión global del 99.73% y una pérdida mínima de 0.78%.

Los resultados obtenidos son comparables con estudios previos como [7], [69], [70], [71] los cuales utilizaron arquitecturas profundas específicamente VGG16 para tareas similares de clasificación de enfermedades en plantas de tomate, estos presentaron un desempeño óptimo con porcentaje de precisión entre el 90% y 96%. Sin embargo, la presente investigación supera los valores mencionados al incorporar un esquema de optimización de hiperparámetros y un proceso robusto de aumento de datos, lo que permitió mejorar la generalización del modelo logrando adquirir mayor precisión al clasificar las imágenes.

El uso de técnicas como Zero-Shot Learning (ver sección **6.2.6**) permitió analizar el desempeño del modelo con datos de entornos reales presentando una exactitud del 75% al clasificar imágenes de hojas de tomate, lo que sugiere que el modelo es confiable para identificar la mancha bacteriana ya que clasificó correctamente el 100% de las imágenes ingresadas. Sin embargo, su desempeño disminuyó al clasificar imágenes de hojas sanas lo que podría ser causado por factores externos como la calidad y fondo de las imágenes.

Finalmente, la implementación de arquitecturas profundas junto a estrategias avanzadas de preprocesamiento y evaluación demostraron ser efectivas sentando las bases para futuras investigaciones que busquen mejorar la robustez del modelo. Los resultados obtenidos permitieron determinar que el modelo desarrollado logra clasificar con alta precisión las imágenes de hojas afectadas por la mancha bacteriana demostrando su utilidad en entornos agrícolas.

8. Conclusiones

A partir de los resultados obtenidos durante el desarrollo del trabajo de integración curricular se concluye lo siguiente:

- La implementación del modelo preentrenado VGG16 ajustado para la clasificación de imágenes de la mancha bacteriana en hojas de tomate de riñón permitió obtener un porcentaje de precisión global del 99.73% con una pérdida mínima del 0.78% durante la fase de validación, estos resultados evidencian que la arquitectura VGG16 al ser optimizada con técnicas de ajuste fino y adaptaciones específicas para la tarea en cuestión es altamente efectiva en la clasificación de imágenes superando estudios previos con configuraciones similares. Además, la capacidad del modelo para reconocer patrones visuales distintivos en las imágenes afectadas demuestra la viabilidad del uso de redes neuronales convolucionales en el monitoreo de enfermedades en entornos agrícolas.
- El ajuste fino del modelo VGG16 se basó en la metodología CRISP-ML(Q) asegurando un enfoque sistemático en la preparación y optimización del modelo, se desarrollaron distintas estrategias de preprocesamiento de datos como la limpieza de imágenes con resolución deficiente, balanceo del conjunto de datos para evitar sesgos en la clasificación y aplicación de técnicas de aumento de datos como ajuste de brillo y contraste, rotación y volteo, esto con el objetivo de incrementar la variabilidad del conjunto de entrenamiento. Además, el modelo fue ajustado mediante la modificación de hiperparámetros clave como la tasa de aprendizaje, optimizador de gradientes y el uso de técnicas de regularización como el decaimiento de peso permitiendo reducir el sobreajuste y mejorar la generalización del modelo.
- La evaluación del modelo entrenado se realizó mediante métricas de rendimiento como la precisión, pérdida y matriz de confusión obteniendo valores que respaldan su eficiencia en la clasificación de imágenes de la mancha bacteriana en hojas de tomate. Durante la fase de validación el modelo alcanzó una precisión del 99.73% lo que indica su capacidad para identificar correctamente la presencia de la enfermedad en un entorno de pruebas controlado, sin embargo, al aplicar la técnica Zero-Shot Learning en la evaluación del modelo con imágenes provenientes de entornos reales la precisión disminuyó al 75% evidenciando la influencia de factores como iluminación, ángulos de captura, calidad de la imagen y ruido en los datos afectan la capacidad de generalización del modelo.

9. Recomendaciones

Teniendo en cuenta los resultados obtenidos en este trabajo se proponen las siguientes recomendaciones:

- Mejorar la capacidad de generalización del modelo en entornos reales ampliando y diversificando el conjunto de datos mediante la incorporación de imágenes capturadas en diferentes condiciones ambientales, tipos de iluminación y con diversas cámaras. A su vez, se sugiere implementar técnicas de segmentación y preprocesamiento avanzadas para destacar las características relevantes de las hojas y mitigar problemas causados por fondos irrelevantes.
- Es esencial realizar pruebas extensivas en entornos reales para evaluar el desempeño del modelo bajo condiciones agrícolas auténticas ya que este proceso permitirá validar su aplicabilidad, detectar limitaciones adicionales y realizar ajustes necesarios para garantizar su efectividad en la práctica.
- Analizar la posibilidad de integrar modelos de clasificación de enfermedades con tecnologías agrícolas como drones, dispositivos móviles y estaciones de monitoreo lo que facilitará su implementación directa en los campos de cultivo.
- Realizar estudios para evaluar el impacto del uso de estos modelos en la productividad agrícola enfocándose en cómo contribuyen a mejorar tanto la productividad como la calidad de los cultivos.

10. Bibliografía

[1] P. Adhikari, T. B. Adhikari, F. Louws and D. R. Panthee, "Advances and Challenges in Bacterial Spot Resistance Breeding in Tomato (Solanum lycopersicum L)," *Internacional Journal of Molecular Sciences,* p. 1734, 2020.

[2] F. López Saca, "Clasificación de imágenes usando redes neuronales convolucionales," Universidad Autónoma Metropolitana, México, 2019.

[3] H. Yang, J. Ni, J. Gao, Z. Han and T. Luan, "A novel method for peanut variety identification and classification by Improved VGG16," *Scientific Reports,* vol. 11, no. 1, p. 15756, 2021.

[4] A. Hernández M, "Sistema de visión artificial para la detección de plantas enfermas mediante aprendizaje profundo," Tecnológico Nacional de México, México, 2022.

[5] A. Pombo Sudré da Silva, F. Olivares Lopes and C. Sudré Pombo, "Attenuations of bacterial spot disease Xanthomonas euvesicatoria on tomato plants treated with biostimulants," *Chemical and Biological Technologies in Agriculture,* pp. 1-9, 2021.

[6] S. U. Habiba and M. K. Islam, "Tomato Plant Diseases Classification Using Deep Learning Based Classifier From Leaves Images," in *2021 International Conference on Information and Communication Technology for Sustainable Development (ICICT4SD)*, 2021.

[7] L. P. Lin and J. S. Leong, "Detection and Categorization of Tomato Leaf Diseases Using Deep Learning," *International Journal of Application on Sciences, Technology and Engineering,* vol. 1, no. 1, pp. 282-291, 2023.

[8] O. Hengxuan Chi, G. Denton and D. Gursoy, "Artificially intelligent device use in service delivery: A systematic review, synthesis, and research agenda," *Journal of Hospitality Marketing & Management,* vol. 29, no. 7, pp. 757-786, 2020.

[9] Y. Jiang, X. Li, H. Luo, S. Yin and O. Kaynak, "Quo vadis artificial intelligence?," *Discover Artificial Intelligence,* vol. 2, no. 1, p. 4, 2022.

[10] I. Ahmed, G. Jeon and F. Piccialli, "From artificial intelligence to explainable artificial intelligence in industry 4.0: A survey on what, how, and where," *IEEE Transactions on Industrial Informatics,* vol. 18, no. 8, pp. 5031-5042, 2022.

[11] G. L. Hart, T. Mueller, C. Toher and S. Curtarolo, "Machine learning for alloys," *Nature Reviews Materials,* vol. 6, no. 8, pp. 730-755, 2021.

[12] W. Jing, C. Xuan, S. Xiang-Yu, D. Hui-Xion and C. Jigen, "Machine learning in materials science," *InfoMat,* vol. 1, no. 3, pp. 338-358, 2019.

[13] C. Janiesch, P. Zschech and K. Heinrich, "Machine learning and deep learning," *Electronic Markets,* vol. 31, no. 3, pp. 685-695, 2021.

[14] M. Soori, B. Arezoo and R. Dastres, "Machine learning and artificial intelligence in CNC machine tools, a review," *Sustainable Manufacturing and Service Economics,* vol. 2, p. 100009, 2023.

[15] E. K. Hee, A. Linan, N. Santhanam, M. Jannesari and T. Ganslandt , "Transfer learning for medical image classification: a literature review," *BMC medical imaging,* vol. 22, no. 1, p. 69, 2022.

[16] I. Mohammadreza, H. Arabnia and K. Rasheed, "A review of deep transfer learning and recent advancements," *Technologies,* vol. 11, no. 2, p. 40, 2023.

[17] Z. Zhu, K. Lin, A. K. Jain and J. Zhou, "Transfer learning in deep reinforcement learning: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 45, no. 11, pp. 13344-13362, 2023.

[18] A. Mathew, P. Amudha and S. Sivakumari, "Deep Learning Techniques: An Overview," in *Advanced Machine Learning Technologies and Applications: Proceedings of AMLTA 2020*, Singapore, 2021.

[19] I. Pérez Borrero and M. E. Gegúndez Arias, Deep learning : fundamentos, teoría y aplicación, Universidad de Huelva, 2021, p. 1–261.

[20] S. X. Jun Wang, W. S. Tuan Ngo, A. Manan Sadick and X. Wang, "Computer vision techniques in construction: A critical review," *Archives of Computational Methods in Engineering,* vol. 28, pp. 3383-3397, 2021.

[21] J. Jaramillo, "Detección del Tizón Foliar en las hojas del cultivo de maíz (Zea Mays L.) mediante un modelo de visión por computador," 2024.

[22] D. Ameijeiras Sánchez, H. R. González Diaz and Y. Hernández Heredia, "Revisión de algoritmos de detección y seguimiento de objetos con redes profundas para videovigilancia inteligente," *Revista Cubana de Ciencias Informáticas,* pp. 165-195, 2020.

[23] N. Ketkar and J. Moolayil, "Convolutional neural networks," *Deep learning with Python: learn best practices of deep learning models with PyTorch*, pp. 197-242, 2021.

[24] Á. Artola Moreno, "Clasificación de imágenes usando redes neuronales convolucionales en Python," 2019.

[25] C. Bonilla Carrión, "Redes convolucionales," 2020.

[26] G. Prerepa, D. Aiswarya, I. Sufyan, P. Rahul and G. Reena, "Exploring the Potential of VGG-16 Architecture for Accurate Brain Tumor DetectionUsing Deep Learning," *Journal of Computers, Mechanical and Management*, vol. 2, no. 2, 2023.

[27] H. Qassim, A. Verma and D. Feinzimer, "Compressed residual-VGG16 CNN model for big data places image recognition," in *2018 IEEE 8th annual computing and communication workshop and conference (CCWC)*, 2018.

[28] A. Tomás Martín, "Investigación de detección de bordes a radiografías con deep learning," E.T.S. de Ingenieros Informáticos (UPM), 2023.

[29] L. González, "Matriz de confusión - Aprende ia," 2019.

[30] R. Borja Robalino, A. Monleón Getino and J. Rodellar, "Estandarización de métricas de rendimiento para clasificadores Machine y Deep Learning," *Revista Ibérica de Sistemas y Tecnologias de Información*, no. 30, pp. 184-196, 2020.

[31] K. Y. Jiménez Cueva, "Determinación de la madurez de frutos del café mediante el reconocimiento de imágenes utilizando un modelo basado en redes neuronales," 2023.

[32] F. Mohameth, C. Bingcai and K. Amath Sada, "Plant disease detection with deep learning and feature extraction using plant village," *Journal of Computer and Communications*, vol. 8, no. 6, pp. 10-22, 2020.

[33] Kaggle, "PlantVillage," 2021. [Online].

[34] J. P. Rodríguez Villamar, "Detección de ataques de tipo Fuzzer en redes IoT empleando algoritmos de clasificación de Machine Learning," Universidad de los Andes, 2023.

[35] P. Chlap, H. Min, L. Holloway and A. Haworth, "A review of medical image data augmentation techniques for deep learning applications," *Journal of Medical Imaging and Radiation Oncology*, vol. 65, no. 5, pp. 545-563, 2021.

[36] A. Mumuni and F. Mumuni, "Data augmentation: A comprehensive survey of modern approaches," *Array*, vol. 16, pp. 100-258, 2022.

[37] K. Alomar and X. Cai, "Data augmentation in classification and segmentation: A survey and new strategies," *Journal of Imaging,* vol. 9, no. 2, p. 46, 2023.

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

[39] X. Wang, K. Wang and S. Lian, "A survey on face data augmentation for the training of deep neural networks," *Neural computing and applications,* vol. 32, no. 29, pp. 15503-15531, 2020.

[40] S. Bock and M. Weib, "A proof of local convergence for the Adam optimizer," in *2019 International Joint Conference on Neural Networks (IJCNN)*, 2019.

[41] A. Hamza Khan, X. Cao, S. Li, V. N. Katsikis and L. Liao, "BAS-ADAM: an ADAM based approach to improve the performance of beetle antennae search optimizer," *IEEE/CAA Journal of Automatica Sinica,* vol. 7, no. 2, pp. 461-471, 2020.

[42] P. Corral Saiz, "Deep learning aplicado a espectroscopia láser," 2022.

[43] R. Susmita, "A quick review of machine learning algorithms," in *2019 International conference on machine learning, big data, cloud and parallel computing (COMITCon)*, 2019.

[44] Z. Xie, I. Sato and M. Sugiyama, "Stable weight decay regularization," in *International Conference on Learning Representations*, Vienna, 2020.

[45] X. Li, S. Chen and J. Yang, "Understanding the disharmony between weight normalization family and weight decay," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

[46] J. M. Sierra Ramos, "Introducción a las redes neuronales artificiales," Universidad Complutense Madrid, 2022.

[47] ml-ops.org, "MLOps," 2024. [Online].

[48] R. Alegre Veliz, P. Gaspar Ortiz, J. Gamboa Cruzado, L. Rodríguez Baca, R. Menéndez Mueras, W. Grandez Pizarro and C. Chávez Herrera, "Machine learning for feeling analysis in twitter communications: A case study in HEYDRU," *International Journal of Interactive Mobile Technologies (iJIM),* vol. 16, no. 24, pp. 126-142, 2022.

[49] J. B. Magdaong, A. B. Culaba, A. T. Ubando and N. S. Lopez, "Generating synthetic building electrical load profiles using machine learning based on the CRISP-ML (Q) framework," in *IOP Conference Series: Earth and Environmental Science*, 2024.

[50] S. Studer, T. Binh Bui, C. Drescher, A. Hanuschkin, S. Peters and K. R. Muller, "Towards CRISP-ML (Q): a machine learning process model with quality assurance methodology," *Machine learning and knowledge extraction*, vol. 3, no. 2, pp. 392-413, 2021.

[51] M. A. Salazar Franco, "Plataforma para desarrollo y evaluaci{\'o}n de modelos de reconocimiento de gestos de la mano: modelo de reconocimiento de 11 gestos de la mano usando deep learning y celdas de memoria desarrollado en Python.," Escuela Politécnica Nacional, 2023.

[52] V. Sharma, K. G. Gupta and M. Gupta, "Performance benchmarking of GPU and TPU on google colaboratory for convolutional neural network," in *Applications of Artificial Intelligence in Engineering: Proceedings of First Global Conference on Artificial Intelligence and Applications (GCAIA 2020)*, Springer, Singapore, 2021.

[53] E. Bodero, M. Lopez, A. Congacha, E. Cajamarca and C. Morales, "Google Colaboratory como alternativa para el procesamiento de una red neuronal convolucional," *Revista Espacios*, vol. 41, no. 7, 2020.

[54] D. A. Blubaugh, S. D. Harbour, S. D. Sears and B. Findler, "OpenCV and Perception," in *Intelligent Autonomous Drones with Cognitive Deep Learning*, 2022, pp. 327-361.

[55] A. Sharma, J. Pathak, M. Prakash and J. N. Singh, "Object Detection using OpenCV and Python," in *2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*, 2021.

[56] T. Domínguez Mínguez, Visión artificial: Aplicaciones prácticas con OpenCV-Python, Marcombo, 2021.

[57] A. K. Mohaideen , "Image Processing Using OpenCV," in *Industrial Vision Systems with Raspberry Pi*, 2024, pp. 87-140.

[58] T. S. Jalolov, "Artificial intelligence python (PYTORCH)," *Oriental Journal of Academic and Multidisciplinary Research*, vol. 1, no. 3, pp. 123-126, 2023.

[59] N. Ketkar and J. Moolayil, "Introduction to PyTorch," in *Deep Learning with Python*, 2021, pp. 27-91.

[60] S. Imambi, K. Prakash and G. R. Kanagachidambaresan, "PyTorch," in *Programming with TensorFlow*, EAI/Springer Innovations in Communication and Computing, 2021, pp. 87-104.

[61] S. Delgado Quintero, "Aprende python," 2023.

[62] A. M. Hernández, *Sistema de visión artificial para la detección de plantas enfermas mediante aprendizaje profundo,* México: Tecnológico Nacional de México, 2022.

[63] T. H. Nguyen, T. N. Nguyen and B. V. Ngo, "A VGG-19 model with transfer learning and image segmentation for classification of tomato leaf disease," *AgriEngineering,* vol. 4, no. 4, pp. 871-887, 2022.

[64] A. Natarajan, M. Babitha and S. Kumar, "Detection of disease in tomato plant using Deep Learning Techniques," *International Journal of Modern Agriculture,* vol. 9, no. 4, pp. 525-540, 2020.

[65] V. Maeda Gutierrez, C. Guerrero Méndez, C. A. Olvera Olvera and G. Espinoza García, "Redes neuronales convolucionales para la detección y clasificación de enfermedades de plantas basadas en imágenes digitales," Repositorio Institucional Caxcán , 2018.

[66] V. Maeda Gutierrez, "Comparación de arquitecturas de redes neuronales convolucionales para la clasificación de enfermedades en tomate," Repositorio Institucional Caxcán, 2019.

[67] E. Huerta Mora, V. González Huitrón, H. Rodriguez Rangel and L. E. Amabilis Sosa, "Detección de enfermedades foliares con arquitecturas de redes neuronales convolucionales," *RINDERESU,* vol. 5, no. 1, 2021.

[68] A. Alatawi, A. Shahd Maadi, N. I. Alhawiti and A. Muhammad, "Plant disease detection using AI based VGG-16 model," *International Journal of Advanced Computer Science and Applications,* vol. 13, no. 4, 2022.

[69] M. Agarwal and S. Gupta, "Development of Efficient CNN model for Tomato crop disease identification," *Sustainable Computing: Informatics and Systems,* vol. 28, p. 100407, 2020.

[70] V. P. Gaikwad and V. Musande, "Plant leaf damage detection using convolution neural network models," in *AIP Conference Proceedings*, 2023.

[71] P. Borugadda, L. Ramasami and S. Satyasa, "Transfer Learning VGG16 Model for Classification of Tomato Plant Leaf Diseases: A Novel Approach for Multi-Level Dimensional Reduction," *Pertanika Journal of Science & Technology,* vol. 31, no. 2, pp. 813 - 841, 2023.

[72] S. Wagle, "A Deep Learning-Based Approach in Classification and Validation of Tomato Leaf Disease," *Traitement du signal,* vol. 38, no. 3, 2021.

[73] A. Meraz Hernández, "Sistema de visión artificial para la detección de plantas enfermas mediante aprendizaje profundo," Tecnológico Nacional de México, 2022.

[74] P. Adhikari, T. B. Adhikari, F. Louws and D. R. Panthee, "Advances and Challenges in Bacterial Spot Resistance Breeding in Tomato (Solanum lycopersicum L.)," *International Journal of Molecular Sciences,* vol. 21, no. 5, p. 1734, 2020.

[75] S. Somashetty, "A COMPREHENSIVE REVIEW ON RECENT ADVANCES IN THE RESEARCH ON BACTERIAL LEAF SPOT IN TOMATO CAUSED BY XANTHOMONAS SPECIES," *Plant Archives,* vol. 23, no. 2, 2023.

[76] W. Jing, C. Xuan, S. Xiang-Yu, D. Hui-Xion and C. Jigen, "Machine learning in materials science," *InfoMat,* pp. 338-358, 2019.

[77] A. Mathew, P. Amudha and S. Sivakumari, "Deep learning techniques: an overview," in *Advanced Machine Learning Technologies and Applications: Proceedings of AMLTA 2020*, Singapore, 2021.

[78] I. Pérez Borrero and M. E. Gegúndez Arias, Deep learning: Fundamentos, teoría y aplicación, Universidad de Huelva, 2021.

[79] M. D. Montalvo Cusi, "Comparación de técnicas de balanceo de datos para la clasificación de fraude en transacciones bancarias," Universidad Nacional Mayor de San Marcos, 2023.

[80] N. Sharma, R. Sharma and N. Jindal, "Machine learning and deep learning applications a vision," *Global Transitions Proceedings,* vol. 2, no. 1, pp. 24-28, 2021.

[81] D. Ameijeiras Sánchez, H. R. González Diez and Y. Hernández Heredia, "Revisión de algoritmos de detección y seguimiento de objetos con redes profundas para videovigilancia inteligente," *Revista Cubana de Ciencias Informáticas,* vol. 14, no. 3, pp. 165-195, 2020.

[82] I. Katsushi, Computer vision: A reference guide, Springer, 2021.

[83] A. P. S. da Silva, F. L. Olivares and C. P. Sudré, "Attenuations of bacterial spot disease Xanthomonas euvesicatoria on tomato plants treated with biostimulants," *Chemical and Biological Technologies in Agriculture,* vol. 8, no. 42, pp. 1-9, 2021.

[84] L. Huang, W. Pan, Y. Zhang and L. Qian, "Data augmentation for deep learning-based radio modulation classification," *IEEE,* vol. 8, pp. 1948-1506, 2019.

11. Anexos

Anexo 1 Entrevista experto

Datos personales:

- Nombre: Jennifer Elizabeth Quizhpe Hurtado
- Cargo: Estudiante de la Carrera de computación
- E-mail: jennifer.quizhpe@unl.edu.ec
- Teléfono: 0995913572

Información del entrevistado:

- Nombre: Ing. Oscar Cumbicus, Mg. Sc
- Cargo: Docente carrera Computación de la Universidad Nacional de Loja"

Muestra: Docente

Fecha: 01/02/2024

Duración: 40 minutos

Objetivos de la entrevista:

- Obtener información detallada acerca de la situación actual de los sistemas inteligentes para la clasificación de imágenes.
- Conocer acerca de trabajos que se hayan realizados utilizando modelos de clasificación de imágenes.

PREGUNTAS:

1. ¿Cuál es la importancia de la clasificación de imágenes en la actualidad?

Una de las cosas más importantes es la ayuda en diferentes áreas, por ejemplo, en medicina permite detectar enfermedades, en la agricultura ayuda a clasificar y detectar enfermedades dentro de cultivos. Es decir, su importancia radica en la automatización de tareas que comúnmente se pueden hacer de forma natural.

2. ¿Con que modelos usted ha trabajado para clasificación de imágenes?

Existen algunos modelos, en su caso ha utilizado VGG16 para clasificar imágenes de COVID, para clasificar imágenes de enfermedades implementó ResNet como modelo principal, finalmente trabajó para clasificación de objetos con Inception, el cual tiene algunas versiones del modelo.

3. **¿Cuáles son los principales desafíos técnicos que enfrenta el desarrollo de modelos para la clasificación de imágenes?**

Uno de los principales desafíos técnicos radica en la disponibilidad y particularidad de los conjuntos de datos, por lo que contar con datasets que permitan realizar tareas específicas es un gran inconveniente, disminuyendo la posibilidad de crear de manera rápida modelos adaptados a estas tareas. Actualmente existen numerosos datasets para problemas generales, tales como COCO y MNIST, esto representa un obstáculo significativo para la resolución de problemas específicos. Aunque existen modelos que realizan tareas generales como detectar la presencia de un gato, estos no tienen la capacidad de identificar aspectos específicos como la especie del gato.

4. **¿Cómo se evalúa la precisión y el rendimiento de estos modelos de clasificación de imágenes?**

La forma más común de evaluación es mediante la precisión o pérdida, esto implica comparar la predicción del modelo con lo que debía haber predicho, evaluando cuanto se equivoca en relación a la verdad conocida. Otra métrica es el porcentaje global de aciertos, esto permite conocer cuantas muestras fueron clasificadas correctamente. Utilizando la matriz de confusión a través de sus métricas, se puede obtener una evaluación más detallada sobre las predicciones realizadas por el modelo.

5. **¿Cómo se abordan las preocupaciones relacionadas con la precisión y la confiabilidad de los resultados en modelos de clasificación de imágenes?**

El análisis de datos no se limita únicamente a tener un porcentaje, sino implica el conocer el motivo que existe detrás de ese resultado. El incremento de datos o uso de muestras sintéticas, en su mayoría son implementadas con el fin de incrementar el valor de precisión y sensibilidad de los modelos, además a través de técnicas de Fine Tuning y uso de transfer learning estos problemas pueden ser abordados.

6. **Conoce usted si, ¿se han desarrollado modelos específicos para detección de enfermedades en cultivos?**

Sí, dentro de la UNL se han desarrollado dos modelos de visión por computadora para la clasificación de imágenes. Uno para la detección de la mancha negra en la hoja de maíz y otro para la clasificación de muestras de café, sean café maduro, semi-maduros y prodridos.

7. **En caso de conocer modelos específicos, ¿Con qué arquitecturas fueron desarrollado los modelos?**

Se han implementado redes neuronales convolucionales con modelos pre-entrenados como VGG16 y ResNet, además se han utilizado redes transformers.

8. ¿Cuáles son las tendencias futuras que se anticipan en la mejora de clasificación de imágenes?

En la actualidad hay una arquitectura llamada “Transformer”, es una de las tendencias de clasificación que se encuentra en investigación con el fin de alcanzar mejores resultados respecto a la CNNs.

Oscar M. Cumbicus-Pineda, Mg. Sc.

EXPERTO ENTREVISTADO

Anexo 2 Evaluación de desempeño de los modelos generados: Matriz de confusión

Experimento 1

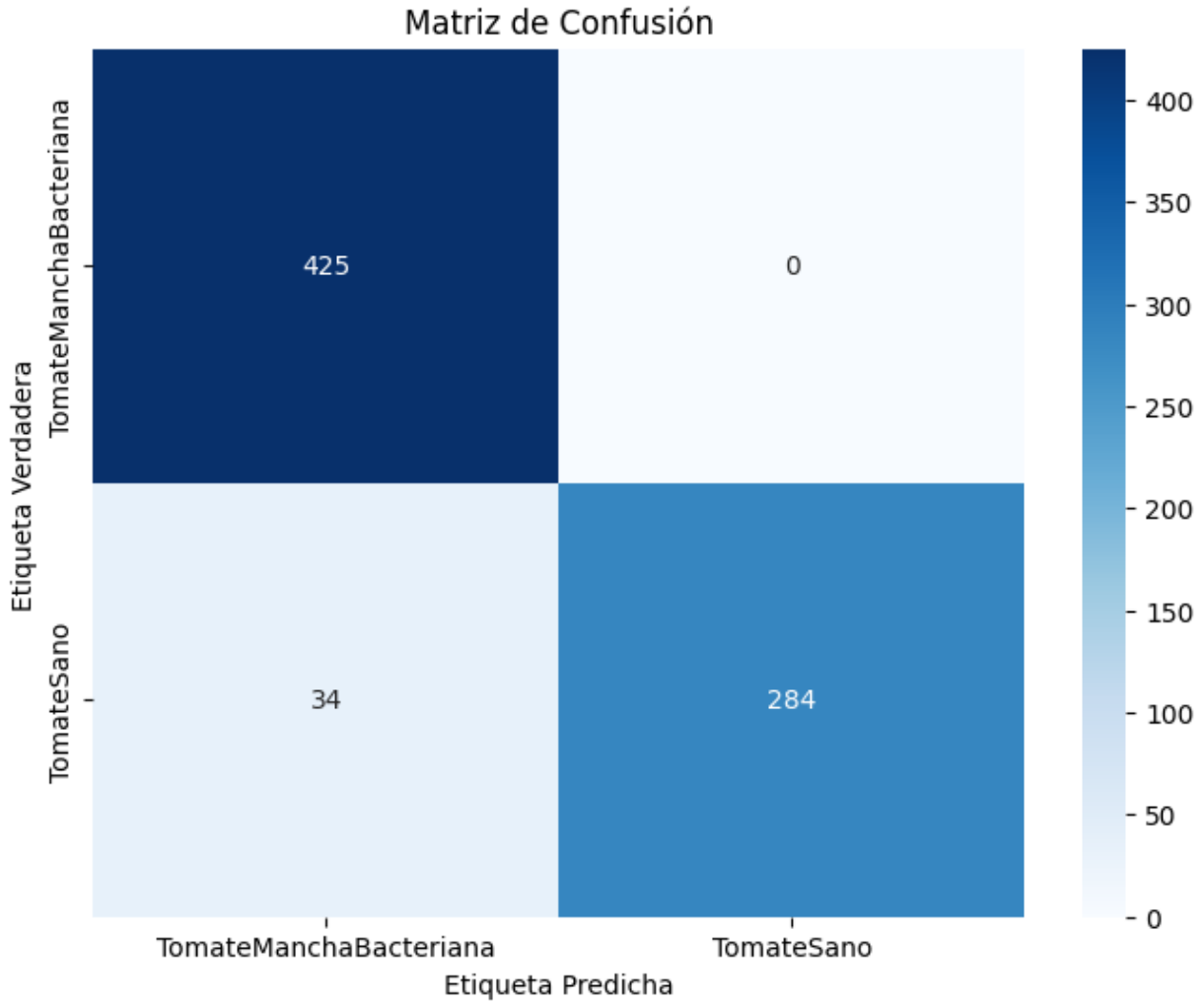

Figura 45 Matriz de confusión del experimento 1

Experimento 2

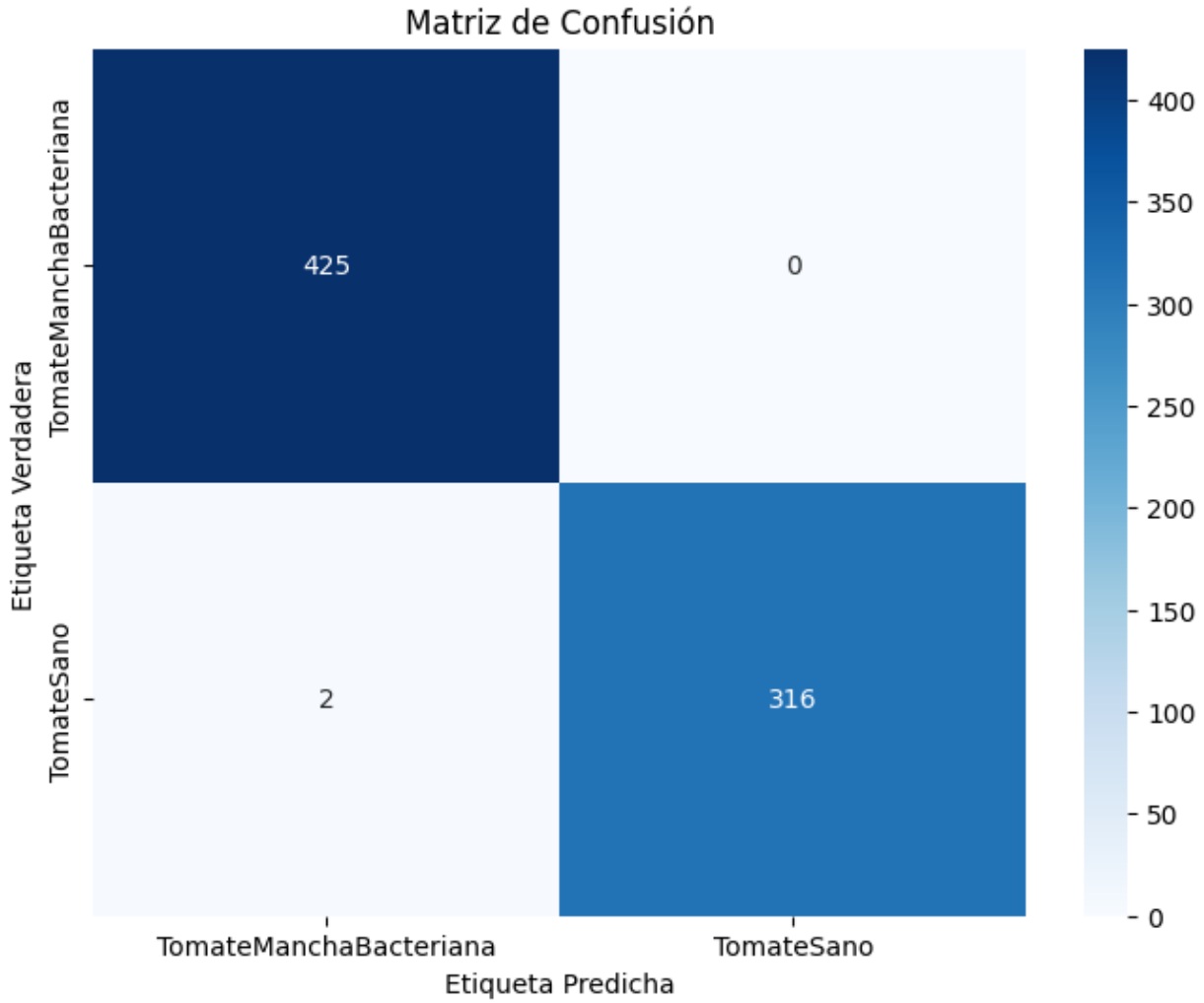

Figura 46 Matriz de confusión del experimento 2

Experimento 3

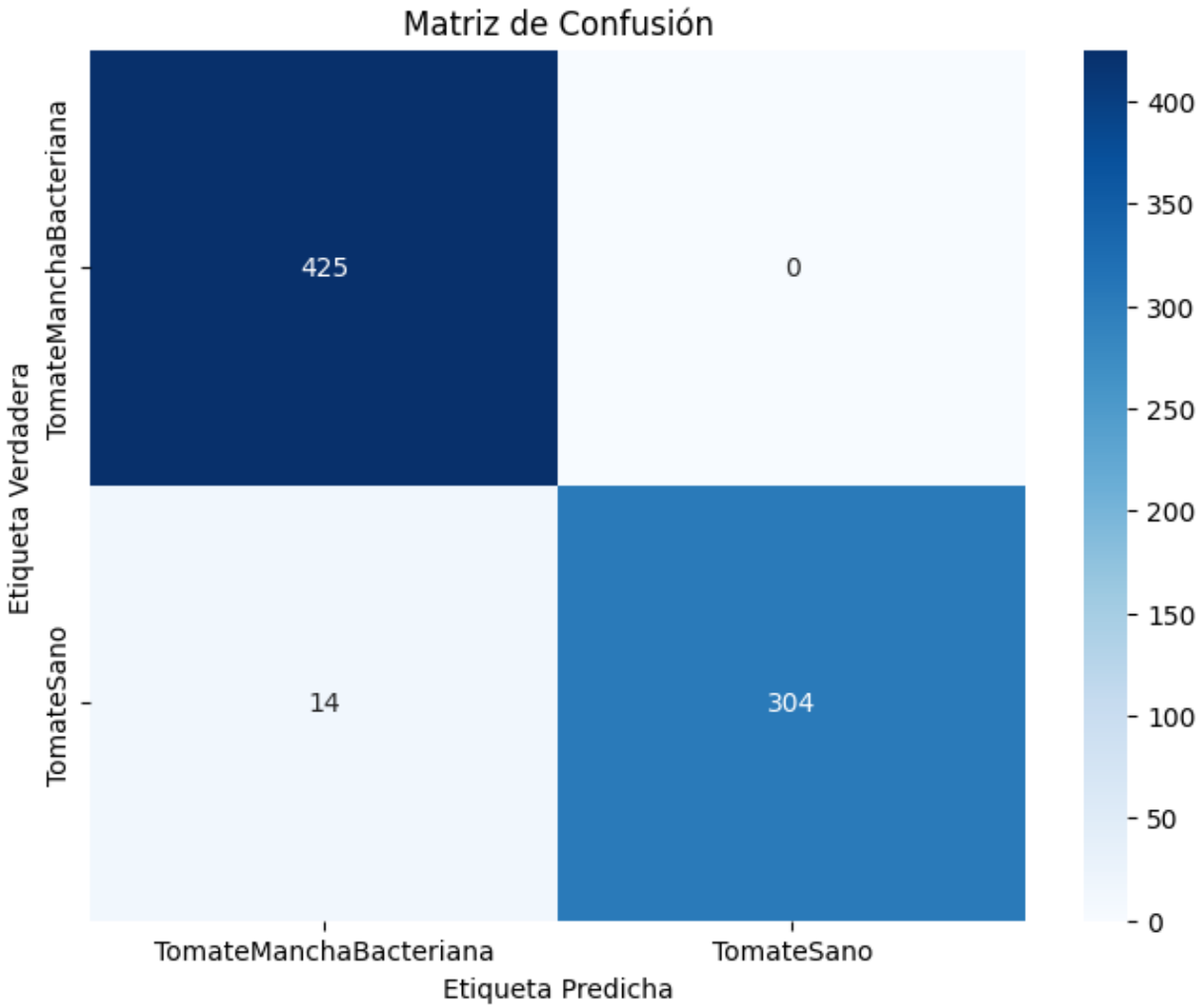

Figura 47 Matriz de confusión del experimento 3

Experimento 4

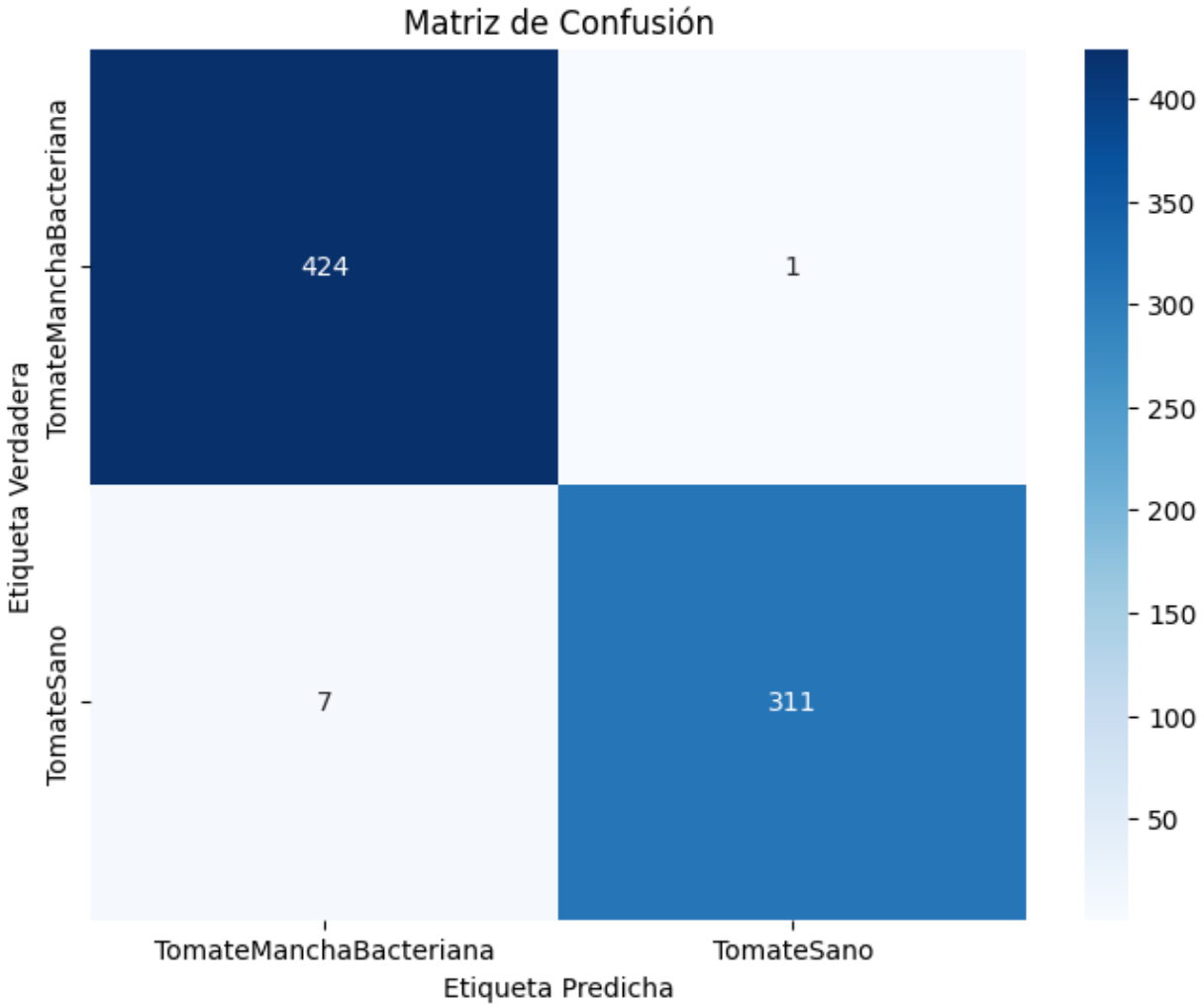

Figura 48 Matriz de confusión del experimento 4

Experimento 5

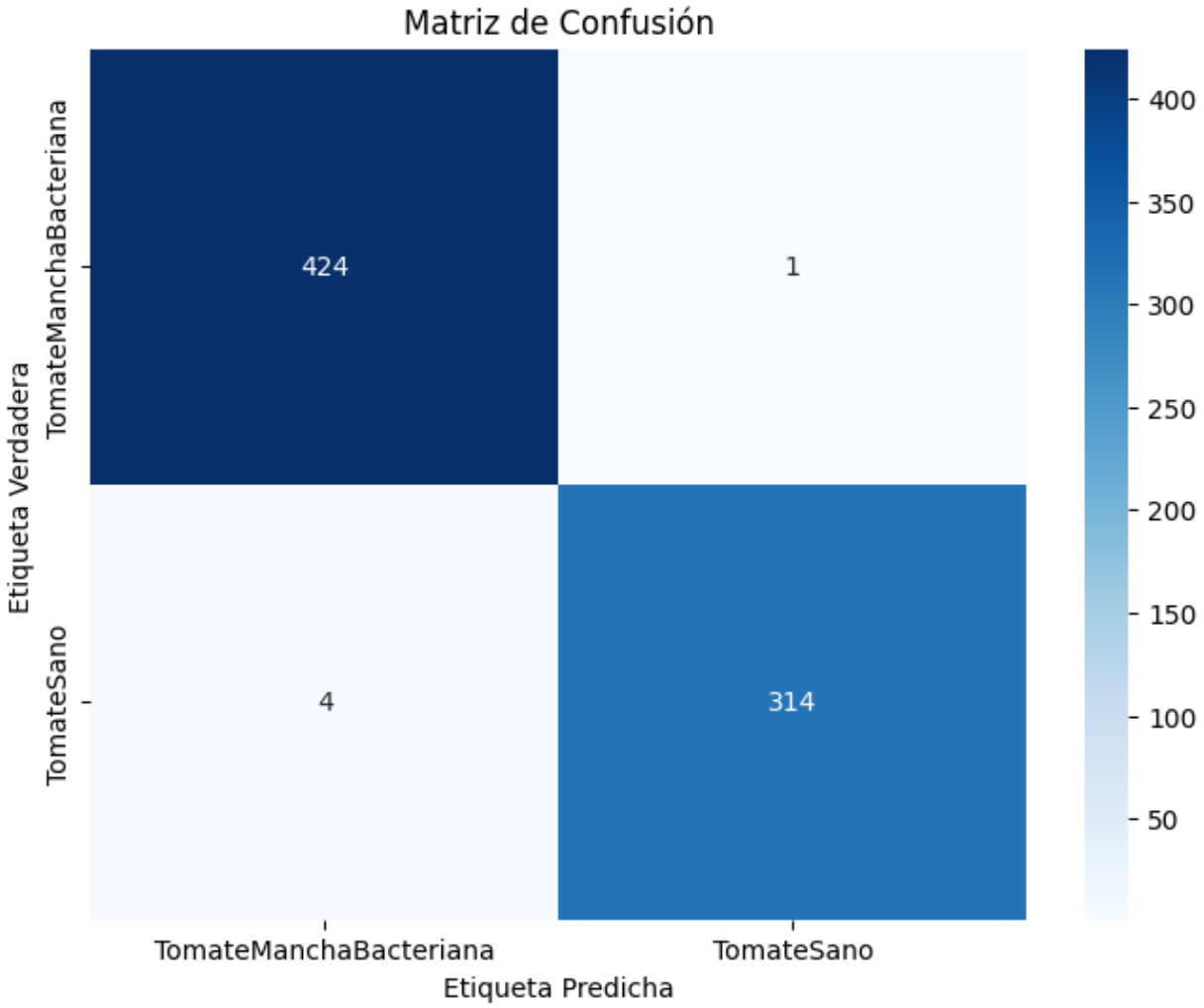

Figura 49 Matriz de confusión del experimento 5

Anexo 3 Certificado avalando traducción del resumen

Universidad Nacional de Loja

Loja, 30 de enero de 2025

Lic. Pedro Geovanny Calva Jiménez
LICENCIADO EN PEDAGOGÍA DEL IDIOMA INGLÉS

CERTIFICO:

Que el resumen del Trabajo de Integración Curricular cuyo título es: **Creación de un modelo para la clasificación de imágenes de la mancha bacteriana en la hoja de tomate utilizando el modelo VGG16**, del aspirante **Jennifer Elizabeth Quizhpe Hurtado**, con cédula de identidad Nro. **1105653172**, de la Carrera de Computación de la Universidad Nacional de Loja, ha sido traducido al inglés y cumple con las características propias del idioma extranjero.

Lo certifico en honor a la verdad y autorizo hacer uso del presente en lo que a sus intereses convenga.

Lic. Pedro Geovanny Calva Jiménez
1150428496
Nro. Reg. Senecyt: 1031-2022-2421774
LICENCIADO EN PEDAGOGÍA DEL IDIOMA INGLÉS