Universidad Nacional de Loja

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

Carrera de Computación

Impacto de los hiperparámetros en arquitecturas YOLO para identificar el tizón tardío en las hojas del cultivo de papa (Solanum tuberosum)

Impact of hyperparameters on YOLO architectures to identify late blight on potato (Solanum tuberosum) crop leaves

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

AUTOR:

Gerardo Manuel Quizhpe Chocho

DIRECTORA:

Ing. María del Cisne Ruilova Sánchez, Mg. Sc.

Loja – Ecuador

2025

Certificación de director

Loja, 05 de marzo del 2025

Ing. María del Cisne Ruilova Sánchez, Mg. Sc.
DIRECTORA DEL TRABAJO DE INTEGRACIÓN CURRICULAR

CERTIFICO:

Que he revisado y orientado todo el proceso de elaboración del Trabajo de Integración Curricular denominado: **Impacto de los hiperparámetros en arquitecturas YOLO para identificar el tizón tardío en las hojas del cultivo de papa (Solanum tuberosum)**, previo a la obtención del título de **Ingeniero en Ciencias de la Computación**, de la autoría del estudiante **Gerardo Manuel Quizhpe Chocho**, con **cedula de identidad Nro. 1106078833**, 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:
MARIA DEL CISNE
RUILOVA SANCHEZ

Ing. María del Cisne Ruilova Sánchez, Mg. Sc.
DIRECTORA DEL TRABAJO DE INTEGRACIÓN CURRICULAR

Autoría

Yo, **Gerardo Manuel Quizhpe Chocho**, declaro ser autor 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: 1106078833

Fecha: 05 de marzo del 2025

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

Teléfono: 0994451236

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

Yo, **Gerardo Manuel Quizhpe Chocho**, declaro ser el autor del Trabajo de Integración Curricular denominado: **Impacto de los hiperparámetros en arquitecturas YOLO para identificar el tizón tardío en las hojas del cultivo de papa (Solanum tuberosum)**, 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 que realice un tercero.

Para constancia de esta autorización, en la ciudad de Loja, a los cinco días del mes de marzo de dos mil veinticinco.

Firma: 

Autor: Gerardo Manuel Quizhpe Chocho
Cédula de identidad: 1106078833
Dirección: Loja - Ecuador
Correo electrónico: gerardo.quizhpe@unl.edu.ec
Teléfono: 0994451236

DATOS COMPLEMENTARIOS:

Directora del Trabajo de Integración Curricular: Ing. María del Cisne Ruilova Sánchez, Mg. Sc.

Dedicatoria

Quiero dedicar el presente trabajo de integración curricular con mucho cariño a mis padres ya que son el principal motivo de inspiración, por su apoyo incondicional, carisma y ánimos positivos fueron las razones que impulsaron a seguir adelante, sin importar las complicaciones que se presenten durante la ejecución.

Gerardo Manuel Quizhpe Chocho

Agradecimiento

Agradezco a mis padres, hermanos, amigos por su apoyo incondicional y consejos de aliento durante todo el proceso académico. Así mismo, quiero agradecer a mi directora, Ing. María del Cisne Ruilova Sánchez, Mg. Sc., por su guía, apoyo, motivación y compartir sus conocimientos adquiridos para culminar el trabajo de integración curricular de manera exitosa. De la misma manera agradezco a los docentes de la carrera de computación quienes aportaron en mi formación académica, especialmente a la Ing. Genoveva Jackelinne Suing Albito, Mg. Sc., quien aportó sus conocimientos y orientación en el desarrollo del TIC.

De igual manera expreso mis agradecimientos al Ing. Agrónomo Angel Rolando Robles Carrión PhD., por su apoyo brindado en el proceso de captura de imágenes.

Gerardo Manuel Quizhpe Chocho

Índice de Contenidos

Índice de Tablas

Índice de Figuras

Índice de Anexos

1. Título

Impacto de los hiperparámetros en arquitecturas YOLO para identificar el tizón tardío en las hojas del cultivo de papa (Solanum tuberosum).

Impact of hyperparameters on YOLO architectures to identify late blight on potato (Solanum tuberosum) crop leaves.

2. Resumen

El cultivo de papa es una actividad agrícola importante en el Ecuador y uno de los tubérculos más consumidas a nivel mundial, no obstante, se ve afectada por enfermedades foliares como el tizón tardío, la cual causa pérdidas económicas y afecta la producción. El Trabajo de Integración Curricular (TIC) tuvo como objetivo general "Determinar la precisión que se obtiene al ajustar YOLOv5 y YOLOv8, mediante la configuración de los hiperparámetros para identificar la enfermedad del tizón tardío en imágenes de hojas de papa". Para lo cual se ejecutaron las fases, comprensión de los datos y negocio, ingeniería de los datos, ingeniería de los modelos y evaluación de los modelos de la metodología CRISP-ML(Q). En la primera fase, se recopilaron cuatro datasets y se capturaron imágenes de hojas de papa infectadas; en la segunda fase se, creó el dataset personalizado, al cual se aplicó limpieza de datos, redimensionamiento, estandarización, etiquetado, aumento de datos y división; en la tercera fase se seleccionó los modelos YOLOv5s y YOLOv8s, ajustando los hiperparámetros; learning rate, epochs, batch size, weight decay y optimizer, a través del método "Un Factor a La Vez" (OFAT). En las métricas de: precisión, recall, mAP@0.50, mAP@0.50:0.90 YOLOv5s logró un 94.01%, 91.94%, 96.59%, 76.99%, mientras que YOLOv8s obtuvo 93.66%, 94.27%, 97.09%, 79.91%, al identificar la enfermedad; en la cuarta fase se evaluó los modelos con el conjunto de pruebas, donde YOLOv8s obtuvo el mejor rendimiento con una precisión del 97.09%; posteriormente se desarrolló un prototipo web utilizando la metodología Cascada y el framework Flask para evaluar el modelo YOLOv8s. Por lo tanto, se logró desarrollar un prototipo que permite la identificación del tizón tardío en las hojas de papa en menor tiempo comparado al proceso manual y obteniendo una precisión del 97.65% en la detección de la enfermedad.

Palabras clave: *Ajuste de hiperparámetros, detección de objetos, OFAT, tizón tardío, YOLOv5s, YOLOv8s.*

Abstract

The potato crop is an important agricultural activity in Ecuador and one of the most consumed tubers worldwide; however, it is affected by foliar diseases such as late blight, which causes economic losses and affects production. The general objective of the Curricular Integration Work (TIC) was “To determine the accuracy obtained by adjusting YOLOv5 and YOLOv8, through the configuration of hyperparameters to identify late blight disease in potato leaf images”. For which the phases, data and business understanding, data engineering, model engineering and model evaluation of the CRISP-ML(Q) methodology were executed. In the first phase, four datasets were collected and images of infected potato leaves were captured; in the second phase, the customized dataset was created, to which data cleaning, resizing, standardization, labeling, data augmentation and splitting were applied; In the third phase, the YOLOv5s and YOLOv8s models were selected, adjusting the hyperparameters: learning rate, epochs, batch size, weight decay and optimizer, through the “One Factor at a Time” (OFAT) method. In the metrics of: precision, recall, mAP@0.50, mAP@0.50:0.90 YOLOv5s achieved 94.01%, 91.94%, 96.59%, 76.99%, while YOLOv8s obtained 93.66%, 94.27%, 97.09%, 79. 91%, when identifying the disease; in the fourth phase, the models were evaluated with the test suite, where YOLOv8s obtained the best performance with an accuracy of 97.09%; subsequently, a web prototype was developed using the Cascada methodology and the Flask framework to evaluate the YOLOv8s model. Therefore, it was possible to develop a prototype that allows the identification of late blight in potato leaves in less time compared to the manual process and obtaining an accuracy of 97.65% in the detection of the disease.

Key words: *Hyperparameter adjustment, object detection, OFAT, late blight, YOLOv5s, YOLOv8s.*

3. Introducción

El cultivo de papa (Solanum tuberosum L.) es una de las principales actividades agrícolas en el Ecuador y uno de los tubérculos más consumidos no solo a nivel nacional, sino mundialmente, no obstante, se ve afectado por diversas enfermedades foliares, entre las que se destaca el tizón tardío, que es causado por el hongo Phytophthora infestans [1]. La principal característica es una mancha necrótica de color negro y se presenta como un tipo de quemazón en las hojas de papa [2]. Esta enfermedad ataca las hojas hasta causar la muerte de la planta [3], por lo que la detección temprana es de suma importancia para mitigar su propagación, lo que ayudaría a reducir pérdidas económicas como del producto.

El Ing. Agrónomo Angel Robles experto en enfermedades foliares hace hincapié que una de las principales limitantes para identificar la enfermedad es que se realiza de manera manual por expertos, guiándose por manchas que surgen en las hojas de la papa, sin embargo, no sabe con certeza si pertenece a la plaga, en tal sentido realizan pruebas en laboratorios lo que conlleva un transcurso entre 3 a 5 días; además, al tratarse de áreas extensas el proceso es lento debido a que se requiere de varias horas, lo que aumenta el riesgo de la pérdida total del cultivo. Por lo tanto, la integración de tecnologías automatizadas, constituye gran aporte para el mejorar el proceso del método tradicional (ver **Anexo 2**), ya que en la actualidad el uso de inteligencia artificial y modelos de visión por computador permiten automatizar las tareas tradicionales llevadas a cabo en la, siembra, cuidado y cosecha.

Además, al momento de entrenar cualquier modelo de aprendizaje profundo es importante tener en cuenta la configuración de los hiperparámetros a usar, ya que influye directamente en el rendimiento del modelo, de tal manera se destaca la importancia de realizar varias pruebas experimentales con la finalidad de determinar la mejor configuración que logre un equilibrio entre convergencia y generalización [4] [5] [6]; en este contexto la Ing. Genoveva Suing docente de Machine Learning hace hincapié que no se puede garantizar el mismo rendimiento logrado con otros modelos, todo depende de la experimentación en la configuración de los hiperparámetros y el conjunto de datos utilizado (ver **Anexo 1**). Por lo tanto, al no realizar varias pruebas se corre el riesgo de tender al sobreajuste o limitar el aprendizaje.

El ajuste de los hiperparámetros es fundamental para lograr un rendimiento óptimo en los modelos de aprendizaje profundo, ya que la configuración adecuada mejora la precisión y generalización del modelo, además la falta de modelos de visión por computador que permitan automatizar tareas como la detección del tizón tardío en los cultivos de papa son limitados. Por lo antes mencionado en el presente TIC, se plantea la siguiente pregunta de investigación "¿Qué porcentaje de precisión se obtiene al ajustar YOLOv5 y YOLOv8, mediante la

configuración de hiperparámetros para identificar la enfermedad del tizón tardío en las hojas de papa?".

Para dar respuesta a la pregunta se establece el objetivo general "Determinar la precisión que se obtiene al ajustar YOLOv5 y YOLOv8, mediante la configuración de hiperparámetros para identificar la enfermedad del tizón tardío en imágenes de hojas de papa" y se plantean los siguientes objetivos específicos: Implementar un modelo de detección de objetos para identificar la infestación de las hojas de papa por el tizón tardío utilizando YOLOv5 y YOLOv8, y Evaluar el modelo mediante pruebas de validación al identificar el tizón tardío en las hojas de papa.

La relación del TIC con otros trabajos se manifiesta en la necesidad de integrar inteligencia artificial como redes neuronales convolucionales en la agricultura, para desarrollar modelos de visión por computador con la finalidad de automatizar el proceso de identificación de enfermedades foliares en los cultivos. Sin embargo, el presente trabajo ofrece un aporte adicional al crear un conjunto de datos personalizado que abarca más de un set, el cual puede ser usado para detectar el tizón tardío en las hojas de papa en entornos reales, reduciendo el tiempo y recursos en la tarea; además a diferencia de trabajos similares que prueban como máximo 3 configuraciones, el trabajo se centra en ajustar los hiperparámetros de los modelos YOLOv5s y YOLOv8s estableciendo la configuración que genera los mejores resultados en las métricas de rendimiento en cada modelo.

El TIC se destaca por la creación de un dataset personalizado de la enfermedad, el cual contiene 5 conjuntos de datos (PlantVillage, Potato disease img_classif, Potato Leaf Disease Dataset in Uncontrolled Enviroment, Potato Leaf (Healthy and Late Blight), dataset propio), con la finalidad de cubrir la variedad de casos en que se puede presentar el tizón tardío. En cuestión a los modelos el trabajo se expande a usar modelos de detección de objetos, es decir se identifican varias zonas en una misma imagen, ya que la mayoría de trabajos relacionados en la tarea de predecir la enfermedad en las hojas de papa se limitan solo a clasificar; además, se destaca por utilizar la metodología CRISP-ML(Q) para la implementación de los modelos de aprendizaje profundo y el método OFAT para el ajuste de hiperparámetros, lo que garantiza un enfoque sistemático y replicable; finalmente se desarrolla un prototipo que permite a usuarios finales evaluar el modelo en un entorno simulado.

Como principales limitaciones del TIC se puede mencionar: la baja disponibilidad de conjuntos de datos con imágenes que se encuentren en diferentes condiciones reales; y la limitada capacidad de recursos computacionales, que restringieron la experimentación con tamaños de lote mayor a 16.

4. Marco teórico

4.1. Antecedentes

El cultivo de papa es uno de los productos más consumidos a nivel internacional y una de las principales producciones en el Ecuador, por esta razón un 89% de la población se dedican al cultivo de la papa. El sembrío se ve afectado principalmente por diversas enfermedades foliares como: Tizón Temprano, Tizón Tardío, Sarna Negra y Fusariosis [1]. Se tiene una estimación de pérdidas económicas de 6.7 mil millones de dólares a nivel mundial por año causadas por el Tizón Tardío en la cosecha y adquisición de fungicidas para su respectivo control (15% de la inversión es para insumos y remedios); en especial causan mayor impacto en los productores pequeños, ya que es su única fuente de ingreso [7] [8].

Los países que se encuentra en desarrollo, pierden 3.5 mil millones de dólares al año, considerándose más de la mitad del porcentaje total que se estima en todo el mundo, al contrario de las potencias, pues este tipo de naciones emplean semillas certificadas y fungicidas de calidad en todas las etapas de desarrollo [7]; de tal manera se debe considerar dichas recomendaciones, con la finalidad de evitar conflictos en las cosechas de los cultivos.

En Ecuador el uso de tecnologías automatizadas para aumentar el rendimiento de los procesos en diferentes áreas es baja, en comparación con los países cercanos (Argentina, Brasil, México, Uruguay, Colombia) o las grandes naciones (Estados Unidos, Rusia, China, Canadá, entre otros), limitándose a las técnicas tradicionales [9] [10]. Pues, el análisis manual de las enfermedades es laborioso y se necesita de tiempo considerable, ya que al contar con grandes cantidades de sembríos se dificulta realizar dicha tarea.

En la actualidad se están incorporando nuevas tecnologías en la agricultura, como: aplicaciones basadas en inteligencia artificial, dispositivos inteligentes y tecnologías para detección, permitiendo automatizar los diversos procesos que se llevan en todas las fases del cultivo, por ejemplo: integrar apps que detecten las enfermedades foliares, usar sensores para riego automático, uso de drones para aplicar químicos en las plantas, entre otros, por ende, el uso de tecnologías visión por computador constituyen aportes, no solo en la agricultura sino en otros campos.

Una posible solución para identificar el Tizón Tardío es automatizar el proceso mediante un modelo de detección de objetos que permita detectar la infestación de la enfermedad en las hojas del cultivo de papa.

4.1.1. Tizón Tardío

También conocida como lancha, es considerada la enfermedad foliar que comúnmente se da en el cultivo de papa, no solo a nivel nacional, sino internacional, es causado por el hongo Phytophthora Infestans, se desarrolla cuando hay temperaturas húmedas y cálidas no

mayores a 24 grados; su principal característica es una mancha necrótica de color marrón que surge en el plantaje [1] [2].

En la **Figura 1** se visualiza el Tizón Tardío, se presenta como un tipo quemazón en las hojas, que es causada por Phytophthora Infestans.

Figura 1. Tizón Tardío en la hoja de papa

La mayoría de veces llega a causar daños en la siembra, que van desde el 41% al 100% cuando no se le aplican fungicidas, principalmente causa la muerte de las hojas y luego de la planta, afectando en el desarrollo, por lo tanto, no carga. Por ende, en un lapso de 7 a 12 días se llega a perder la mayor parte del cultivo, generando pérdidas económicas muy grandes [3].

4.1.2. Agricultura 4.0

En la actualidad se integra la agricultura 4.0, integran algoritmos basados en inteligencia artificial que apoyan en la toma de decisiones, implementación de dispositivos IoT (Internet de las cosas) para mejorar y optimizar las tareas de siembra, cosecha y cuidado con una mejor precisión; además se introducen ramas como: computación en la nube, drones autónomos, big data, robótica. Surge con el pasar del tiempo, pues los métodos tradicionales de siembra son muy trabajosos y en las tareas de cuidados son muy difíciles de analizar las diversas plagas y enfermedades que se pueden dar, por lo que los agricultores se van acoplando a las nuevas tecnologías automatizadas [11] [12].

En la **Figura 2** se presenta una comparativa entre el enfoque tradicional llevado en la agricultura y la integración de avances de la agricultura 4.0 [11].

Figura 2. Comparativa entre la agricultura tradicional frente a la agricultura 4.0

- **Ventajas**

A continuación, se mencionan algunas ventajas que se obtienen al integrar la agricultura 4.0 en los cultivos [9] [13] [14]:

- Optimizar los procesos mediante la aplicación de algoritmos basados en aprendizaje automático.
- Los dispositivos IoT generan un gran impacto en el ámbito ambiental, médico, industrial, etc. y optimizan las tareas lo que ayuda a reducir la mano de obra en personal, por lo tanto, se invierte menos dinero.
- Permiten integrar modelos de visión por computador en invernaderos para tener un control o monitoreo constante del cultivo y mantener en condiciones óptimas, seguimiento en el desarrollo por ende se tienen buenas cosechas.
- Uso de sensores facilitan el control de las condiciones ambientales, por ejemplo, son muy usados para las tareas de riego automatizado.
- Ayuda en la toma de decisiones, agilizando los procesos en los sembríos.
- Predecir enfermedades en base a las condiciones climáticas.

4.2. Fundamentación Teórica

4.2.1. Aprendizaje automático

También conocido como Machine Learning, permite a los ordenadores tener la capacidad de aprender características o patrones sin la necesidad de ser programados, es decir un programa aprende de la experiencia, en función de una acción y una métrica de rendimiento para evaluar el algoritmo [15]. Integra varios algoritmos que le permiten aprender de datos y poder dar sugerencias o predicciones [16].

- **Tipos de aprendizaje automático**

  El aprendizaje automático posee diversos tipos de algoritmos [16] [17]:

  - **Aprendizaje supervisado**

    Se trata de algoritmos que usan datos etiquetados, es decir cada dato tiene asignado una etiqueta o clase que permite identificar a qué categoría pertenecen las nuevas instancias.

  - **Aprendizaje no supervisado**

    No usa datos etiquetados, sino que el algoritmo aprende y busca patrones para agrupar o clasificar de acuerdo a las características.

  - **Aprendizaje por refuerzo**

    Los algoritmos aprenden de la experiencia, es decir se da una recompensa o se penaliza dependiendo del resultado arrojado.

4.2.2. Redes neuronales artificiales (ANN)

Son redes altamente interconectadas entre ellas de manera paralela, que tratan de simular el comportamiento del cerebro humano, mediante la implementación de modelos matemáticos, para lo cual adquieren aprendizaje, a través de la experiencia y extracción de información de un dataset (base de todo proyecto de inteligencia artificial) [18]. También se destaca lo siguiente: extienden el conocimiento automáticamente y abstraen características de un objeto que no son comunes en relación con los demás datos [19].

En la **Figura 3** se aprecia la estructura de una ANN, es similar a la neurona del cerebro, está compuesta por las dendritas o sinapsis que son las entradas, y el axón que es la salida [20].

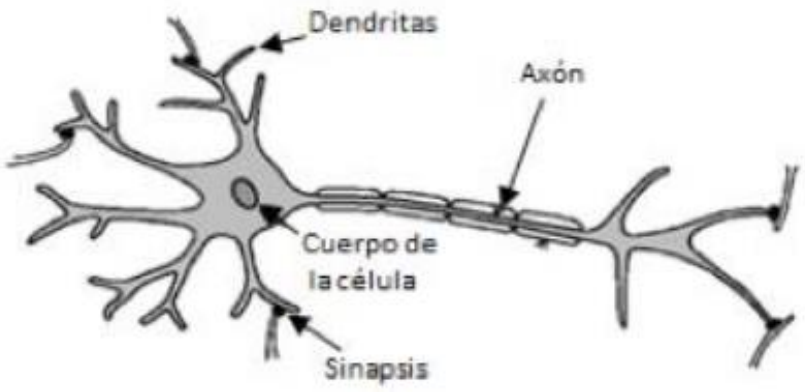


Figura 3. Neurona biológica

En la **Figura 4** se muestra los elementos básicos que componen una ANN, a continuación, se describe cada una [21]:

- Xi son las entradas que poseen valores numéricos y envían señales.

- Wj simbolizan los pesos sinápticos que determinan cómo la neurona se va a comportar.
- Un sesgo φ que permite activar la neurona si supera un umbral.
- Yj se encarga de sumar todas las entradas multiplicadas por los pesos
- Una función de activación no lineal (Escalón, Sigmoide, ReLu, Foftmax, Tangente hiperbólica), que permite extender el aprendizaje de la neurona.
- Una salida v1j, v2j.

$$salida = \sum_{k=0}^{n} wj * xi \quad (1)$$

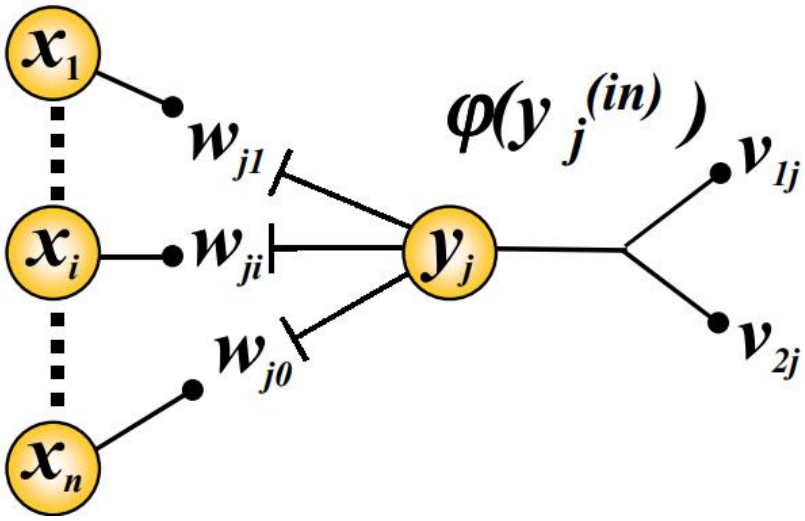


Figura 4. Esquema de una red neuronal artificial

4.2.3. Aprendizaje profundo

Integra diversos algoritmos que usan el aprendizaje por capas o arquitecturas profundas, cada capa tiene una función en específico y están conectadas entre sí, permitiendo dividir las tareas complejas en las diversas capas, aumentando el aprendizaje de los modelos durante la fase de entrenamiento [22]. Específicamente las redes neuronales convolucionales (CNN) son las más usadas, en aplicaciones de Visión por Computador como: reconocimiento de imágenes o vídeo, procesamiento de lenguaje natural, entre otras. Debido a su gran desempeño este tipo de arquitecturas solucionan los problemas relacionados con el análisis complejo que presentan las imágenes, audios y vídeos [23].

4.2.4. Visión por computador

Se refiere a la capacidad que tienen los dispositivos informáticos para procesar diversas cantidades de contenido; las computadoras poseen la capacidad de ver, a través de los periféricos como: cámaras, mouse, teclado, entre otros. Además, es considerada una rama enfocada en la implementación de algoritmos y la fundamentación de la teoría mediante el

análisis de características [24]. El campo de Visión por Computador es frecuentemente usado en tareas de análisis datos en imágenes, sistemas demográficos, clasificación automatizada de una tarea en específico, reconocimiento facial, procesamiento, entre otros [25].

En la **Figura 5** se aprecia como un ordenador interactúa con los diversos periféricos de entrada para extraer la información esencial de una imagen o texto y poder dar un resultado en base a los datos [26].

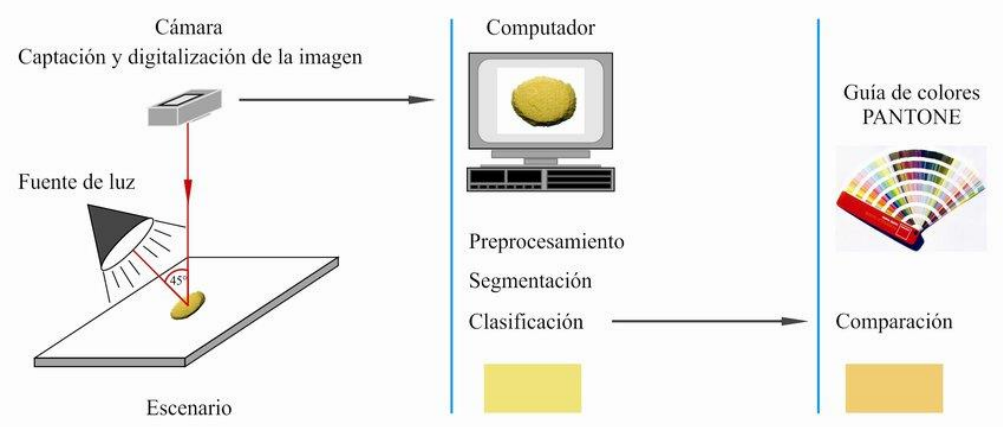


Figura 5. Etapas de un sistema de visión por computador

4.2.4.1. Redes Neuronales Convolucionales (CNN)

CNN por sus siglas en inglés Convolutional Neuronal Network, se trata de un modelo de aprendizaje profundo que consta de múltiples capas o redes multicapa, módulos convolucionales y de submuestreo (Pooling), formando un perceptrón multicapa [27]. El propósito que tienen, es extraer las diversas características de una imagen, audio o vídeo, para posteriormente usar dicha información en tareas como: detección de patrones, clasificación o predicciones. Las CNN como YOLO (You Only Look Once), Faster R-CNN (Region-Based Convolutional Neural Networks) y SSD (Single Multibox Detector) [23], son frecuentemente usadas en tareas de detección de objetos.

En la **Figura 6** se presenta la arquitectura de una CNN, está compuesta por dos capas convolucionales, dos capas de agrupamiento (Max pooling), una capa de aplanamiento, una capa oculta compuesta por 128 neuronas y una capa de salida con 10 neuronas [28].

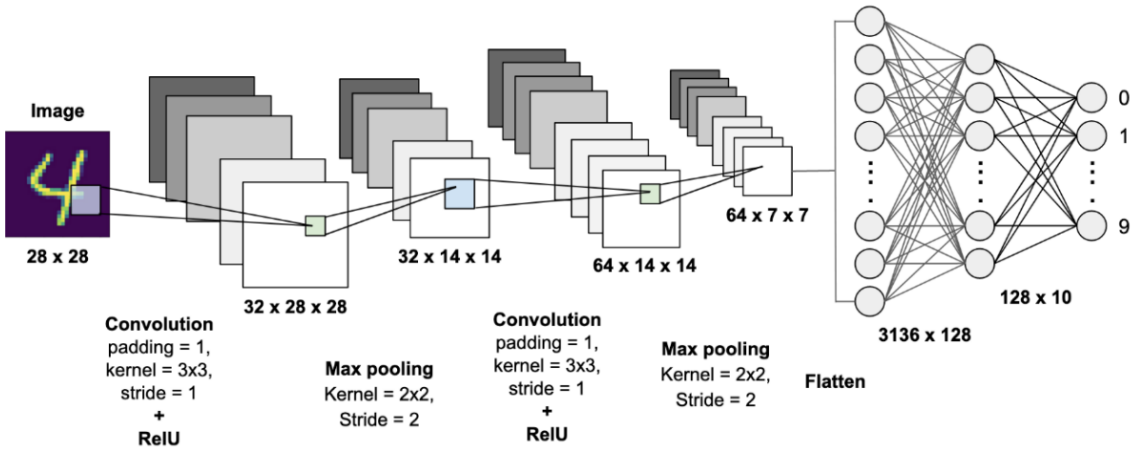


Figura 6. Arquitectura de una Red Neuronal Convolucional

4.2.5. Detección de objetos

Es una tarea muy importante dentro del campo de la visión por computador, surge como una solución que enfrentan los modelos de Machine Learning, ya que la detección permite detectar varios objetos (personas, animales, autos, entre otros), en una imagen o video [29]. Permite determinar la zona o desplazamiento del objeto, mediante un cuadro delimitador asociado a una etiqueta que identifica a la zona de interés, es decir el modelo posee la capacidad de predecir la ubicación exacta y asignar la clase a la que pertenece [30].

En la **Figura 7** se muestra los pasos que lleva a cabo un modelo basado de detección de objetos [25].

Figura 7. Proceso para la detección de objetos

En la **Figura 8** se presenta un ejemplo de detección de objetos, donde se asigna una etiqueta al cuadro delimitador que anteriormente fue predicho por una CNN [31].

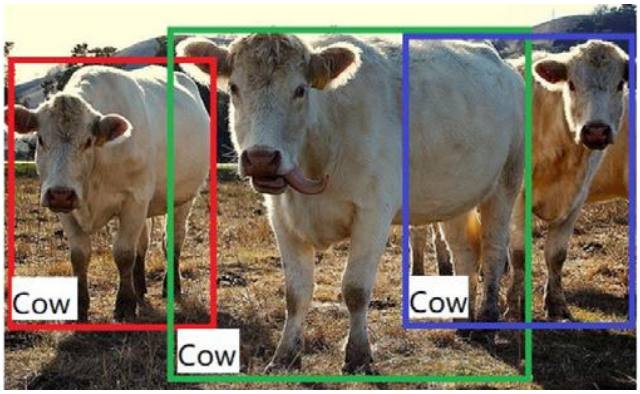


Figura 8. Ejemplo de detección de objetos

4.2.6. You Only Look Once (YOLO)

Desarrollado 2016 por Joseph Redmon et al., es el primer framework enfocado para la detección de objetos en tiempo real [32], también es muy usado para la clasificación y es más eficiente que las redes neuronales DPM (Deep Probabilistic Models) y R-CNN (Region-Based Convolutional Neural Networks) en cuestión de recursos computacionales [33]. YOLO integra el uso de cuadros delimitadores asociada una clase, al momento de hacer una predicción realiza la extracción de características para luego dibujar un cuadro delimitador del objeto analizado. Además, permite la detección en tiempo real, diferenciándolo de otras redes neuronales orientadas a la detección y es muy rápido en cuestión de procesamiento [34].

En la **Figura 9** se presenta la arquitectura de YOLO, consta de 24 capas convolucionales, continuadas de 2 capas totalmente conectadas [33].

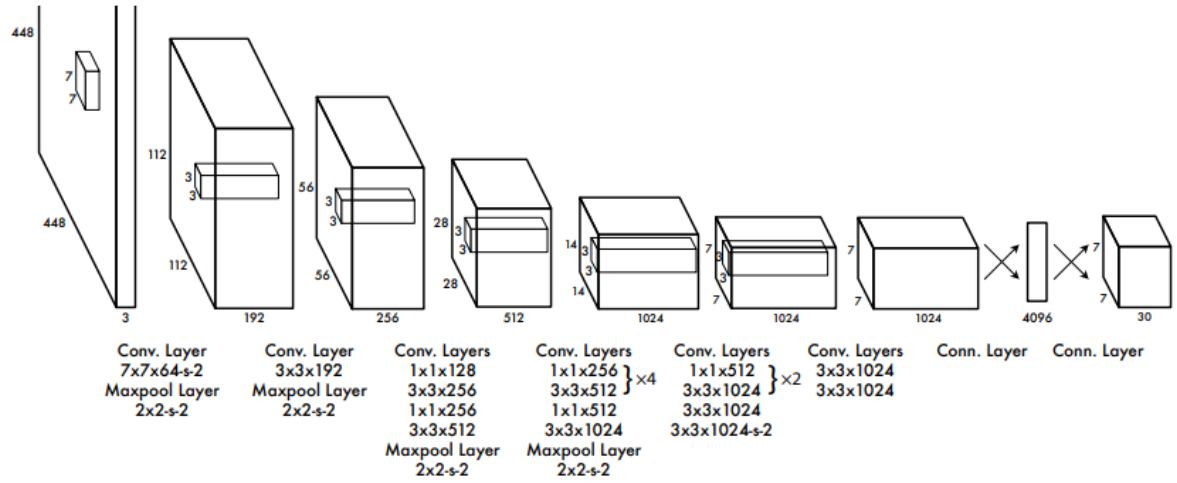


Figura 9. Arquitectura del framework YOLO

4.2.6.1. YOLOv5

YOLOv5 desarrollado por Glenn Jocher, incorpora un nuevo módulo CSPNet (Cross Stage Partial Network); integra nuevas mejoras con respecto a las versiones anteriores (YOLOv3, YOLOv4), aumenta la capacidad de detección en las imágenes pequeñas, ajusta el espacio RGB y mejora la predicción [35]. Está compuesta por 3 capas: red troncal (Backbone), red de cabeza (Head) y red de cuello (Neck) mejorado con la red de agregación de rutas (PAN) y el módulo de agrupación de pirámide espacial (SPP), las cuales permiten mejorar el rendimiento de YOLOv5 [36].

En la **Figura 10** se muestra la arquitectura de YOLOv5 (ver a detalle la arquitectura[1]); la capa Neck integra la red SPP (Spatial Pyramid Pooling) y la red BottleNeckCSP, estas se encargan de extraer las diversas características de las entradas, mejorando la extracción con

[1] Arquitectura detallada de YOLOv5:
https://docs.ultralytics.com/yolov5/tutorials/architecture_description/#1-model-structure

la implementación con CSPNet (Cross Stage Partial Network), optimizando la precisión al localizar los pixeles de las imágenes. Por lo tanto, YOLOv5 aborda los problemas relacionados con los gradientes redundantes y mejora la eficacia en la detección [37].

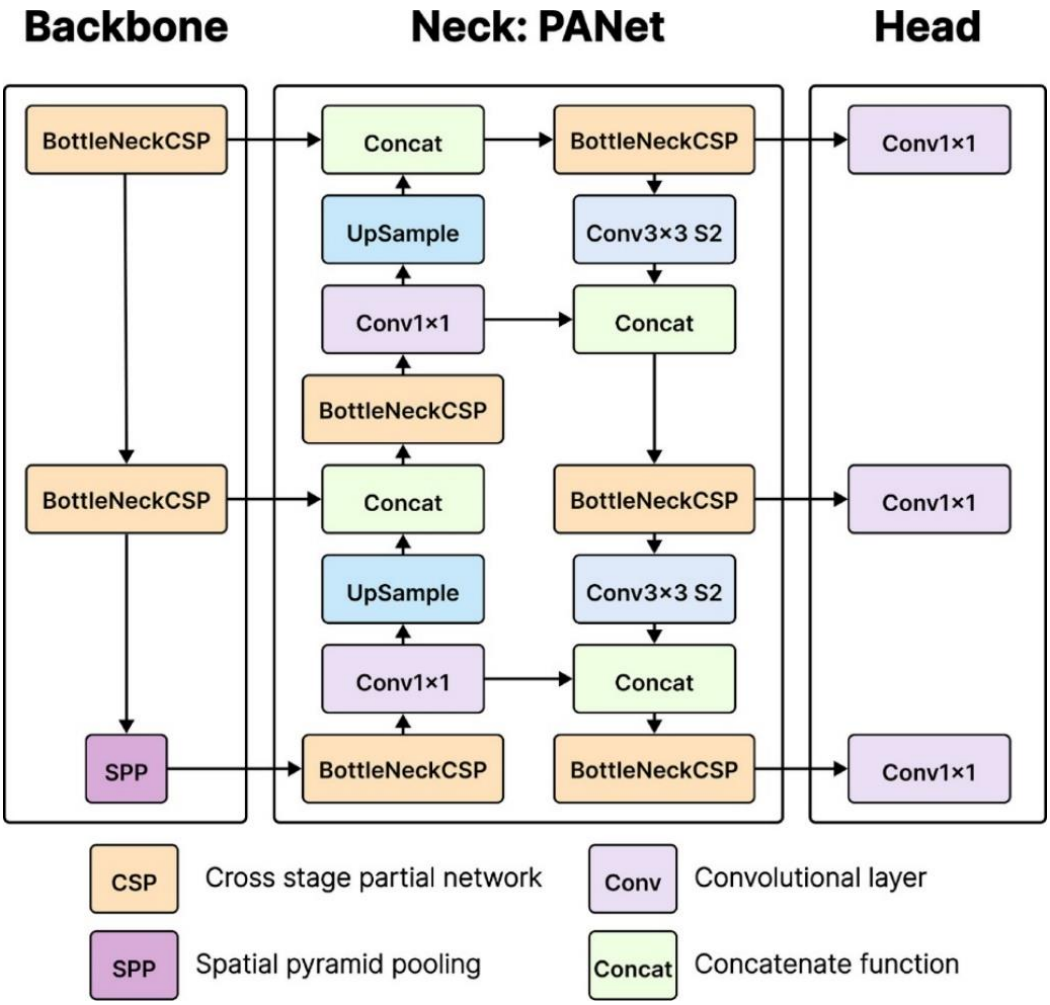


Figura 10. Arquitectura del modelo YOLOv5

En la **Tabla 1**, se presenta un resumen las variantes del modelo YOLOv5[2] para la detección de objetos entrenado con el dataset COCO.

Tabla 1. Resumen de las variantes de YOLOv5 entrenadas con el dataset COCO

Modelo	Tamaño de imagen (píxeles)	Parámetros (Millones)	mAP val 50-95
Yolov5nu	640	2.6	34.3
Yolov5su	640	9.1	43
Yolov5mu	640	25.1	49
Yolov5lu	640	53.2	52.2
Yolov5xu	640	97.2	53.2

Precisión media promedio (mAP) con intersección sobre unión 50-95

[2] Versiones de YOLOv5: https://docs.ultralytics.com/models/yolov5/

Yolov5s (small), es adecuada para apps móviles y sistemas que no requieren de muchos recursos computacionales; se diferencia de las demás versiones por el número de capas de convolución y los parámetros que posee. En específico la distribución s consta de 9.1 millones que son los pesos y sesgos aprendidos en la etapa de entrenamiento, además el modelo ajusta estos parámetros para aumentar el rendimiento, no obstante, la velocidad en la detección se reduce. Además, agrega el módulo C3 en la red troncal (Backbone) que se integra con SPP para fusionar las características de la imagen en mapas, asignando la ubicación del objeto y la confianza de la predicción en el cuadro delimitador con mayor precisión [38].

4.2.6.2. YOLOv8

YOLOV8 lanzado por Ultralytics, es la mejora de YOLOv5 en todas sus versiones y el sucesor de YOLOv7, en especial optimiza el módulo CSPDarknet (Cross Stage Partial Network) de YOLOv5 y el módulo PANet, utiliza la función de activación SILU (Swish) en vez de Leaky ReLU que usa YOLOv5 [39], descarta la capa convolucional de la fase de muestreo ascendente, es más rápido y efectivo para detectar la zona objetivo y clasificar con mayor precisión [40].

En la **Figura 11** se muestra la arquitectura de YOLOv8 (ver a detalle la arquitectura[3]), donde el bloque Conv está compuesta de cuatro capas, se encargan de realizar las operaciones de convolución, luego el módulo C2f soluciona los problemas con el gradiente, una vez realizada esta tarea pasa al módulo SPPF (Spatial Pyramid Pooling Fast) está encargado de transformar las características en varios tamaños para luego asignarlos a un vector de tamaño fijo, reduciendo el costo computacional y finalmente pasa a la capa de detección que se encarga de clasificar y predecir la ubicación del objeto, asignando una etiqueta al cuadro delimitador del resultado [41].

[3] Arquitectura detallada de YOLOv8: https://github.com/ultralytics/ultralytics/issues/189

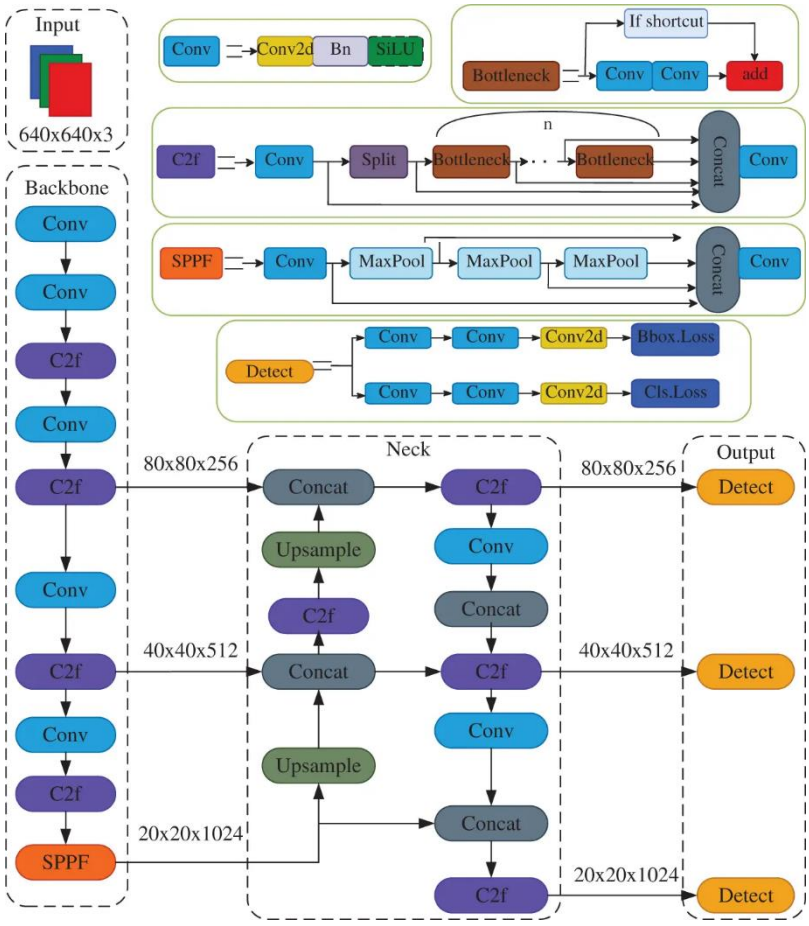


Figura 11. Arquitectura del modelo YOLOv8

En la **Tabla 2** se muestra un resumen de las distribuciones del modelo YOLOv8[4] para la detección de objetos entrenado con el dataset COCO.

Tabla 2. Resumen de las variantes de YOLOv8 entrenadas con el dataset COCO

Modelo	Tamaño de imagen (píxeles)	Parámetros (Millones)	mAP val 50-95
Yolov8n	640	3.2	37.3
Yolov8s	640	11.2	44.9
Yolov8m	640	25.9	50.2
Yolov8l	640	43.7	52.9
Yolov8x	640	68.2	53.9

Precisión media promedio (mAP) con intersección sobre unión 50-95

YOLOv8s (small), es un modelo preentrenado liviano, que no requiere de muchos recursos para su ejecución en comparación con sus versiones sucesoras, también varían los parámetros (11.2 millones) y el número de capas convolucionales. Además, usa varias formas de extraer las características en la capa de entrada como: anclaje adaptativo y escalado de imágenes; durante la etapa de entrenamiento genera diversos cuadros de la región de interés y usa la supresión no máxima (NMS), para determinar la predicción más cercana [42].

[4] Versiones de YOLOv8: https://docs.ultralytics.com/models/yolov8/

4.2.6.3. Hiperparámetros de configuración para los modelos YOLO

- **Tasa de aprendizaje (lro)**

Determina la velocidad en que un modelo actualiza sus pesos durante la etapa de entrenamiento, se debe tomar en cuenta que el uso de una tasa de aprendizaje alta tiende a converger más rápido, lo que afecta el rendimiento, no obstante, al usar una tasa baja se tiende a demorar más al encontrar la mejor solución, pero se garantiza que el uso de un lro bajo tiende a una convergencia más estable [43] [44] [45].

- **Épocas**

Es un parámetro que hace referencia al número total de iteraciones que da un modelo durante el entrenamiento; la mayoría de veces mientras mayor es el número de épocas el modelo aprende más, no obstante, puede tender al sobreajuste, así mismo un número menor de épocas se considera una mejor opción ya que se ejecuta en menos tiempo [43].

- **Optimizador**

Se tratan de los algoritmos que usa un modelo en la fase de entrenamiento para actualizar los parámetros, con la finalidad de reducir la función de pérdida. Hay diversos optimizadores y cada uno aplica un método diferente [43]. A continuación, se mencionan los algoritmos de optimización estocástica más usados:

- **SGD (Stochastic Gradient Descent):** Se refiere a un algoritmo de optimización por ascenso de gradiente estocástico usado para entrenamiento de modelos de aprendizaje profundo. Utiliza tamaños de pasos fijos en cada iteración [46] [47].
- **Adam (Adaptive Moment Estimation):** Es un método para la optimización basada en gradientes de primer orden, combina las ventajas de dos métodos: AdaGrad (Adaptative Gradient) funciona bien manejando gradientes dispersos y RMSProp trabaja bien en entornos no estacionarios y en línea [48]. Calcula tasas de aprendizaje adaptativos basados en las iteraciones anteriores [46], no necesita de requerimientos computacionales altos y ni mucha memoria para ejecutarse [48].
- **AdamW (Adaptive Moment Estimation Weight Decay):** Desacopla el decaimiento de peso y las actualizaciones de gradiente basadas del optimizador Adam, aumentando la capacidad de velocidad en la etapa de entrenamiento generando tasas de pérdidas bajas [49]. Altera los pesos de la actualización del gradiente evitando que el modelo tienda al sobreajuste [50]
- **AdaMax (Adaptative Max):** Es una variante de Adam basada en la norma del infinito o norma máxima para la optimización, pues la magnitud de las actualizaciones de los pesos es más simple que Adam [48], generando un

rendimiento eficaz del modelo entrenado. Frecuentemente es usado cuando el conjunto de datos posee mucho ruido o los datos poseen valores atípicos [51].

- **RMSProp (Root Mean Squared Propagation):** Es método de optimización de la tasa de aprendizaje adaptativo (aumentar la tasa de aprendizaje) lo que permite dar pasos más grandes, donde la magnitud de los gradientes difiere para diversos pesos y pueden variar durante el entrenamiento, dificultando el aprendizaje del algoritmo [52], por lo tanto, RMSProp soluciona esta adversidad ajustando los pesos en base a la magnitud y conservando un promedio del gradiente al cuadrado [53].

- **Batch size**

Hace referencia a la cantidad de muestras de entrenamiento usadas en una iteración lo que actualiza los pesos del modelo. Un tamaño de lote pequeño genera más actualizaciones de peso por cada época ejecutada, pero son muy propensos a generar ruido por otra parte, un tamaño de lote grande proporciona una estimación de gradiente más eficaz no obstante se requiere de recursos computacionales bastante altos [43] [54].

- **Weigth decay**

Decaimiento de peso o regulador de pesos, es una técnica que permite regular la función de pérdida mediante el uso de penalización, es controlada por la degradación de peso, permitiendo evitar el sobreajuste del modelo y aumentando el rendimiento con datos nuevos [43] [55] [50].

4.2.7. Métricas de evaluación para la clasificación binaria

Las diferentes métricas como la matriz de confusión, precisión, recall, F1-score permiten evaluar el rendimiento de un modelo entrenado para una tarea en específico, por lo tanto, se puede evaluar que tan bien un modelo predice nuevas instancias. A continuación, se detalla cada una de estas métricas:

- **Matriz de confusión:** Permite visualizar a través de una tabla como un modelo es capaz de predecir nuevos datos y asignarlos a cada una de sus filas o columnas como: Verdaderos Positivos (TP), Verdaderos Negativos (TN), Falsos Positivos (FP) y Falsos Negativos (FN), son usados en la clasificación binaria.

En la **Figura 12** se presentan las columnas de la matriz que representan la cantidad de predicciones que realiza el modelo mientras que las filas son las instancias reales de la clase [56].

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

Figura 12. Matriz de confusión

Dónde [57]:

- **TP:** Verdaderos Positivos, son las instancias positivas que son predichas de manera correcta como positivas.
- **TN:** Verdaderos Negativos, son las instancias negativas, que son predichas de manera correcta como negativos.
- **FP:** Falsos Positivos, corresponden a las instancias que son negativas y el modelo los clasifica como correctas.
- **FN:** Falsos Negativos, se tratan de las instancias positivas y que el modelo las clasifica como negativas.

De la matriz de confusión se pueden evaluar diferentes métricas, como las siguientes:

- **Precisión**

Determina como el modelo es capaz de predecir nuevas instancias correctamente [58].

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

- **Recall**

También conocida como recuperación, determina la capacidad del modelo para identificar las instancias correctas en relación con todos los datos que pertenecen a los verdaderos positivos [59][60].

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

- **F1-Score**

F1-Score o media armónica entre la precisión y recall [59], permite evaluar el rendimiento en general, oscila entre 0 y 1. Si el valor es 1 indica que el modelo es confiable en las predicciones mientras que si es 0 determina que no hay una confianza en las predicciones [60].

$$F1 - Score = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (4)$$

4.2.8. Métricas de detección de objetos

En el campo de la detección de objetos las diferentes métricas de evaluación son diferentes a las métricas de una clasificación, a continuación, se detallan:

- **AP (Average Precision)**

Precisión promedio, permite medir el área bajo la curva entre la precisión y recall. Su ecuación se aprecia [61]:

$$AP = \int_0^1 P(R)dR \quad (5)$$

- **mAP (Mean Average Precision)**

Precisión media promedio, evalúa el rendimiento global de un modelo, brindando información pertinente sobre cuál es la capacidad para detectar y recuperar de manera eficaz objetos de diversas clases. Por lo tanto, un mAP alto determina un buen rendimiento en la detección de las clases [60]. La ecuación 5 permite calcular el mAP [60]:

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i \quad (6)$$

- **mAP50 (Mean Average Precision IoU 50)**

mAP50 indica el porcentaje que se obtiene en un umbral de IoU (Intersection over Union) 0.5 de la detección. A continuación, se muestra la ecuación para calcular mAP50 [41]:

$$mAP50 = \frac{1}{N} \sum_{i=1}^{N} AP50_i \quad (7)$$

- **mAP50-95 (Mean Average Precision IoU 50-95)**

mAP50-95 indica el porcentaje que se obtiene entre el umbral de IoU 0.5 a 0.95, mediante el aumento de 0.05. Tal como se aprecia en la ecuación [41]:

$$mAP50 - 95 = \frac{1}{10}\,\text{mAP50} + \frac{1}{10}\,\text{mAP55} + \cdots + \frac{1}{10}\,\text{mAP95} \quad (8)$$

4.2.9. Herramientas de etiquetado de imágenes

Al momento de trabajar con modelos basados en la detección de objetos, es necesario etiquetar las imágenes, donde se asigna una clase y un cuadro a la región de interés. A continuación, se describen algunas herramientas que facilitan este tipo de tarea:

- **Roboflow**

Es una herramienta que no requiere ser instalada, se encuentra en la web y es de acceso libre, simplemente se necesita registrar y poder hacer uso de sus funcionalidades. Roboflow ayuda en el etiquetado de imágenes para la detección de objetos de manera intuitiva es decir se agrega un cuadro delimitador de la zona y una etiqueta [62].

- **LabelIMG**

Es una herramienta de acceso libre, permite etiquetar o anotar las imágenes, fue creado por Tzulin, actualmente forma parte de Label Studio; para hacer uso del software requiere ser instalado, se encuentra disponible para diversos sistemas operativos como: Windows, Linux, Ubuntu y Mac). Permite guardar las anotaciones en archivo XML específicamente en el formato PASCAL VOC formato admitido por ImageNet, también permite generar archivos txt los cuales son aceptados por YOLO [63].

En la **Tabla 3** se presenta una comparación entre las herramientas de etiquetado Roboflow y LabelIMG.

Tabla 3. Comparativa entre Roboflow y LabelIMG

Roboflow	LabelIMG
No requiere ser instalado.	Requiere instalar.
Permite dos formas de etiquetar: cuadro o polígono.	Solo admite etiquetar con un cuadro.
Software en línea.	Aplicación.
Permite realizar la división del dataset etiquetado.	No permite hacer la división.
Iniciar sesión.	No requiere iniciar sesión.
No requiere comandos.	Ejecutar comando “python labelImg.py” para abrir la interfaz.
Facilita la opción de entrenar un modelo con el dataset etiquetado.	No tiene una función de entrenar modelos.
Integra la opción de aplicar técnicas de aumento de datos.	No posee la funcionalidad de aplicar el aumento de datos.

En base a la comparativa descrita en la **Tabla 3**, se destaca la herramienta Roboflow por sus diversas ventajas y funciones que permite realizar como: etiquetado de diversas formas, técnicas de aumento de datos, dividir el dataset y entrenar modelos con los datos generados en comparación con LabelIMG que permite opciones básicas de etiquetado.

4.2.10. Proceso Estándar Intersectorial para el Aprendizaje Automático con Garantía de Calidad (CRISP-ML(Q))

CRISP-ML(Q) por sus siglas en inglés Cross Industry Standard Process for Machine Learning with Quality Assurance, comúnmente usada para planificar y ejecutar proyectos de

Machine Learning. Es una metodología iterativa, que hace énfasis en la corrección de errores (listar riesgos y mitigarlos), asegurando la calidad en cada fase ejecutada [64].

En la **Figura 13** se visualiza el ciclo de vida que debe seguir un proyecto de Machine Learning y las fases que se deben llevar a cabo [65].

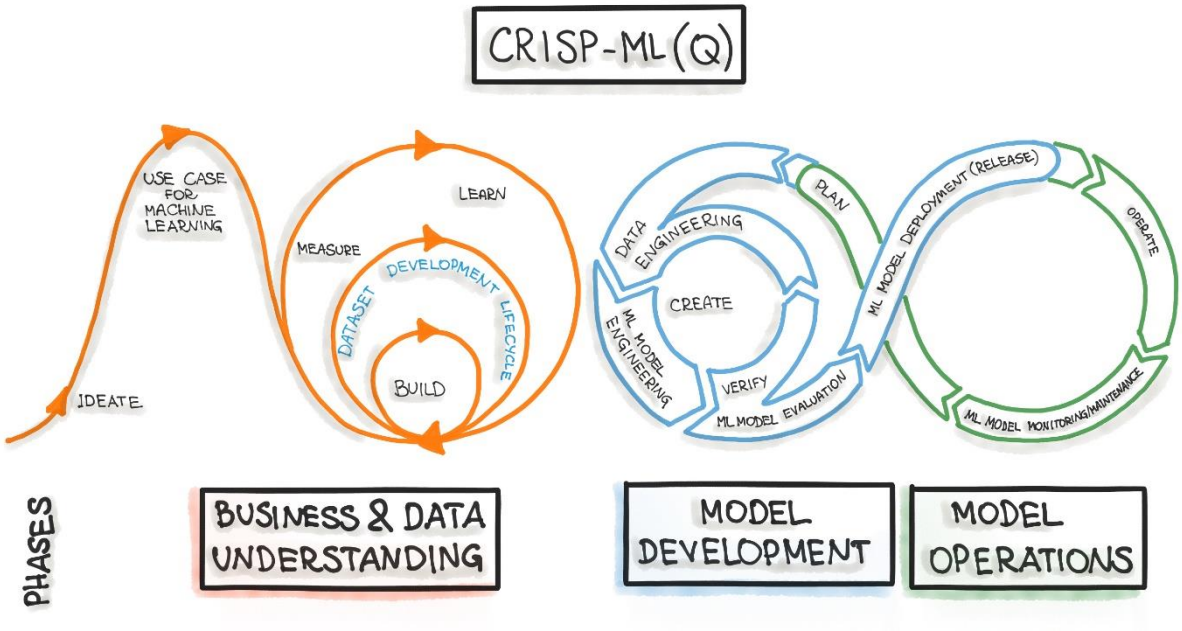


Figura 13. Ciclo de vida de la metodología CRISP-ML(Q)

4.2.10.1. Fases de la metodología CRISP-ML(Q)

En la **Tabla 4**, se presenta un resumen de las tareas que se deben llevar a cabo en cada una las fases de la metodología [64] [65]:

Tabla 4. Resumen de las fases de la metodología CRISP-ML(Q)

Fase	Descripción	Tareas
Comprensión de los datos y negocio	Definir los objetivos y el alcance del proyecto de Machine Learning (ML).	• Definir objetivos comerciales. • Definir el alcance de la aplicación de ML. • Recopilar y verificar los datos. • Evaluar la viabilidad del proyecto. • Crear prueba de concepto (POC).
Ingeniería de los datos	Aplicar tareas necesarias para preparar los datos y obtener el dataset listo para el modelo.	• Seleccionar los datos. • Equilibrio de clases. • Limpieza de datos. • Ingeniería de características (construcción de datos). • Aumento de datos. • Estandarización de datos.
Ingeniería del modelo	Establecer los modelos de aprendizaje automático que se	• Definir la medida de calidad del modelo. • Selección de modelos. • Agregar conocimiento del dominio para especializar el modelo. • Entrenamiento del modelo.

Fase	Descripción	Tareas
	van a implementar en el proyecto.	• Compresión del modelo. • Aprendizaje en conjunto. • Documentar el modelo ML y los experimentos.
Pruebas y validación del modelo	Determinar el rendimiento del modelo mediante pruebas de validación.	• Validar el rendimiento del modelo • Determinar la robustez • Aumentar la aplicabilidad del modelo. • Tomar una decisión sobre si implementar el modelo. • Documentar la fase de evaluación.
Despliegue	Poner en producción el modelo entrenado ya sea en un aplicativo web o móvil.	• Evaluar el modelo en condiciones de producción. • Asegurar la aceptación y usabilidad del usuario. • Gobernanza del modelo. • Implementar de acuerdo con la estrategia seleccionada.
Monitoreo y Mantenimiento	Descubrir errores en la implementación del modelo en un campo práctico, para posteriormente solucionar las falencias y prevenir el deterioramiento de la aplicación.	• Supervisar la eficiencia y eficacia del servicio de predicción del modelo. • Volver a entrenar el modelo si es necesario • Recopilar nuevos datos. • Realizar el etiquetado de los nuevos puntos de datos. • Repetir las tareas de las fases: Ingeniería del modelo y Evaluación del modelo. • Continuidad, integración, entrenamiento e implementación del modelo.

4.2.11. Metodología Cascada

Establecido por Winston Royce, que se empezó a diseñar en 1966 y fue presentado en 1970 [66]; es considerado como el primer modelo para el desarrollo de software [67]; también conocido como Waterfall debido a la ubicación de las fases que caen en cascada hacia las demás etapas, es una de las metodologías más usadas en proyectos de desarrollo de software [67] [68]; principalmente el marco de trabajo es consecutivo (secuencial) ya que las fases se ejecutan por separado, por tanto, se debe culminar la etapa anterior para seguir con la siguiente [69], al final de cada una se documenta todo lo realizado, con la finalidad de asegurar que se cumple con todo los requerimientos antes de pasar a la siguiente fase [66].

En la **Figura 14** se presenta las fases de la metodología cascada para el desarrollo de software, la cual consta de 5 fases [70].

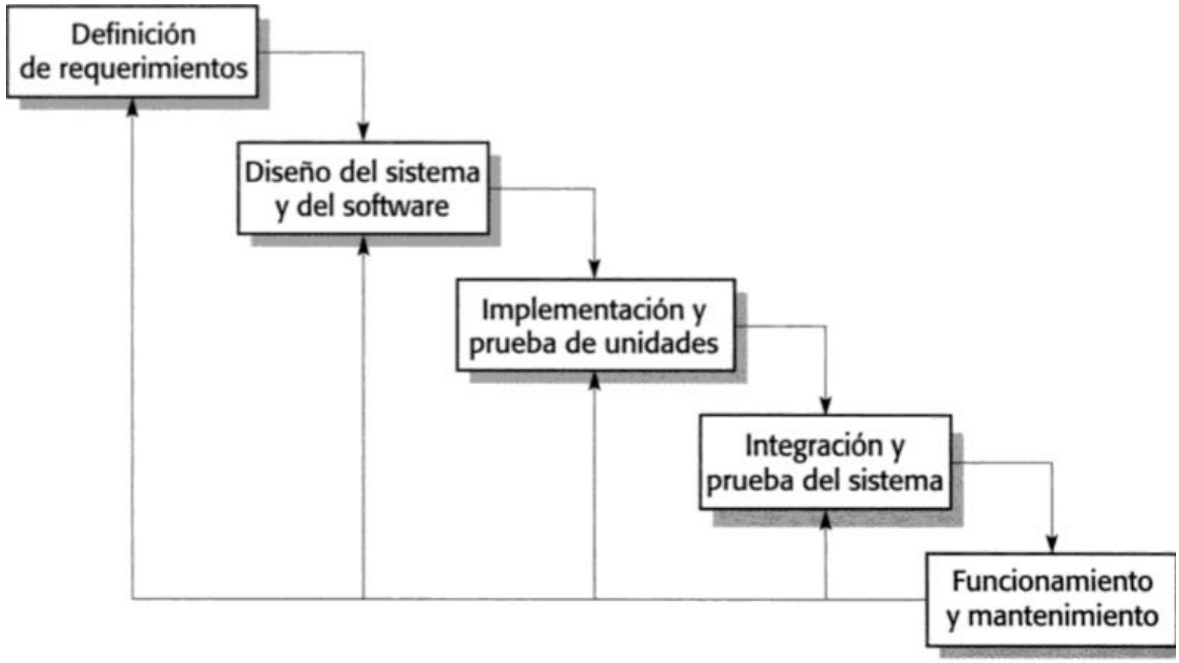


Figura 14. Fases de la metodología en cascada

A continuación, se describe cada fase [71]:

- **Definición de requerimientos:** Se determinan los objetivos, condiciones y servicios a través de una entrevista con el usuario, para posteriormente definir los requisitos.
- **Diseño del sistema y del software:** Se establece la arquitectura del sistema en base a los requisitos, también se representan las abstracciones primordiales del sistema y las relaciones.
- **Implementación y Prueba de unidades:** Se lleva a cabo con la codificación del software y se contrasta que cada método cumpla con sus requerimientos.
- **Integración y Prueba del Sistema:** Los diversos métodos se complementan en uno solo para realizar pruebas con el objetivo de determinar que se cumplan con los requisitos del software, luego se implementa en un entorno para el usuario final.
- **Operación y mantenimiento:** En esta fase se pone en producción el software para su posterior uso. El mantenimiento abarca detectar falencias para corregirlas con la finalidad de mejorar el sistema e integrar nuevas funcionalidades según surjan nuevas funcionalidades.

4.2.12. Tecnologías y métodos

En la actualidad hay diversas herramientas y tecnologías que facilitan el desarrollo de modelos de aprendizaje profundo, como: uso de la GPU en caso de contar con una, frameworks de desarrollo, lenguajes de programación y uso de entornos en la nube. A continuación, se detallan las tecnologías usadas para llevar a cabo con los objetivos del presente TIC:

- **Anaconda**

Es un IDE gratuito y de código abierto que gestiona diversos entornos de desarrollo como Jupyter Notebook, JupyterLab, Spyder y PyCharm. Admite lenguajes de programación como Python y R, incluye diversas bibliotecas relacionadas en el campo de inteligencia artificial lo que facilitan la implementación de códigos, además facilita la gestión de entornos virtuales [72].

- **Google Colab**

Es una herramienta en línea que permite crear proyectos en un cuaderno Jupyter, crear prototipos inteligentes, permite el uso de entornos como: GPU, TPU, pues este tipo de modelos requieren de recursos computacionales altos, por lo tanto, las TPU facilitan este tipo de tareas [73]. Además, no requiere de instalar dependencias y es muy útil para personas enfocadas al campo de la programación en lenguaje Python [74].

- **Python**

Es un lenguaje de programación de alto nivel, es sencillo, tiene una ejecución rápida, es flexible y abarca una gran cantidad de bibliotecas disponibles, lo que facilita el desarrollo; además su sintaxis permite la programación orientada a objetos y el desarrollo de modelos de aprendizaje automático [75]. Puede ejecutarse en cualquier plataforma y tiene una gran capacidad de campos como: desarrollo web, ejecución de scripts, crear juegos, implementar modelos de inteligencia artificial y programas de seguridad [76].

- **PyTorch**

Se trata de una biblioteca de código abierto desarrollado por Facebook. Inc, permite entrenar modelos de aprendizaje profundo, enfocando los recursos computacionales (GPU) para aumentar la capacidad de procesamiento en paralelo y flexibilidad, ya que cuenta con un entorno de ejecución basado en C++ de gran capacidad, permitiendo a los desarrolladores aprovechar la compilación y así evitar la inferencia, además es muy usado en diversas aplicaciones que requieren de minimización de tiempo [77][78].

- **CUDA**

Es una plataforma de computación paralela y un framework de programación implementado por NVIDIA; principalmente se usa para no trabajar con la CPU de la computadora, ya que procesa una sola tarea a la vez, sino usar la GPU de NVIDIA, ya que permite ejecutar varios procesos de manera paralela, lo cual depende de la capacidad en la GPU y poder aumentar el procesamiento de las aplicaciones o cualquier programa que requiere de altos recursos para ejecutarse sin problemas [79].

- **OpenCV**

Se trata de una librería de código abierto iniciada en 1999 por una comunidad enfocada en la Inteligencia Artificial, desarrollada en C++ [80]. Su principal tarea es el procesamiento de imágenes y visión por computadora; por ende, permite aplicar diversas técnicas de

preprocesamiento de imágenes y también facilita la extracción de características de manera fácil en vídeos e imágenes. Se encuentra disponible para Windows, Linux y Mac, comúnmente es usado para apps en tiempo real [81].

- **Método OFAT (One Factor at Time)**

Permite variar una variable a la vez, manteniendo fijas las demás variables; se recomienda usar este método cuando se desea analizar más de dos factores [82]. Su principal objetivo es determinar el impacto de cada valor individual en los resultados y comparar si dicha variable influye positivamente o negativamente en el resultado [83] [84]. No obstante, es que se requiere ejecutar varios experimentos, por lo tanto, se requiere de tiempo y recursos [85].

Especialmente se usa este procedimiento ya que al no contar con una gran capacidad de recursos computacionales no se puede ejecutar de manera secuencial todas las configuraciones, por ende, cada experimento se realiza de manera individual tomando en cuenta el cambio del hiperparámetro que se está analizando para determinar los mejores valores.

- **Flask**

Se trata de un framework de desarrollo web liviano basado en el lenguaje de programación Python [86], se basa en paquetes o módulos lo que facilita la implementación de aplicaciones, además, Flask presenta simplicidad y flexibilidad lo que permite al desarrollador personalizar y ajustar según las necesidades [87]. También, se utiliza para desarrollo de backend y frontend [88]. Las principales dependencias de esta herramienta son: subsistemas de enrutamiento, depuración y web server Gateway interface (WSGI); soporte de plantillas proporcionado por Jinja2; y la línea de comandos proveniente de Click [89].

4.2.13. Dataset

Conocido como un conjunto de datos, pueden ser números, texto, imágenes, audios, videos, entre otros; además son muy usados en proyectos de machine learning, big data, análisis de datos, etc., pues toda la información que contiene un dataset contribuye al aprendizaje de un modelo [90].

Por ejemplo, Kaggle es un repositorio muy conocido que contiene una gran cantidad de datasets y es de acceso libre, facilitando las tareas de recopilación de datos para una tarea en específico. Además, permite alojar conjuntos de datos, códigos y facilita el uso de cuadernos para entrenar modelos de aprendizaje profundo.

4.3. Trabajos Relacionados

A continuación, en la **Tabla 5** se presenta los trabajos relacionados en la tarea de detección del Tizón Tardío en las hojas de papa y ajuste de hiperparámetros en la fase de entrenamiento u otros trabajos similares, los cuales permiten comprender y tener una idea de cómo desarrollar el TIC:

Tabla 5. Trabajos Relacionados

Código	Título	Descripción
TR1	Deep learning hyperparameter's impact on potato disease detection	El presente trabajo experimenta con diferentes configuraciones para evaluar el impacto de los hiperparámetros en una CNN para detectar enfermedades en el tizón tardío. Utiliza el conjunto de datos de PlantVillage con un total de 2 152 imágenes; aplica preprocesamiento de imágenes, como: cambio de tamaño y normalización [0, 1], dividen los datos en un 80:10:10 entrenamiento, prueba y validación. Para el entrenamiento del modelo usan las siguientes configuraciones, primer experimento: batch size (16,32,64,128, 25), optimizador SGD, lr0 0,001 y 100 épocas; segundo experimento: función de activación (ReLU, Leaky ReLU, tanh, PreLU); tercer experimento: optimizador (SGD, Adam, AdaGrad, Adamax, RMSProp). Destacando que el batch size de 32, optimizador Leaky ReLU y la función de activación AdaGrad son las mejores configuraciones para entrenar el modelo, alcanzando una precisión de 91.3%, 97.3% y 98% respectivamente [91].
TR2	Hyperparameter optimization of apple leaf dataset for the disease recognition based on the YOLOv8	El estudio usa YOLOv8 para determinar enfermedades en las hojas de manzana. El dataset comprende 17 470 imágenes; aplican técnicas de aumento de datos (rotación 15°, volteo, contraste y aumento del canal de color) alcanzando un total de 23 750 datos distribuido: 80% entrenamiento, 10% validación y 10% pruebas. Durante la etapa de entrenamiento usa el método OFAT para determinar los mejores hiperparámetros, valores: optimizer (SGD, Adam, Adamax, NAdam, RAdam, RMSProp), learning rate (0.01, 0.0001, 0.001, 0.005, 0.015), batch size (8, 16, 32, 48, 64, 80), weight decay (0.00005, 0.0001, 0.001, 0.005, 0.01, 0.05), momentum (0.859, 0.885, 0.911, 0.937, 0.963, 0.989), drop out (0.0, 0.1, 0.3, 0.5, 0.7) y epochs (200, 300, 400). Utilizan una configuración predeterminada (batch= 16, optimizer= SGD, lr0= 0.01, weight decay= 0.0005, momentum = 0.937, drop out= 0, and epochs= 100), tomando en cuenta el mejor valor para los siguientes experimentos. Determinando batch 48, optimzer SGD, lro 0.01, weight decay 0.0005, momentum 0.937, dorp out 0.0 y 200 epochs, alcanzando una precisión 96.747%, 96.747%, 96.124%, 96.747%, 96.747%, 96.386% y 96.931% respectivamente [83].
TR3	Hyperparameter optimization of YOLOv8 for smoke and	El presente trabajo implementa YOLOv8 para detectar humo e incendios forestales en el cultivo. Usa el dataset etiquetado de 9 796 imágenes, dividido en 70:20:10. En la etapa de entrenamiento usan el método OFAT para experimentar con

Código	Título	Descripción
	wildfire detection: Implications for agricultural and environmental safety	los siguientes hiperparámetros: lr0 (0.01,0.001, 0.0001), epochs (100, 200, 300), batch (16, 32, 64, 128, 256), optimizer (SGD, Adam, AdamW, Adamax, NAdam, RAdam, RMSProp), weight decay (0.00005, 0.0005, 0.005, 0.05). Para lo cual establecen una configuración inicial (lr0= 0.01, optimizer= SGD, batch= 16, weight decay= 0.0005, and epochs= 100), tomando en cuenta los mejores resultados, es decir el mejor valor del hiperparámetro se usa en el siguiente entrenamiento hasta llegar al ajuste de YOLOv8. Determinando la mejor configuración lr0 0.01, optimizer SGD, batch 64, weight decay 0.0005 y epochs 200, alcanzando un 90.9% de precisión, 79.2% recall y 87.7 mAP en todas las clases [43].
TR4	Revolutionizing potato late blight surveillance: uav-driven object detection innovations	El presente trabajo usa modelos de detección de objetos como YOLOv6, YOLOv7, YOLOv8, Faster R-CNN con ResNet-50, VGG16, VGG19, para detectar la zona infectada del tizón tardío en las hojas de papa. El conjunto de datos consta de 2 280 imágenes etiquetadas con" Late Blight"; aplican técnicas de aumento de datos como: rotaciones, zoom, volteos y modificaciones geométricas con parámetros aleatorios; la división de datos es 70% entrenamiento, 10% pruebas y 20% validación. Para el entrenamiento usan una configuración de 100 épocas para todos los modelos, alcanzando una precisión de 93.92% para Faster R-CNN con ResNet-50, Faster R-CNN con VGG19 91.15%, Faster R-CNN con VGG16 91.96%, YOLOv6 84.59%, YOLOv7 84.78% y YOLOV8 87.18%; destacando a YOLOV8 el más eficiente para aplicaciones en tiempo real y Faster R-CNN en cuestión de precisión [92].
TR5	Detection and identification of plant leaf diseases using YOLOv4	El trabajo utiliza YOLOv4 para identificar enfermedades en las hojas de las plantas. Utilizan el conjunto de datos de PlantVillage, que contiene hojas sanas y enfermas de 14 distintas especies; para el etiquetado de imágenes usan LabelImg, donde se delimita la zona en un cuadro. Para el entrenamiento destina un 90% y un 10% validación, integra Darknet el cual posee pesos previamente entrenados y poder optimizar YOLOv4. Configura el tamaño de lote 64, 16 subdivisiones y la tasa de aprendizaje 0.001, alcanzando un 99.99% de precisión y F1-Score obtuvo un 99% [93].
TR6	Plant Disease Detection and Classification Method Based on the Optimized Lightweight YOLOv5 Model	El trabajo usa YOLOv5 para detectar y clasificar enfermedades del maní, para lo cual usan el dataset de PlantVillage, con un total de 3 265 imágenes. Para el etiquetado usan la herramienta LabelImg. El conjunto de datos se dividió en 80:10:10. Para entrenar el modelo integran el submódulo de atención mejorado (IASM) para mejorar la precisión, también usan la estructura Ghostnet para reducir la cantidad de cálculo y la estructura BiFPN para mejorar la eficiencia del modelo; configuran el optimizador Adam y la tasa de aprendizaje 0.0001. Alcanzando una precisión del modelo optimizado en un 11.8% y un 3.98% más altos que el modelo original YOLOv5 (89.75 precisión, 90.25 recall, 9068 F1-Score), obteniendo una precisión 93.73%, recall 92.94% y F1 Score 92.97% [94].

Código	Título	Descripción
TR7	Deep learning pest detection on Indonesian red chili pepper plant based on fine-tuned YOLOv5	El trabajo implementa el modelo YOLOv5s para detectar plagas en la planta de chile, emplean un dataset de 4 994 imágenes; aplica técnicas de aumento de datos como rotación, cortar, brillo, ruido y desenfoque obteniendo un total de 10 683 imágenes. La división del conjunto de datos es 80% entrenamiento, 10% pruebas y 10% validación. En el entrenamiento del modelo se toma en cuenta el siguiente ajuste fino: tamaño del lote 16, tamaño de imagen 1 216, 1 200 épocas, patience 100 y función de optimización SGD. Alcanzando 74% F1-Score con 445 épocas y paciencia 100 [95].
TR8	Advancing common bean (Phaseolus vulgaris L.) disease detection with YOLO driven deep learning to enhance agricultural AI	El trabajo usa diferentes arquitecturas basadas en detección de objetos, YOLOv7, YOLOv8 y YOLO-NAS para detectar enfermedades foliares del frijol, alcanzando un total de 9 500 imágenes tomadas de manera manual. Aplican técnicas de aumento de datos, inversión de imágenes sobre ejes verticales y horizontales, ajuste del brillo de los píxeles. Asignan el 70% entrenamiento, 20% validación y 10% pruebas. En el entrenamiento configuran la versión 7 y 8 con 100 épocas y la versión NAS con 70 épocas. Alcanzando un mAP superior al 95% y 93% recall respectivamente YOLOv7, YOLOv8 y YOLO-NAS, destacando a YOLO-NAS como la mejor opción obteniendo un mAP 97.9% y un recall de 98.8% [60].
TR9	A mobile-based system for maize plant leaf disease detection and classification using deep learning	El trabajo implementa diversas versiones de YOLO (YOLOv3-tiny, YOLOv4, YOLOv5s, YOLOv7s y YOLOv8n), para detectar y clasificar enfermedades en las hojas de maíz, recopilan 2 675 imágenes. Implementa YOLOv3- Tiny con activación leaky RELU, lr0 0.001, batch size 64 y subdivisions 16; YOLOv4, lr0 0.001, activación Mish y batch size 64; YOLOv5s optimzer SGD, momentum 0.937, activación Mish, batch size 64, lr0 0.01 y weight decay 0.005; YOLOv7s optimzer SGD, momentum 0.937, activación leaky RELU, batch size 64, lr0 0.01, weight decay 0.0005, warmup epochs 3 y warmup bias lr 0.1; YOLOV8n optimizer Adam, momentum 0.937, activación SiLU, batch size 64, lr0 0.01, weight decay 0.0005, warmup momentum 3 y warmup bias lr 0.1. Obteniendo una precisión 69.40%, 97.50%, 88.23%, 93.30% y 99.04% respectivamente, demostrando que YOLOv8 es la mejor arquitectura para detectar las enfermedades [58].
TR10	Tea leaf disease detection and identification based on YOLOv7 (YOLO-T)	El trabajo tiene como objetivo identificar enfermedades de las hojas de té, con la red neuronal YOLOv7, para el conjunto de datos recopilan 4 000 imágenes de cinco tipos de enfermedades del té. Para el etiquetado de imágenes emplean la herramienta LabelImg, normalizan las imágenes. Establece la división de los datos: 70% entrenamiento, 20% pruebas y 10% validación. Configuran el número de épocas entre 100 y 150; obteniendo un mAP 98.2% y una precisión de 97.3% [96].
TR11	Comparison Study of Corn Leaf Disease Detection based	El presente trabajo realiza una comparación entre YOLOv5 y YOLOv8n, para detectar enfermedades de las hojas de maíz, recopilan 4 225 imágenes del repositorio de Kaggle, para lo cual sólo etiquetan las imágenes infectadas. Antes de

Código	Título	Descripción
	on Deep Learning YOLO-v5 and YOLO-v8	realizar la división del conjunto de datos aplican técnicas de preprocesamiento como: rotación de 90 grados, redimensionamiento, transformar a RGB y normalizar. Implementan una sola configuración para los dos modelos: optimizador AdamW, tasa de aprendizaje 0.01, épocas 100, batch size 16, tamaño de imagen 416 y función de pérdida BCE, DFL y CIoU. Obteniendo un mAP de 96.5% para YOLOv8x y 90.6% YOLOv5x, 74.1% YOLOv5n, 79.2% YOLOv8n, demostrando que YOLOv8 es la mejor arquitectura para detectar las enfermedades, pero YOLOv5 detecta de manera más rápida las zonas infectadas [97].

La revisión de diversos trabajos relacionados en la detección de enfermedades foliares en los cultivos o enfocados en la temática de la configuración de hiperparámetros, permiten establecer que recursos se pueden usar, modelos, métricas de evaluación y configuraciones para entrenar, además toda esta información es de gran ayuda para llevar a cabo el TIC.

Los trabajos relacionados TR1, TR2, TR4, TR6, TR7, TR8, TR9, TR10, TR11 y TR12 destacan que la configuración de los diversos parámetros permite aumentar el rendimiento de un modelo, pero así mismo hacen hincapié que no siempre generan resultados aceptables, por lo que se debe experimentar hasta encontrar una solución adecuada. De la misma manera se establecen las versiones que se implementan en el TIC, para lo cual se analizan los trabajos relacionados que usan modelos YOLO, destacando YOLOv5 y YOLOv8.

Otro enfoque que ayuda en la creación del dataset, es considerar aplicar técnicas de aumento de datos como: rotar, voltear la imagen horizontal y vertical, agregar ruido gaussiano, desenfocar, cortar, desplazar y aplicar zoom. Además, se establece una idea para la división del conjunto de datos, ya que en cada trabajo aplican diferente división, destacando la separación 80:20. Finalmente facilita determinar las herramientas de etiquetado usadas en los mismos, de las cuales Roboflow, LabelIMG son los que frecuentemente se utilizan, destacando Roboflow como la opción más factible (ver **Tabla 3** comparativa entre las dos herramientas).

5. Metodología

En la presente sección se describen los métodos, técnicas y materiales usados para llevar a cabo el Trabajo de Integración Curricular (TIC). En la sección 5.1 se detalla el área de estudio y en la sección 5.2 se establece el procedimiento para la ejecución de cada objetivo.

5.1. Área de estudio

El presente TIC se llevó a cabo en la “Quinta Experimental La Argelia” de la Universidad Nacional de Loja. En la **Figura 15** se aprecia el mapa del área estudio ubicada en las coordenadas: latitud -4.039391, longitud -79.199796.

Dicha zona es usada por los estudiantes y docentes de la carrera de Agronomía de la Facultad Agropecuaria y de Recursos Naturales Renovables, para realizar prácticas de campo con diferentes cultivos, como la siembra de la papa. De tal manera se realizó un acercamiento con el Ing. Angel Robles docente de la carrera de Agronomía y especialista en enfermedades foliares, en las instalaciones del área de estudio mencionada, para la realizar la recolección de las fotografías.

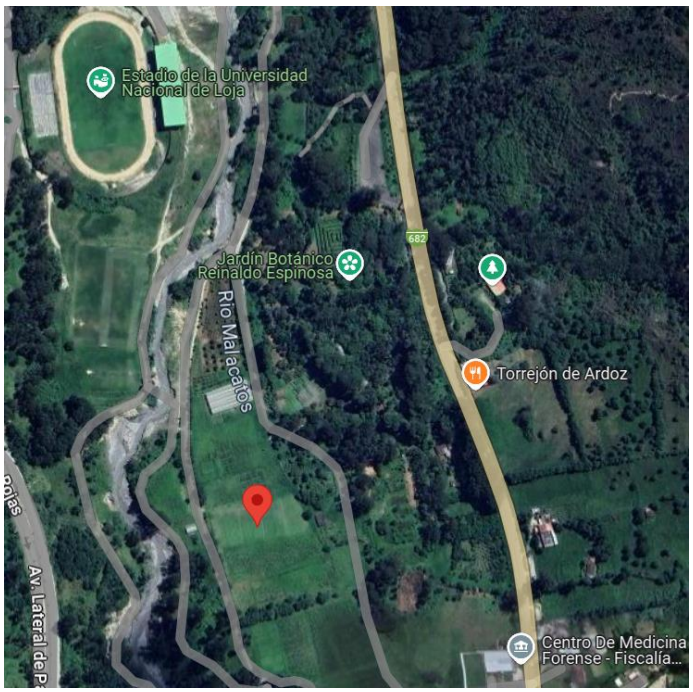


Figura 15. Mapa del área de estudio, Quinta Experimental La Argelia

5.2. Procedimiento

Para cumplir con los objetivos se usó recursos de hardware y software, específicamente tecnologías como: lenguaje de programación python, pytorch, CUDA, ultralytics, roboflow, anaconda, flask y recursos científicos: método experimental, técnicas de

aumento de datos, entrevistas y método One Factor At a Time (OFAT). A continuación, se describe las fases aplicadas de la metodología para proyectos de Machine Learning CRISP-ML(Q) y las tareas necesarias para llevar a cabo los objetivos planteados del presente TIC:

5.2.1. Objetivo 1: Implementar un modelo de detección de objetos para identificar la infestación de las hojas de papa por el tizón tardío utilizando YOLOv5 y YOLOv8

Para abarcar el primer objetivo se ejecutaron las tres primeras fases de la metodología CRISP-ML(Q), que son la comprensión de los datos y negocio, ingeniería de los datos, ingeniería del modelo. En tal sentido son la base para empezar a crear el dataset de la enfermedad del Tizón Tardío, estandarizar, etiquetar, realizar la división del conjunto de datos y entrenar los modelos de YOLOv5, YOLOv8.

5.2.1.1. Fase 1: Comprensión de los datos y negocio

Se inició comprendiendo el método tradicional que emplean los docentes y estudiantes para detectar la enfermedad del Tizón Tardío en las hojas del cultivo de papa, para lo cual se aplicó la técnica de la entrevista al Ing. Agrónomo Angel Robles especialista en enfermedades foliares, con la finalidad de conocer los síntomas o características que se presentan en la hoja y determinar el tiempo requerido para el análisis manual de la enfermedad. Se hizo énfasis que la integración de tecnologías modernas sería de gran ayuda, ya que se automatizará el proceso típico de detección (ver **Anexo 2**).

- **Tarea 1: Recopilación de datos**

Para la recopilación de datos se usó lo repositorios Kaggle[5] y Mendeley Data[6] ya que tiene una amplia variedad de datasets que otros no tienen, además son de acceso libre para cualquier usuario. De tal manera se realizó la búsqueda de conjuntos de datos, con la finalidad de obtener sets de imágenes con la enfermedad del Tizón Tardío en las hojas de papa, mediante el filtrado de palabras claves, late blight, potato diseases.

Se seleccionó los conjuntos que cuenten con fotografías infectadas por enfermedad mencionada, teniendo en cuenta que no estén duplicadas en otros datasets; en Kaggle se obtuvo el conjunto Potato disease img_classif[7], que contiene fotografías con extensión .jpg y Potato Leaf (Healthy and Late Blight)[8] está conformado por imágenes en formato .jpg. Asimismo, en Mendeley Data se consiguió el set de datos Potato Leaf Disease Dataset in

[5] Repositorio Kaggle: https://www.kaggle.com/
[6] Repositorio Mendeley Data: https://data.mendeley.com/
[7] Dataset 1: https://www.kaggle.com/datasets/juliancortes2/potato-disease-img-classif
[8] Dataset 2: https://data.mendeley.com/datasets/v4w72bsts5/1

Uncontrolled Enviroment[9], dicho conjunto se encuentra estructurado en diferentes enfermedades de la papa con imágenes en formato .jpg.

Además, se tomó como base los Trabajos Relacionados (TR) para inferir los dataset empleados en los mismos; los TR1, TR4, TR6, TR7 usaron el conjunto de datos PlantVillage[10], que consta de varias enfermedades incluyendo imágenes acordes a lo requerido (ver sección 6.1.1.1).

- **Tarea 2: Captura de imágenes**

Para realizar la captura de fotografías se tomó en cuenta diferentes condiciones, para asegurar la calidad de las imágenes y construir un set variado; dicho proceso se llevó a cabo por el autor del TIC con la ayuda del Ing. Angel Robles, en el área de estudio Quinta Experimental La Argelia (ver **Figura 15**). Se utilizó el dispositivo móvil Infinix (ver especificaciones en la **Tabla 6**) para realizar la captura; la toma de las imágenes se efectuó desde el 13 de junio hasta el 6 de agosto del 2024, alcanzando un total de 266 fotografías con una dimensión de 3968 x 2976 (ver **Anexo 3,** ver resultados sección 6.1.1.2).

En la **Tabla 6** se presenta un resumen de las especificaciones del dispositivo Infinix Note 30 Pro, usado para la toma de fotografías.

Tabla 6. Especificaciones del dispositivo usado para la recolección de imágenes

Marca del dispositivo móvil	Modelo de cámara	Apertura de cámara	Tamaño (Megapíxeles)	Procesador
Infinix Note 30 Pro	Samsung S5KHM6	f/1.8	108	Helio G99

5.2.1.2. Fase 2: Ingeniería de los datos

La presente fase permitió construir y preparar el conjunto de datos, facilitando establecer la división del dataset, además dicho set es usado para la etapa de entrenamiento y validación.

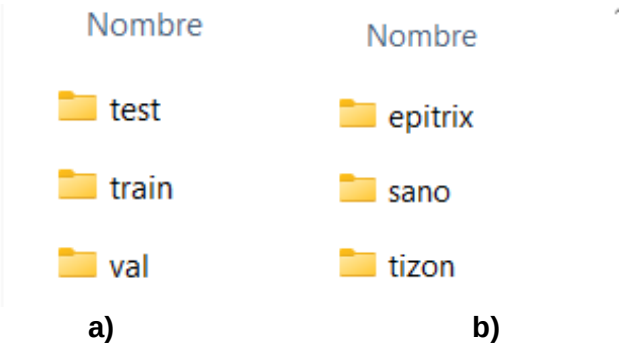
- **Tarea 3: Selección de datos**

Se realizó la selección de los datos de los conjuntos de datos obtenidos en la Fase 1, para lo cual se hizo una exploración de todos los dataset con la finalidad de determinar las imágenes acordes a lo requerido (ver sección 6.1.2.1). En primera instancia se procedió a descartar las carpetas que no son necesarias, ya que los directorios cuentan con imágenes relacionadas a otras enfermedades de tal manera se eligieron las carpetas que presenten el Tizón Tardío, a continuación, se presenta el proceso llevado a cabo:

- Potato disease img_classif está compuesto por 3 directorios (test, train, val), cada una contiene 3 subcarpetas (epitrix, sano, tizon), de las cuales se

[9] Dataset 4: https://data.mendeley.com/datasets/ptz377bwb8/1
[10] Dataset 3: https://www.kaggle.com/datasets/emmarex/plantdisease

seleccionaron las carpetas denominadas tizón, alcanzando un total de 61 imágenes. En la **Figura 16** se muestran los directorios que compone el dataset analizado.



a) **b)**

Figura 16. Dataset Potato disease img_classif. a) Directorios. b) Subdirectorios.

- Potato Leaf (Healthy and Late Blight) se encuentra divido en 2 directorios (Healthy, Late Blight), por lo tanto, se escogió el directorio Late Blight y el otro se descartó debido a que tiene imágenes que no presentan la enfermedad, obteniendo un total de 68 fotografías. En la **Figura 17** se presenta los directorios que posee el conjunto de datos mencionado.

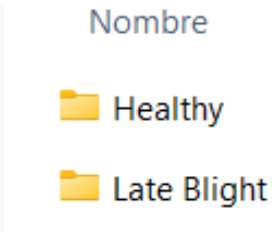


Figura 17. Directorios del dataset Potato Leaf (Healthy and Late Blight)

- PlantVillage contiene 15 directorios, que pertenecen a diversas enfermedades de las plantas, como: papa, tomate, fresa, entre otros. De tal manera se descartó las carpetas que son para otros cultivos y que no presenten la enfermedad, ya que son innecesarias para el TIC; por ende, se eligió la carpeta Potato_Late_blight, consiguiendo 1000 imágenes. En la **Figura 18** se muestran los directorios del set de datos PlantVillage.

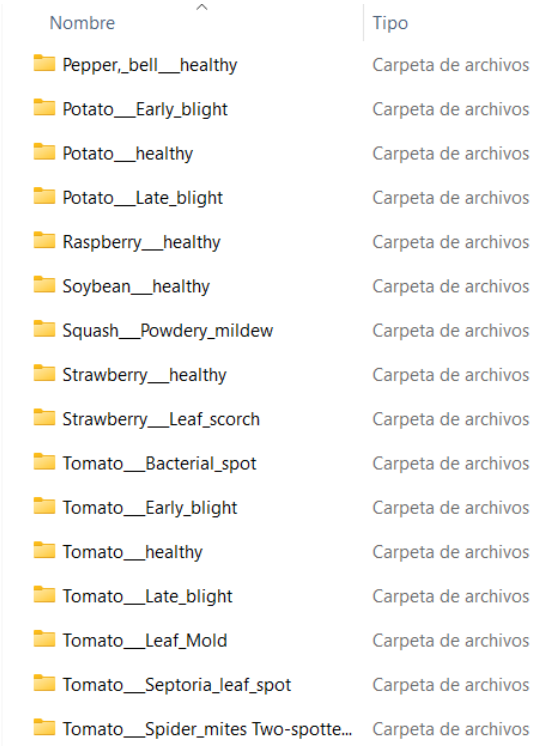


Figura 18. Directorios del dataset PlantVillage

- Potato Leaf Disease Dataset in Uncontrolled Enviroment, está categorizado en: Bacterias, Fungi, Healthy, Nematode, Pest, Phytophthora y Virus. De la misma manera se realizó el mismo proceso ejecutado en los anteriores conjuntos de datos, por lo tanto, se descartó los directorios que no presentan la enfermedad del Tizón Tardío en las hojas de papa y se seleccionó la carpeta Phytophthora, obteniendo un total 347 imágenes. En la **Figura 19** se presentan los directorios que componen el dataset mencionado.

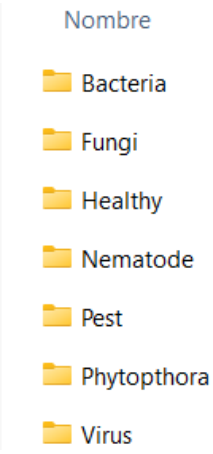


Figura 19. Directorios del dataset Potato Leaf Disease Dataset in Uncontrolled Enviroment

Finalmente se realizó la unión de todas las imágenes seleccionadas de los diferentes conjuntos de datos y del dataset propio, con la finalidad de obtener un solo set de datos,

alcanzado un total de 1741 imágenes para el dataset personalizado con la enfermedad del Tizón Tardío en las hojas papa.

- **Tarea 4: Limpieza de datos**

Para tener una dataset óptimo, se realizó la limpieza de los datos (imágenes) aplicando criterios de inclusión y exclusión, a los conjuntos de datos obtenidos, por lo tanto, se seleccionaron las fotografías que cumplieron con los criterios de inclusión y se descartaron las que no cumplen, obteniendo un total de 1661 fotografías (ver sección 6.1.2.2).

- **Tarea 5: Estandarización**

Para llevar a cabo la tarea se tomó en cuenta la dimensión de entrada de las imágenes de los modelos de YOLOv5 (ver **Tabla 1**) y YOLOv8 (ver **Tabla 2**), destacando que el tamaño de 640 x 640 píxeles es requerido, sin embargo, el conjunto de datos obtenido posee diferentes tamaños de imágenes, como: 2556 x 256, 255 x 255, 1500 x 1500 y 3968 x 2976 píxeles, por lo tanto se usó la herramienta iLoveIMG para realizar la tarea. Dicho proceso fue necesario para manejar un solo tamaño del dataset.

En la **Figura 20** se presenta una muestra del redimensionado de las fotografías, para lo cual se subió las imágenes en la interfaz de la herramienta, limitando el tamaño y posteriormente descargarlas.

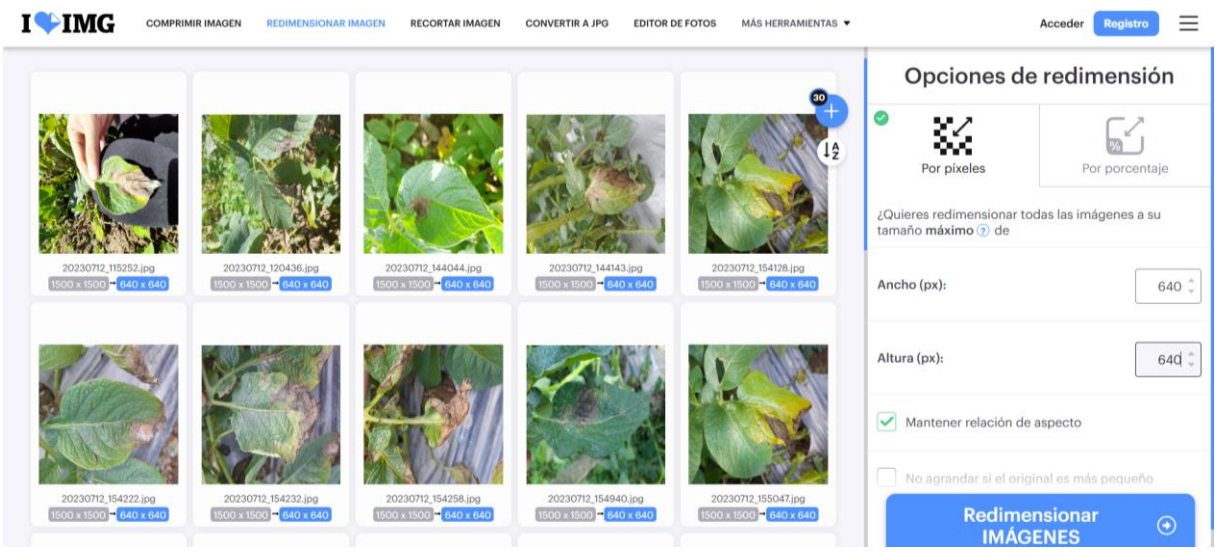


Figura 20. Configuración en iLoveIMG para el redimensionado de imágenes

Además, se estandarizó el nombre del set de datos a un solo formato IMG_# (donde # es el número que va incrementando de manera ascendente), mediante la ejecución de un script en JupyterLab. Las tareas realizadas son necesarias para tener una mejor organización de los datos y facilitar el entrenamiento de los modelos YOLOv5 y YOLOv8 (ver sección 6.1.2.3).

- **Tarea 6: Etiquetado de datos**

El etiquetado (en particular formato YOLO está compuesto por archivos .txt) es necesario para acoplarse a los requerimientos de los modelos YOLOv5 y YOLOv8, facilitando el entrenamiento de los modelos mencionados o cualquier arquitectura basada en la detección de objetos. De tal manera se usó la herramienta web Roboflow[11], la misma ofrece una interfaz intuitiva para realizar etiquetado de las imágenes.

Al momento de etiquetar se tomó en cuenta que hay dos maneras de realizar la tarea mediante un cuadro o en polígono, de tal manera, se analizó la mejor opción para la cual se basó en los trabajos relacionados. En tal sentido se usó la primera opción, debido a que la otra alternativa requiere de mayor esfuerzo y tiempo, pues se debe de dibujar la forma de la zona objetivo.

En la **Figura 21** se presenta el proceso realizado para crear un proyecto, donde se estableció el nombre de la clase ***TizonTardio***, con la tarea detección de objetos. Posteriormente se cargó el dataset en el proyecto creado, donde se dibujó el cuadro delimitador en la zona objetivo y se asignó la etiqueta en dicha área (ver sección 6.1.2.4).

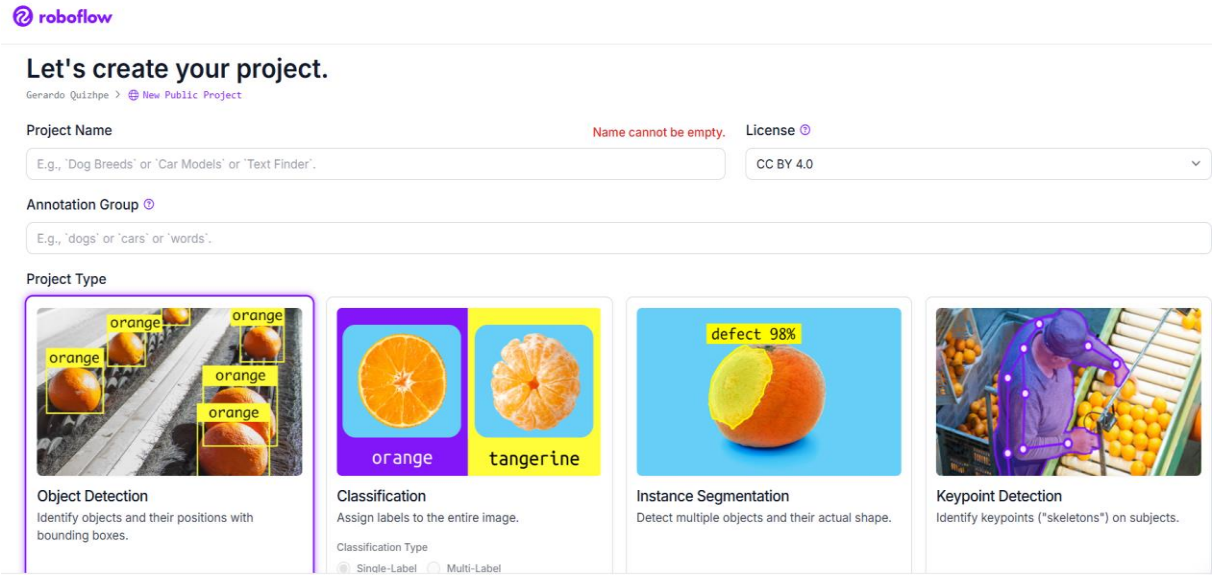


Figura 21. Configuración en Roboflow para crear un proyecto

- **Tarea 7: Aumento de datos**

El aumento de datos se empleó porque permite generar nuevos escenarios en las imágenes, por ende, ayuda al modelo a aprender y mejorar su rendimiento. De tal manera se usó la herramienta Roboflow ya que permitió generar nuevas fotografías etiquetadas, donde se aplicó las siguientes técnicas: voltear verticalmente, agregar ruido (0.5 % de los píxeles) y voltear horizontalmente.

[11] Herramienta web para etiquetar imágenes: https://roboflow.com/

Se realizó la división del conjunto de datos en un 80:10:10, para lo cual el software toma en cuenta el conjunto de entrenamiento para aplicar las técnicas antes mencionadas, generando 3987 imágenes en dicho set, finalmente se alcanzó un total de 4319 fotografías (ver sección 6.1.2.5).

- **Tarea 8: División del dataset**

Es necesario para dividir las imágenes en 3 conjuntos, con la finalidad de facilitar la tarea de entrenamiento y pruebas que realiza el modelo. Para esto los trabajos relacionados TR1, TR2, TR6 y TR7 fueron de apoyo para determinar la proporción más usada, estableciendo la división 80 % entrenamiento, 10 % pruebas y 10 % validación la más común. Una vez establecido la división, se descargó el nuevo dataset generado con el aumento de datos, por lo tanto, se utilizó un script creado en JupyerLab, para realizar la separación del conjunto de datos; distribuyendo 3455 imágenes para el set de entrenamiento, 432 pruebas y 432 validación (ver sección 6.1.2.6).

5.2.1.3. Fase 3: Ingeniería del modelo

La presente fase permitió determinar las versiones de YOLO y los valores para los hiperparámetros, con la finalidad de establecer el modelo que obtiene un rendimiento óptimo para detectar la enfermedad del Tizón Tardío, haciendo énfasis las métricas precisión, recall y mAP (0.50, 050:0.90).

- **Tarea 9: Selección de los modelos**

Se realizó una revisión de las versiones de YOLO implementadas en los trabajos relacionados (TR), con el fin de determinar los modelos que lograron un rendimiento óptimo en las métricas de precisión y mAP@0.50. Destacando las versiones de YOLOv5 y YOLOV8, en específico la distribución small (s), ya que alcanzaron resultados eficaces en las métricas mencionadas, pues para realizar el entrenamiento no se requiere de GPUs superiores a 4GB, ni RAMs con más de 16 GB (ver sección 6.1.3.1).

- **Tarea 10: Entrenamiento de los modelos**

Se realizó una exploración en los trabajos relacionados (TR) para determinar los hiperparámetros y valores para los mismos; destacando el learning rate, optimizer, batch size, weigth_decay y epochs.

En la **Figura 22** se presenta el flujo que se realizó para entrenar y ajustar los hiperparámetros en las distribuciones de YOLOv5s y YOLOv8s; primero se seleccionó el modelo que se va a implementar, luego se configuró los valores iniciales para los hiperparámetros; se aplicó el método OFAT para variar un parámetro a la vez, posteriormente se analizó el rendimiento del modelo con las métricas obtenidas, con la finalidad de determinar el valor que generó resultados óptimos, luego se seleccionó dicho valor, para posteriormente

usarlo en la siguiente iteración, hasta finalizar con todos valores definidos para cada hiperparámetro. El resultado del flujo fue el ajuste de los hiperparámetros del modelo (ver sección 6.1.3.2).

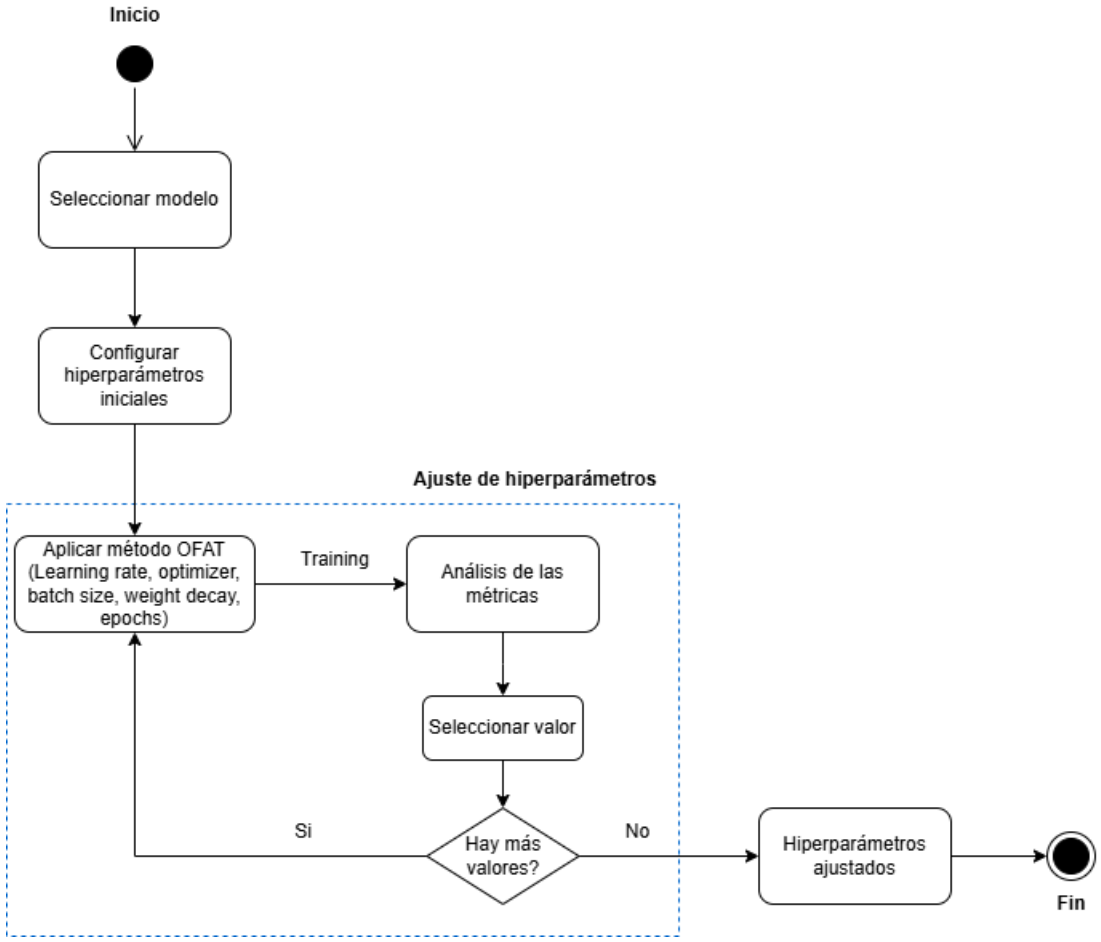


Figura 22. Flujo de trabajo propuesto para el ajuste de hiperparámetros

Se usó la guía de Ultralytics para el entrenamiento de YOLOv5s[12] y YOLOv8s[13], para lo cual se utilizó el entorno de desarrollo JupyerLab, con la finalidad de crear y ejecutar los scripts, donde se configuró los hiperparámetros y se utilizó el archivo data.yaml el cual contiene las rutas de los conjuntos train, valid, y test (división del dataset etiquetado 80 % entrenamiento, 10% validación, 10% pruebas).

- **Tarea 11: Documentar los experimentos del modelo**

Se analizó la influencia de los hiperparámetros, learning rate, optimizer, batch size, weigth_decay y epochs en el rendimiento de YOLOv5s, YOLOv8s, haciendo énfasis en las métricas de precisión, recall, mAP (0.50, 0.50:0.90) obtenidas en cada modelo, con la finalidad

[12] Guía para entrenar YOLOv5: https://docs.ultralytics.com/es/models/yolov5/
[13] Guía para entrenar YOLOv8: https://docs.ultralytics.com/models/yolov8/

de determinar los mejores valores para los parámetros establecidos en las dos versiones de YOLO (ver sección 6.1.3.3).

5.2.2. Objetivo 2: Evaluar el modelo mediante pruebas de validación al identificar el tizón tardío en las hojas de papa

5.2.2.1. Fase 4: Pruebas y validación del modelo

En esta fase se analizó el rendimiento de YOLOv5s y YOLOv8s, en el conjunto de pruebas, con la finalidad de determinar la eficacia de los modelos mencionados al detectar la enfermedad del Tizón Tardío en nuevas imágenes.

- **Tarea 1: Evaluación de los modelos con el conjunto de pruebas**

Se evaluó YOLOv5s y YOLOv8s utilizando el conjunto pruebas, que no fue empleado en la etapa de entrenamiento ni validación. Dicho set consta de 432 imágenes con un total de 812 áreas etiquetadas, lo que permitió determinar la eficacia de los modelos para detectar el Tizón Tardío en datos que no ha visto, para lo cual se tomó en cuenta las métricas de precisión, mAP (0.50 y 0.50:0.90) y recall. Destacando YOLOv8s como el mejor modelo para identificar la enfermedad en las hojas del cultivo de papa, por ende, se usó para crear el prototipo (ver sección 6.2.1.1).

- **Tarea 2: Evaluar el modelo en un entorno simulado**

Se desarrolló un prototipo web para evaluar el modelo (YOLOv8s) en un entorno simulado, con el objetivo de facilitar el uso a un usuario final; el proceso de desarrollo del prototipo se basó en la metodología para desarrollo de software cascada, enfocado en las fases: Definición de requerimientos, Diseño del software e Integración y pruebas.

En primera instancia se realizó una entrevista (ver **Anexo *2***) al Ingeniero Angel Robles con la finalidad de obtener los requisitos necesarios para el prototipo; posteriormente se creó el prototipo que cumple con los requisitos definidos que son cargar imágenes, eliminar y ver resultado de la predicción. Finalmente se evaluó el rendimiento del modelo.

Para lo cual se recolectó imágenes de hojas de papa en diferentes condiciones, obtenido un total de 110 imágenes, de las cuales 55 presentan Tizón Tardío, denominadas tizon_# y 55 corresponden a hojas sanas identificadas como sana_#, donde # es el número que empieza en 1 e incrementa en uno.

En la evaluación participó el Ing. Angel Robles, docente de la carrera de Agronomía, quien analizó cada imagen y determino si están sanas o infectadas por Tizón Tardío, para posteriormente comparar y corroborar los resultados con las predicciones realizadas por el modelo (ver sección 6.2.1.2).

5.2.3. Recursos

5.2.3.1. Recursos científicos

Para el desarrollo del presente TIC se aplicaron las siguientes técnicas:

- **Método experimental**

Permitió aplicar los diferentes conocimientos adquiridos a lo largo de la carrera, experimentar con varias configuraciones de hiperparámetros durante la etapa de entrenamiento. Por otra parte, facilitó la adaptación de las fases de la metodología CRISP-ML(Q) como: comprensión de los datos y negocio, ingeniería de los datos, ingeniería del modelo, pruebas y validación del modelo.

- **Entrevista**

Se utilizó para obtener información necesaria acerca de las posibles soluciones que se pueden dar al problema identificado. Por lo tanto, se aplicó una entrevista referente al campo técnico (ver **Anexo 1**), para conocer ideas de que modelo es la mejor opción, que tipo de técnicas se recomienda aplicar en los datos y que tipo de configuraciones se debe emplear en las arquitecturas de aprendizaje profundo. También se aplicó una entrevista al docente especialista en enfermedades foliares (ver **Anexo 2**), donde se conoció el método tradicional para detectar el Tizón Tardío y el tiempo que toma el análisis, además de tener una idea de cómo se desarrolla la enfermedad en las hojas de las plantas.

- **Técnicas de aumento de datos**

Estas técnicas permitieron generar nuevas imágenes con diferentes escenarios. Varios trabajos relacionados destacan que el aumento de datos ayuda a mejorar el rendimiento de un modelo de aprendizaje profundo. La aplicación de estas técnicas permitió crear una gran variedad de fotografías del conjunto inicial (1314) a 4136 en formato .jpg.

- **Método One Factor at a Time (OFAT)**

Se usó para realizar la configuración de los hiperparámetros, donde se mantuvo constantes todos los parámetros excepto uno (se empezó con learning rate y se finalizó con las epochs), el mismo se varió en cada experimento, luego se tomó el valor que generó resultados óptimos, con la finalidad de emplearlo en el siguiente entrenamiento. Además, permitió analizar cómo influyó dichos valores en el rendimiento de YOLOv5s y YOLOv8s.

5.2.3.2. Recursos tecnológicos

Para el desarrollo del TIC se emplearon los siguientes recursos:

- **Python**

Se usó como el lenguaje de programación base, pues permite ejecutar los scripts para entrenar las versiones de YOLO; en específico se usó la versión de python 3.12.4.

- **Anaconda**

Facilitó las herramientas JupyterLab o Jupyter Notebooks que sirvieron como un gestor de código y de compilación. Se usó el entorno de desarrollo Jupyerlab para programar los scripts de entrenamiento con las respectivas configuraciones de los modelos, implementar la estandarización y división de datos.

- **Roboflow**

Herramienta web que permitió el etiquetado de datos mediante la asignación de una clase y un cuadro delimitador en la zona objetivo de las imágenes. Además, permitió aplicar el aumento de datos (voltear horizontalmente y verticalmente, agregar ruido de 0.5%).

- **Pytorch – CUDA**

Librerías necesarias que permitieron la configuración de la GPU del computador y hacer uso de la misma en la ejecución del entrenamiento de los modelos de aprendizaje profundo sin inconvenientes; la versión usada de Pytorch 2.4.1 y CUDA 12.6.

- **Ultralytics**

Biblioteca que permitió descargar los modelos preentrenados: YOLOv5s y YOLOv8s; la versión usada en el TIC es 8.2.64.

- **Flask**

Framework que facilitó las herramientas necesarias para implementar el frontend y backend del prototipo web, gestionar rutas, procesar datos.

- **Visual Studio Code**

Editor de código que sirvió como el entorno de desarrollo para organizar los directorios, codificar y depurar el código del prototipo web.

5.2.3.3. Participantes

El presente TIC fue llevado a cabo por los siguientes está involucrados:

- Gerardo Manuel Quizhpe Chocho como el autor del TIC, que llevó a cabo todas las tareas para culminar y cumplir los objetivos planteados.
- Ingeniera María del Cisne Ruilova Sánchez, Mg. Sc, como directora del TIC y encargada de las respectivas revisiones y correcciones.
- Ingeniero Angel Rolando Robles Carrión PhD, como especialista en el área de enfermedades foliares y docente de la carrera de Agronomía que facilitó información básica acerca del Tizón Tardío.

6. Resultados

En la presente sección se detallan los resultados por cada objetivo específico planteado y tareas. En la sección 6.1 se presentan la comprensión de los datos, ingeniería de los datos e ingeniería del modelo, los cuales permitieron crear, preparar el conjunto de datos y entrenar YOLOv5s, YOLOv8s mediante la configuración de los hiperparámetros. En la sección 6.2 se presentan las pruebas y evaluación del modelo usando el conjunto de pruebas y tomando en cuenta las métricas de rendimiento como la precisión y mAP.

6.1. Objetivo 1: Implementar un modelo de detección de objetos para identificar la infestación de las hojas de papa por el tizón tardío utilizando YOLOv5 y YOLOv8

6.1.1. Fase 1: Comprensión de los datos y del negocio

6.1.1.1. Tarea 1: Recopilación de datos

- **Búsqueda de datasets en repositorios Kaggle y Mendeley Data**

Los conjuntos de datos se obtuvieron de los repositorios Kaggle[14] y Mendeley Data[15] mediante la búsqueda de datasets con imágenes que presenten el Tizón Tardío. A partir de los resultados obtenidos en la búsqueda, se seleccionó los datasets Potato disease img_classif, Potato Leaf (Healthy and Late Blight) y Potato Leaf Disease Dataset in Uncontrolled Enviroment.

- **Datasets utilizados en los trabajos relacionados**

Se tomó como referencia los trabajos relacionados (ver sección 4.3) con la finalidad de determinar los sets de datos empleados. De tal manera TR1, TR4, TR6, TR7 usaron el dataset PlantVillage.

En la **Tabla 7** se presentan los resultados obtenidos luego de identificar los datasets que fueron usados en los trabajos relacionados y conseguidos en la búsqueda; a continuación, se muestra el nombre del conjunto de datos, criterio de selección del porque se seleccionó dichos sets y la fuente donde se encuentran almacenados.

[14] Repositorio Kaggle: https://www.kaggle.com/
[15] Repositorio Mendeley Data: https://data.mendeley.com/

Tabla 7. Datasets seleccionados

Nombre	Criterio de selección	Fuente
Potato disease img_classif	Dataset variado con respecto a la planta.	Kaggle
Potato Leaf (Healthy and Late Blight)	El dataset posee imágenes con la enfermedad en diferentes partes.	Kaggle, Mendeley Data
PlantVillage	La mayoría de trabajos relacionados (TR1, TR4, TR5, TR8, TR9, TR10, TR11) utilizaron el set.	Kaggle
Potato Leaf Disease Dataset in Uncontrolled Enviroment	Dataset con imágenes que están en diferentes condiciones.	Mendeley Data

6.1.1.2. Tarea 2: Captura de datos del dataset propio

A partir de la captura de fotografías realizadas por el autor del TIC con apoyo del Ing. Angel Robles, en la Quinta Experimental La Argelia (ver sección 5.1 área de estudio), bajo condiciones referentes a clima, distancia, ángulo y momentos del día, se logró un conjunto de datos de 266 fotografías. El proceso completo se detalla en el **Anexo 3**.

En la **Tabla 8** se presentan las condiciones consideradas para la captura de imágenes.

Tabla 8. Condiciones para la captura de imágenes

Condiciones	**Distancia:** mínimo 5 cm y máximo 15 cm.
	Condiciones climáticas: días soleados, lluviosos y nublados.
	Momentos del día: mañana, mediodía y tarde.
	Diferentes ángulos: frontal, lateral y superior.

6.1.2. Fase 2: Ingeniería de los datos

6.1.2.1. Tarea 3: Selección de los datos

- **Recopilación de imágenes**

En la recolección de imágenes, se realizó la selección de los directorios que contienen fotografías que presenten el Tizón Tardío en las hojas de papa. A continuación, se presenta los resultados obtenidos, por cada dataset seleccionado:

- **Potato disease img_classif;** se obtuvo un total de 61 imágenes, dicho conjunto presenta fotografías con las siguientes características: cubre un área más extensa del plantaje, fotografías con diferentes ángulos, en ciertos casos el fondo se encuentra desenfocado, resaltando la zona objetivo y constan de un fondo que incluye tierra, vegetación u otras hojas de la planta. En la **Figura 23** se presenta una muestra de las imágenes del dataset mencionado.

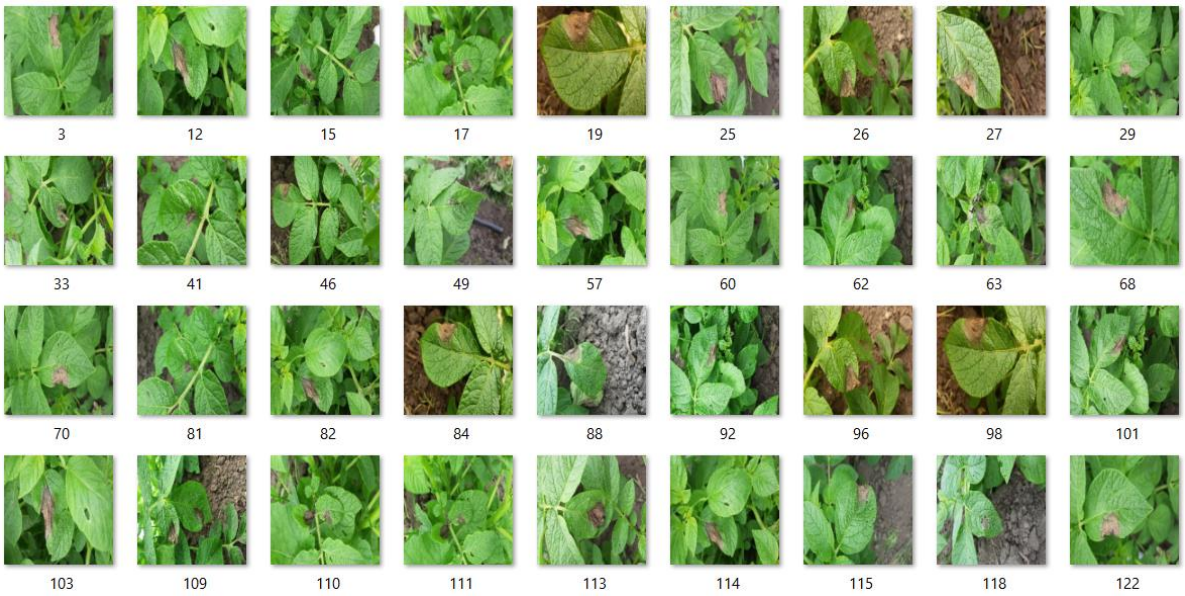


Figura 23. Muestra de imágenes del dataset Potato disease img_classif

- **Potato Leaf (Healthy and Late Blight):** se alcanzó un total de 67 fotografías, que presentan las siguientes particularidades: presentan una hoja en específico, fondo de color blanco, imágenes en diferentes ángulos. En la **Figura 24** se presenta una muestra del dataset indicado.

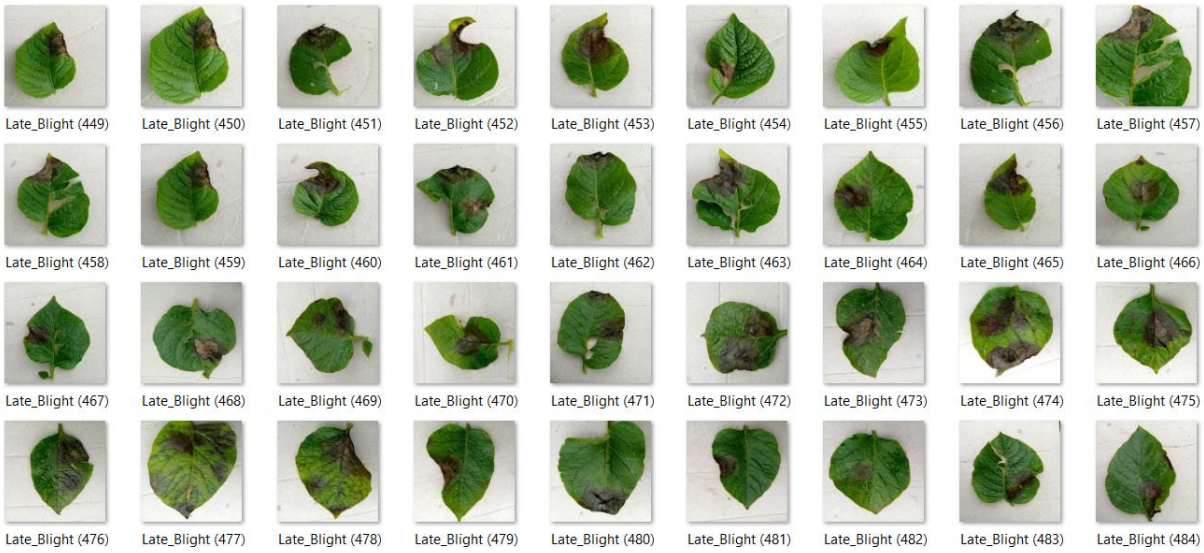


Figura 24. Muestra de imágenes del dataset Potato Leaf (Healthy and Late Blight)

- **PlantVillage:** se consiguió un total de 1000 imágenes con las siguientes peculiaridades: fotografías con diferentes ángulos y ubicaciones de la enfermedad, constan de un fondo gris, muestras de la hoja de papa en específico. En la **Figura 25** se presenta una muestra del dataset descrito.

Figura 25. Muestra de imágenes del dataset PlantVillage

- **Potato Leaf Disease Dataset in Uncontrolled Enviroment**, se obtuvo un total 347 imágenes que presentan las siguientes características: fotos que abarcan más el área de la planta de la papa, muestran varias zonas infectadas por el Tizón en las hojas, fotografías con diferentes ángulos y posiciones, incluyen un fondo donde se aprecia la tierra. En la **Figura 26** se presenta una muestra del dataset descrito.

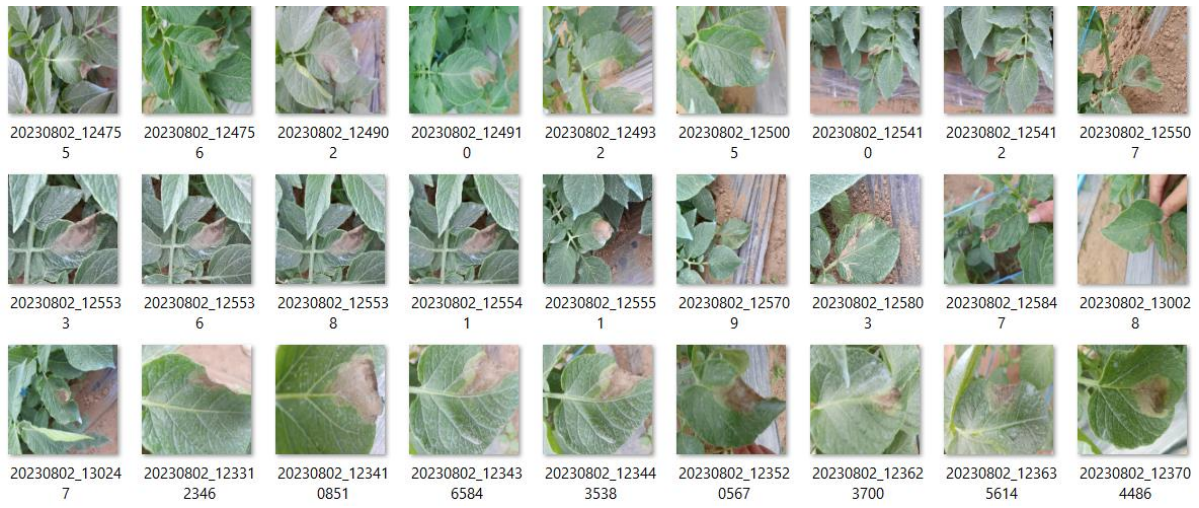


Figura 26. Muestra de imágenes del dataset Potato Leaf Disease Dataset in Uncontrolled Enviroment

- **Dataset Propio:** se alcanzó un total de 266 fotografías con las siguientes características: imágenes con diferentes zonas infectadas por la enfermedad, fotos tomadas en diferentes ángulos y horas del día. En la **Figura 27** se presenta una muestra de imágenes que forma parte del dataset propio, obtenido a partir del proceso de la captura fotográfica.

Figura 27. Muestra de imágenes del dataset propio

Luego de seleccionar las imágenes requeridas de los diferentes datasets y haber realizado la captura de fotografías en el área de estudio, se logró nuevas versiones por cada set de datos. En la **Tabla 9** se presenta los datasets finales seleccionados, destacando el nombre, especificaciones (formato, dimensión), año de publicación y tipo de acceso.

Tabla 9. Resumen de los datasets seleccionados

Nombre	Especificaciones	Imágenes	Año	Acceso
Potato disease img_classif[16]	255 x 255 píxeles Formato: JPG	61	2021	Libre
Potato Leaf (Healthy and Late Blight)[17]	256 x 256 píxeles Formato: JPG	67	2020	Libre
PlantVillage[18]	255 x 255 píxeles Formato: JPG	1000	2019	Libre
Potato Leaf Disease Dataset in Uncontrolled Enviroment[19]	1500 x 1500 píxeles Formato: JPG	347	2023	Libre
Dataset Propio[20]	3968 x 2976 píxeles Formato: JPG	266	2024	-
Total		1741		

[16] Dataset_1: https://drive.google.com/drive/folders/1uJ12WwXnp_do3gCN9jcigZz66h2ipCs0?usp=sharing
[17] Dataset_2: https://drive.google.com/drive/folders/1DfaR2oAfEe1SnPxRnMnw6xFTS9dVRFCE?usp=sharing
[18] Dataset_3: https://drive.google.com/drive/folders/1R7Ya6yS2ZYbGGzivo8CTuhwdQeVib3BM?usp=sharing
[19] Dataset_4: https://drive.google.com/drive/folders/1_Pn2Z0JJTsuQkkuj17oRMmMBQsoeJdwK?usp=sharing
[20] Dataset_5: https://drive.google.com/drive/folders/1YHDhwYUWk3VXh70yn42S7AoAqGub1KGL?usp=sharing

- **Creación del dataset personalizado de la enfermedad del Tizón Tardío en las hojas de papa**

De acuerdo a los datasets establecidos en la **Tabla 9** se unió dichos sets en uno solo, obteniendo un conjunto de datos personalizado[21] que posee un total de 1741 imágenes, dicho set se utilizó para aplicar las técnicas de limpieza y estandarización.

6.1.2.2. Tarea 4: Limpieza de datos

A partir del dataset personalizado obtenido se realizó la limpieza de los datos (imágenes) por el autor del TIC, mediante la aplicación de criterios de inclusión y exclusión detalladas en la **Tabla 10**, con el objetivo de mejorar la calidad del dataset.

En la **Tabla 10** se presenta los criterios de inclusión y exclusión establecidos para la limpieza de las imágenes.

Tabla 10. Criterios de inclusión y exclusión para la limpieza de datos

Criterios de inclusión	Criterios de exclusión
• Imágenes que presentan la enfermedad del Tizón Tardío en las hojas. • Imágenes con un tamaño mayor a 250 píxeles. • Imágenes con formato .jpg.	• Imágenes duplicadas. • Imágenes pixeladas. • Imágenes a blanco y negro.

El número inicial del dataset personalizado contaba con 1741 imágenes, al realizar la limpieza se descartaron 80, obteniendo un nuevo conjunto de datos[22], con un total de 1661 fotografías.

En la **Tabla 11** se presenta el resultado obtenido luego de aplicar la limpieza de datos, alcanzando un total de 1661 fotografías.

Tabla 11. Resultados de la limpieza de datos

Dataset Personalizado	1741
Descartadas	80
Total	1661

6.1.2.3. Tarea 5: Estandarización

- **Redimensionamiento de imágenes**

Se usó la herramienta web iLoveIMG[23] para redimensionar el conjunto de datos a un tamaño de 640 x 640 píxeles, pues es la dimensión de entrada que sugiere YOLOv5 y

[21] Dataset personalizado V1: https://drive.google.com/drive/folders/18MvULSl_Qq3j44xtrZiOhr6wTWcq1NjG?usp=sharing
[22] Dataset limpio V2: https://drive.google.com/drive/folders/1hbjpKmgPHqC5fPaIBhfFSjXvWFPRLMMC?usp=sharing
[23] Herramienta usada para la redimensión: https://www.iloveimg.com/es/redimensionar-imagen

YOLOv8, con la finalidad de tener un dataset[24] que posea el tamaño adecuado para el entrenamiento de los modelos mencionados.

En la **Figura 28** se presenta el resultado luego de redimensionar las imágenes a un tamaño de 640 x 640 píxeles.

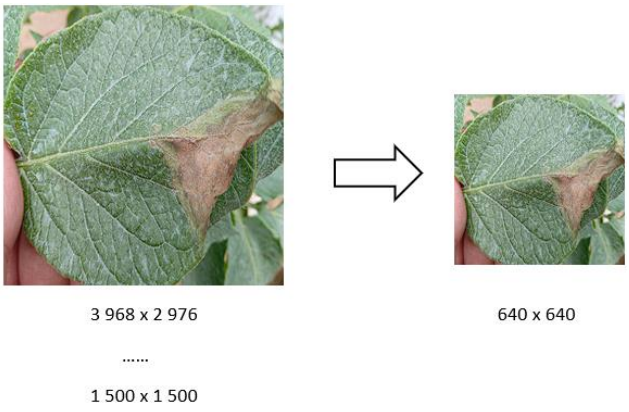


Figura 28. Redimensionamiento de imágenes

- **Estandarización del nombre de las imágenes**

Una vez establecido el conjunto de datos y aplicado la redimensión, se estandarizó el nombre para todas las imágenes, con la finalidad de tener una mejor organización del dataset. En la **Figura 29** se presenta el script ejecutado en el entorno de desarrollo JupyterLab, para establecer un solo nombre de todas las fotografías del dataset, manejando la estructura IMG_#, incrementando en uno, donde # es el número que se asigna a la etiqueta definida con la extensión .jpg y posteriormente se guardan en la carpeta de destino.

```
def convert_and_save_images_to_jpg(src_directory, dst_directory):
    # Crea el directorio de destino si no existe
    os.makedirs(dst_directory, exist_ok=True)

    for count, filename in enumerate(os.listdir(src_directory)):
        # Nombre del archivo
        new_name = f"IMG_{count + 1}.jpg"

        # Obtener la ruta completa del directorio de origen y destino
        src_path = os.path.join(src_directory, filename)
        dst_path = os.path.join(dst_directory, new_name)

        try:
            # Abrir el archivo original y convertir a .jpg
            with Image.open(src_path) as img:
                rgb_img = img.convert('RGB')
                rgb_img.save(dst_path, 'JPEG')

        except Exception as e:
            print(f"Failed to convert {src_path}: {e}")
```

Figura 29. Script para estandarizar el nombre y formato de las imágenes

[24] Dataset redimensionado V3: https://drive.google.com/drive/folders/1ffCwU7vTB7-ex7gnpxEiZMciGt7jCMZV?usp=sharing

En la **Figura 30** se presenta una muestra del resultado obtenido del conjunto de datos[25] luego de haber aplicado el script anteriormente descrito, las imágenes siguen el formato IMG_1 y así sucesivamente de manera ascendente hasta la última fotografía del dataset.

Figura 30. Muestra de imágenes con los nombres estandarizados del dataset

6.1.2.4. Tarea 6: Etiquetado de datos

Se utilizó el conjunto de datos estandarizado que posee un total de 1661 imágenes; para realizar la tarea de etiquetado se realizó una comparativa entre LabelImg y Roboflow (ver **Tabla 3** sección de Marco Teórico), destacando la segunda opción ya que permitió realizar la división del dataset y aplicar técnicas de aumento de datos. El etiquetado se realizó de manera manual donde se asignó el nombre de la clase ***TizonTardio***.

En la **Figura 31** se presenta el resultado de cómo se aplicó el etiquetado a la imagen (IMG_1333) que compone el conjunto de datos, donde se dibujó un cuadro delimitador en la zona objetivo y la clase ***TizonTardio*** establecida.

[25] Dataset estandarizado V4: https://drive.google.com/drive/folders/1LvmgcH9UNoc3tXVxbYBEqQbDVE-bKl5T?usp=sharing

Figura 31. Resultado del etiquetado de la imagen IMG_1333 generado por Roboflow

En la **Figura 32** se presenta el resultado obtenido del etiquetado de la imagen (IMG_1333), donde se generó un archivo con extensión .txt de la fotografía, con las coordenadas del cuadro delimitador y el número de clase (en este caso 0 por que se maneja solo una clase), obteniendo el dataset con las imágenes etiquetadas[26].

```
0 0.34375 0.30546875 0.48984375 0.2359375
```

Figura 32. Archivo txt generado para la imagen etiquetada IMG_1333

6.1.2.5. Tarea 7: Aumento de datos

La finalidad de esta tarea es aumentar el número de imágenes en diferentes contextos permitiendo mejorar la calidad del conjunto de datos y que el modelo entrenado con dichos datos sea capaz de detectar la enfermedad sin importar las condiciones en la que se encuentre. Para el presente caso de estudio se aplicó: voltear verticalmente, agregar ruido (0.5 % de los píxeles) y voltear horizontalmente.

El aumento de datos se realizó en Roboflow, ya que permitió generar nuevas imágenes etiquetadas, ayudando a reducir el tiempo de etiquetado con las nuevas fotografías, por lo tanto, se empleó la división del dataset en 80:10:10, ya que Roboflow aplica las técnicas antes mencionadas al set de entrenamiento, alcanzando un total de 3987 imágenes en dicho conjunto, de tal manera se obtuvo el dataset final[27] que posee una cantidad de 4319 fotografías.

[26] Dataset etiquetado V5: https://universe.roboflow.com/gerardo-quizhpe/dataset-tizontardio/
[27] Dataset con aumento de datos final V6: https://drive.google.com/drive/folders/1RC_HWNGETpSFZCdYlRCsJCLggnMo0PG5?usp=sharing

En la **Figura 33** se visualiza el resultado obtenido de haber aplicado las técnicas de aumento de datos a la imagen IMG_1525 utilizando Roboflow.

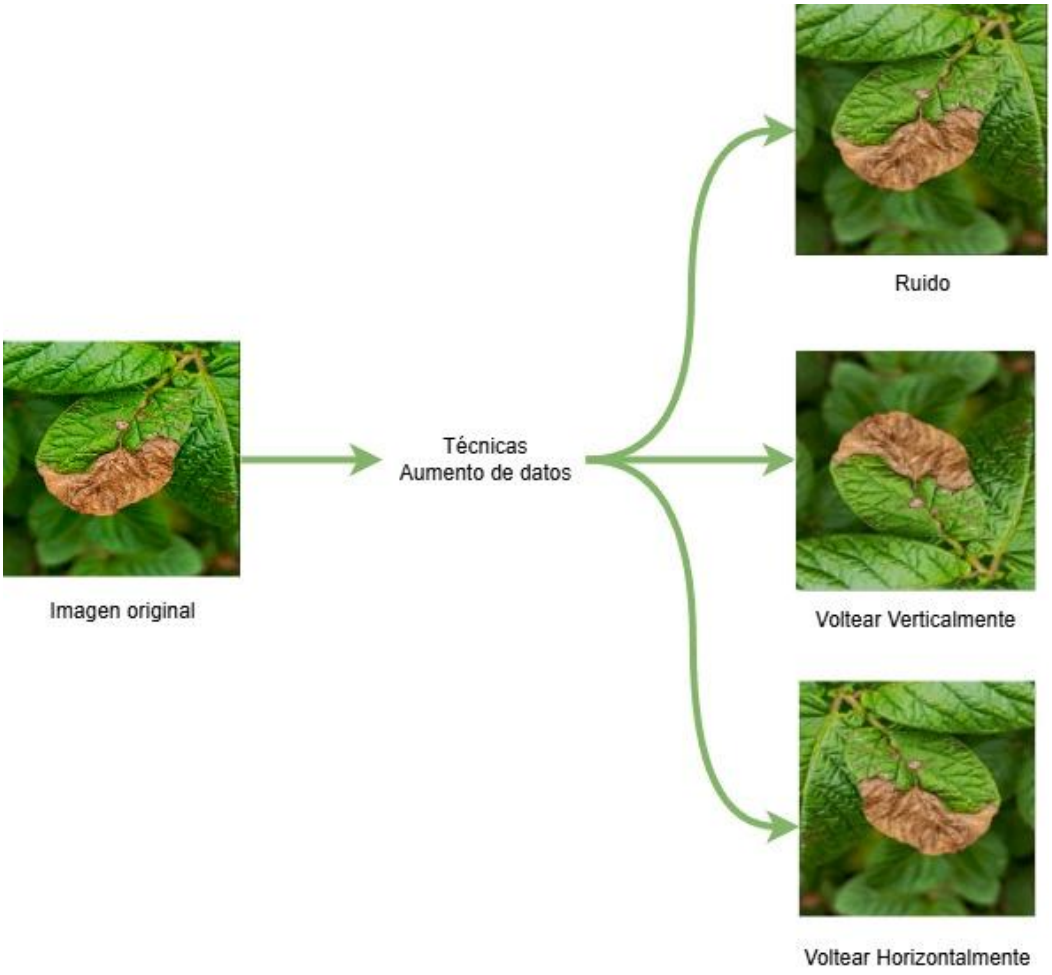


Figura 33. Técnicas de aumento de datos

6.1.2.6. Tarea 8: División del dataset

Una vez creado el conjunto de datos etiquetado y aplicado las técnicas de aumento de datos se realizó la división del mismo, pues en la anterior división se modificó tras aplicar el aumento de datos, por lo tanto, se descargó la nueva versión generada. Luego se aplicó la proporción 80:10:10 ya que fue la división más usada en los trabajos relacionados (TR1, TR2, TR6 y TR7), por ende, dicha división se aplicó al set de datos final.

En la **Figura 34** se presenta el script ejecutado en JupyterLab, para realizar la separación del conjunto de datos, el código obtiene todas las imágenes y etiquetas, luego compara las imágenes y etiquetas con la finalidad de asegurarse de que coincidan, seguidamente se realizó la división del dataset, donde se destina un 80% entrenamiento, 10% pruebas y 10% validación; finalmente guarda los resultados en los respectivos directorios (train, test, valid).

```python
# Obtener lista de todas las imágenes y etiquetas en el directorio
all_images = os.listdir(images_dir)
all_labels = os.listdir(labels_dir)

# Asegurarse de que la lista de imágenes y etiquetas coincidan
all_images.sort()
all_labels.sort()
assert len(all_images) == len(all_labels)

# Dividir las imágenes y etiquetas en conjunto de entrenamiento y el resto (validación + prueba
train_images, temp_images, train_labels, temp_labels = train_test_split(
    all_images, all_labels, test_size=0.2, random_state=42)  # 80% train, 20% (test, valid)

# Dividir las imágenes y etiquetas restantes en validación y prueba
valid_images, test_images, valid_labels, test_labels = train_test_split(
    temp_images, temp_labels, test_size=0.5, random_state=42)  # 50% valid, 50% test (10% cada

# Función para copiar imágenes y etiquetas a las carpetas de destino
def copy_files(files, source_dir, dest_dir):
    for file in files:
        src = os.path.join(source_dir, file)
        dst = os.path.join(dest_dir, file)
        shutil.copyfile(src, dst)

# Copiar imágenes y etiquetas a las carpetas de entrenamiento, validación y prueba
copy_files(train_images, images_dir, train_images_dir)
copy_files(valid_images, images_dir, valid_images_dir)
copy_files(test_images, images_dir, test_images_dir)

copy_files(train_labels, labels_dir, train_labels_dir)
copy_files(valid_labels, labels_dir, valid_labels_dir)
copy_files(test_labels, labels_dir, test_labels_dir)
```

Figura 34. Script para la división del conjunto de datos

La **Figura 35** presenta el porcentaje 80:10:10 asignado a cada conjunto del dataset original, donde se dividen en 3 conjuntos[28]: entrenamiento se asignan 3455 imágenes, pruebas y validación 432 imágenes a cada uno.

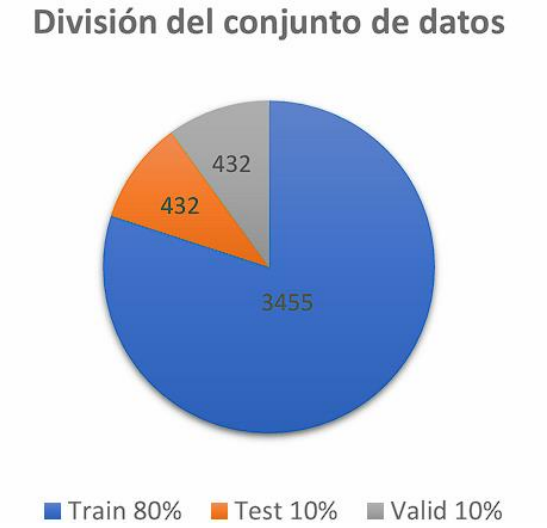


Figura 35. División del conjunto de datos

[28] División del dataset versión final:
https://drive.google.com/drive/folders/1qEaALcZd0A510ap4j8I-X22xHLcGop3M?usp=sharing

6.1.3. Fase 3: Ingeniería del modelo

6.1.3.1. Tarea 9: Selección de los modelos

Se realizó una revisión de las versiones de YOLO usadas en los TR, con la finalidad de determinar los modelos que obtuvieron mejor rendimiento en la detección de objetos, tomando en cuenta las métricas de Precisión, Recall y mAP@0.50. En la **Tabla 12** se muestran los modelos usados por cada TR con los valores obtenidos en las métricas definidas anteriormente.

Tabla 12. Rendimiento de modelos YOLO utilizados en los Trabajos Relacionados (TR)

Trabajo relacionado	Modelo	Resultados obtenidos		
		Precisión	**Recall**	**mAP@0.50**
TR2	YOLOv8n	0.96071	0.96569	0.97884
	YOLOv8s	0.97082	0.96837	0.98016
	YOLOv8m	0.96983	0.973	0.98076
	YOLOv8l	0.97133	0.97389	0.98293
	YOLOv8x	0.96981	0.97003	0.98092
TR3	YOLOv8n	0.787	0.733	0.788
	YOLOv8s	0.837	0.781	0.834
	YOLOv8m	0.865	0.79	0.857
	YOLOv8l	0.864	0.809	0.862
	YOLOv8x	0.86	0.812	0.864
TR4	YOLOv6	0.8459	0.848	0.8912
	YOLOv7	0.8478	0.8499	0.9029
	YOLOv8	0.8718	0.8834	0.9145
TR5	YOLOv4	0.99	0.99	-
TR6	YOLOv5s	0.9373	0.9294	-
TR7	YOLOv5s	0.792	0.826	0.813
TR8	YOLO-NAS	0.706	0.988	0.979
	YOLOv7	0.954	0.958	0.967
	YOLOv8	0.962	0.925	0.955
TR9	YOLOv3-tiny	0.81	0.43	0.694
	YOLOv4	0.89	0.98	0.9750
	YOLOv5s	0.95	0.89	0.8823
	YOLOv7s	1.0	0.95	0.933
	YOLOv8n	0.88	0.8766	0.9904
TR10	YOLOv7s	0.967	0.964	0.982
TR11	YOLOv5n	-	-	0.741
	YOLOv5s	-	-	0.847
	YOLOv5m	-	-	0.895
	YOLOv5l	-	-	0.889
	YOLOv5x	-	-	0.906
	YOLOv8n	-	-	0.792
	YOLOv8s	-	-	0.909
	YOLOv8m	-	-	0.947
	YOLOv8l	-	-	0.953
	YOLOv8x	-	-	0.965

Proporción de predicciones positivas (Precision); Proporción de casos positivos (Recall); Precisión media promedio obtenida en un umbral de intercesión sobre unión (IOU) de 0.5 (mAP@0.5); nano (n); small (s); medium (m); large (l); extra large (x).

A partir de los resultados obtenidos en la **Tabla 12**, se seleccionó las versiones YOLOv5s y YOLOv8s, ya que son los más usados en los TR; además al explorar los estudios, se determinó que los modelos elegidos, requieren de un tiempo entre 2.49 a 3 horas para entrenar, obteniendo resultados óptimos; en el caso de YOLOv5s alcanzó un mAP@0.50 (0.813-0.8823), Precisión (0.792-0.95) y Recall (0.826-0.9294); mientras tanto YOLOv8s obtuvo un mAP@0.50 (0.834-0.98016), Recall (0.781-0.96837) y Precisión (0.837-0.97082).

Por lo tanto, en comparación con las distribuciones grandes (m, l, x), necesitan recursos computacionales altos, tales como GPUs superiores a 8 GB, memoria RAM mayor a 32 GB, debido que requieren más capacidad para el procesamiento durante el entrenamiento, mayor consumo de energía y la inferencia de las tasas de cuadro por segundo (FPS) es más lenta. De tal manera los modelos anteriormente mencionados son usados, debido a que no requieren de hardware y software especializado, ya que son consideradas versiones ligeras y pequeñas.

6.1.3.2. Tarea 10: Entrenamiento de los modelos

- **Selección de hiperparámetros**

Una vez seleccionado los modelos de detección de objetos YOLOv5s y YOLOv8s, se definió los hiperparámetros para el entrenamiento, tomando en cuenta los trabajos relacionados TR1, TR2 y TR3, los cuales proporcionaron información útil para la configuración y valores usados en los parámetros. En la **Tabla 13** se detalla la configuración general establecida para el entrenamiento inicial y los valores usados en los diferentes experimentos.

Tabla 13. Hiperparámetros implementados en los Trabajos Relacionados

Trabajo relacionado	Configuración inicial	Hiperparámetro	Valores
TR1	(optimizer= SGD, lr0= 0,001, epochs = 100)	Batch size	6, 32, 64, 128, 25
		Activation function	ReLU, Leaky ReLU, Tanh, PreLU
		Optimizer	SGD, Adam, AdaGrad, Adamax, RMSProp
TR2	(batch size= 16, optimizer= SGD, lr0= 0.01, weight decay= 0.0005, momentum= 0.937, drop out= 0, epochs= 100)	Learning rate	0.0001, 0.001, 0.005, 0.01, 0.015, 0.05
		Optimizer	SGD, Adam, Adamax, AdamW, NAdam, RAdam, RMSProp
		Epochs	100, 200, 300
		Batch size	8, 16, 32, 48, 64, 80
		Weight decay	0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05
		Momentum	0.859, 0.885, 0.911, 0.937, 0.963, 0.989

Trabajo relacionado	Configuración inicial	Hiperparámetro	Valores
TR2		Drop out	0.0, 0.1, 0.3, 0.5, 0.7
TR3	(lr0= 0.01, optimizer= SGD, batch size= 16, weight decay= 0.0005, epochs= 100)	Learning rate	0.001, 0.01, 0.1
		Optimizer	SGD**,** Adam, AdamW, Adamax, NAdam, RAdam, RMSProp
		Batch size	16, 32, 64, 128, 256
		Weight decay	0.00005, 0.0005, 0.005, 0.05
		Epochs	100, 200, 300

A partir de los resultados establecidos en la **Tabla 13** se definió una nueva configuración con los siguientes hiperparámetros: learning rate, optimizer, batch size, weight decay, epochs. En la **Tabla 14** se presenta los valores establecidos para cada hiperparámetro, los mismos se usaron durante la etapa de entrenamiento.

Tabla 14. Hiperparámetros definidos para el entrenamiento

Hiperparámetro	Valores
Learning rate	0.01, 0.001, 0.0001
Optimizer	SGD, Adam, AdaMax, AdamW, RMSProp
Batch size	4, 8, 16
Weight decay	0.05, 0.005, 0.0005, 0.00005
Epochs	50, 100, 200

En la **Tabla 15** se presenta la configuración por defecto de los hiperparámetros usada en el entrenamiento de los modelos YOLOv5s y YOLOv8s.

Tabla 15. Configuración general para el entrenamiento de YOLOv5s y YOLOv8s

Configuración general				
Lerning rate	**Optimizer**	**Batch size**	**Weight decay**	**Epochs**
-	SGD	8	0.0005	50

En la **Tabla 16**, se presenta las especificaciones de los recursos, librerías, hardware y software utilizado en el proceso de entrenamiento de los modelos YOLOv5s y YOLOv8s.

Tabla 16. Hardware y Software utilizado en el TIC

Especificaciones del entorno usado en el entrenamiento	
Procesador	AMD Ryzen 7 5800H with Radeon Graphics 3.20 GHz
Memoria RAM	16 GB
Sistema Operativo	Windows 11
GPU	NVIDIA GeForce RTX 3050 4 GB

- **Entrenamiento de los modelos YOLOv5s y YOLOv8s utilizando el conjunto de datos dividido**

En la **Figura 36** se presenta la configuración del archivo data.yaml, el cual contiene las rutas de los directorios de entrenamiento, validación, pruebas y el nombre de la clase, específicamente las urls hacen referencia a la dirección que contiene las imágenes; dicho archivo es necesario para entrenar los modelos YOLOv5s y YOLOv8s.

```
train: [redacted]DivisionData/train/images
val: [redacted]/DivisionData/valid/images
test: [redacted]/DivisionData/test/images

nc: 1
names: ['TizonTardio']
```

Figura 36. Configuración del archivo data.yaml

A continuación, se presenta el entrenamiento realizado con las distribuciones YOLOv5s y YOLOv8s utilizando el dataset divido:

- **YOLOv5s**

En la **Figura 37**, se presenta el script usado para realizar el entrenamiento de YOLOv5s en el entorno de desarrollo JupyterLab, para lo cual se importó la librería de ultralytics con la finalidad de descargar el modelo preentrenado yolov5s; se seleccionó la tarea train donde se configuró los hiperparámetros con los valores descritos en la **Tabla 15**, de tal manera se asignó el primer valor para learning rate (0.01).

Una vez finalizado el primer entrenamiento, se usó los valores establecidos en la **Tabla 14,** para lo cual se utilizó el mismo script con la variación del nuevo valor, aplicando el método un factor a la vez (OFAT), empezando por learning rate hasta finalizar con las épocas.

```python
from ultralytics import YOLO

# Descargar variante yolov5s
model = YOLO('yolov5su.pt')

# Configurar parámetros para el entrenamiento
model.train(data='[redacted]/DivisionData/data.yaml',
            epochs=50, batch=8, optimizer='SGD', lr0=0.01, weight_decay=0.0005)

# Validar el modelo y obtener las métricas
metrics = model.val()

# Mostrar métricas del entrenamiento
print(f"Precisión: {metrics.results_dict['metrics/precision(B)']}")
print(f"Recall: {metrics.results_dict['metrics/recall(B)']}")
print(f"mAP50: {metrics.results_dict['metrics/mAP50(B)']}")
print(f"mAP50-95: {metrics.results_dict['metrics/mAP50-95(B)']}")
```

Figura 37. Script para entrenar YOLOv5s

En la **Figura 38** se muestra una visión general de la estructura de YOLOv5s con la configuración de los hiperparámetros; la arquitectura del modelo consta número de 262 capas, 9 122 579 millones de parámetros, 9 122 563 gradientes y 24.0 GFLOPs (operaciones de coma flotante); además se transfirieron 421 de 427 elementos de los pesos preentrenados y congeló la capa model.24.dfl.conv.weight de manera automática con la finalidad de mejorar la estabilidad del entrenamiento.

```
engine\trainer: task=detect, mode=train, model=yolov5su.pt, data=
patience=100, batch=8, imgsz=640, save=True, save_period=-1, cache=False, device=None, workers=8, project
=True, optimizer=SGD, verbose=True, seed=0, deterministic=True, single_cls=False, rect=False, cos_lr=Fals
action=1.0, profile=False, freeze=None, multi_scale=False, overlap_mask=True, mask_ratio=4, dropout=0.0,
rid=False, conf=None, iou=0.7, max_det=300, half=False, dnn=False, plots=True, source=None, vid_stride=1,
=False, agnostic_nms=False, classes=None, retina_masks=False, embed=None, show=False, save_frames=False,
se, show_labels=True, show_conf=True, show_boxes=True, line_width=None, format=torchscript, keras=False,
plify=False, opset=None, workspace=4, nms=False, lr0=0.01, lrf=0.01, momentum=0.937, weight_decay=0.0005,
p_bias_lr=0.1, box=7.5, cls=0.5, dfl=1.5, pose=12.0, kobj=1.0, label_smoothing=0.0, nbs=64, hsv_h=0.015,
0.1, scale=0.5, shear=0.0, perspective=0.0, flipud=0.0, fliplr=0.5, bgr=0.0, mosaic=1.0, mixup=0.0, copy_
0.4, crop_fraction=1.0, cfg=None, tracker=botsort.yaml, save_dir=runs\detect\train8
Overriding model.yaml nc=80 with nc=1

                   from  n    params  module                                       arguments
  0                  -1  1      3520  ultralytics.nn.modules.conv.Conv             [3, 32, 6, 2, 2]
  1                  -1  1     18560  ultralytics.nn.modules.conv.Conv             [32, 64, 3, 2]
  2                  -1  1     18816  ultralytics.nn.modules.block.C3              [64, 64, 1]
  3                  -1  1     73984  ultralytics.nn.modules.conv.Conv             [64, 128, 3, 2]
  4                  -1  2    115712  ultralytics.nn.modules.block.C3              [128, 128, 2]
  5                  -1  1    295424  ultralytics.nn.modules.conv.Conv             [128, 256, 3, 2]
  6                  -1  3    625152  ultralytics.nn.modules.block.C3              [256, 256, 3]
  7                  -1  1   1180672  ultralytics.nn.modules.conv.Conv             [256, 512, 3, 2]
  8                  -1  1   1182720  ultralytics.nn.modules.block.C3              [512, 512, 1]
  9                  -1  1    656896  ultralytics.nn.modules.block.SPPF            [512, 512, 5]
 10                  -1  1    131584  ultralytics.nn.modules.conv.Conv             [512, 256, 1, 1]
 11                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 12             [-1, 6]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 13                  -1  1    361984  ultralytics.nn.modules.block.C3              [512, 256, 1, False]
 14                  -1  1     33024  ultralytics.nn.modules.conv.Conv             [256, 128, 1, 1]
 15                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 16             [-1, 4]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 17                  -1  1     90880  ultralytics.nn.modules.block.C3              [256, 128, 1, False]
 18                  -1  1    147712  ultralytics.nn.modules.conv.Conv             [128, 128, 3, 2]
 19            [-1, 14]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 20                  -1  1    296448  ultralytics.nn.modules.block.C3              [256, 256, 1, False]
 21                  -1  1    590336  ultralytics.nn.modules.conv.Conv             [256, 256, 3, 2]
 22            [-1, 10]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 23                  -1  1   1182720  ultralytics.nn.modules.block.C3              [512, 512, 1, False]
 24        [17, 20, 23]  1   2116435  ultralytics.nn.modules.head.Detect           [1, [128, 256, 512]]
YOLOv5s summary: 262 layers, 9,122,579 parameters, 9,122,563 gradients, 24.0 GFLOPs

Transferred 421/427 items from pretrained weights
Freezing layer 'model.24.dfl.conv.weight'
```

Figura 38. Estructura del modelo YOLOv5s: Configuración y Arquitectura

- **YOLOv8s**

En la **Figura 39** se muestra el scritp usado para entrenar el modelo, para lo cual se cargó el modelo preentrenado yolov8s, posteriormente se usó la configuración inicial detallada en la **Tabla 15**, asignando el primer valor (0.01) al hiperparámetro learning rate. Al finalizar el entrenamiento, se aplicó el método OFAT para variar un valor del parámetro a la vez, mediante

la asignación del nuevo valor descrito en la **Tabla 14**; se tomó en cuenta la variable que generó mejores resultados en las métricas para los siguientes entrenamientos hasta terminar con las épocas.

```
# Importar libreria de ultralytics
from ultralytics import YOLO

# Descargar variante yolov8s
model = YOLO('yolov8s.pt')

# Configuración para el entrenamiento
model.train(data='                                    /DivisionData/data.yaml',
            epochs=50, batch=8, optimizer='SGD', lr0=0.001, weight_decay=0.0005)

# Validar el modelo y obtener las métricas
metrics = model.val()

# Mostrar métricas del entrenamiento
print(f"Precisión: {metrics.results_dict['metrics/precision(B)']}")
print(f"Recall: {metrics.results_dict['metrics/recall(B)']}")
print(f"mAP50: {metrics.results_dict['metrics/mAP50(B)']}")
print(f"mAP50-95: {metrics.results_dict['metrics/mAP50-95(B)']}")
```

Figura 39. Script para entrenar YOLOv8s

En la **Figura 40** se muestra un resumen de la estructura general del modelo YOLOv8s con su respectiva configuración de hiperparámetros; referente a la arquitectura consta de 225 capas, 11 135 987 millones de parámetros, 28.6 GFLOPs y 11 135 971 millones de gradientes; transfiere 349 de 355 elementos desde los pesos preentrenados y congela la capa model.22.dfl.conv.weight para estabilizar el entrenamiento.

```
engine\trainer: task=detect, mode=train, model=yolov8s.pt, data=                                                    /
patience=100, batch=8, imgsz=640, save=True, save_period=-1, cache=False, device=None, workers=8, project=
=True, optimizer=SGD, verbose=True, seed=0, deterministic=True, single_cls=False, rect=False, cos_lr=False
action=1.0, profile=False, freeze=None, multi_scale=False, overlap_mask=True, mask_ratio=4, dropout=0.0,
rid=False, conf=None, iou=0.7, max_det=300, half=False, dnn=False, plots=True, source=None, vid_stride=1,
=False, agnostic_nms=False, classes=None, retina_masks=False, embed=None, show=False, save_frames=False,
se, show_labels=True, show_conf=True, show_boxes=True, line_width=None, format=torchscript, keras=False,
plify=False, opset=None, workspace=4, nms=False, lr0=0.001, lrf=0.01, momentum=0.937, weight_decay=0.0005
up_bias_lr=0.1, box=7.5, cls=0.5, dfl=1.5, pose=12.0, kobj=1.0, label_smoothing=0.0, nbs=64, hsv_h=0.015,
0.1, scale=0.5, shear=0.0, perspective=0.0, flipud=0.0, fliplr=0.5, bgr=0.0, mosaic=1.0, mixup=0.0, copy_
0.4, crop_fraction=1.0, cfg=None, tracker=botsort.yaml, save_dir=runs\detect\train4
Overriding model.yaml nc=80 with nc=1

                   from  n    params  module                                       arguments
  0                  -1  1       928  ultralytics.nn.modules.conv.Conv             [3, 32, 3, 2]
  1                  -1  1     18560  ultralytics.nn.modules.conv.Conv             [32, 64, 3, 2]
  2                  -1  1     29056  ultralytics.nn.modules.block.C2f             [64, 64, 1, True]
  3                  -1  1     73984  ultralytics.nn.modules.conv.Conv             [64, 128, 3, 2]
  4                  -1  2    197632  ultralytics.nn.modules.block.C2f             [128, 128, 2, True]
  5                  -1  1    295424  ultralytics.nn.modules.conv.Conv             [128, 256, 3, 2]
  6                  -1  2    788480  ultralytics.nn.modules.block.C2f             [256, 256, 2, True]
  7                  -1  1   1180672  ultralytics.nn.modules.conv.Conv             [256, 512, 3, 2]
  8                  -1  1   1838080  ultralytics.nn.modules.block.C2f             [512, 512, 1, True]
  9                  -1  1    656896  ultralytics.nn.modules.block.SPPF            [512, 512, 5]
 10                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 11             [-1, 6]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 12                  -1  1    591360  ultralytics.nn.modules.block.C2f             [768, 256, 1]
 13                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 14             [-1, 4]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 15                  -1  1    148224  ultralytics.nn.modules.block.C2f             [384, 128, 1]
 16                  -1  1    147712  ultralytics.nn.modules.conv.Conv             [128, 128, 3, 2]
 17            [-1, 12]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 18                  -1  1    493056  ultralytics.nn.modules.block.C2f             [384, 256, 1]
 19                  -1  1    590336  ultralytics.nn.modules.conv.Conv             [256, 256, 3, 2]
 20             [-1, 9]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 21                  -1  1   1969152  ultralytics.nn.modules.block.C2f             [768, 512, 1]
 22        [15, 18, 21]  1   2116435  ultralytics.nn.modules.head.Detect           [1, [128, 256, 512]]
Model summary: 225 layers, 11,135,987 parameters, 11,135,971 gradients, 28.6 GFLOPs

Transferred 349/355 items from pretrained weights
Freezing layer 'model.22.dfl.conv.weight'
```

Figura 40. Estructura del modelo YOLOv8s: Configuración y Arquitectura

6.1.3.3. Tarea 11: Documentar los experimentos de los modelos

Una vez finalizado los entrenamientos con YOLOv5s y YOLOv8s aplicando el método OFAT, se procede a describir los resultados obtenidos en cada versión y como los hiperparámetros influyen en las métricas de: precision, recall, mAP@0.50, mAP@0.50:0.90.

A. YOLOv5s

- **Tasa de aprendizaje (Learning rate)**

En esta sección se presenta los resultados obtenidos al variar la tasa de aprendizaje, utilizando una configuración fija que incluye, optimizer SGD, batch size 8, weight decay 0.0005 y 50 epochs. A continuación, se describe el impacto del hiperparámetro en el rendimiento del modelo:

- Con un valor inicial de **tasa de aprendizaje (lr0) de 0.01**, el modelo alcanzó una precisión máxima de 0.9276 y un mAP@0.50 de 0.9549. Esto indica que las

predicciones fueron altamente efectivas y que, en la mayoría de los casos, identificó correctamente la clase. Por otro lado, el recall fue de 0.8655, lo que evidenció un rendimiento óptimo del modelo YOLOv5s.

- Al utilizar un **lr0 de 0.001**, se observó un equilibrio entre la precisión (0.888) y el recall (0.894). Esto sugiere que el modelo mejoró en la identificación de instancias verdaderas positivas; sin embargo, la precisión (0.888) fue inferior en comparación con la obtenida con lr0 = 0.01. Además, los valores de mAP resultaron ligeramente menores.
- Para un **lr0 de 0.0001**, el modelo mostró un rendimiento inferior en comparación con los valores anteriores, ya que todas las métricas fueron más bajas. Esto destacó una convergencia más lenta, lo que limitó el aprendizaje y evitó alcanzar un punto óptimo en el rendimiento del modelo.

A partir de los resultados obtenidos, se evidenció que al variar la tasa de aprendizaje generó un impacto significativo en el rendimiento del modelo, ya que se encarga de regular la magnitud con la que se actualizan los pesos, específicamente el valor de 0.01 demostró ser adecuado para detectar la enfermedad alcanzando un 94% (734 de 785) de predicciones correctas.

En la **Tabla 17** se presenta los resultados obtenidos al utilizar diferentes valores para Learning Rate (tasa de aprendizaje) en la validación del modelo YOLOv5s.

Tabla 17. Resultados de la validación de YOLOv5s con diferentes Tasas de aprendizaje

Learning Rate	Precision	Recall	mAP@0.50	mAP@0.50:0.90	Time (hours)
0.01	**0.9276**	**0.8655**	**0.9549**	**0.7093**	1.312
0.001	0.8880	0.8994	0.9489	0.7002	1.275
0.0001	0.8618	0.8338	0.9147	0.6415	1.244

Configuración: optimizer = SGD, batch = 8, weight_decay = 0.0005, epochs = 50.

- **Optimizador (Optimizer)**

A continuación, se analizan los resultados obtenidos por cada optimizador, utilizando una configuración constante: learning rate 0.01, batch size 8, weight decay 0.0005 y epochs 50:

- **AdamW** obtuvo una precisión de 0.87 y un recall de 0.8227, lo que reflejó un equilibrio entre ambas métricas. Sin embargo, aunque los valores fueron aceptables, el modelo no logró sobresalir en mAP@0.50 ni en precisión, en comparación con los optimizadores SGD y AdaMax.
- **Adam** presentó un rendimiento ligeramente inferior al de AdamW en todas las métricas, debido a su limitada capacidad para generalizar con datos complejos. Esto lo posicionó por debajo de AdamW, SGD y AdaMax, aunque mostró un buen balance general.

- **AdaMax** alcanzó un rendimiento aceptable, con una precisión de 0.8979 y un mAP@0.50 de 0.9283, lo que indicó que el modelo fue capaz de adaptarse bien en la identificación de la enfermedad. Además, demostró un balance adecuado entre precisión y recall; no obstante, su rendimiento fue ligeramente inferior al del optimizador SGD.
- **SGD** obtuvo el mejor desempeño en todas las métricas, logrando una precisión de 0.9276, un recall de 0.8655 y un mAP@0.50 de 0.9549. Esto sugiere que el modelo tuvo una gran capacidad para actualizar los pesos, demostrando ser altamente eficaz en la detección de la enfermedad.
- **RMSProp** mostró un desempeño inferior en comparación con los demás optimizadores, con métricas notablemente más bajas. Esto indicó que el modelo tuvo dificultades tanto en la convergencia como en la generalización para capturar patrones relevantes en los datos.

El optimizador juega un papel crucial en la regulación del ajuste de los pesos del modelo basándose en el gradiente calculado. Se destacó SGD como el mejor optimizador, logrando una tasa de aciertos del 94% y obteniendo los mejores resultados en las métricas clave. Esto evidencia una minimización eficaz de los errores en las predicciones.

En la **Tabla 18** se muestra los resultados obtenidos en la validación del modelo YOLOv5s, al utilizar diferentes Optimizer (optimizadores).

Tabla 18. Resultados de la validación de YOLOv5s con diferentes Optimizadores

Optimizer	Precision	Recall	mAP@0.50	mAP@0.50:0.90	Time (hours)
AdamW	0.87	0.8274	0.9170	0.6405	1.257
Adam	0.8563	0.8044	0.9029	0.6168	1.253
AdaMax	0.8979	0.8288	0.9283	0.6611	1.269
SGD	**0.9276**	**0.8655**	**0.9549**	**0.7093**	1.312
RMSProp	0.00197	0.296	0.0019	0.0005	1.225

Configuración: lr0 = 0.01, batch = 8, weight_decay = 0.0005, epochs = 50.

- **Tamaño de lote**

Se analiza la influencia del batch size en el rendimiento del modelo, utilizando una configuración estática: learning rate 0.01, optimizer SGD, weight decay 0.0005 y 50 epochs:

- **Batch size = 4**, el modelo logró un equilibrio entre precisión de 0.8754 y recall de 0.8772. Además, el mAP@0.50 alcanzó 0.9408, lo que sugiere una detección eficaz. Sin embargo, el mAP@0.50:0.90 fue el más bajo en comparación con otros tamaños de lote, indicando que un batch pequeño genera más ruido en las actualizaciones y dificulta la generalización en casos complejos.
- **Batch size = 8**, logró los mejores resultados con una precisión de 0.9276 y mAP@0.50 de 0.9549. Esto sugiere que el modelo tiene la capacidad de identificar correctamente la enfermedad en la mayoría de los casos. Aunque el recall de

0.8655 es ligeramente inferior, el modelo mantiene un buen equilibrio entre precisión y generalización.

- **Batch size = 16**, alcanzó el mayor valor de recall, 0.9185, y un ligero aumento en mAP@0.50:0.90 a 0.7188, lo que indica un desempeño eficaz en la identificación de instancias positivas. Sin embargo, la precisión de 0.8674 fue inferior en comparación con el batch de 8, lo que sugiere un leve aumento en los falsos positivos.

El tamaño de lote influye significativamente en la eficacia del modelo al determinar cuántas muestras procesa antes de actualizar los pesos. En este caso, un batch de 8 ofreció el mejor rendimiento en la detección de la enfermedad, alcanzando un 94% de aciertos y reduciendo los falsos positivos. Aunque el recall no es el valor más alto, es adecuado para identificar los casos positivos.

En la **Tabla 19** se presenta los resultados obtenidos al usar diferentes Batch Sizes (tamaños de lote) en la validación del modelo YOLOv5s.

Tabla 19. Resultados de la validación de YOLOv5s con diferentes Tamaños de Lote

Batch Size	Precision	Recall	mAP@0.50	mAP@0.50:0.90	Time (hours)
4	0.8754	0.8772	0.9408	0.6934	1.533
8	**0.9276**	**0.8655**	**0.9549**	**0.7093**	1.312
16	0.8674	0.9185	0.9525	0.7188	2.521

Configuración: lr0 = 0.01, optimizer = SGD, weight_decay = 0.0005, epochs = 50.

- **Decaimiento de peso o Regulador de pesos**

En la sección, se analiza el impacto del weight decay en el rendimiento del modelo, utilizando una configuración constante: learning rate 0.01, optimizer SGD, batch 8 y 50 epochs:

- **Weight decay = 0.05**, obtuvo las métricas más bajas, con una precisión de 0.8504, recall de 0.8328 y mAP@0.50 de 0.9036. Esto sugiere que un alto decaimiento de peso limita la capacidad del modelo para aprender patrones complejos.

- **Weight decay = 0.005**, logró un rendimiento aceptable, alcanzando un equilibrio entre precisión de 0.9040 y recall de 0.8634. Su mAP@0.50 fue óptimo, con un valor de 0.9401, indicando que el modelo es adecuado para la identificación de la enfermedad. Sin embargo, no logró superar los valores obtenidos con un decaimiento de peso de 0.0005.

- **Weight decay = 0.0005**, generó los mejores resultados en precisión (0.9276) y mAP@0.50 (0.9549), mostrando un equilibrio adecuado entre generalización y precisión, y reduciendo los falsos positivos.

- **Weight decay = 0.00005**, alcanzó el mayor valor de recall, 0.8904, y un mAP@0.50:0.90 de 0.7116, destacando que el modelo identificó correctamente los casos positivos. Sin embargo, la precisión fue de 0.8884, menor en comparación con el valor de 0.0005, lo que sugiere un aumento en los falsos positivos.

La caída de peso ayuda a penalizar los pesos grandes para evitar el sobreajuste del modelo. En este caso, un decaimiento de peso de 0.0005 demostró ser el valor óptimo para la detección de la enfermedad, ya que generó resultados adecuados en la precisión y mAP@0.50, lo que reflejó una eficaz generalización en las detecciones y una tasa de aciertos del 94%. Aunque el recall no alcanzó el mejor valor, sigue siendo aceptable, lo que indica una predicción aceptable de los casos positivos.

En la **Tabla 20** se presenta los resultados alcanzados en la validación del modelo YOLOv5s al usar diferentes valores para el hiperparámetro Weight Decay (regulador de peso o caída de peso).

Tabla 20. Resultados de la validación de YOLOv5s con diferentes Reguladores de Pesos

Weight Decay	Precision	Recall	mAP@0.50	mAP@0.50:0.90	Time (hours)
0.05	0.8504	0.8328	0.9036	0.6059	1.265
0.005	0.9040	0.8634	0.9401	0.6901	1.256
0.0005	**0.9276**	**0.8655**	**0.9549**	**0.7093**	1.312
0. 00005	0.8884	0.8904	0.9504	0.7116	1.271

Configuración: lr0 = 0.01, optimizer = SGD, batch = 8, epochs=50.

- **Épocas**

A continuación, se describe la influencia de epochs en las métricas usando una configuración fija: learning rate 0.01, optimizer SGD, batch 8 y weight decay 0.0005:

- Cuando **epochs = 50**, alcanzó una precisión adecuada de 0.9276 y un mAP@0.50 de 0.9549, lo que indica que el modelo generaliza bien y detecta de manera aceptable los casos positivos. Sin embargo, el recall fue de 0.8655 y el mAP@0.50:0.90 alcanzó 0.7093, por lo tanto, son valores inferiores en comparación con otros números de iteraciones.
- Al usar **epochs = 100**, se observó una mejora en el recall, alcanzando 0.9159, así como en mAP@0.50 (0.9593) y mAP@0.50:0.90 (0.7342). Esto indica una mejor capacidad del modelo para identificar los casos positivos. Sin embargo, la precisión disminuyó ligeramente a 0.925 en comparación con 50 épocas, lo que determinó un leve incremento en los falsos positivos.
- Con **epochs = 200**, se lograron los valores más altos en todas las métricas: precisión de 0.9401, recall de 0.9194, f1-score de 0.9296, mAP@0.50 de 0.9659

y mAP@0.50:0.90 de 0.7699. Esto demuestra que el modelo mejora su capacidad para realizar detecciones en casos más complejos.

El número de épocas afecta directamente el rendimiento del modelo, ya que representa las veces que el modelo recorre el conjunto de entrenamiento, esto influye en el aprendizaje de los patrones de los datos. En este caso, 200 épocas fueron adecuadas, ya que lograron los mejores resultados en todas las métricas. Esto indica que el modelo detectó correctamente un 95% (747 de 784) de los casos positivos.

En la **Tabla 21** se presenta los resultados obtenidos en la validación del modelo YOLOv5s al utilizar diferentes valores para el parámetro Epochs (épocas).

Tabla 21. Resultados de la validación de YOLOv5s con diferentes Épocas

Epochs	Precision	Recall	mAP@0.50	mAP@0.50:0.90	Time (hours)
50	0.9276	0.8655	0.9549	0.7093	1.312
100	0.925	0.9159	0.9593	0.7342	2.478
200	**0.9401**	**0.9195**	**0.9659**	**0.7699**	4.957

Configuración: lr0 = 0.01, optimizer = SGD, batch = 8, weight_decay = 0.0005.

Los resultados obtenidos de los entrenamientos se pueden ver en el **Anexo *4***, donde se muestran las gráficas de rendimiento del modelo YOLOv5s.

- **Resultados con mayor rendimiento en las métricas de YOLOv5s**

Una vez finalizado los entrenamientos del modelo, se desatacó los siguientes valores para los hiperparámetros: lr0=0.01, optimizer=SGD, batch=8, weight_decay=0.0005, epochs=200.

En la **Tabla 22** se presenta un resumen de las funciones de pérdida obtenidas durante la validación y el entrenamiento de YOLOv5s. Durante el entrenamiento, la pérdida de caja comenzó en 1.4161 y finalizó en 0.49885, mientras que la pérdida de clasificación comenzó en 1.5223 y disminuyó a 0.24808 con las iteraciones. En la validación, la pérdida de clasificación comenzó en 1.2659 y terminó en 0.84543, y la clasificación comenzó en 1.0302 y finalizó en 0.44302.

De este modo, el modelo demostró ser capaz de detectar y clasificar de manera adecuada la enfermedad del Tizón Tardío, ya que los valores de pérdida disminuyeron constantemente tanto en el entrenamiento como en la validación, indicando una mejora continua en la capacidad del modelo para realizar la predicción.

Tabla 22. Funciones de pérdida de YOLOv8s en el entrenamiento y validación

Funciones de pérdida del entrenamiento			
Epoch	**box_loss**	**cls_loss**	**dfl_loss**
1/200	1.4161	1.5223	1.4992
2/200	1.3309	1.0917	1.3343
….	….	….	….
196/200	0.50608	0.25556	0.87339

Epoch	box_loss	cls_loss	dfl_loss
197/200	0.5022	0.25278	0.86987
198/200	0.50397	0.25153	0.87441
199/200	0.49911	0.25002	0.86839
200/200	0.49885	0.24808	0.86763
Funciones de pérdida de la validación			
Epoch	**box_loss**	**cls_loss**	**dfl_loss**
1/200	1.2659	1.0302	1.2874
2/200	1.3583	1.099	1.3346
….	….	….	….
196/200	0.84994	0.43814	1.0305
197/200	0.84942	0.43852	1.0314
198/200	0.84775	0.44233	1.0314
199/200	0.84625	0.44298	1.0309
200/200	0.84543	0.44302	1.0304

Numero de época (Epochs); Pérdida de caja (box_loss); Pérdida de clasificación (cls_loss); Pérdida de distribución (dfl_loss).

A continuación, en la **Figura 41** se muestra cómo las funciones de perdida para el entrenamiento y validación disminuyeron considerablemente. Estos resultados están detallados en la tabla anterior.

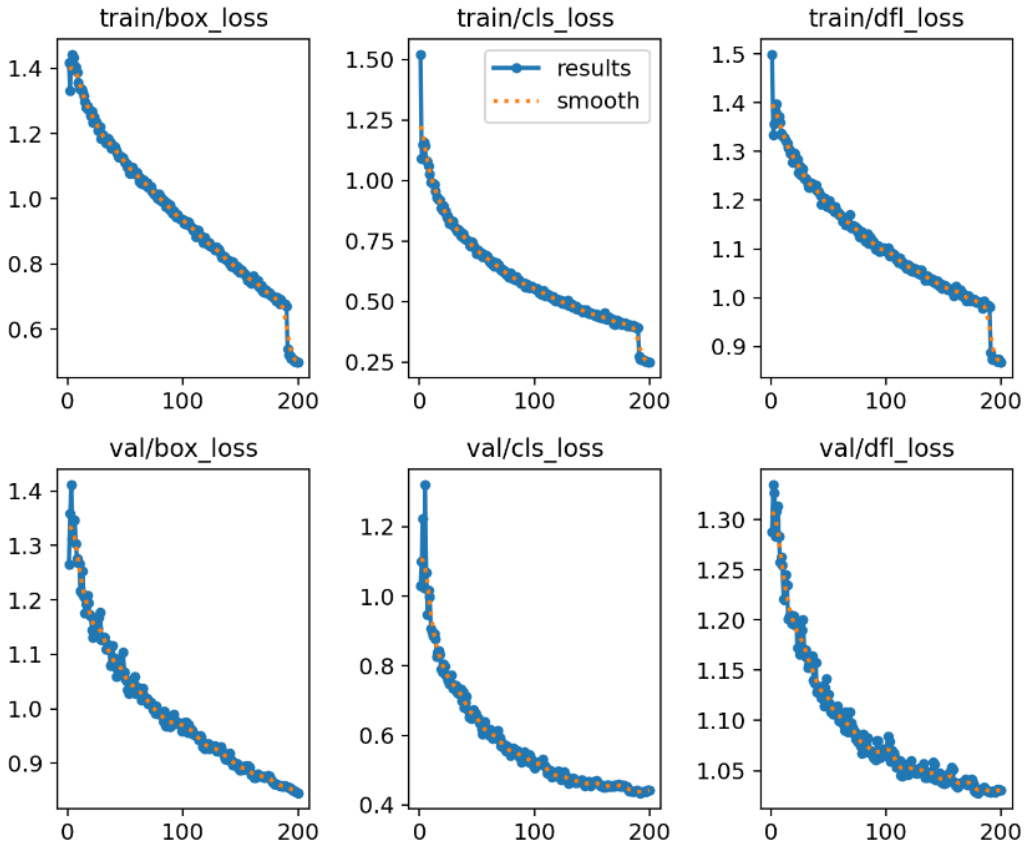


Figura 41. Rendimiento de YOLOv5s durante el entrenamiento y validación

B. YOLOv8s

- **Tasa de aprendizaje**

En esta sección se analiza el impacto del learning rate en el rendimiento de YOLOv8s bajo una configuración especifica: optimizer SGD, batch de 8, weightn decay 0.0005, epochs 50:

- Al usar **lr0 = 0.01**, YOLOv8s alcanzó los mejores resultados en todas las métricas: precisión de 0.9025, recall de 0.9096, mAP@0.50 de 0.9533 y mAP@0.50:0.90 de 0.7208. Esto indica que el modelo tiene un buen desempeño para detectar los casos positivos y minimizar los falsos positivos.
- Cuando el **lr0 = 0.001**, mostró un leve descenso en el recall de 0.037 y en mAP@0.50 de 0.0155. Sin embargo, esto no significa que la detección sea errónea; por el contrario, la precisión es aceptable, lo que indica una predicción estable.
- Con un **lr0 = 0.0001**, el rendimiento de YOLOv8s disminuyó significativamente, pues los resultados alcanzados fueron los más bajos en comparación con los demás valores. En particular, el mAP@0.50:0.90 se vio más afectado con 0.6538, indicando que el modelo no ajustó los pesos lo suficientemente bien durante el entrenamiento, lo que impactó negativamente en la detección.

A partir de los resultados analizados, se determinó que un learning rate de 0.01 logró el mejor equilibrio entre convergencia y un desempeño adecuado. Esto minimizó los falsos positivos y permitió que el modelo detectara correctamente un 94% (737 de 785) de los casos verdaderos.

En la **Tabla 23** se presenta los resultados obtenidos en la validación del modelo YOLOv8s, al variar el Learning Rate (tasa de aprendizaje).

Tabla 23. Resultados de la validación de YOLOv8s con diferentes Tasas de Aprendizaje

Learning Rate	Precision	Recall	mAP@0.50	mAP@0.50:0.90	Time (hours)
0.01	**0.9025**	**0.9096**	**0.9533**	**0.7208**	1.325
0.001	0.8938	0.8726	0.9378	0.7152	1.325
0.0001	0.8513	0.8293	0.9154	0.6538	1.837

Configuración: optimizer = SGD, batch = 8, weight_decay = 0.0005, epochs = 50

- **Optimizador**

En este apartado, se detalla el impacto que generó el optimizador en el rendimiento de YOLOv8s, utilizando una configuración constante, lr0 = 0.01, batch = 8, weight_decay = 0.0005, epochs = 50:

- **AdamW** logró un rendimiento equilibrado con una precisión de 0.8616 y un recall de 0.8564. Sin embargo, los valores de mAP@0.50 (0.9233) y

mAP@0.50:0.90 (0.6556) son inferiores a los de otros optimizadores como SGD y AdaMax, indicando que el ajuste no fue adecuado.

- **Adam** presentó un leve descenso en la precisión (0.8367) y el recall (0.8483) en comparación con AdamW. Además, las métricas de mAP@0.50 y mAP@0.50:0.90 fueron menores, lo que sugiere que el modelo no tuvo un rendimiento adecuado para detectar la enfermedad.
- **AdaMax** alcanzó un rendimiento aceptable al lograr un equilibrio entre precisión y recall con 0.8777. Destaca su capacidad para identificar adecuadamente los casos positivos, aunque las métricas de mAP@0.50 y mAP@0.50:0.90 (0.6739) son buenas, no superan los resultados de SGD.
- **SGD** proporcionó el mejor rendimiento de YOLOv8s, ya que alcanzó los mejores valores en todas las métricas. Esto indica que es el más adecuado para identificar la enfermedad y minimizar los falsos negativos.
- **RMSProp** mostró un desempeño inferior en comparación con los demás optimizadores. Las métricas fueron bajas debido a una tendencia a inestabilidades con el learning rate y una convergencia ineficaz.

En base a los resultados obtenidos, se determinó que el optimizador SGD manejó bien el ajuste de los pesos, logrando los mejores resultados en todas las métricas. Esto minimizó los falsos positivos y permitió que el modelo detectara correctamente un 94% de los casos de la enfermedad.

En la **Tabla 24** se presenta los resultados alcanzados en la validación del modelo YOLOv5s al modificar el hiperparámetro Optimizer (optimizador).

Tabla 24. Resultados de la validación de YOLOv8s con diferentes Optimizadores

Optimizer	Precision	Recall	mAP@0.50	mAP@0.50:0.90	Time (hours)
AdamW	0.8616	0.8564	0.9233	0.6556	1.339
Adam	0.8367	0.8483	0.9120	0.6349	2.016
AdaMax	0.8592	0.8777	0.9307	0.6739	2.016
SGD	**0.9025**	**0.9096**	**0.9533**	**0.7208**	1.325
RMSProp	0.3549	0.3847	0.3017	0.1181	1.326

Configuración: lr0 = 0.01, batch = 8, weight_decay = 0.0005, epochs = 50

- **Tamaño de lote**

Se analiza cómo influye el optimizador en el rendimiento de YOLOv8s, utilizando una configuración fija, lr0 = 0.01, optimizer = SGD, weight_decay = 0.0005, epochs = 50:

- Un **batch size = 4** generó resultados adecuados, aunque las métricas fueron inferiores a las de otros tamaños de lote, como el recall. Esto indica que el modelo no detectó correctamente los casos positivos, limitando el aprendizaje durante el entrenamiento.

- Al usar un **batch size de 8** demostró un equilibrio entre la frecuencia de las actualizaciones de los pesos y estabilidad en los gradientes, ya que logró un rendimiento adecuado en las métricas: precisión, recall, mAP@0.50 y mAP@0.50:0.90, lo que indica que el modelo identificó adecuadamente la enfermedad.
- El **batch size de 16** alcanzó los mejores valores en precisión (0.9303) y mAP@0.50 (0.9534), lo que sugiere una reducción de falsos positivos y una detección adecuada de la enfermedad. Aunque el recall fue ligeramente menor, esto no significa que las predicciones sean erróneas.

El tamaño de lote de 16 se determinó como la mejor opción, ya que proporcionó un equilibrio entre frecuencia de actualización de pesos y estabilidad de los gradientes, logrando una mejora significativa en las métricas de precisión y mAP, alcanzando un 94% (737 de 785) de detecciones correctas.

En la **Tabla 25** se muestra los resultados obtenidos en la validación del modelo YOLOv8s al variar el hiperparámetro Batch Size (tamaño de lote).

Tabla 25. Resultados de la validación de YOLOv8s con diferentes Tamaños de Lote

Batch Size	Precision	Recall	mAP@0.50	mAP@0.50:0.90	Time (hours)
4	0.8915	0.8764	0.9449	0.7049	1.66
8	0.9025	0.9096	0.9533	0.7208	1.325
16	**0.9303**	**0.8841**	**0.9534**	**0.7286**	1.833

Configuración: lr0 = 0.01, optimizer = SGD, weight_decay = 0.0005, epochs = 50.

- **Decaimiento de peso o Regulador de pesos**

En esta sección se detalla el impacto del decaimiento de pesos en el rendimiento de YOLOv8s, utilizando una configuración constante, lr0 = 0.01, batch = 16, optimizer = SGD, epochs = 50:

- Al usar un **weight decay de 0.05**, se generaron resultados aceptables, aunque fueron inferiores a los demás valores de weight decay debido a que aplica una regularización alta, restringiendo la capacidad del modelo para aprender patrones complejos debido a la excesiva penalización de los pesos.
- Con un **weight de 0.005**, se logró un balance entre precisión (0.887) y recall (0.9185). Esto indica que el modelo se ajustó mejor a los datos sin comprometer su capacidad de generalización. Además, se mejoraron las métricas mAP@0.50 (0.9537) y mAP@0.50:0.90 (0.7214), lo que sugiere una detección aceptable de la enfermedad.
- **Weight decay de 0.0005**, alcanzó los mejores valores en precisión (0.9303) y mAP@0.50:0.90 (0.7286). Esto indica que el modelo es capaz de predecir adecuadamente los casos positivos. Aunque el recall y mAP@0.50 son

ligeramente menores que los resultados obtenidos con un weight decay de 0.005, se logró un equilibrio entre precisión y detección en general.

- Al reducir el **weight decay a 0.00005**, se mejoró el recall a 0.9209, destacando una detección adecuada de los casos positivos. Sin embargo, la precisión disminuyó significativamente, lo que sugiere un aumento en las predicciones de falsos positivos.

A partir de los resultados un weight decay de 0.0005 alcanzó un mejor rendimiento en YOLOv8s ya que logró un equilibrio entre la generalización y la regularización de los pesos. Esto se destacó principalmente en la precisión, minimizando los falsos positivos y logrando una detección correcta del 94% (737 de 785) de los casos de la enfermedad.

En la **Tabla 26** se presenta los resultados obtenidos en la validación del modelo al alterar el hiperparámetro Weight Decay (caída de peso o regulador de peso).

Tabla 26. Resultados de la validación de YOLOv8s con diferentes Reguladores de Pesos

Weight Decay	Precision	Recall	mAP@0.50	mAP@0.50:0.90	Time (hours)
0.05	0.8427	0.8548	0.9119	0.6340	1.773
0.005	0.887	0.9185	0.9537	0.7214	1.636
0.0005	**0.9303**	**0.8841**	**0.9534**	**0.7286**	1.823
0. 00005	0.8937	0.9209	0.9534	0.7236	1.558

Configuración: lr0 = 0.01, optimizer = SGD, batch = 16, epochs = 50.

- **Épocas**

En este apartado se detalla la influencia del número de épocas en el rendimiento de YOLOv8s, utilizando una configuración especifica, lr0 = 0.01, optimizer = SGD, batch = 8, weight_decay = 0.0005:

- Con **50 épocas**, se obtuvo resultados aceptables en la precisión 0.9303 y mAP@0.50 0.9534, no obstante, el recall 0.8841 es inferior, lo que indica que el modelo aún no es capaz de aprender bien de los patrones de la enfermedad, afectando en la detección.
- Al usar **100 épocas**, las métricas mejoraron ligeramente en comparación con el valor anterior. Específicamente, el recall aumentó a 0.9159 y el mAP@0.50:0.90 a 0.7541. Esto indica que el modelo aprendió mejor de los patrones, mejorando así la detección de la enfermedad y aumentando las predicciones de los casos positivos.
- Con **200 iteraciones** el modelo logró los mejores resultados en todas las métricas. Esto destacó que el modelo se ajustó a los datos, aumentando la detección de casos positivos y minimizando los falsos positivos.

En base a los resultados obtenidos, 200 épocas es el mejor valor, ya que ofrece mayores resultados en las métricas y mejora el rendimiento general. Esto garantiza que el

95% (743 de 785) de los casos sean detectados correctamente, ya sea en diferentes niveles de confianza.

En la **Tabla 27** se muestra los resultados alcanzados en la validación del modelo al modificar el hiperparámetro Epochs (épocas).

Tabla 27. Resultados de la validación de YOLOv8s con diferentes Épocas

Epochs	Precision	Recall	mAP@0.50	mAP@0.50:0.90	Time (hours)
50	0.9303	0.8841	0.9534	0.7286	1.823
100	0.9305	0.9042	0.9582	0.7541	3.557
200	**0.9366**	**0.9427**	**0.9709**	**0.7991**	7.107

Configuración: lr0 = 0.01, optimizer = SGD, batch = 16, weight_decay = 0.0005.

Los resultados obtenidos de los entrenamientos se pueden ver en el **Anexo 5**, donde se muestran las gráficas de rendimiento del modelo YOLOv8s.

- **Resultados con mayor rendimiento en las métricas de YOLOv8s**

Al finalizar con el entrenamiento del modelo YOLOv8s, se destacó los siguientes valores para los hiperparámetros, lr0=0.01, optimizer=SGD, batch=16, weight_decay=0.0005, epochs=200.

En la **Tabla 28** se muestra un resumen de las funciones de pérdida obtenidas durante el entrenamiento y validación, dichos resultados atribuyen a las pérdidas en la clasificación, caja y distribución.

Lo que demuestra que YOLOv8s se adaptó bien a los datos durante el entrenamiento e identificó adecuadamente la enfermedad en imágenes nuevas, debido a que las funciones de pérdida disminuyeron constantemente, por ejemplo, en el entrenamiento la clasificación empezó con un valor de 1.5784 y término en 0.19464, por otra parte, en la validación comenzó en 1.1373 se redujo a 0.3719.

Tabla 28. Funciones de pérdida de YOLOv8s en el entrenamiento y validación

Funciones de pérdida del entrenamiento			
Epoch	**box_loss**	**cls_loss**	**dfl_loss**
1/200	1.4074	1.5784	1.4772
2/200	1.2896	1.0075	1.3136
….	….	….	….
197/200	0.38487	0.1971	0.8357
198/200	0.38615	0.19546	0.83539
199/200	0.38754	0.19654	0.83813
200/200	0.38301	0.19464	0.83779
Funciones de pérdida de la validación			
Epoch	**box_loss**	**cls_loss**	**dfl_loss**
1/200	1.3057	1.1373	1.348
2/200	1.2645	1.2369	1.2956
….	….	….	….
197/200	0.76551	0.37338	1.0369
198/200	0.76669	0.3732	1.0391

Epoch	box_loss	cls_loss	dfl_loss
199/200	0.76727	0.37214	1.0403
200/200	0.77078	0.3719	1.0455

Número de época (Epoch); Pérdida de caja (box_loss); Pérdida de clasificación (cls_loss); Pérdida de distribución (dfl_loss).

En la **Figura 42** se visualiza cómo las funciones de pérdida fueron disminuyendo constantemente en el entrenamiento y validación del modelo YOLOv8s.

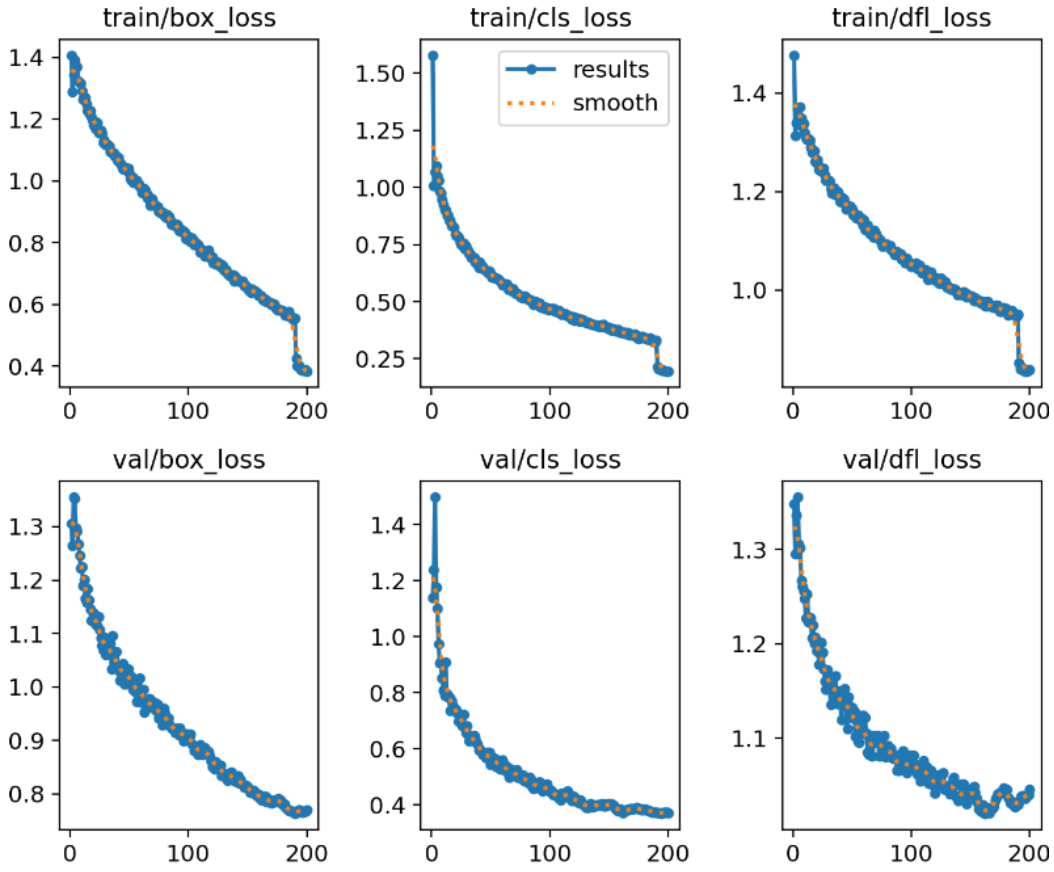


Figura 42. Rendimiento del modelo YOLOv8s durante el entrenamiento y validación

- **Documentación de los modelos de aprendizaje profundo**

Arquitectura YOLOv5

En la **Figura 43** se presenta la arquitectura de YOLOv5, la misma está compuesta por las siguientes capas:

- **Entrada:** recibe una imagen que puede presentar el Tizón Tardío, la misma se redimensiona a un tamaño de 640 x 640 píxeles en formato RGB (color).
- **Capa Backbone:** también conocida como columna vertebral, se encarga de extraer las características (patrones de la enfermedad como manchas oscuras o bordes dañados) de la entrada utilizando bloques convolucionales, específicamente Croos Stage Partial BotleNeck (BottleNeckCSP) procesa las particularidades tales como, color, bordes o texturas específicas de las lesiones

del tizón, por otra parte, Spatial Pyramid Pooling (SPP) se encarga de acoplar los patrones en diferentes tamaños (lesiones pequeñas o grandes) mejorando el aprendizaje del modelo.

- **Capa Neck:** o cuello, usa Path Aggregation Network (PANet), para fusionar las características extraídas en diferentes escalas (pequeño, grande) con la finalidad de optimizar la detección de las áreas infectadas, mediante la integración de operaciones de concatenación, Upsample y convoluciones 1x1 (tamaño de kernel 1x1), 3x3, facilitando la propagación efectiva de la información de la enfermedad, teniendo en cuenta eliminar ruido (patrones similares).
- **Head:** conocida como cabeza, se encarga de dibujar los cuadros delimitadores, a través de convoluciones 1x1 las cuales ubican las coordenadas en las salidas detectadas y asignar el nombre de la clase ***TizonTardio*** con su respectivo puntaje de confianza.

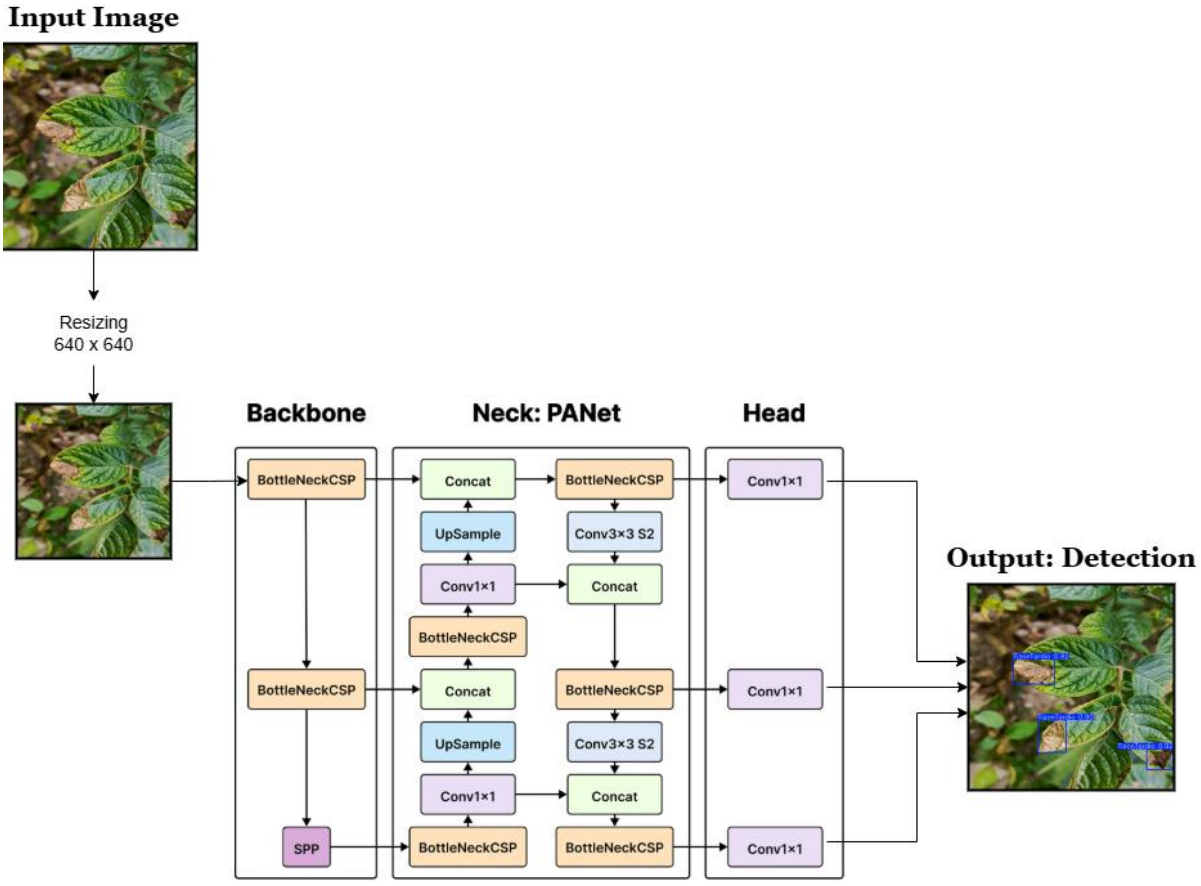


Figura 43. Arquitectura YOLOv5 para detectar el Tizón Tardío en las hojas de papa

Arquitectura YOLOv8

En la **Figura 44** se visualiza la arquitectura de YOLOv8 para detectar el Tizón Tardío. A continuación, se describen cada una de las capas que compone a la arquitectura:

- **Entrada:** procesa la imagen de una hoja de papa con posible presencia de la enfermedad, se redimensiona a un tamaño de 640 x 620 píxeles.
- **Backbone:** usa convoluciones iniciales (Conv) para identificar los patrones (manchas o lesiones de la enfermedad, color, etc) de la imagen, luego el bloque C2f procesa la información para combinar las características y la red Spatial Pyramid Pooling Fast (SPPF) captura las particularidades en múltiples escalas, lo que permite detectar el Tizón Tardío en manchas pequeñas o grandes e independientemente de la posición.
- **Neck:** emplea el método Upsample para aumentar la resolución de las características extraídas, posteriormente combina las particularidades de baja y alta resolución, con la finalidad de asegurar que los patrones (manchas grandes o pequeñas) sean ubicadas en la salida, en tal sentido el bloque C2f tiene el objetivo de distinguir la infección de otros patrones similares, minimizando las detecciones erróneas.
- **Head**: integra capas de detección (pueden ser en diferentes escalas: 80x80 para detecciones pequeñas y 40x40 o 20x20 áreas grandes) para generar las coordenadas de los cuadros delimitadores y asignar la etiqueta ***TizonTardio***, con su respectivo porcentaje de confianza.

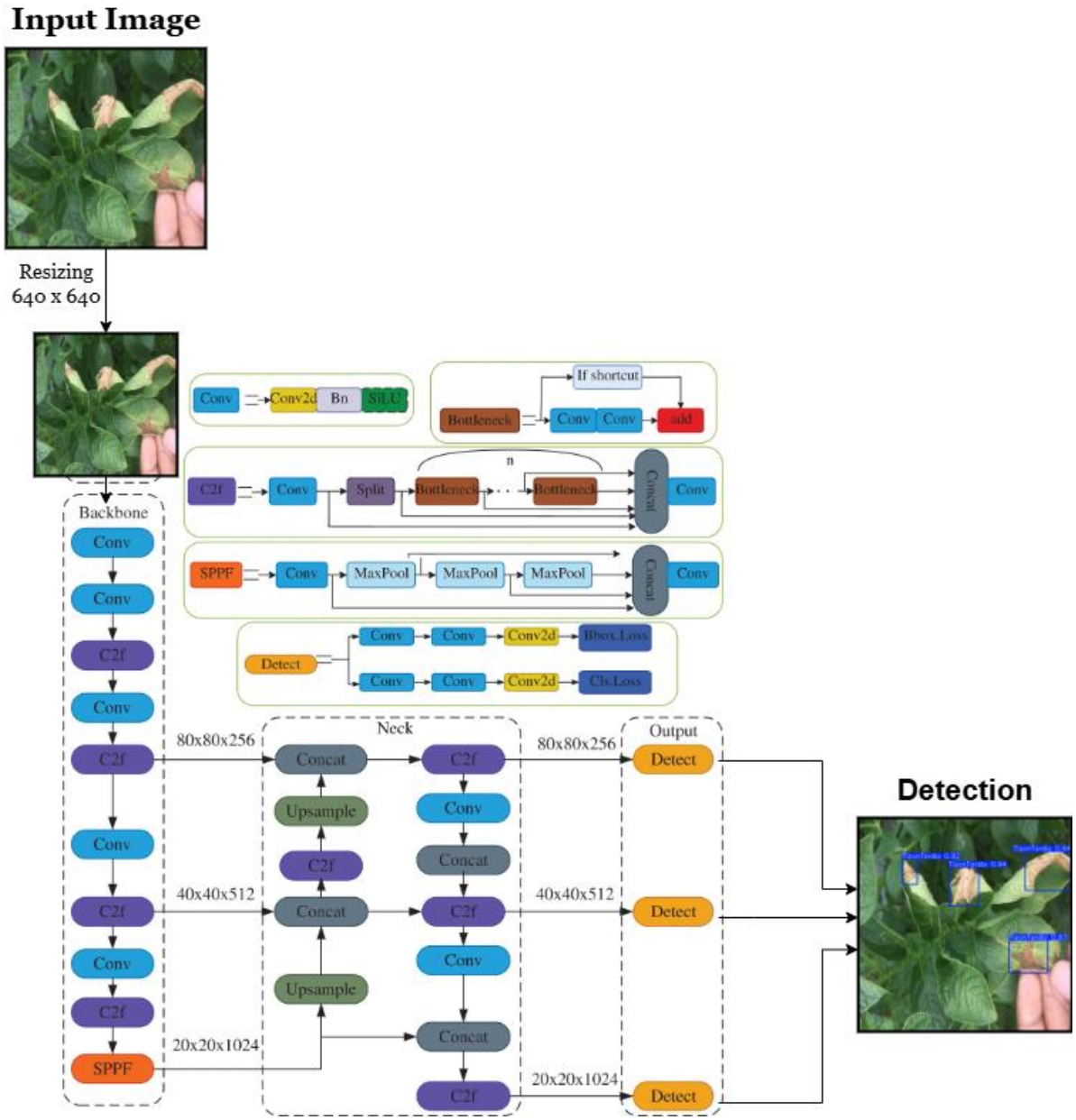


Figura 44. Arquitectura YOLOv8 para la detección del Tizón Tardío en las hojas de papa

Al finalizar con el entrenamiento de las dos versiones se determinaron los valores que generaron un mayor rendimiento en las métricas, en el caso de YOLOv5s la configuración final es: lr0 = 0.01, optimizer = SGD, batch = 8, weight_decay = 0.0005, epochs = 200. Para YOLOv8s se estableció: lr0 = 0.01, optimizer = SGD, batch = 16 weight_decay = 0.0005, epochs = 200. Por lo tanto, se destacó una diferencia entre las dos versiones, específicamente en el tamaño del lote, ya que en el primer modelo el tamaño de lote es de 8, mientras que en el segundo se estableció un batch de 16.

De tal manera se valoró el rendimiento de las dos versiones, bajo las mismas condiciones, es decir se usaron los mismos valores en ambos contextos. En el caso de YOLOv5s se utilizó un tamaño de lote de 16 y en YOLOv8s se usó un batch de 8, con el objetivo de determinar el rendimiento de los modelos en las mismas circunstancias.

En la **Tabla 29** se presenta los resultados finales alcanzados en las métricas de rendimiento después de haber ajustado los hiperparámetros en los dos modelos y con la nueva valoración para el batch size.

Tabla 29. Comparativa de rendimiento entre YOLOv5s y YOLOv8s en la validación

Modelo	Precision	Recall	mAP@0.50	mAP@0.50:0.90	Batch
YOLOv5s	0.9401	0.9194	0.9659	0.7699	8
YOLOv5s	0.**9583**	0.9414	0.**9746**	0.7874	16
YOLOv8s	0.9321	0.9325	0.9635	0.7742	8
YOLOv8s	0.9366	0.**9427**	0.9709	0.**7991**	16

6.2. Objetivo 2: Evaluar el modelo mediante pruebas de validación al identificar el tizón tardío en las hojas de papa

6.2.1. Fase 4: Pruebas y validación del modelo

En la fase de entrenamiento se determinó los valores para los hiperparpámetros (learning rate, optimizer, batch size, weight decay, epochs), dando como resultado los modelos YOLOv5s y YOLOv8s ajustados para la detección del Tizón Tardío en las hojas de papa, por lo tanto, se evaluó ambas versiones con el conjunto de pruebas.

6.2.1.1. Tarea 1: Evaluación de los modelos con el conjunto de pruebas

En la **Figura 45** se muestra el script usado para evaluar los modelos con el conjunto de pruebas; para lo cual se utilizó el mejor modelo[29] (best.pt) generado en la fase de entrenamiento (ver sección 6.1.3.2) de YOLOv5s y YOLOv8s, además se especificó usar el conjunto de pruebas (test) con split. Dicho set consta de 432 imágenes que presentan el Tizón Tardío con un total de 812 instancias (zonas etiquetadas de la enfermedad).

```
from ultralytics import YOLO
model = YOLO("                                    /runs/detect/train29/weights/best.pt")
results = model.val(
    data="                                    /DivisionData/data.yaml",
    split="test"
)

# Métricas
print(f"Precision: {results.results_dict['metrics/precision(B)']:.4f}")
print(f"Recall: {results.results_dict['metrics/recall(B)']:.4f}")
print(f"mAP50: {results.results_dict['metrics/mAP50(B)']:.4f}")
print(f"mAP50-95: {results.results_dict['metrics/mAP50-95(B)']:.4f}")
```

Figura 45. Script para evaluar YOLOv5s y YOLOv8s con el conjunto de pruebas

[29] Mejores modelos:
https://drive.google.com/drive/folders/1p6n-tcJuL4eHsP999KnX6NkDYR6m2bjo?usp=sharing

A. YOLOv5s

- **Batch 8**

En la **Figura 46** se presenta la matriz de confusión obtenida de la evaluación del modelo en el conjunto de pruebas, donde las columnas son las clases reales y las filas son las predicciones realizadas por YOLOv5s; la etiqueta background se trata del fondo, es decir dichos casos no corresponden a ninguna clase objetivo. De la matriz se identificó:

- TP (Verdaderos Positivos) instancias identificadas correctamente como ***TizonTardio***: 767 de 812.
- FN (Falsos Negativos) regiones de TizonTardio que fueron clasificadas erróneamente como background: 45.
- FP (Falsos Positivos) áreas que fueron clasificadas incorrectamente como ***TizonTardio***: de los cuales 59.

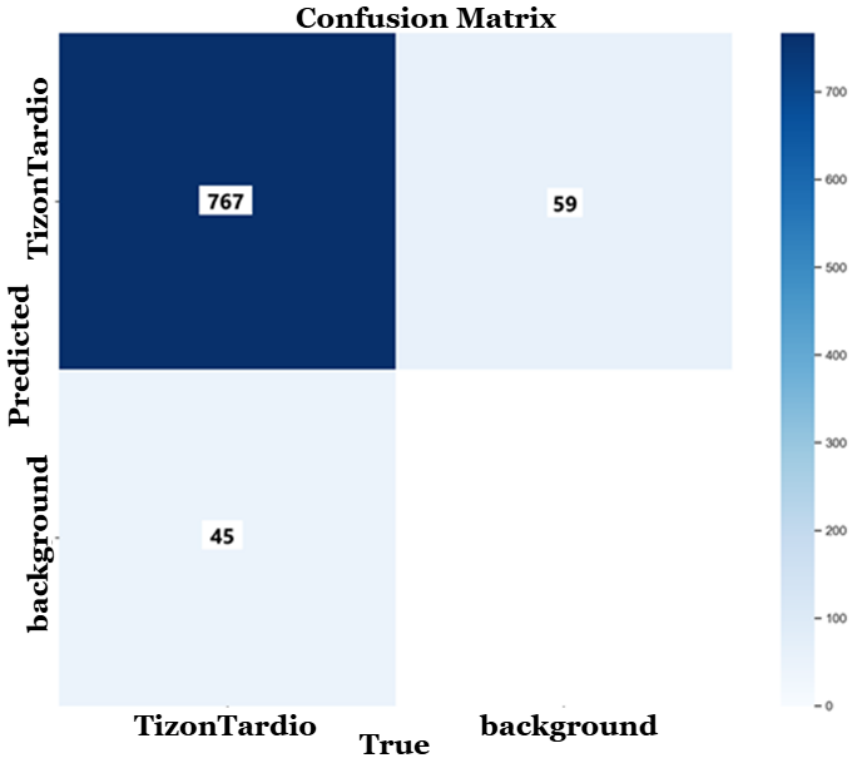


Figura 46. Matriz de confusión de YOLOv5s (batch 8) obtenida en la evaluación

- **Métricas de clasificación**
  - **Precisión**

$$\text{Precisión} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{767}{767 + 59} = 0.9286 \longrightarrow 92.86\%$$

  - **Recall (Sensibilidad)**

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{767}{767 + 45} = 0.9446 \longrightarrow 94.46\%$$

Del total de las 812 instancias etiquetadas, YOLOv5s (batch de 8) obtuvo una precisión del 92.86% y un recall de 94.46% en la clasificación de la enfermedad.

- **Bacth 16**

En la **Figura 47** se muestra la matriz de confusión obtenida de YOLOv5s (batch de 16) al detectar el Tizón Tardío en el conjunto de pruebas, donde se determinó:

- 766 (TP) áreas infectadas fueron correctamente identificados como ***TizonTardio***.
- 46 (FN) regiones que tienen ***TizonTardio*** y el modelo los clasificó incorrectamente como background (fondo).
- 41 (FP) instancias que no pertenecen a la clase y el modelo las etiqueto como ***TizonTardio***.

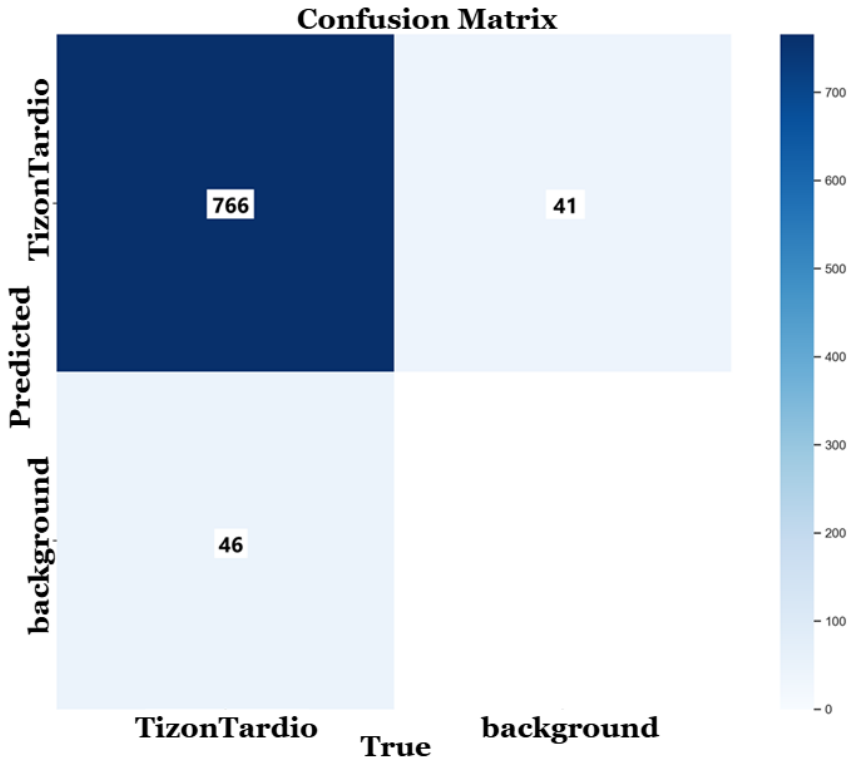


Figura 47. Matriz de confusión de YOLOv5s (batch 16) obtenida en la evaluación

- **Métricas de clasificación**
  - **Precisión**

$$\text{Precisión} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{766}{766 + 41} = 0.9492 \longrightarrow 94.92\%$$

  - **Recall (Sensibilidad)**

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{766}{766 + 46} = 0.9433 \longrightarrow 94.33\%$$

Del total de las 812 áreas que presentan la enfermedad el modelo logró clasificar un 94.92% de los casos positivos y alcanzó un recall del 94.33%.

B. YOLOv8s

- **Batch 8**

En la **Figura 48** se presenta la matriz de confusión obtenida en la evaluación de YOLOv8s (batch de 8) al detectar la enfermedad en el conjunto de pruebas. Principalmente se desatacan las siguientes características de la matriz:

- **Verdaderos Positivos (TP):** 765 casos fueron detectados correctamente como ***TizonTardio*** por el modelo.
- **Falsos negativos (FN):** 47 instancias infectadas fueron identificados como que no pertenecen a la clase (background).
- **Falsos positivos (FP):** 52 áreas que no pertenecen a la clase objetivo, el modelo detecto incorrectamente como ***TizonTardio***.

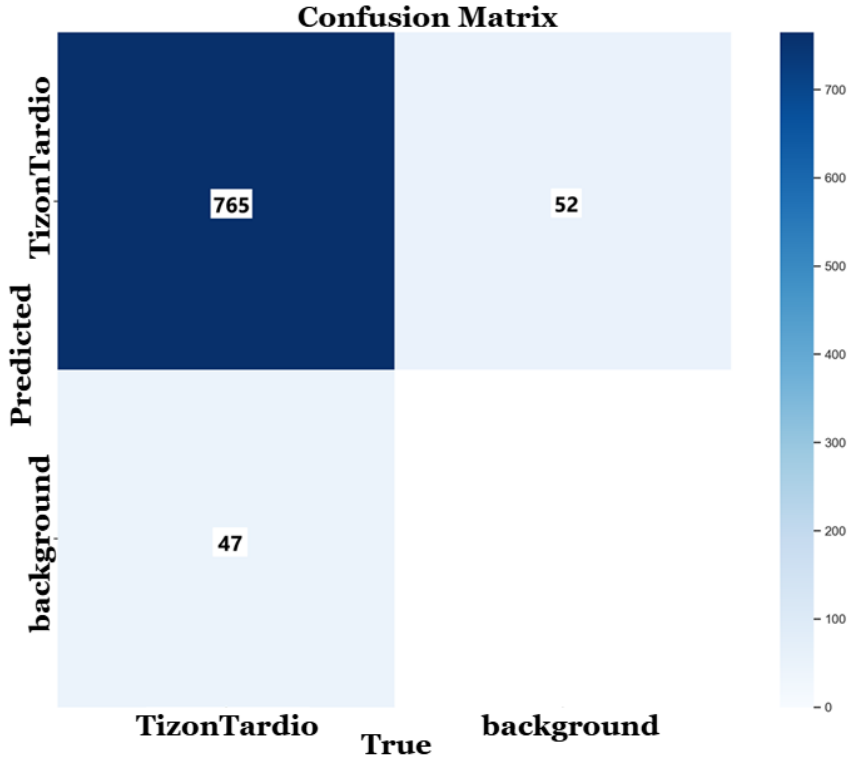


Figura 48. Matriz de confusión de YOLOv8s (batch 8) obtenida en la evaluación

- **Métricas de clasificación**
  - **Precisión**

$$\text{Precisión} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{765}{765 + 52} = 0.9364 \longrightarrow 93.64\%$$

  - **Recall (Sensibilidad)**

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{765}{765 + 47} = 0.9421 \longrightarrow 94.21\%$$

El modelo obtuvo una precisión del 93.64% al clasificar los casos positivos (áreas que presentan Tizón Tardío) y un recall del 94.21%.

- **Bacth 16**

En la **Figura 49** se muestra la matriz de confusión generada al evaluar YOLOv8s (batch de 16) en el conjunto de pruebas. Donde se destacó la siguiente información:

- **TP:** 764 instancias fueron detectadas correctamente como ***TizonTardio.***
- **FN:** 48 regiones etiquetadas enfermas el modelo las identifico erróneamente como background.
- **FP:** 50 casos que no pertenecen a la clase fueron clasificados como ***TizonTardio.***

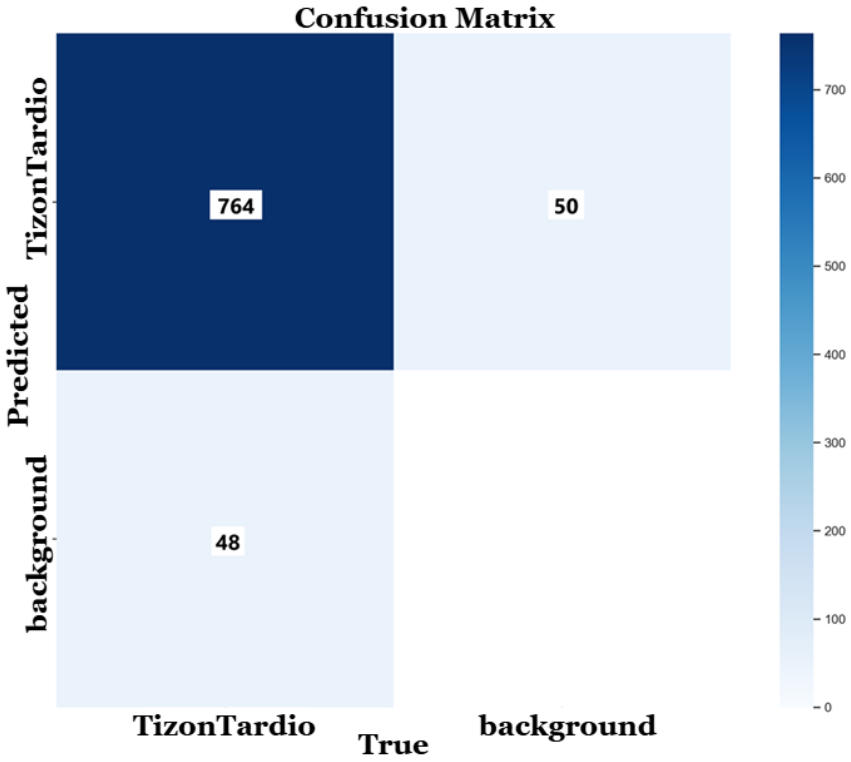


Figura 49. Matriz de confusión de YOLOv8s (batch 16) obtenida en la evaluación

- **Métricas de clasificación**
  - **Precisión**

$$\text{Precisión} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{764}{764 + 50} = 0.9386 \longrightarrow 93.86\%$$

  - **Recall (Sensibilidad)**

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{764}{764 + 48} = 0.9409 \longrightarrow 94.09\%$$

Del total de las 812 instancias etiquetadas del conjunto de pruebas, el modelo demostró que es capaz de clasificar un 93.86% de aciertos y un 94.01% de recall.

En la **Tabla 30** se presenta un resumen de los valores de la matriz de confusión obtenidos en la evaluación de YOLOv5s y YOLOv8s utilizando el conjunto de pruebas, como:

Verdaderos Positivos, Verdaderos Negativos, Falsos Positivos, Falsos Negativos y las métricas.

Tabla 30. Resumen de los valores de la matriz confusión y métricas obtenidas en la evaluación

Modelo	Valores de la Matriz				Métricas	
	TP	TN	FP	FN	Precisión	Recall
YOLOv5s (batch 8)	767	0	59	45	0.9286	0.9446
YOLOv5s (batch 16)	766	0	41	46	0.9492	0.9433
YOLOv8s (batch 8)	765	0	52	47	0.9364	0.9421
YOLOv8s (batch 16)	764	0	50	48	0.9386	0.9409

Verdaderos Positivos (TP); Verdaderos Negativos (TN); Falsos Positivos (FP); Falsos Negativos (FN).

- **Análisis de las métricas de detección**

A partir de los resultados obtenidos en la evaluación de los modelos, se basó en las métricas de detección de objetos con la finalidad de determinar el mejor modelo.

En la **Tabla 31** se presenta un resumen de los valores obtenidos en la evaluación de YOLOv5s y YOLOv8s utilizando el conjunto de pruebas, como: Verdaderos Positivos, Verdaderos Negativos, Falsos Positivos, Falsos Negativos y las métricas.

Tabla 31. Resumen de las métricas de detección obtenidas en la evaluación

Modelo	Precision	Recall	mAP@0.50	mAP@0.50:0.90
YOLOv5s (batch 8)	0.9599	0. 9138	0.9703	0.7845
YOLOv5s (batch 16)	0.9618	0.93	**0.9717**	0.7966
YOLOv8s (batch 8)	0.9545	**0.9304**	0.9709	0.7927
YOLOv8s (batch 16)	**0.9709**	0.9113	0.9709	**0.8113**

En la **Figura 50** se muestra las métricas de detección obtenidas en la evaluación con el conjunto de pruebas de los modelos YOLOv5s y YOLOv8s con diferentes tamaños de lote.

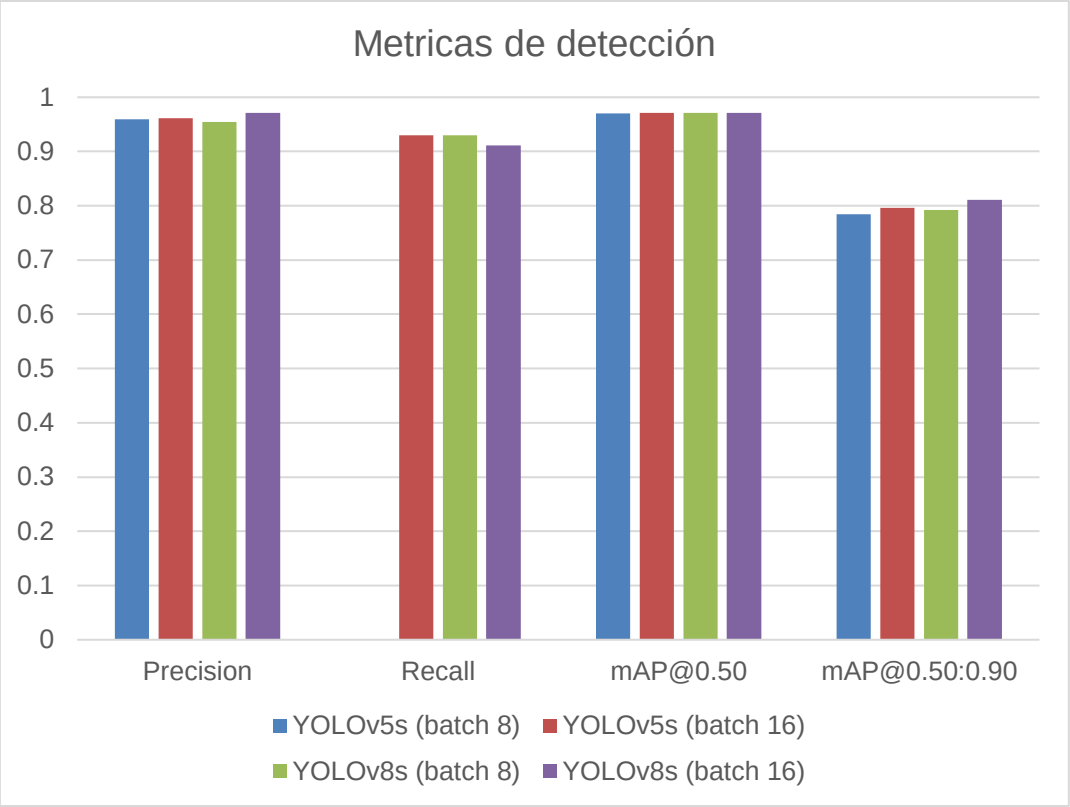


Figura 50. Métricas de detección de YOLOv5s y YOLOv8s

En base a los resultados obtenidos YOLOv8s (batch de 16) destacó, ya que logró la mayor precisión (97.09%) y mAP@0.50:0.90 (81.13%) en comparación con los otros modelos, indicando que realiza predicciones más precisas y localiza mejor las zonas de interés en diferentes umbrales de Intersection over Union (IoU).

6.2.1.2. Tarea 2: Evaluar el modelo en un entorno simulado

En la tarea anterior se determinó YOLOv8s como el modelo que generó mejores resultados en la detección de la enfermedad, por tanto, se utilizó para crear el prototipo web, con la finalidad de facilitar el uso del modelo al usuario en la identificación del Tizón Tardío en las hojas de papa.

El prototipo se desarrolló en base a las fases: definición de requerimientos, diseño del software, integración y pruebas, de la metodología de desarrollo de software Cascada; a continuación, se presentan los resultados obtenidos:

- **Fase de definición de requerimientos**
  - **Requisitos de prototipo para evaluación del modelo**

En la **Tabla 32** se presentan los requisitos funcionales establecidos para el desarrollo del prototipo.

El sistema debe permitir al usuario:

Tabla 32. Requisitos funcionales para el prototipo

ID	Requisito	Descripción
RF01	Subir imágenes	El usuario podrá subir imágenes en formato JPG, PNG y JPEG.
RF02	Eliminar imagen	Se podrá eliminar la imagen previamente cargada.
RF03	Visualizar resultado	Mostrar la imagen con la respectiva detección de la enfermedad.

- **Diagrama de casos de uso**

En la **Figura 51** se presenta el diagrama de casos de uso general del prototipo web, el cual muestra la interacción del usuario con el sistema, donde puede cargar imágenes, eliminar y visualizar el resultado de la detección.

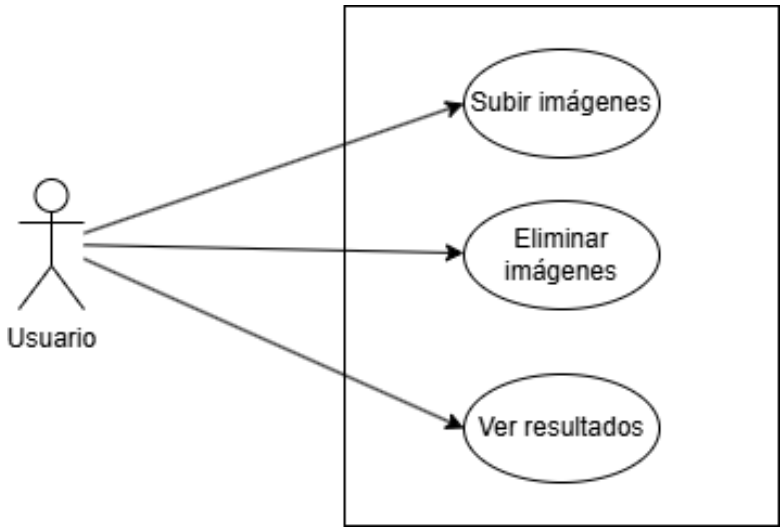


Figura 51. Diagrama de caso de uso general

- **Fase de diseño del software**

En la **Figura 52** se muestra la arquitectura del prototipo web que permite al usuario interactuar con el sistema a través del navegador donde carga imágenes, posteriormente el servidor web recibe las entradas y las envía al modelo YOLO mediante solicitud POST, además gestiona las solicitudes de eliminar, finalmente el modelo procesa las imágenes recibidas para luego retornar al servidor los resultados de las detecciones, que los muestra al usuario.

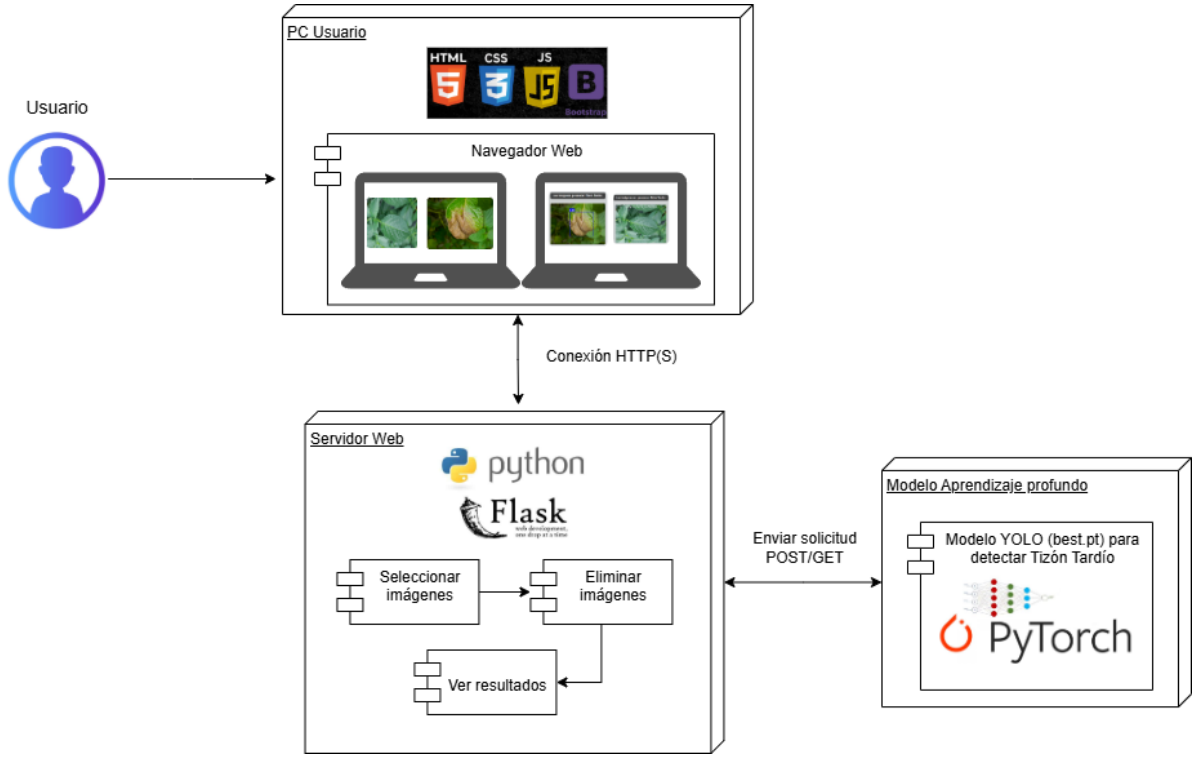


Figura 52. Arquitectura del prototipo web para detectar el Tizón Tardío

En la **Figura 53** se presenta la página principal del prototipo web[30], la cual permite realizar las siguientes acciones:

- Seleccionar imágenes.
- Ver mensaje informativo “Máximo 10 imágenes en formato PNG, JPG o JPEG”.
- Eliminar una o varias imágenes después de haber sido cargadas.
- Ejecutar la acción de detectar, la cual muestra los resultados de las detecciones realizadas por el modelo de aprendizaje profundo.

[30] Repositorio: https://github.com/GerardoQuizhpe/DetectorTizonTardio.git

Figura 53. Página principal del prototipo

En la **Figura 54** se presenta la interfaz para visualizar los resultados de las detecciones. La página divide en dos secciones las predicciones realizadas: si la predicción del Tizón Tardío, muestra las imágenes con los cuadros delimitadores con el respectivo porcentaje de confianza; en el caso que no identifica la enfermedad se presenta el resultado con un mensaje “No hay presencia de la enfermedad”. Además, la interfaz incluye un botón de Salir, que redirecciona a la página principal del prototipo web.

Figura 54. Resultados de la detección

- **Fase de integración y pruebas**
  - **Evaluación del prototipo web**

En la **Figura 55** se presenta el resultado de los casos de prueba obtenidos en la evaluación del prototipo web por parte del usuario.

ID	Nombre CP	Datos de entrada	Resultado esperado	Correcto	Fallido	Retroalimentación
CP1	Subir imágenes	Imágenes en formato JPG, PNG y JPEG.	El usuario sube las imágenes y el sistema muestra el nombre de las fotografías.	☑	☐	Ninguna
CP2	Eliminar imágenes	Identificador de la imagen a eliminar.	El usuario elimina las imágenes correspondientes y el sistema las quita.	☑	☐	Ninguna
CP3	Visualizar resultado	Solicitud del usuario para ver el resultado.	El sistema muestra los resultados de la predicción en dos secciones: con presencia del Tizón Tardío y sin presencia de la enfermedad.	☑	☐	Ninguna

Figura 55. Casos de prueba del prototipo web

De acuerdo a los resultados obtenidos en la evaluación del prototipo web, se destaca que el software desarrollado fue aprobado por el Ing. Angel Robles, ya que cumplió satisfactoriamente con todas las funcionalidades descritas (ver **Anexo 7**).

- **Evaluación del modelo por parte del experto**

La evaluación del modelo fue realizado por el Ing. Angel Robles a través del manejo del prototipo web, la **Figura 56** presenta una parte de las detecciones realizadas por el modelo en las imágenes de prueba[31].

Las predicciones completas se encuentran almacenadas en Google Drive[32].

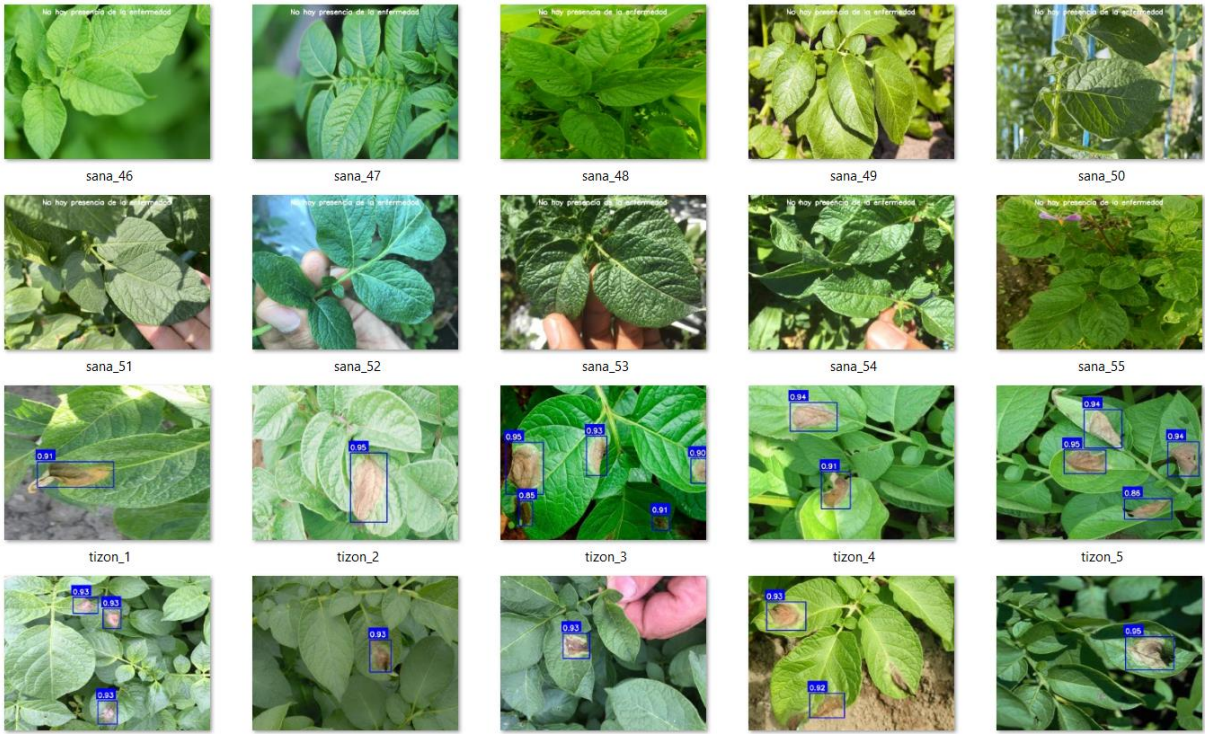


Figura 56. Muestra de detecciones realizadas por el modelo

[31] Imágenes de prueba: https://drive.google.com/drive/folders/1rfyBhOKzlCL7Fa1lc62Wbb0M287CJR6u?usp=sharing
[32] Detecciones realizadas un entorno simulado: https://drive.google.com/drive/folders/1sGPDq0nylHeErLkria8ZCpEO25NdOz79?usp=sharing

En la **Tabla 33** se presenta los resultados obtenidos durante la evaluación del modelo, detallando las zonas identificadas con Tizón Tardío según el criterio del experto, así como la cantidad de imágenes que son sanas. Además, se muestra las predicciones correctas, incorrectas e incompletas realizadas por el modelo YOLOv8s (ver **Anexo 6**).

Tabla 33. Resultados de la evaluación en un entorno simulado

Clase	Conteo real (Experto)	Predicciones correctas	Predicción incompleta	Predicciones incorrectas
Tizón Tardío	91 (100%)	83 (91 %)	8 (9 %)	2
Sana	0	54	0	1

$$\text{Precisión} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{83}{83 + 2} = 0.9765 \ \longrightarrow 97.65\%$$

En base a los resultados obtenidos se demostró que el modelo alcanzó una precisión del 97.65% lo que significa que es confiable para la tarea de identificar el Tizón Tardío en las hojas del cultivo de papa.

7. Discusión

7.1 Primer objetivo: Implementar un modelo de detección de objetos para identificar la infestación de las hojas de papa por el tizón tardío utilizando YOLOv5 y YOLOv8

La metodología CRISP-ML(Q) es un enfoque que organiza y facilita las tareas necesarias en cada una de sus fases, a diferencia de los trabajos relacionados (TR1 al TR11) los cuales no usan ningún tipo de metodología, para proyectos de Machine Learning, sino que se rigen a tareas establecidas por los autores. En la ejecución del presente trabajo, CRISP-ML(Q), constituyó un pilar fundamental para lograr un enfoque iterativo centrado en la calidad del desarrollo de cada una de sus fases, garantizando un marco estructurado y riguroso, además es una de las metodologías que comúnmente se usan en el desarrollo de modelos de aprendizaje profundo a comparación de KDD o CRISP-DM que también son usadas para minería de datos.

Por otra parte, los trabajos TR1 al TR11 usan un dataset en concreto, lo que puede conllevar a limitar el aprendizaje de los modelos; el TR1 y TR5 en cambio usan el set PlantVillage para detectar la enfermedad antes mencionada, no obstante, dicho conjunto muestra una hoja en específico, mientras que el TR4 emplea un dataset con fotografías que abarca más el área de la planta de papa; sin embargo, dicho conjunto no es de acceso público. Ante lo mencionado se recolectó un set propio, con un total de 266 fotografías que presentan el Tizón Tardío; logrando crear un dataset personalizado que contiene múltiples conjuntos de datos, destacando la diversidad de situaciones en la que se presenta la enfermedad. El conjunto de datos creado consta de 1661 imágenes, al cual se aplicó técnicas de aumento de datos como voltear horizontalmente y verticalmente, agregar ruido, ya que los trabajos TR4, TR5, TR6, TR8, TR9, TR10 resaltan que la aplicación de dichas técnicas, entre otras, ayudan a mejorar la calidad del dataset. Sin embargo, la mayor parte del conjunto utilizado son imágenes que muestra una hoja de papa con el tizón tardío, por lo que puede haber una brecha en la detección de la enfermedad en fotografías que abarque más de una hoja y con diferentes áreas infestadas.

Se usó la división de los datos más utilizada en los trabajos relacionados (TR1, TR2, TR6, TR7) destacando el 80% entrenamiento, 10% validación, 10% pruebas, ya que dicha proporción garantiza una adecuada generalización del modelo y evaluación del rendimiento en datos no usados. Al revisar los trabajos TR2 al TR11, se determinó que YOLOv5s y YOLOv8s son los modelos que obtuvieron un rendimiento adecuado en la detección de enfermedades foliares, resaltando en las métricas de precisión y mAP (0.50, 0.50:0.90), por tal razón se usaron estos modelos en el TIC.

Por otra parte, acorde a los trabajos TR1, TR2 y TR3 donde emplean el método OFAT (One Factor At a Time) como una técnica de búsqueda y ajuste de hiperparámetros en

modelos de aprendizaje profundo (CNN, YOLO), en el presente TIC, se utilizó dicho enfoque, para analizar cómo los valores de los parámetros influyen en el rendimiento de los modelos; el método OFAT es un enfoque sencillo, no obstante, en la práctica presentó la limitante de no permitir analizar la relación entre parámetros a diferencia de métodos avanzados como optimización bayesiana, GWO (optimizador de lobo gris), WOA (algoritmo de optimización de ballenas), PSO (optimización de enjambre de partículas), entre otras, lo que puede llevar a establecer una configuración no óptima para los modelos de aprendizaje profundo.

En el entrenamiento de YOLOv5s y YOLOv8s se determinó la configuración que generó los mejores resultados en las métricas, learning rate=0.01, optimizer=SGD, epochs=200, weight decay=0.0005, con la diferencia en el batch size=8 para YOLOv5s y 16 YOLOv8s, corroborando con lo que muestra el TR2, al aplicar la técnica OFAT donde obtuvo los mismos valores en el learning rate, epochs, weight decay, optimizer, con la diferencia en el batch=48; además se comprobó que el optimizador RMSProp genera los resultados más bajos a comparación de Adam, SGD, Adamax, AdamW; mientras que el TR3 obtuvo el mejor rendimiento con 100 epochs, batch=64 y weight decay=0.00005.

Finalmente, YOLOv5s obtuvo una precisión del 94.01%, recall de 91.95%, mAP@0.50 del 96.59% y mAP@0.50:0.90 de 76.99%, mientras que YOLOv8s logró un 93.66%, 94.27%, 97.09% y 79.91%, respectivamente en las métricas.

7.2 Segundo objetivo: Evaluar el modelo mediante pruebas de validación al identificar el tizón tardío en las hojas de papa

La fase de evaluación constituyó la base para determinar el rendimiento de los modelos entrenados al detectar el tizón tardío en imágenes que no han visto durante el entrenamiento, ni en la validación. Por lo tanto, al evaluar YOLOv5s y YOLOv8s con el conjunto de pruebas, se determinó que ambas versiones lograron resultados adecuados para detectar el tizón tardío en las hojas de papa con una precisión del 96.18% y 97.09% respectivamente; no obstante, YOLOv8s demostró mejor rendimiento tanto en precisión como en mAP@0.50:0.90 del 81.83%, que YOLOv5s con un 79.66%.

De igual manera el TR11 corrobora que YOLOv8 logra la mayor puntuación en las métricas de mAP@0.50 (90.9%), mAP@0.50:0.90 (60.2%), debido a que integra una arquitectura optimizada, en concreto mejora la capa de extracción de características CSPDarknet al reemplazar CSPLayer (Capa de conexiones parciales entre etapas) por C2f (Cuello de botella parcial de etapa cruzada con dos convoluciones) mejorando la precisión de la detección e incluye anclajes adaptativos lo que permite detectar objetos en diferentes tamaños; el TR9 compara diferentes versiones de YOLO, destacando YOLOv8 con un mAP@0.50 del 99.04% al identificar hojas afectadas por plagas, lo que demuestra los resultados obtenidos en la evaluación con la variante (small) de la versión 8.

Es importante mencionar que los TR se limitan a evaluar el modelo con el conjunto de pruebas y no experimentan con nuevos datos (imágenes), careciendo de la integración de aplicativos para validar el modelo en un entorno real o simulado; se ha identificado un trabajo que si implementa un prototipo móvil TR9, no obstante no aplica una metodología para desarrollar la aplicación; ante lo mencionada en el presente trabajo se logra desarrollar un prototipo web basado en la metodología de desarrollo de software Cascada que permite la evaluación del modelo YOLOv8s por parte de usuarios finales en un entorno simulado.

Las pruebas desarrolladas por el experto en enfermedades foliares, permitió evaluar el rendimiento del modelo YOLOv8s con 110 imágenes nuevas (55 sanas, 55 infectadas) a través del prototipo; las fotografías fueron analizadas por el técnico y posteriormente comparó su análisis con las predicciones del modelo, logrando una precisión del 97.65% en la detección, al contrario de los TR donde no interviene una persona experimentada. Se demuestra que el modelo es capaz y adecuado para identificar el tizón tardío, ya sea en imágenes que están en condiciones controladas, como también en fotografías que presentan escenarios reales. Por lo tanto, el modelo desarrollado puede ser integrado en un sistema lo que facilitaría el desarrollo de aplicaciones para detectar la enfermedad y puede ser usado como un punto de referencia para extender su generalización con más enfermedades foliares.

A partir de los resultados obtenidos en las fases de entrenamiento y evaluación de los modelos YOLOv5s, YOLOv8s, se aborda la pregunta de investigación planteada “¿Qué porcentaje de precisión se obtiene al ajustar YOLOv5 y YOLOv8 mediante la configuración de hiperparámetros para identificar la enfermedad del tizón tardío en las hojas de papa?”. Utilizando el método One Factor At a Time (OFAT) para el ajuste de hiperparámetros durante la etapa de entrenamiento, el modelo YOLOv5s logró una precisión del 94.01% y YOLOv8s alcanzó un 93.66%, demostrando que ambos modelos son efectivos para identificar el tizón tardío en las hojas de papa; no obstante, aunque YOLOv5s fue superior en el entrenamiento con un 0.35% en la precisión que YOLOv8s, en la fase de evaluación se demostró que la versión más reciente logró mejores resultados alcanzando una precisión del 97.09%, en comparación con el 95.99% que obtuvo YOLOv5s, debido a la arquitectura de YOLOv8s optimizada y la mayor capacidad para manejar datos complejos o variados lo que permite mayor generalización del modelo. Por lo tanto, el modelo y el prototipo desarrollado constituye un recurso valioso para las agricultores o profesionales dedicados al cultivo de papa para identificar el tizón tardío de manera temprana, ya que reduce el tiempo de detección.

8. Conclusiones

- La implementación del método OFAT (One Factor At a Time) como una técnica de ajuste de hiperparámetros permitió identificar la configuración adecuada; para YOLOv8s se estableció un learning rate de 0.01, optimizer SGD, weight decay de 0.0005, 200 epochs y un batch size de 16; en el caso de YOLOv5s, se determinaron los mismos valores, con la diferencia en el batch size que fue de 8. Además, los modelos mencionados lograron un rendimiento adecuado en las métricas de: precisión, recall, mAP@0.50 y mAP@0.50:0.90, donde YOLOv5s alcanzó un 94.01%, 91.95%, 96.59% y 76.99%, mientras que YOLOv8s logró un 93.66%, 94.27%, 97.09% y 79.91% respectivamente en las métricas; los resultados indican que ambos modelos presentan un desempeño adecuado, sin embargo, YOLOv8s muestra una ligera ventaja en recall y mAP (@0.50, @0.50:0.90), lo que determina que tiene mejor capacidad para detectar correctamente el tizón tardío en diferentes condiciones.
- La evaluación permitió analizar el rendimiento de los modelos YOLOv5s y YOLOv8s bajo las mismas condiciones al detectar el tizón tardío en las hojas de papa, con el conjunto de pruebas. Los resultados muestran que la versión más reciente ofrece una mejor capacidad en la detección y localización de la enfermedad con mayor precisión en diferentes umbrales de intersección, determinándolo la opción más eficiente para la tarea, ya que obtuvo una precisión del 97.09%, recall de 91.13%, mAP@0.50 97.09% y mAP@0.50:0-90 81.13%, a comparación de las métricas obtenidas con YOLOv5s que logró una precisión del 96.18%, recall de 93%, mAP@0.50 de 97.17% y mAP@0.50:0.90 del 79.66%.
- Los hiperparámetros influyen directamente en la velocidad de convergencia, capacidad de generalización y estabilidad del entrenamiento en los modelos de aprendizaje profundo. La tasa de aprendizaje controla el tamaño de ajuste de pesos; el optimizador establece la estrategia de actualización de los pesos; el tamaño de lote afecta la estabilidad y eficiencia del entrenamiento; el decaimiento de pesos actúa como regulador previniendo el sobreajuste y el número de épocas determina cuántas veces el modelo recorre el conjunto de datos de entrenamiento, estos parámetros son los que tienen mayor impacto en el aprendizaje e inferencia en los modelos.

9. Recomendaciones

- Utilizar conjuntos de datos que incluyan imágenes capturadas en diferentes condiciones como brillo, ángulo, momentos del día y la presencia de múltiples zonas infectadas por el tizón tardío en una misma fotografía.
- Implementar versiones ligeras/pequeñas de YOLO, como las variantes nano o small, ya que los modelos están diseñados para ser más eficientes debido a su menor tamaño de parámetros que posee la arquitectura, permitiendo su ejecución en dispositivos con menor capacidad de cómputo.
- Usar herramientas de etiquetado que facilite aplicar técnicas de aumento de datos y etiquetado automatizado como Roboflow, con la finalidad de reducir el tiempo y esfuerzo al realizar las tareas antes mencionadas.
- Hacer uso de plataformas en la nube como Google Colab para entrenar modelos de aprendizaje profundo de manera más eficiente y sin inconvenientes, ya que proporciona acceso gratuito a GPUs y TPUs.
- Implementar métodos de optimización avanzados para el ajuste de hiperparámetros, como: WOA (algoritmo de optimización de ballenas), GWO (optimizador de lobo gris), optimización bayesiana, PSO (optimización de enjambre de partículas).
- Implementar un aplicativo móvil, para facilitar el acceso a usuarios, especialmente en áreas rurales o de campo, donde el uso de dispositivos portátiles puede ser limitado.
- Ampliar la cantidad de enfermedades foliares en el conjunto de datos para mejorar la capacidad de generalización del modelo, con la finalidad de que no se limite a la detección de una sola clase.

10. Bibliografía

[1] S. J. Z. Chila, C. M. Espinoza, and M. E. E. Martínez, “Cultivo de la papa y sus condiciones climáticas,” Gestión ingenio y sociedad, vol. 2, no. 2, pp. 140–152, 2017.

[2] F. Yumisaca, V. López, D. Peñaherrera, J. Camacho, and C. Tello, “ALERTA TEMPRANA PARA EL MANEJO DEL TIZÓN TARDIO DE LA PAPA. ATN/RF 16678 RG PRODUCTO# 7. SISTEMA DE ALERTA TEMPRANA DE TIZÓN TARDÍO EN ECUADOR,” 2022.

[3] G. del M. Chanco Casahuilca, “Validación del Sistema de Apoyo a la Toma de Decisiones Para el Manejo del Tizón Tardío de la Papa (Solanum Tuberosum L.),” 2022.

[4] Y. Wang, H. Zhang, and G. Zhang, “cPSO-CNN: An efficient PSO-based algorithm for fine-tuning hyper-parameters of convolutional neural networks,” Swarm Evol Comput, vol. 49, pp. 114–123, 2019, doi: https://doi.org/10.1016/j.swevo.2019.06.002.

[5] M. D. Siddiqi, B. Jiang, R. Asadi, and A. Regan, “Hyperparameter Tuning to Optimize Implementations of Denoising Autoencoders for Imputation of Missing Spatio-temporal Data,” Procedia Comput Sci, vol. 184, pp. 107–114, 2021, doi: https://doi.org/10.1016/j.procs.2021.04.001.

[6] W.-Y. Lee, S.-M. Park, and K.-B. Sim, “Optimal hyperparameter tuning of convolutional neural networks based on the parameter-setting-free harmony search algorithm,” Optik (Stuttg), vol. 172, pp. 359–367, 2018, doi: https://doi.org/10.1016/j.ijleo.2018.07.044.

[7] A. Sevilla Rivadeneira and W. A. Vásquez Castillo, “Evaluación de componentes de resistencia genética de papa (Solanum tuberosum) al tizón tardío (Phytophthora infestans) en condiciones controladas,” Universidad de las Américas, Quito, 2018. [Online]. Available: Vásquez Castillo, Wilson Arturo

[8] R. C. Solano Porras, “Aplicación de herramientas de apoyo de toma de decisiones sostenible para el manejo del tizón tardío de la papa en Huasahuasi-Tarma 2018,” 2020.

[9] L. E. R. Enríquez, J. L. M. Portilla, and R. C. L. Pozo, “Inteligencia artificial e innovación:: campos de aplicación para la industria del Ecuador,” Visión empresarial, no. 9, pp. 163–172, 2019.

[10] P. Mooney and E. T. C. Grupo, “La insostenible agricultura 4.0,” Digitalización y poder corporativo en la cadena alimentaria. Ver en< https://bit. ly/3ep6Rxh, 2019.

[11] S. Santos Valle and J. Kienzle, “Agricultura 4.0-Robótica agrícola y equipos automatizados para la producción agrícola sostenible,” 2021.

[12] A. U. Castaneda, “¿ Será la Agricultura 4.0 la solución al hambre global?,” Realidad y Reflexión, no. 57, pp. 39–58, 2023.

[13] F. Fouquet Calderón, “Inteligencia Artificial aplicada a la agricultura de precisión. Control de hongos en la planta de tomate,” 2021.

[14] J. L. Pérez, “Impacto de las tecnologías disruptivas en la percepción remota: big data, internet de las cosas e inteligencia artificial,” UD y la Geomática, no. 14, 2019.

[15] J. Díaz-Ramírez, “Aprendizaje automático y aprendizaje profundo,” Ingeniare. Revista chilena de ingeniería, vol. 29, no. 2, pp. 180–181, 2021.

[16] L. Rouhiainen, “Inteligencia artificial,” Madrid: Alienta Editorial, pp. 20–21, 2018.

[17] L. J. Sandoval Serrano, “Algoritmos de aprendizaje automático para análisis y predicción de datos,” Revista Tecnológica; no. 11, 2018.

[18] R. F. López and J. M. F. Fernández, Las redes neuronales artificiales. Netbiblo, 2008.

[19] W. R. Asanza and B. M. Olivo, “Redes neuronales artificiales aplicadas al reconocimiento de patrones,” Editorial UTMACH, vol. 1, no. 4, p. 5, 2018.

[20] W. R. Asanza and B. M. Olivo, “Redes neuronales artificiales aplicadas al reconocimiento de patrones,” Editorial UTMACH, vol. 1, no. 4, p. 5, 2018.

[21] F. Izaurieta and C. Saavedra, “Redes neuronales artificiales,” Departamento de Física, Universidad de Concepción Chile, 2000.

[22] A. González Muñiz, “Aplicaciones de técnicas de inteligencia artificial basadas en aprendizaje profundo (deep learning) al análisis y mejora de la eficiencia de procesos industriales,” Feb. 2018, Accessed: Sep. 28, 2024. [Online]. Available: http://hdl.handle.net/10651/45097

[23] M. Massiris, C. Delrieux, and J. Á. Fernández Muñoz, “Detección de equipos de protección personal mediante red neuronal convolucional YOLO,” in XXXIX Jornadas de Automática, Área de Ingeniería de Sistemas y Automática, Universidad de Extremadura, 2018, pp. 1022–1029.

[24] D. Mery, “Visión por computador,” Jan. 2004.

[25] A. Khan, “Machine Learning in Computer Vision,” Procedia Comput Sci, vol. 167, Apr. 2020, doi: 10.1016/j.procs.2020.03.355.

[26] C. A. P. Pereira, “Sistema de visión computarizada y herramientas de diseño gráfico para la obtención de imágenes de muestras de alimentos segmentadas y promediadas en coordenadas CIE-L* a* b,” Agronomía Costarricense, vol. 33, no. 2, pp. 283–301, 2009.

[27] J. Durán Suárez, “Redes neuronales convolucionales en R: Reconocimiento de caracteres escritos a mano,” 2017.

[28] D. A. Silva Carnero, “ANÁLISIS DE ARQUITECTURAS CNN DEL ESTADO DEL ARTE EN EL RECONOCIMIENTO DE ACTIVIDADES POR VIDEO,” Universidad Autónoma de Chihuahua, 2021. [Online]. Available: http://repositorio.uach.mx/id/eprint/357

[29] J. Ma, M. Lin, G. Zhou, and Z. Jia, “Joint Image Restoration For Domain Adaptive Object Detection In Foggy Weather Condition,” in 2024 IEEE International Conference on Image Processing (ICIP), 2024, pp. 542–548. doi: 10.1109/ICIP51287.2024.10647560.

[30] K. G. Nalbant and Ş. Uyanık, “Computer Vision in the Metaverse,” Journal of Metaverse, vol. 1, no. 1, pp. 9–12, 2021, [Online]. Available: https://dergipark.org.tr/en/pub/jmv/issue/67581/1051377

[31] X. Wu, D. Sahoo, and S. C. H. Hoi, “Recent advances in deep learning for object detection,” Neurocomputing, vol. 396, pp. 39–64, 2020, doi: https://doi.org/10.1016/j.neucom.2020.01.085.

[32] J. Terven, D.-M. Córdova-Esparza, and J.-A. Romero-González, “A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS,” Mach Learn Knowl Extr, vol. 5, no. 4, pp. 1680–1716, Nov. 2023, doi: 10.3390/make5040083.

[33] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 779–788. doi: 10.1109/CVPR.2016.91.

[34] D. Horcajada Jiménez, “Metodología para la detección de objetos en imágenes basada en la librería YOLO con aplicación a la detección de carros,” 2021.

[35] F. Fahim and M. S. Hasan, “Enhancing the reliability of power grids: A YOLO based approach for insulator defect detection,” e-Prime - Advances in Electrical Engineering, Electronics and Energy, vol. 9, p. 100663, 2024, doi: https://doi.org/10.1016/j.prime.2024.100663.

[36] R. Rajamohanan and B. C. Latha, “An Optimized YOLO v5 Model for Tomato Leaf Disease Classification with Field Dataset,” Engineering, Technology & Applied Science Research, vol. 13, no. 6, pp. 12033–12038, 2023.

[37] E. Casas, L. Ramos, C. Romero, and F. Rivas-Echeverría, “A comparative study of YOLOv5 and YOLOv8 for corrosion segmentation tasks in metal surfaces,” Array, vol. 22, p. 100351, 2024, doi: https://doi.org/10.1016/j.array.2024.100351.

[38] G. Wen et al., “YOLOv5s-CA: A modified YOLOv5s network with coordinate attention for underwater target detection,” Sensors, vol. 23, no. 7, p. 3367, 2023.

[39] M. Hussain, “YOLOv5, YOLOv8 and YOLOv10: The Go-To Detectors for Real-time Vision,” arXiv preprint arXiv:2407.02988, 2024.

[40] K. Lan, X. Jiang, X. Ding, H. Lin, and S. Chan, “High-Efficiency and High-Precision Ship Detection Algorithm Based on Improved YOLOv8n,” Mathematics, vol. 12, no. 7, p. 1072, 2024.

[41] Z. Huangfu, S. Li, and L. Yan, “Ghost-YOLO v8: An Attention-Guided Enhanced Small Target Detection Algorithm for Floating Litter on Water Surfaces,” Computers, Materials

and Continua, vol. 80, no. 3, pp. 3713–3731, 2024, doi: https://doi.org/10.32604/cmc.2024.054188.

[42] S. Xie and H. Sun, “Tea-YOLOv8s: A tea bud detection model based on deep learning and computer vision,” Sensors, vol. 23, no. 14, p. 6576, 2023.

[43] L. Ramos, E. Casas, E. Bendek, C. Romero, and F. Rivas-Echeverría, “Hyperparameter optimization of YOLOv8 for smoke and wildfire detection: Implications for agricultural and environmental safety,” Artificial Intelligence in Agriculture, vol. 12, pp. 109–126, 2024, doi: https://doi.org/10.1016/j.aiia.2024.05.003.

[44] M. D. Zeiler, “ADADELTA: an adaptive learning rate method,” arXiv preprint arXiv:1212.5701, 2012.

[45] T. Schaul, S. Zhang, and Y. LeCun, “No more pesky learning rates,” in International conference on machine learning, PMLR, 2013, pp. 343–351.

[46] D. J. Im, M. Tao, and K. Branson, “An empirical analysis of the optimization of deep network loss surfaces,” arXiv preprint arXiv:1612.04010, 2016, doi: https://doi.org/10.48550/arXiv.1612.04010.

[47] H. Robbins and S. Monro, “A stochastic approximation method,” The annals of mathematical statistics, pp. 400–407, 1951.

[48] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” arXiv preprint arXiv:1412.6980, 2014, doi: https://doi.org/10.48550/arXiv.1412.6980.

[49] L. K. Meng, L. J. Xin, H. H. Yi, Z. A. A. Salam, and N. B. Wei, “A machine learning approach for face mask detection system with AdamW optimizer,” J Appl Technol Innov, vol. 7, no. 3, p. 25, 2023.

[50] I. Loshchilov and F. Hutter, “Decoupled weight decay regularization,” arXiv preprint arXiv:1711.05101, 2017.

[51] S. Omatu and J. M. Corchado, “Advances in Distributed Computing and Artificial Intelligence Journal,” ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal, vol. 9, no. 2, España, 2020. doi: https://doi.org/10.14201/ADCAIJ202092.

[52] D. Vijendra Babu, C. Karthikeyan, Shreya, and A. Kumar, “Performance Analysis of Cost and Accuracy for Whale Swarm and RMSprop Optimizer,” IOP Conf Ser Mater Sci Eng, vol. 993, no. 1, p. 012080, 2020, doi: 10.1088/1757-899X/993/1/012080.

[53] S. Reddy, K. Reddy, and V. Vallikumari, “Optimization of deep learning using various optimizers, loss functions and dropout,” International Journal of Recent Technology and Engineering, vol. 7, pp. 448–455, Jan. 2018.

[54] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein, “Visualizing the loss landscape of neural nets,” Adv Neural Inf Process Syst, vol. 31, 2018.

[55] A. Krogh and J. Hertz, “A simple weight decay can improve generalization,” Adv Neural Inf Process Syst, vol. 4, 1991.

[56] A. Quevedo López and D. Domínguez, “Métodos de inteligencia artificial aplicados a datos de turismo,” Universidad Autónoma de Madrid, 2019. [Online]. Available: http://hdl.handle.net/10486/689521

[57] C. L. Corso, “Aplicación de algoritmos de clasificación supervisada usando Weka,” Córdoba: Universidad Tecnológica Nacional, Facultad Regional Córdoba, 2009.

[58] F. Khan, N. Zafar, M. N. Tahir, M. Aqib, H. Waheed, and Z. Haroon, “A mobile-based system for maize plant leaf disease detection and classification using deep learning,” Front Plant Sci, vol. 14, May 2023, doi: 10.3389/fpls.2023.1079366.

[59] F. Coenen, “On the use of confusion matrixes,” Department of Computer Science, University of Liverpool, 2012.

[60] D. Gomez et al., “Advancing common bean (Phaseolus vulgaris L.) disease detection with YOLO driven deep learning to enhance agricultural AI,” Sci Rep, vol. 14, no. 1, p. 15596, 2024, doi: 10.1038/s41598-024-66281-w.

[61] L. Chen, G. Li, S. Zhang, W. Mao, and M. Zhang, “YOLO-SAG: An improved wildlife object detection algorithm based on YOLOv8n,” Ecol Inform, vol. 83, p. 102791, 2024, doi: https://doi.org/10.1016/j.ecoinf.2024.102791.

[62] A. D. Pérez Calderón and M. Valero García, “Prototipado de visión por computadora para la detección de objetos en drones autónomos,” Universitat Politècnica de Catalunya, 2024. Accessed: Sep. 28, 2024. [Online]. Available: http://hdl.handle.net/2117/411614

[63] Tzutalin, Bryan Russell, Antonio Torralba, and William T. Freeman, “labelImg.” Accessed: Sep. 29, 2024. [Online]. Available: https://github.com/HumanSignal/labelImg

[64] S. Studer et al., “Towards CRISP-ML(Q): A Machine Learning Process Model with Quality Assurance Methodology,” Mach Learn Knowl Extr, vol. 3, no. 2, pp. 392–413, Apr. 2021, doi: 10.3390/make3020020.

[65] “CRISP-ML(Q). The ML Lifecycle Process.” Accessed: Sep. 29, 2024. [Online]. Available: https://ml-ops.org/content/crisp-ml

[66] L. Egas, J. Jativa, and L. Garcés, “Evolución de la Metodologías de Desarrollo de la Ingeniería de Software en el proceso la Ingeniería de Sistemas Software y Determinación de una metodología adaptable orientada a una organización pequeña,” 2016, EcuCiencia.

[67] C. G. Prieto Álvarez, “Adaptación de las metodologías tradicionales cascada y espiral para la inclusión de evaluación inicial de usabilidad en el desarrollo de productos de software en México,” REPOSITORIO NACIONAL CONACYT, 2015.

[68] M. Rodríguez Parra and G. A. Ortiz Cárdenas, “Diseño de un sistema de gestión de proyectos para el desarrollo de software basado en la metodología híbrida entre marcos de trabajo Cascada y Scrum,” 2022.

[69] J. C. Ojeda and M. del C. G. Fuentes, “Taxonomía de los modelos y metodologías de desarrollo de software más utilizados,” Universidades, no. 52, pp. 37–47, 2012.

[70] I. Sommerville, Ingeniería del software, vol. Tercera edición. Pearson educación, 2005.

[71] I. Sommerville, INGENIERÍA DE SOFTWARE, vol. Novena edición. México: Pearson Education, 2011.

[72] D. Rolon-Mérette, M. Ross, T. Rolon-Mérette, and K. Church, “Introduction to Anaconda and Python: Installation and setup,” Quant. Methods Psychol, vol. 16, no. 5, pp. S3–S11, 2016.

[73] E. Bisong, Building machine learning and deep learning models on Google cloud platform. Springer, 2019.

[74] M. Kuroki, “Using Python and Google Colab to teach undergraduate microeconomic theory,” International Review of Economics Education, vol. 38, p. 100225, 2021, doi: https://doi.org/10.1016/j.iree.2021.100225.

[75] R. González Duque, “Python para todos,” 2011, Creative Commons Reconocimiento.

[76] J. S. Walker, Python: La Guía Definitiva para Principiantes para Dominar Python. Babelcube Inc., 2018.

[77] S. Imambi, K. B. Prakash, and G. R. Kanagachidambaresan, “PyTorch,” in Programming with TensorFlow: Solution for Edge Computing Applications, K. B. Prakash and G. R. Kanagachidambaresan, Eds., Cham: Springer International Publishing, 2021, pp. 87–104. doi: 10.1007/978-3-030-57077-4_10.

[78] N. Ketkar, J. Moolayil, N. Ketkar, and J. Moolayil, “Introduction to pytorch,” Deep learning with python: learn best practices of deep learning models with PyTorch, pp. 27–91, 2021.

[79] C. Nvidia and F. H. Fitzek, “Cuda,” Online http://www. nvidia. com/object/cuda_home_new, vol. 15, 2006.

[80] A. Zelinsky, “Learning OpenCV---Computer vision with the OpenCV library (Bradski, GR et al.; 2008)[On the Shelf],” IEEE Robot Autom Mag, vol. 16, no. 3, p. 100, 2009.

[81] J. Salvatore, J. Osio, and M. Morales, “Detección de objetos utilizando el sensor Kinect,” Guayaquil, Ecuador, LACCEI, 2014.

[82] F. I. G. César, A. F. Fernandes, D. M. B. F. Oian, C. A. Oian, and I. K. Makiya, “MELHORIA DA QUALIDADE UTILIZANDO ONE FACTOR AT A TIME (OFAT) ATRAVÉS DO DESIGN OF EXPERIMENTS (DoE): ESTUDO DE CASO EM UMA INDÚSTRIA DE POLIESTIRENO EXPANDIDO (EPC),” RECIMA21-Revista Científica Multidisciplinar-ISSN 2675-6218, vol. 5, no. 10, pp. e5105760–e5105760, 2024.

[83] Y.-S. Lee, M. P. Patil, J. G. Kim, Y. B. Seo, and G. Do Kim, “Hyperparameter Optimization of Apple Leaf Dataset for the Disease Recognition Based on the Yolov8,” Available at SSRN 5002502.

[84] G. Samaniego Rocha, “Análisis de estrategias de experimentación en columna para biosorción de metales pesados con residuos de bajo coste,” Universitat Politècnica de Catalunya, Barcelona, 2015. [Online]. Available: http://hdl.handle.net/2117/84180

[85] O. D. Gómez Cárdenas, “Optimización de un método de microextracción en gota orgánica flotante (SFODME) de Cu (II) en agua mediante diseño de experimentos,” 2022.

[86] M. Copperwaite and C. Leifer, Learning flask framework. Packt Publishing Ltd, 2015.

[87] D. F. Ningtyas and N. Setiyawati, “Implementasi Flask Framework pada Pembangunan Aplikasi Purchasing Approval Request,” Jurnal Janitra Informatika dan Sistem Informasi, vol. 1, no. 1, pp. 19–34, 2021.

[88] D. Ghimire, “Comparative study on Python web frameworks: Flask and Django,” 2020.

[89] M. Grinberg, Flask web development. “ O’Reilly Media, Inc.,” 2018.

[90] A. H. Renear, S. Sacchi, and K. M. Wickett, “Definitions of dataset in the scientific and technical literature,” Proceedings of the American Society for Information Science and Technology, vol. 47, no. 1, pp. 1–4, Nov. 2010, doi: https://doi.org/10.1002/meet.14504701240.

[91] T. A. Asfaw, “Deep learning hyperparameter’s impact on potato disease detection,” The Scientific Temper, vol. 14, no. 03, pp. 582–590, Sep. 2023, doi: 10.58414/SCIENTIFICTEMPER.2023.14.3.04.

[92] Y. ZARROUK, M. YANDOUZI, M. GRARI, M. BOURHALEB, M. RAHMOUNE, and K. HACHAMI, “REVOLUTIONIZING POTATO LATE BLIGHT SURVEILLANCE: UAV-DRIVEN OBJECT DETECTION INNOVATIONS,” J Theor Appl Inf Technol, vol. 102, no. 7, 2024.

[93] E. A. Aldakheel, M. Zakariah, and A. H. Alabdalall, “Detection and identification of plant leaf diseases using YOLOv4,” Front Plant Sci, vol. 15, Apr. 2024, doi: 10.3389/fpls.2024.1355941.

[94] H. Wang, S. Shang, D. Wang, X. He, K. Feng, and H. Zhu, “Plant Disease Detection and Classification Method Based on the Optimized Lightweight YOLOv5 Model,” Agriculture, vol. 12, no. 7, p. 931, Jun. 2022, doi: 10.3390/agriculture12070931.

[95] I. Agustian, R. Faurina, S. I. Ishak, F. P. Utama, K. D. Dinata, and N. Daratha, “Deep learning pest detection on Indonesian red chili pepper plant based on fine-tuned YOLOv5,” International Journal of Advances in Intelligent Informatics, vol. 9, no. 3, p. 383, Oct. 2023, doi: 10.26555/ijain.v9i3.864.

[96] Md. J. A. Soeb et al., “Tea leaf disease detection and identification based on YOLOv7 (YOLO-T),” Sci Rep, vol. 13, no. 1, p. 6078, 2023, doi: 10.1038/s41598-023-33270-4.

[97] N. Chitraningrum et al., “Comparison Study of Corn Leaf Disease Detection based on Deep Learning YOLO-v5 and YOLO-v8,” Journal of Engineering and Technological Sciences, vol. 56, no. 1, pp. 61–70, Feb. 2024, doi: 10.5614/j.eng.technol.sci.2024.56.1.5.

11. Anexos

Anexo 1. Entrevista realizada a la docente de la asignatura de Machine Learning

Entrevista sobre la Identificación del tizón tardío en las hojas de papa utilizando un modelo basado en detección de objetos

I. PLANIFICACIÓN DE LA ENTREVISTA

- **Duración:** 15 - 30 minutos.
- **Fecha de entrevista:** 18-06-2024
- **Participantes:**
  - **Entrevistado:** Genoveva Jackeline Suing Albito docente de Machine Learning de la carrera de Computación de la Universidad Nacional de Loja, Facultad de la Energía, las Industrias y los Recursos Naturales no Renovables.
  - **Entrevistador:** Gerardo Quizhpe estudiante de la carrera de Computación de la Universidad Nacional de Loja, actualmente me encuentro cursando el 8vo ciclo, estoy investigando acerca de las diferentes métricas que influyen en la evaluación de un modelo para identificar imágenes infectadas por el tizón tardío en las hojas de la papa.
- **Audio de la entrevista (visitar):**

  https://drive.google.com/drive/folders/1-fFywC6eSNr06DxGUEv0kYYmBM0jp3M4?usp=sharing

II. TRANSCRIPCIÓN DE LA ENTREVISTA

Preámbulo:

Entrevistador: El día de hoy nos encontramos, entrevistando a la ingeniería Genoveva Suing docente de la materia Machine Learning, con el objetivo de obtener una comprensión más profunda sobre las mejores prácticas y consideraciones que se deben de llevar a cabo en la identificación de imágenes infectadas por el tizón tardío, el impacto del pre-procesamiento de imágenes y la modificación de los hiperparámetros, especialmente en relación con las métricas de evaluación. Sus respuestas son valiosas puesto que permite obtener un enfoque de cómo llevar a cabo la mejor solución para generar resultados satisfactorios del proyecto a abordar.

1. **¿Considera que la precisión y f1-score son las métricas más relevantes para medir el rendimiento en la clasificación de imágenes? Si no es el caso, ¿cuáles considera más relevantes y por qué?**

De acuerdo a lo que me menciona en su proyecto, está tratando de hacer el reconocimiento de una enfermedad, generalmente en algoritmos de clasificación se quiere identificar si la hoja está sana o no, se suele incluir siempre la métrica de precisión pero en este caso si le toca evaluar un poco más a lo mejor las características hacer alguna representación, ya no únicamente identificación de clases sino identificar algunos escenarios o características de las formas de la hoja. Por lo tanto involucraría un poco más en métricas para detección de

objetos a parte de f1-score y precisión, la identificación de la zona de interés (IoU), que evalúa y compara la zona que se está identificando, así como esas existen algunas otras métricas dentro del proceso para detección de objetos. A mi criterio no se trata solo de clasificación de imágenes, sino detección de objetos, pero las dos métricas mencionadas si son bastante acertadas para poder evaluar un modelo.

2. **¿Considera que el uso de un conjunto de datos donde las clases están desbalanceadas, afectan la precisión y f1-score?**

De acuerdo a la teoría se recomienda trabajar con grupos homogéneos, es decir las clases que se están evaluando, por lo general las métricas si tienden a ser mejores si se trabaja de esa manera. Por ejemplo hay 1000 imágenes de una clase infectadas y 200 la clase sana, en este caso sí afectaría, pues la teoría destaca que siempre se debe trabajar de manera homogénea, de acuerdo a las clases que se está evaluando. Entonces el algoritmo no reconoce la clase menor porque el modelo tendría que aprender por igual de las dos o varias clases que esté identificando, en consecuencia la clase sana que tiene menos cantidad de datos, tiende a predicciones erróneas porque es de la que menos va a aprender.

3. **¿De acuerdo a su experiencia cree que la proporción de división de datos influye en las métricas de un modelo? ¿Cual seria la mejor opción que recomendaría?**

Si influye notablemente cuando uno hace la separación. La teoría especifica poder trabajar con un 60% de entrenamiento y el resto se distribuye en pruebas y validación; esos tres conjuntos siempre se los trabajan, pero las características de separación se las puede definir de acuerdo a la arquitectura de la red con la que se va a trabajar. En ciertos casos se menciona un 60%, 70% o 80% para entrenamientos y el resto se distribuye en los conjuntos test y validación.

4. **¿Considera que es necesario el etiquetado de imágenes para identificar la zona determinada de la enfermedad y el modelo aprenda mejor las características, cual recomienda?**

Para poder hacer la detección de objetos directamente se tendría que incluir el etiquetado; en cuanto a herramientas existe una variedad. Hay herramientas libres, de paga, incluso herramientas de programación que trae consigo el módulo; por ejemplo en matlab cuenta con una sección o módulo (LabelImg), permitiendo hacer el etiquetado dentro de la plataforma y solo hacer la lectura. No obstante hay varias herramientas que puede usar, para lo cual tiene que realizar un breve análisis de ventajas y desventajas para definir la mejor opción.

5. **¿Qué algoritmos de optimización recomienda para aumentar el rendimiento de una red CNN para la clasificación de imágenes?**

Es un poco complejo hablar de optimización dentro de una arquitectura, porque es una caja negra. Por lo general las redes neuronales convolucionales intentar optimizar el modelo, se

tiene que analizar la arquitectura para poder variar puesto que una red neuronal internamente tiene funciones matemáticas y es una combinación de varias funciones; entonces poder optimizar a nivel de modelo o arquitectura, sería bastante complejo en todo caso no se trataría de optimizar sino mejorar las métricas relacionadas. A diferencia de algoritmos de caja blanca que son más flexibles, sí puede optimizar el modelo pues uno puede manejar o ir alterando y moviendo; pero en algoritmos de caja negra como las redes neuronales convolucionales es un poco más trabajoso hablar de optimización dentro de ese modelo. Lo que se puede es tener mejores condiciones de evaluación en métricas modificando los hiperparámetros, porque son cajas negras. A diferencia de que uno pueda crear su modelo desde cero, tiene la flexibilidad de variar internamente y optimizar sería para alcanzar mejores valores en métricas.

6. **¿Considera que la calidad de las imágenes, formato, diferentes condiciones (como fondo, ángulo, iluminación) pueden afectar al rendimiento de un modelo en términos de precisión y f1-score?**

Existen distintos factores porque hay arquitecturas que sí distinguen el objeto sea de día o de noche. Por ejemplo hay arquitecturas que distinguen, una cierta sección en la hoja entera, como una sección en la esquina de la hoja. En cuanto a características como iluminación, fondo, etc., sí influye porque no se visualiza de manera correcta, pero si etiqueta la sección se puede mejorar. También la estandarización de datos en este caso las imágenes se tiene que hacer de manera correcta para que pueda el modelo aprender mejor.

En cuestión de imágenes con factores no controlados tiene que analizar las diferentes características, para que pueda entender el modelo. No obstante se debe tomar en cuenta que el modelo va a funcionar en campo directo, es decir directamente a la planta; además tiene que tener cierta configuración, que le permita entender o ver la forma de la planta, pues la validación siempre la va a considerar en campo.

7. **¿Según su criterio cuál considera la mejor alternativa para aumentar el rendimiento de un modelo, la modificación de los hiperparámetros o el uso de técnicas de pre-procesamiento de imágenes?**

Las técnicas antes mencionadas van de la mano, porque si no configura o no trata de mejorar las métricas serán bajas; puede tomar en consideración aumentar el número de capas, número de épocas y otras configuraciones. Por lo tanto no puede llegar a comprobar o tratar de tener mejores métricas, puesto que también influye mucho el procesamiento, si no hay una limpieza correcta en los datos se va a empezar a confundir el modelo.

Por consiguiente las técnicas de preprocesamiento y configuración de hiperparámetros siempre se usan y forman parte del proceso sistemático es decir lo que requiere para poder entrenar el modelo.

8. **¿Qué técnicas de pre-procesamiento de imágenes recomienda usar para mejorar el rendimiento de un modelo?**

Hoy en día hay varias técnicas pero puede consultar o apegarse a algún tipo de metodología donde especifique cada una de las pautas; pues el uso de una metodología da una guia como iniciar, por ejemplo analizar técnicas referente a tamaño, nitidez, etiquetado, pues es importante apegarse a una metodología y poder cubrir cada una de las etapas de pre-procesamiento que se requiere y acoplara de acuerdo a las necesidades.

Por ejemplo a nivel de Software están vinculados con la metodología XP, por ende también se tiene que hacer el levantamiento de requisitos, considerando ciertas pautas necesarias para llevar a cabo. Aunque se trate de un modelo pequeño o una aplicación pequeña se debe considerar la parte de requerimientos de usuarios; es decir no se acopla toda la metodología pero sí ciertas características.

9. **¿Qué hiperparámetros de una red neuronal convolucional (CNN) tienden a mejorar la precisión y f1-score?**

No se puede garantizar, se determina mediante la experimentación, porque el comportamiento de estas estructuras no son estándar; siempre va a depender de la información que tenga, cantidad de datos distribuidos en cada conjunto y otras configuraciones se están implementado, de esta manera se experimenta y puede que resulte bien. Pero no aplica lo mismo en su información puede que tenga porcentajes un poco bajos y no se puede definir exactamente lo mismo. Cada arquitectura o modelo tiene su comportamiento propio, puede que en un caso sí le ayude y en otro no, un hiperparámetro fijo, simplemente lo puede definir haciendo experimentación.

10. **¿De acuerdo a su experiencia considera que ResNet50 o VGG16 son factibles para identificar el tizón tardío en las hojas de la papa? Si no es el caso, ¿cual recomienda?**

Referente a modelos no se puede generalizar, debido a que el comportamiento de cada arquitectura es distinto, solo mediante experimentos o mediante referencias como trabajos relacionados puede definir la mejor solución.
Se puede vincular con Yolo, hoy en día tiene una buena detección no solo imágenes, sino vídeo, es decir le da al usuario final una detección de imagen, sino que puede trascender hacia la detección de vídeo, también trabaja en tiempo real. Se puede considerar Yolo una arquitectura, pero todo depende de los experimentos que haga, como la tasa de aprendizaje, datos disponibles; en caso donde tiene únicamente imágenes o quiere ampliarlo a video, entonces esa arquitectura se la puede establecida, porque se define esas dos en experimento y le llega una precisión del 50%, puede empezar a buscar y tratar de mejorar esa precisión. En

arquitecturas de caja negra no se puede garantizar que va a tener o va a alcanzar un porcentaje del 90%.

En la semana del estudiante vino el ingeniero Córdova, fue docente de la universidad, actualmente trabaja en Inteligencia artificial en Holanda, presentó algunas arquitecturas y una de ellas fue Yolo, con la que están trabajando en tiempo real, es decir diseñan sus complementos. De hecho el ingeniero Córdova tiene muchos trabajos con esa arquitectura, entonces no es que sea malo, sino que depende de la experimentación.

Por ejemplo: la detección de peces, que es un proyecto para las personas que hacen pesca y mediante la implementación de un modelo para poder hacer el reconocimiento, en donde el pez que es pequeño ya ni siquiera lo llevaban al puerto sino que ese mismo momento se identifica y devuelto al mal.

11. ¿Pero en los trabajos relacionados no destacan el uso de la arquitectura Yolo?

De acuerdo lo nombrado se infiere que los trabajos relacionados solo usan clasificación mediante redes neuronales convolucionales, por lo tanto la clasificación es coger y poner en carpetas cada una de las imágenes y entrenar un modelo; por eso no ve reflejado una arquitectura de Yolo porque se soluciona con dos carpetas y con esa arquitecturas que las está mencionando. Yolo no es que sea malo es cuestión de experimentar y a lo mejor los trabajos relacionados no lo usaron por desconocimiento.

Pero si va un poco más, aumentando el nivel ya entran a otras arquitecturas que están relacionadas con la detección de objetos por consiguiente empieza a etiquetar todas las imágenes y va a obtener las características, entonces ya se vincula la detección de objetos.

Firmado electrónicamente por:
GENOVEVA JACKELINNE
SUING ALBITO

Entrevistador
Gerardo Quizhpe

Entrevistado
Ing. Genoveva Suing

Anexo 2. Entrevista realizada al docente de la carrera de Agronomía

Universidad Nacional de Loja	Proyecto de Integración Curricular	Entrevista de la temática			
		Fecha:	2024-6-13		
		Página:	1	de	5

Entrevista sobre la gestión tradicional para identificar del tizón tardío en el cultivo de papa y la necesidad de automatizar este proceso

I. PLANIFICACIÓN DE LA ENTREVISTA

- **Duración:** 15 - 30 minutos.
- **Fecha de entrevista:** 13-06-2024
- **Participantes:**
  - **Entrevistado:** Angel Rolando Robles Carrión docente de la carrera de agronomía de la Facultad Agropecuaria UNL, especializado en enfermedades de plantas, en la actualidad imparte la materia de botánica y taxonomía vegetal.
  - **Entrevistador:** Gerardo Quizhpe estudiante de la carrera de Computación de la Universidad Nacional de Loja, actualmente me encuentro cursando el 8vo ciclo, estoy investigando acerca del método tradicional actual que tienen los estudiantes y profesionales para identificar el tizón tardío en las hojas de la papa y como se diferencia de otras enfermedades.
- **Audio de la entrevista (visitar):**

  https://drive.google.com/drive/folders/182CjdfBHLS10o4bp8jbHVac-pJRV_oE_?usp=sharing

II. TRANSCRIPCIÓN DE LA ENTREVISTA

Preámbulo:

Entrevistador: El día de hoy nos encontramos, entrevistando al ingeniero Angel Robles docente de la carrera de Agronomía, con el objetivo de obtener una comprensión más profunda sobre las prácticas y consideraciones que llevan en la actualidad para identificar el del tizón tardío y el interés en soluciones tecnológicas automatizadas que permitan ofrecer mejoras significativas. Sus respuestas son valiosas ya que permite obtener un enfoque de cómo llevar a cabo la mejor solución en base al problema a abordar y generar resultados satisfactorios del proyecto.

1. **¿Cuál es el nombre científico de la papa y el tizón tardío?**

La papa es Solanum Tuberosum y el tizón tardío phytophthora infestans.

2. **¿Cuáles son los síntomas iniciales del tizón tardío en las hojas de la papa?**

Los síntomas iniciales del tizón tardío comienza con una mancha necrótica de color negro, pero no muy profunda que comienza en las hojas más jóvenes o a veces también en las hojas bajeras y se presenta como un tipo quemazón en las hojas.

3. **¿Cuáles son las características que diferencian al tizón tardío de otras enfermedades que se pueden dar en las hojas de la papa?**

Por la naturaleza de la lesión o la mancha, el tizón temprano tiene manchas que son circulares mientras que la otra no, por esa características se las identifica.

4. **¿Qué tipo de variedades de papa siembran y cuáles de ellas son más resistentes al tizón tardío?**

Las variedades que se siembra en la universidad son chola, súper chola, bolona, entre otras. Para el tizón tardío no hay variedad de papa resistente, cuando le pega la enfermedad puede alcanzar pérdidas del 100% del cultivo.

5. **¿Qué condiciones ambientales favorecen la aparición y propagación del tizón tardío?**

Las condiciones favorables para el desarrollo del tizón tardío son temperaturas de 12, 17, 10 y 15 grados; sobre todo en las madrugadas que baja la temperatura a 10, 12 grados, es un momento propicio para que se manifieste esta enfermedad y también la alta humedad, es decir presencia de lluvias constantes. En este momento en la ciudad de Loja se tiene un clima que llueve y hace sol, por lo tanto eso influye en la aparición de la enfermedad y es muy destructiva.

6. **¿Según el tipo de papa se diferencia el tizón tardío, indique sus características?**

No el tizón tardío es similar en cualquier variedad de papa es decir se forma la misma mancha en las hojas.

7. **¿Cuáles son las mejores prácticas de manejo para prevenir y mitigar la aparición del tizón tardío en los cultivos de papa?**

Las mejores prácticas son: sembrar semillas resistentes, es decir que la semilla sea certificada por un almacén agropecuario o por el instituto nacional de investigaciones agropecuarias del Ecuador (INIAP). También se debe tomar en cuenta que antes de sembrar hay que eliminar todos los restos de cosechas anteriores, por otra parte antes de sembrar hay que eliminar todos los restos de cosechas anteriores de plantas emparentadas de la papa como tomate, pepino, entre otras; hay que eliminarlas si es posible quemarlas.

Posteriormente hay que hacer una buena desinfección del suelo con cal agrícola y coger la semilla para desinfectarla con un producto químico llamado vitavax, se desinfecta y se siembra. Cuando se está comenzando a sembrar se hacen labores culturales de deshierbe y antes de que se presente la enfermedad, se pone un producto preventivo, pues hay que adelantarse a la enfermedad si no aparece. Referente a remedios para la enfermedad del tizón tardío hay cura lancha y lanchafin que son los más usados.

8. **¿Cuál es el método tradicional que se usa para detectar el tizón tardío en las hojas de la papa y que tiempo les lleva determinar la enfermedad?**

Tanto agricultores como profesionales en cuestión de métodos, lo identificamos por una sospecha, es decir cuando hay presencia de manchas. Los profesionales cogen y enfundan la muestra para mandar a un laboratorio especializado, en donde técnicos capacitados observan la muestra bajo microscopio. En cuestión de tiempo depende por ejemplo si detectamos ahora (jueves) los resultados estarían entre el martes o miércoles (4 a 5 días).

9. **¿Considera que los estudiantes de agronomía son capaces de detectar el tizón tardío sin la presencia de un experto? Si no es el caso desde que ciclo son capaces de identificarlo?**

Del total del 100%, tal vez un 20% sea capaz de detectar la enfermedad. Desde 4 ciclo los estudiantes son capaces de detectar esta enfermedad sin la presencia de un profesional.

10. **¿Considera que es necesario el uso de tecnologías automatizadas e inteligencia artificial para detectar el tizón tardío de manera más rápida y para fines educativos?**

Claro que sería de gran ayuda para nosotros los profesionales o para los agricultores una app. Por ejemplo: teniendo una tecnología automatizada o una una herramienta, que se enfoque a la planta enferma y nos diga que se trata del tizón tardío o lancha negra; así mismo que rechace sino se trata de la enfermedad, porque hay muchas enfermedades bacterianas y otras biroticas que la persona no tenga conocimiento se puede confundir.

11. **¿Cómo considera la ayuda de una aplicación para la identificación del tizón tardío en el cultivo de papa?**

Puede ser de ayuda para fines educativos y fines industriales; por ejemplo nosotros como profesionales tengamos unas 4 o 20 hectáreas de papa, e ir monitoreando uno solo todo el cultivo entonces al tener una aplicación/herramienta tecnológica se puede ir tomando fotos con un dispositivo móvil o por drones.

En otros países tienen tecnologías que utilizan para monitorear mediante drones, toman fotografías y mandan a computadoras especiales para hacer un monitoreo, por tanto se puede decir la falta de esa tecnología en la zona para detectar la enfermedad, puesto que la lancha o tizón tardío del Ecuador no se desarrolla de la misma manera que en otros países. No obstante en Perú van a tener los mismos síntomas pero se desarrollan a veces de otras formas.

Ya que en nuestro país tenemos una temperatura más o menos entre 12 hasta 23, 24 grados y Estados Unidos puede tener temperaturas de 5, 6 grados, estas temperaturas son propicias para que se desarrolle la lancha.

Por ejemplo la historia de la lancha o tizón tardío surge porque los de Irlanda hacían la dieta con la papa, debido a esto les pegó la enfermedad de la lancha en la década de los 1840, como consecuencia todos emigración a los Estado Unidos. Por eso los americanos son descendientes de Irlanda; y los irlandeses se fueron por la hambruna, ocasionada por la aparición de esta enfermedad.

12. ¿Cual seria la mejor opción del prototipo para la aplicación móvil o web?

Sería más factible que la aplicación sea móvil porque todo agricultor o profesional lleva su celular. Para el prototipo de la aplicación se puede considerar que sea sin uso de internet pero sin uso de internet se puede tener desventajas como el acceso a una base de datos; por tanto con uso de internet la aplicación sería más factible. En el contexto de conexión a internet no hay interrupciones de señal; incluso yo tengo una aplicación doctor Live que es americana, sirve para ver enfermedades foliares de las gramíneas; esta aplicación cuesta más o menos entre $500 a $1000.

Como le comentaba no es lo mismo detectar enfermedades en el Ecuador que en otro país, por lo que todos los productos biológicos que vienen de otras partes del mundo no tienen la misma efectividad para controlar la enfermedad. Porque hay que buscar enfermedades o microorganismos o controladores que sean propiamente adaptadas a la zona y en nuestras condiciones.

13. ¿Qué requisitos considera necesarios para el prototipo de la aplicación?

En ese caso se tomaría una foto de la planta supuestamente enferma y la aplicación me indicaría que se trata de lancha o tizón tardío, también se puede considerar que sugiera los posibles controladores químicos, como no químicos es decir biológicos. En la actualidad en la carrera de agronomía enseñamos a los estudiantes a utilizar productos alternativos a los químicos.

Firmado electrónicamente por:
ANGEL ROLANDO ROBLES CARRION

Entrevistador
Gerardo Quizhpe

Entrevistado
Ing. Angel Robles

Anexo 3. Ensayo de adquisición del dataset en la Quinta Experimental La Argelia

Ensayo de adquisición del conjunto de datos mediante la captura de fotografías en la Quinta Experimental la Argelia de la Universidad Nacional de Loja

Introducción

El presente ensayo describe el proceso para la adquisición de imágenes con la enfermedad del tizón tardío en las hojas del cultivo de papa, mediante la captura de imágenes en diversas fechas, específicamente desde el 13 de junio al 6 de agosto del 2024 en la Quinta Experimental La Argelia, la misma fue dirigida por el Ing. Angel Robles docente de la carrera de Agronomía de la UNL.

Hipótesis

El conjunto de datos requerido es útil para el entrenamiento, pruebas y validación de los modelos YOLO, puede ser adquirido mediante fotografías que comprendan la enfermedad del tizón tardío presentes en las hojas del cultivo de papa.

Materiales

- Cámara de teléfono móvil, marca Infinix Note 30 pro; cuenta con las siguientes especificaciones: cámara trasera de 108 Megapíxeles con una apertura de f/1.8.
- Muestra hojas de papa con la enfermedad del tizón tardío.

Procedimiento

- El investigador se ubica en las plantaciones de la Quinta Experimental La Argelia, para recolectar muestras del tizón tardío presentes en las hojas del cultivo de papa.
- Se toma en cuenta una distancia entre 5 cm a 15 cm sobre la cámara y la hoja infestada, para la captura de las fotografías.
- Se capturan diversas imágenes de la enfermedad en diferentes condiciones climáticas: días soleados y nublados.
- Se capturan las fotografías en diferentes momentos del día: por la mañana, al mediodía, por la tarde y desde diferentes ángulos.
- Finalmente se seleccionan las fotografías que tengan buena resolución y nitidez, mediante la revisión del conjunto de imágenes adquiridas.

Resultados

Una vez capturadas las fotografías se obtiene un dataset de 266 imágenes con una dimensión de 3968 x 2976 píxeles con una resolución horizontal y vertical de 72 PPP (puntos por pulgada) mediante la cámara del dispositivo móvil Infinix Note 30 pro. Se transfieren todas las imágenes a un directorio local en el ordenador del investigador y un respaldo en la nube (Google Drive), alcanzando un conjunto de imágenes sólido, donde se aprecia de manera clara la presencia del tizón tardío.

Conclusión

- El ensayo demostró que la adquisición del dataset personalizado mediante la captura de fotografías que presentan el tizón tardío en las hojas de papa, facilita la implementación de las fases como: recolección de los datos, división (entrenamiento, pruebas y validación) y el entrenamiento del modelo de aprendizaje profundo.

Bibliografía

[1] A. Reyes, “Diseño de un sistema semiautomático para lavado de botellas de vidrio tipo III, de 750 ml de capacidad,” Universidad Nacional de Loja, Loja, 2017. [Online]. Available: http://dspace.unl.edu.ec/jspui/handle/123456789/18535

[2] N. Shabrina, S. Indarti, R. Maharani, D. Kristiyanti, Irmawati, N. Prastomo, and T. Mutiara, “A novel dataset of potato leaf disease in uncontrolled environment,” Data in Brief , vol. 52, p. 109955, 2024.

[3] A. M P and A. P. Reddy, “Dataset of groundnut plant leaf images for classification and detection,” Data in Brief, vol. 48, p. 109185, 2023.

[4] J. Gao, J. C. Westergaard, E. H. R. Sundmark, M. Bagge, E. Liljeroth, and E. Alexandersson, “Automatic late blight lesion recognition and severity quantification 88 based on field imagery of diverse potato genotypes by deep learning,” Knowl Based Syst, vol. 214, p. 106723, 2021, doi: https://doi.org/10.1016/j.knosys.2020.106723.

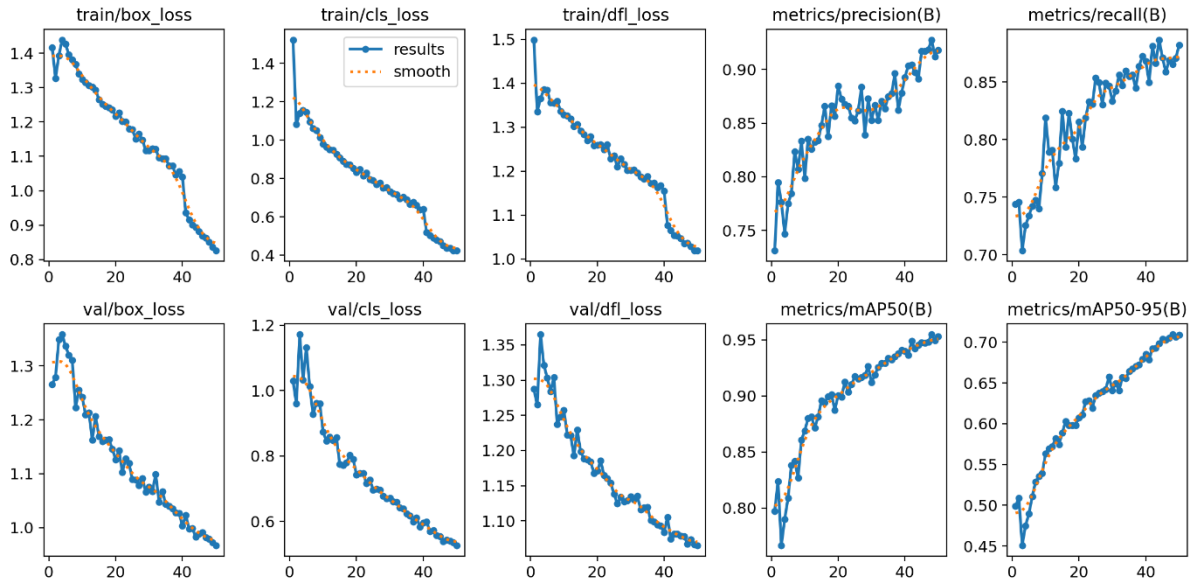
Imágenes de la adquisición del conjunto de datos a través de la captura de fotografías del tizón tardío en las hojas de papa:

Anexo 4. Gráficas de rendimiento del modelo YOLOv5s durante el entrenamiento

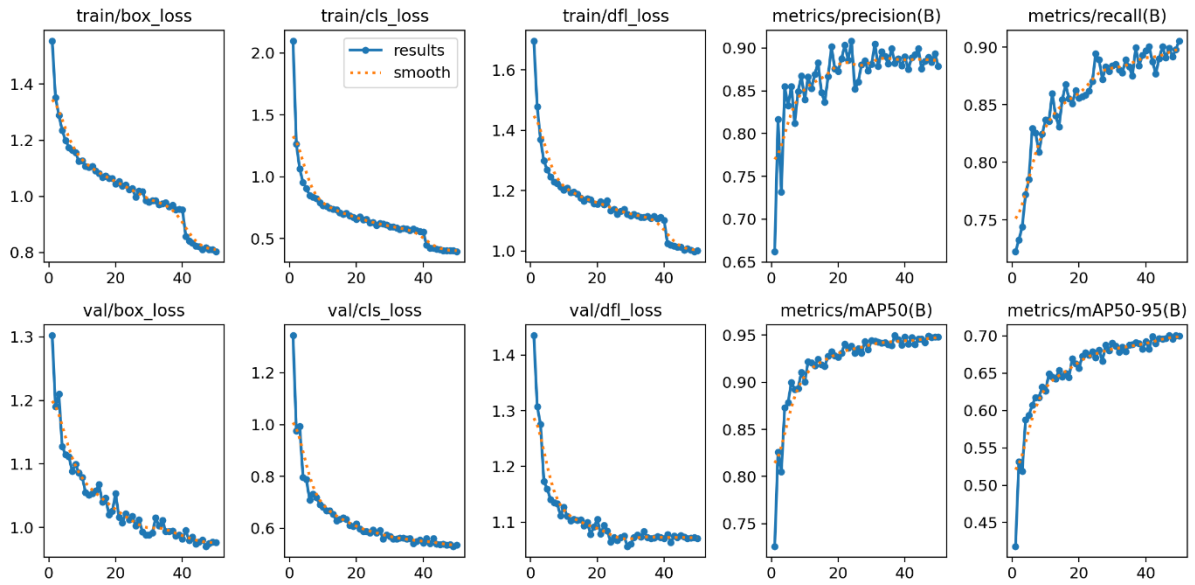
Se presenta a continuación ejemplos representativos de las gráficas de rendimiento al experimentar con los hiperparámetros (Learning rate, Optimizer, Batch size, Weight decay, Epochs) en YOLOv5s durante el entrenamiento del modelo. Las gráficas completas están disponibles en Google Drive[33].

Tasa de aprendizaje (Learning rate)

- **Learning rate (0.01)**

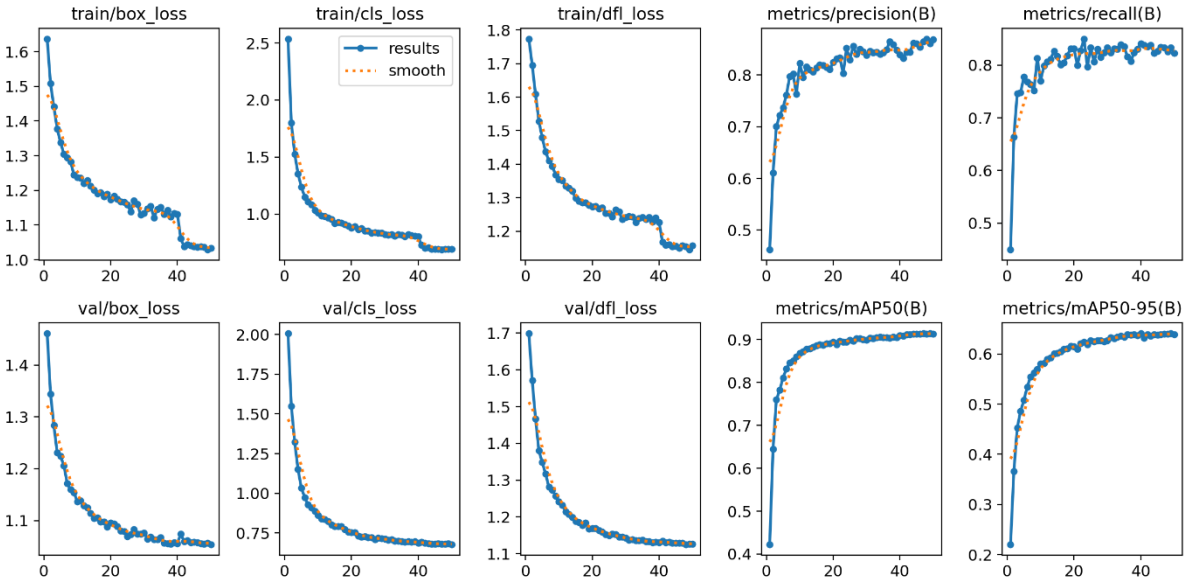


- **Learning rate (0.001)**



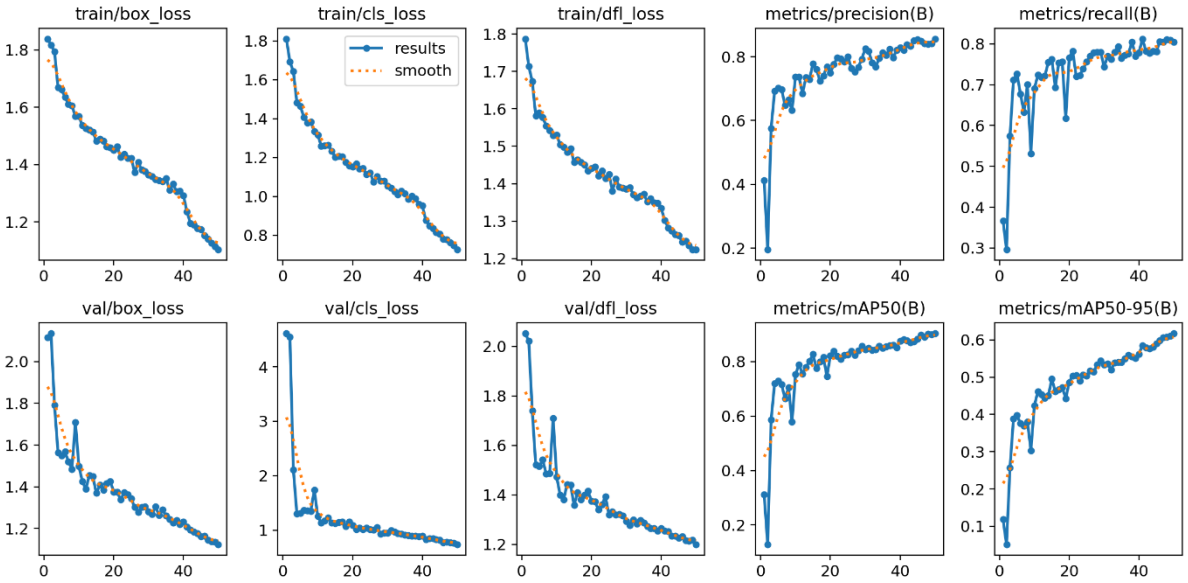
[33] Repositorio de las gráficas de YOLOv5s:
https://drive.google.com/drive/folders/14bj1VLoIPqsSyjT6ainLAKJhyOqmOoSK?usp=sharing

- **Learning rate (0.0001)**

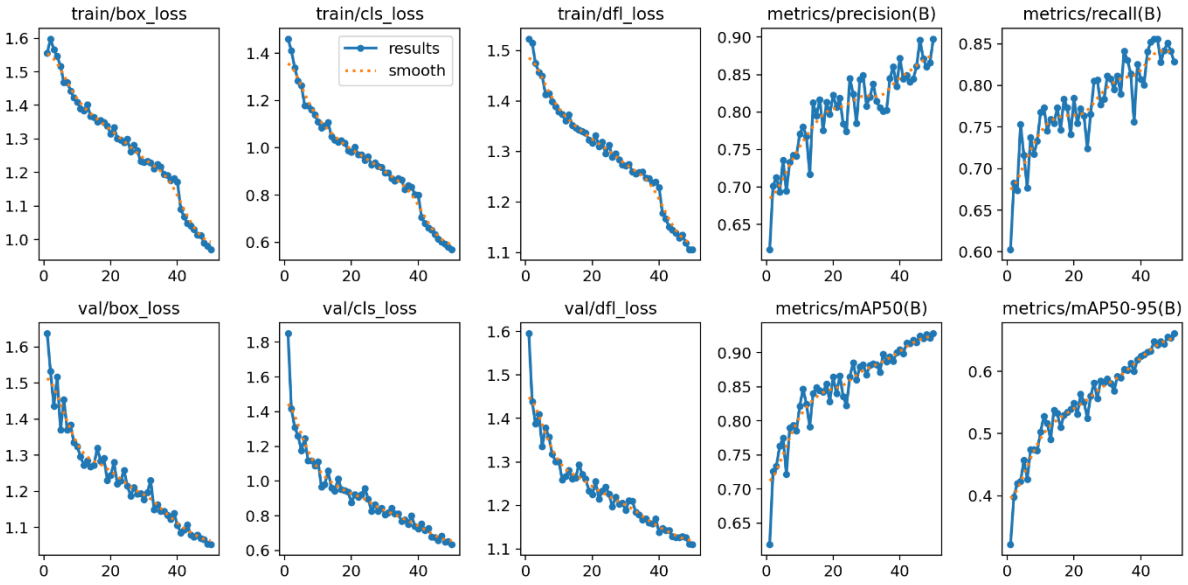


Optimizador (Optimizer)

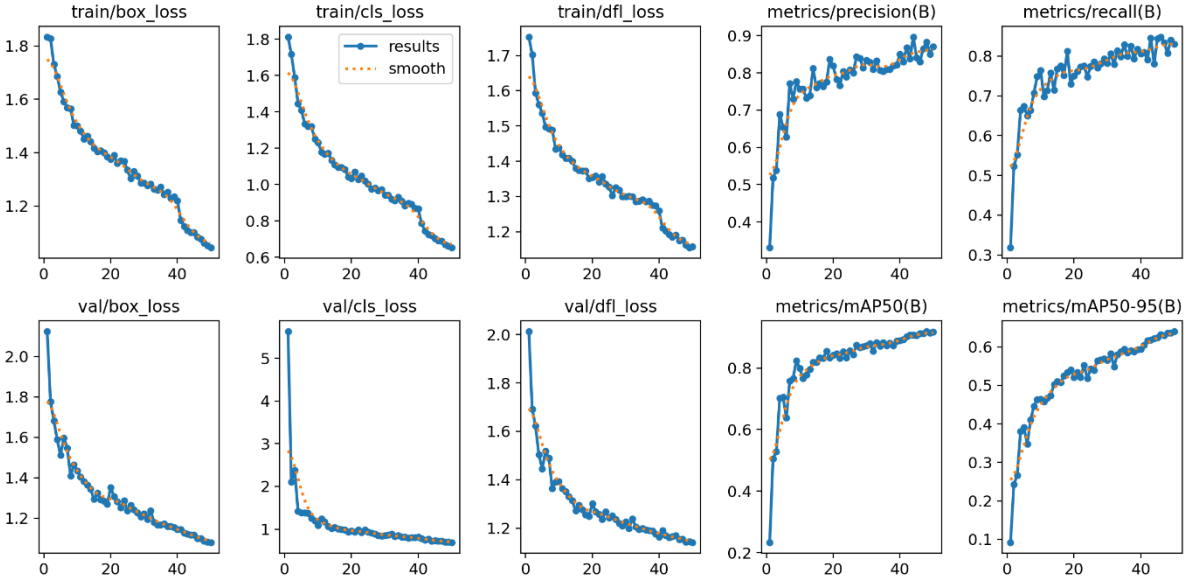
- **Optimizer (Adam)**



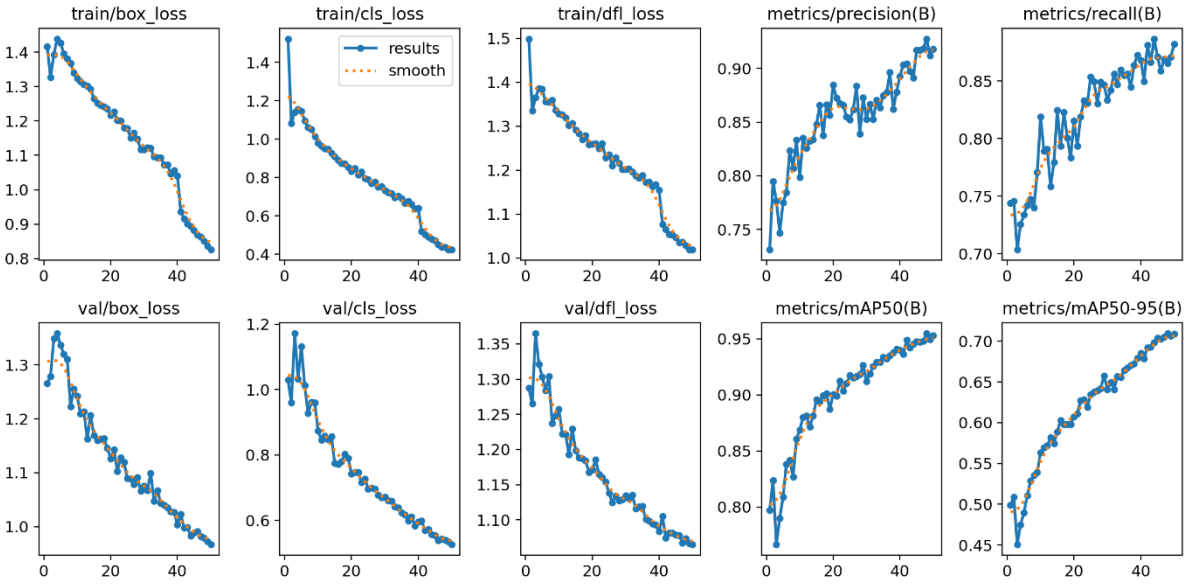
- **Optimizer (AdaMax)**



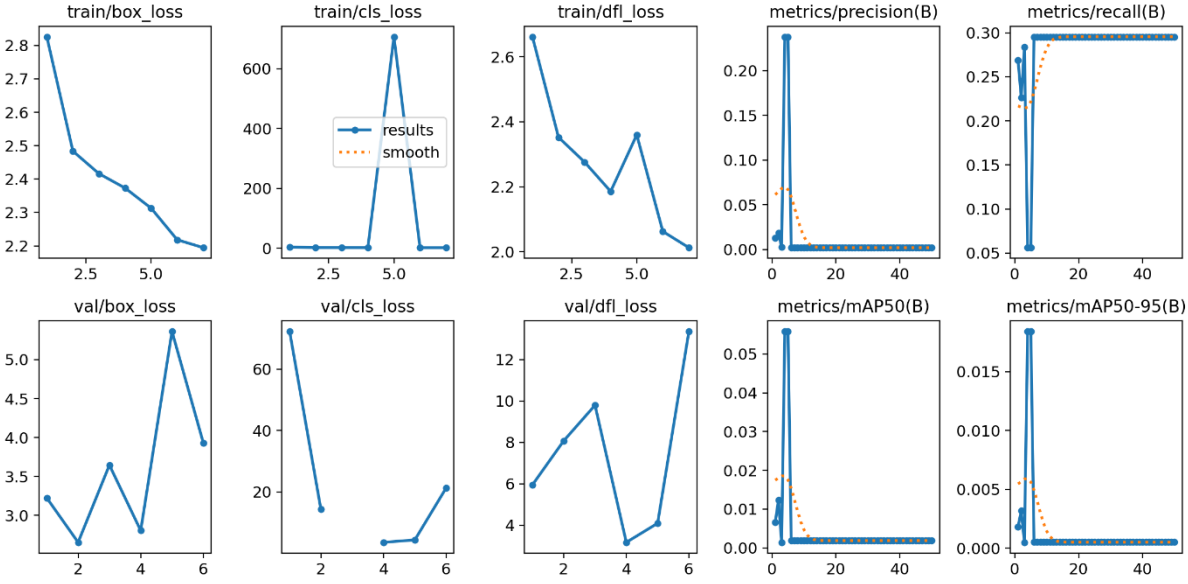
- **Optimizer (AdamW)**



- **Optimizer (SGD)**

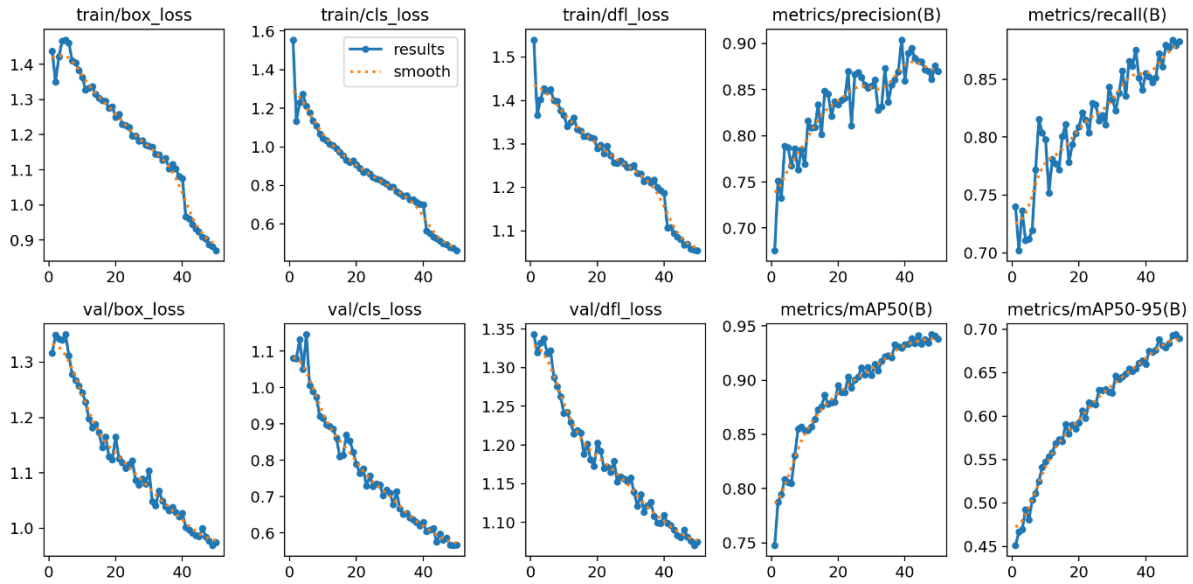


- **Optimizer (RMSProp)**

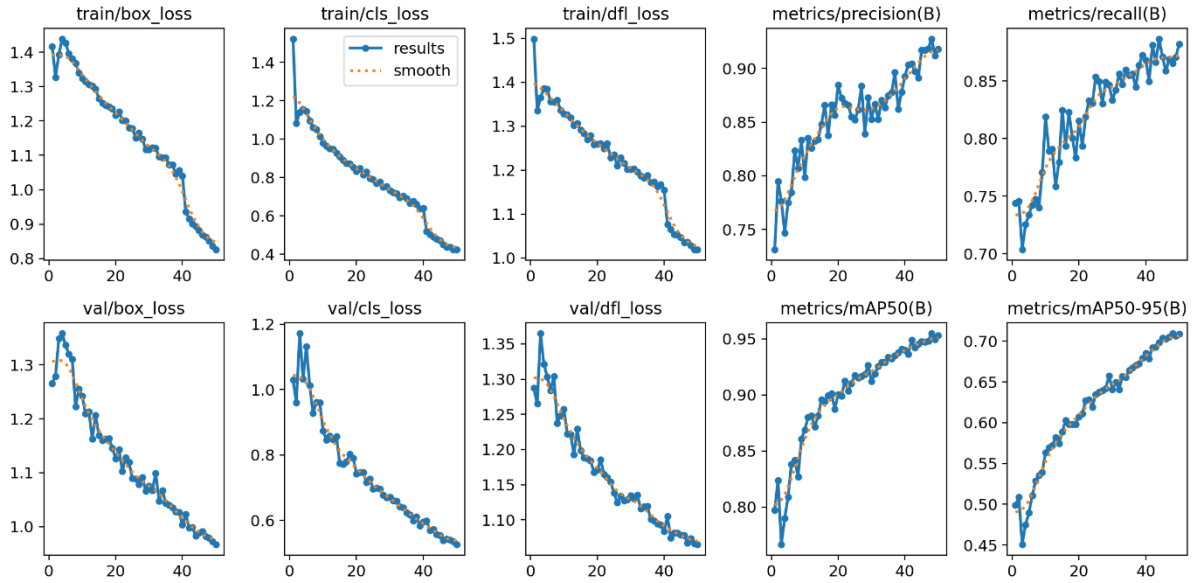


Tamaño de lote (Batch size)

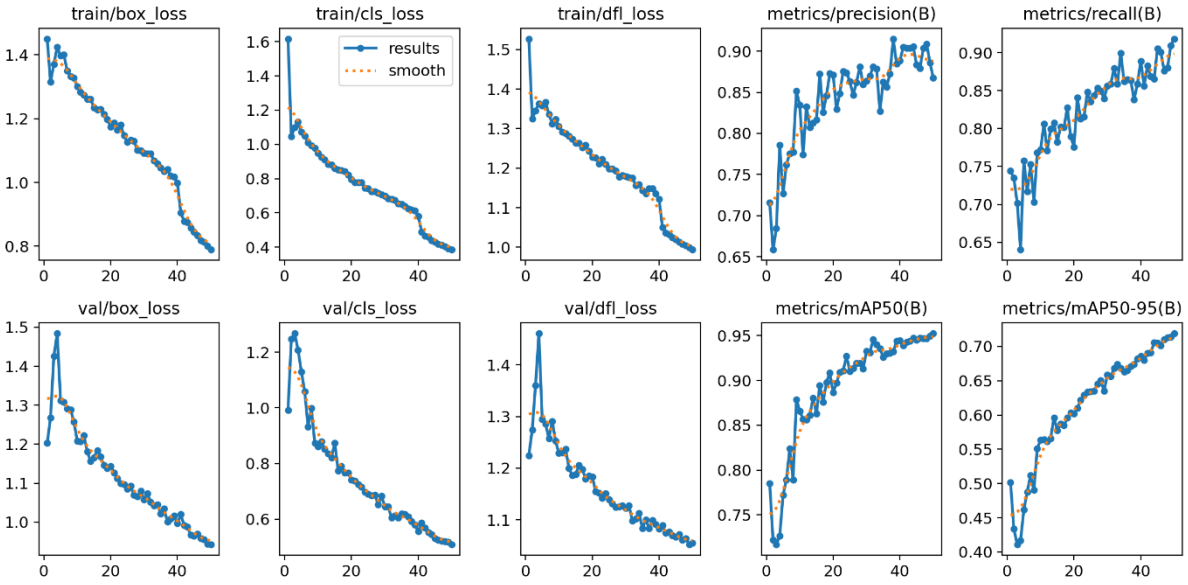
- **Batch size (4)**



- **Batch size (8)**

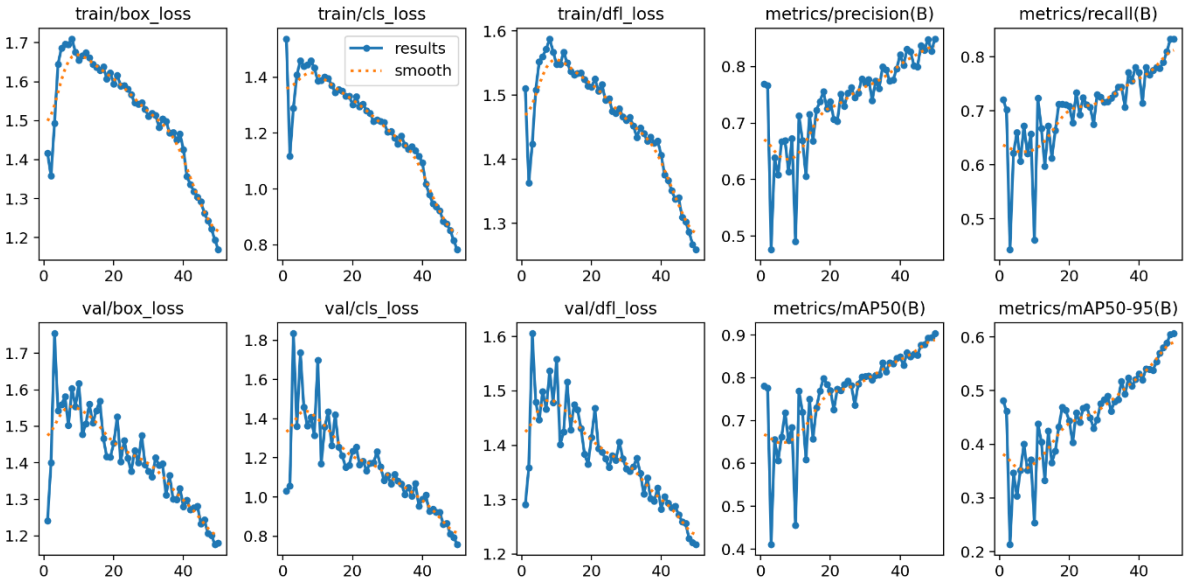


- **Batch size (16)**

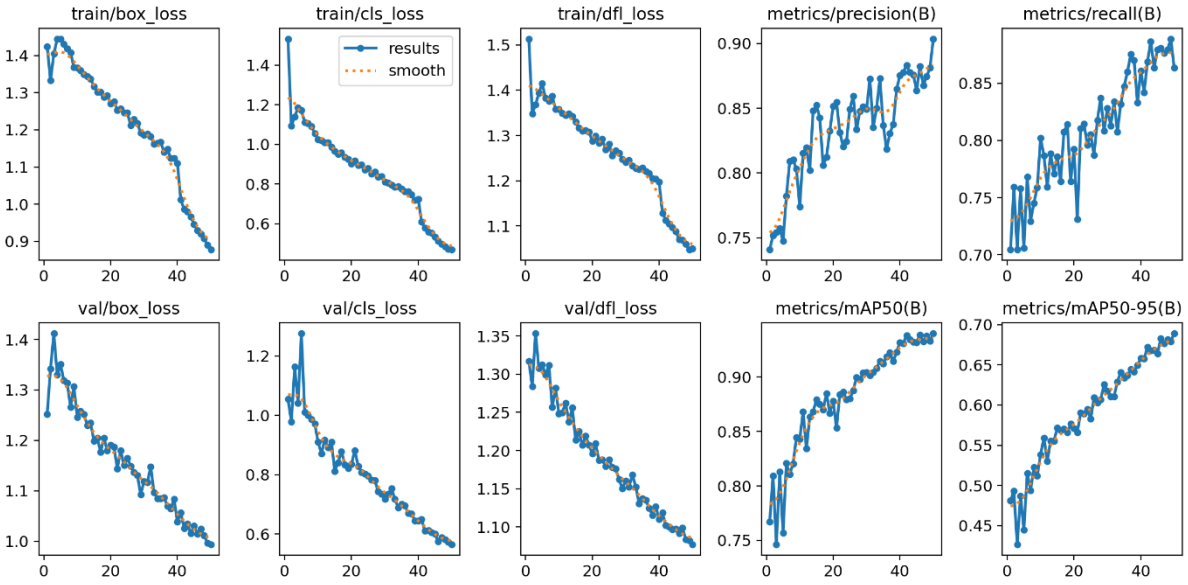


Decaimiento de peso (Weight decay)

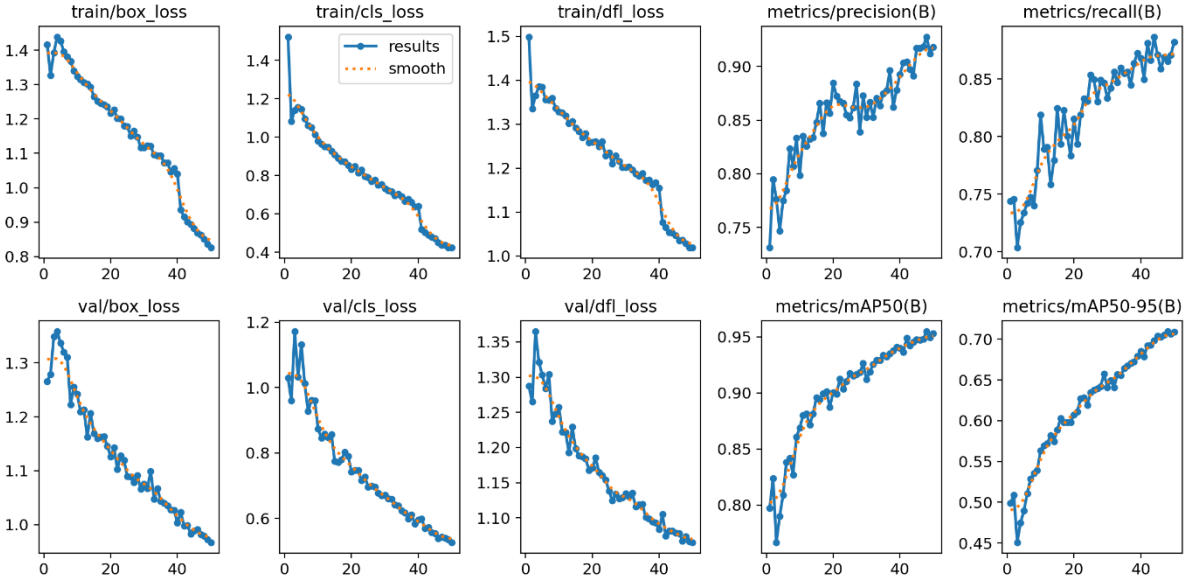
- **Weight decay (0.05)**



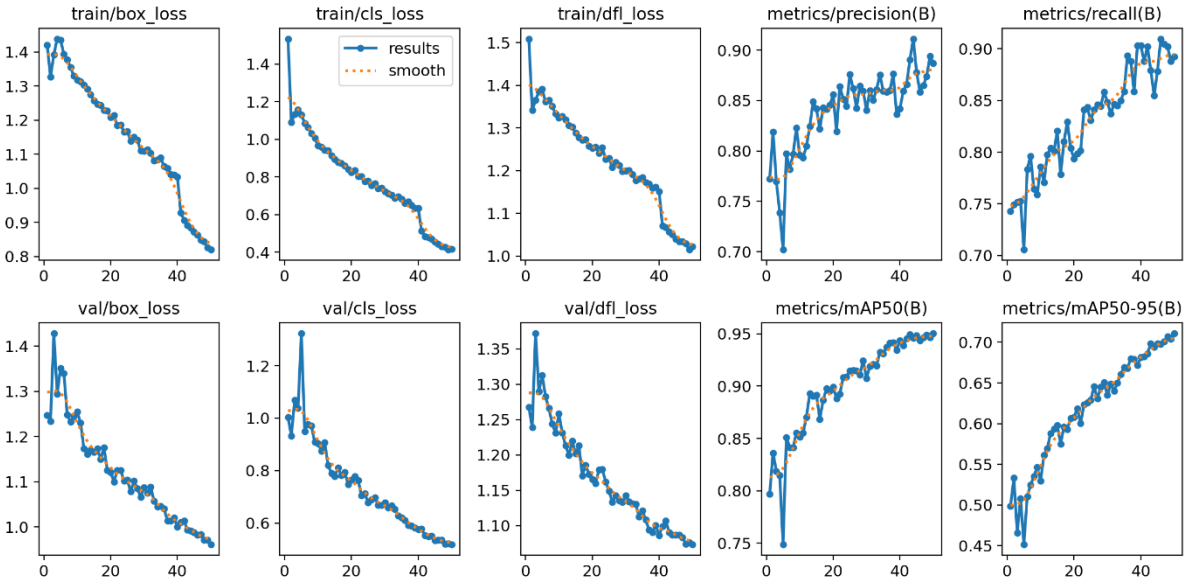
- **Weight decay (0.005)**



- **Weight decay (0.0005)**

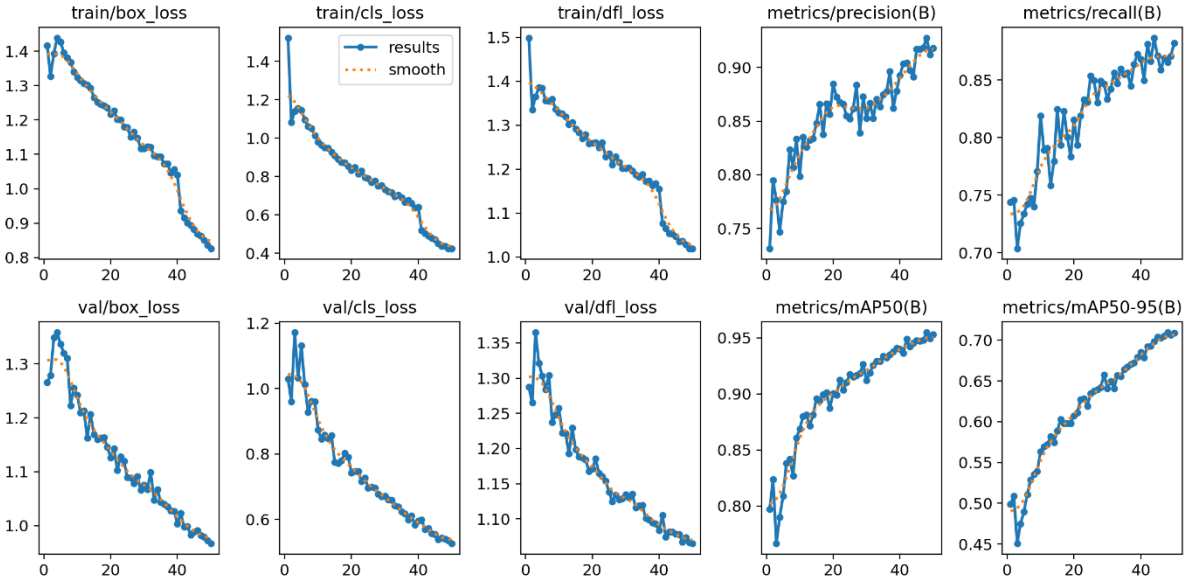


- **Weight decay (0.00005)**

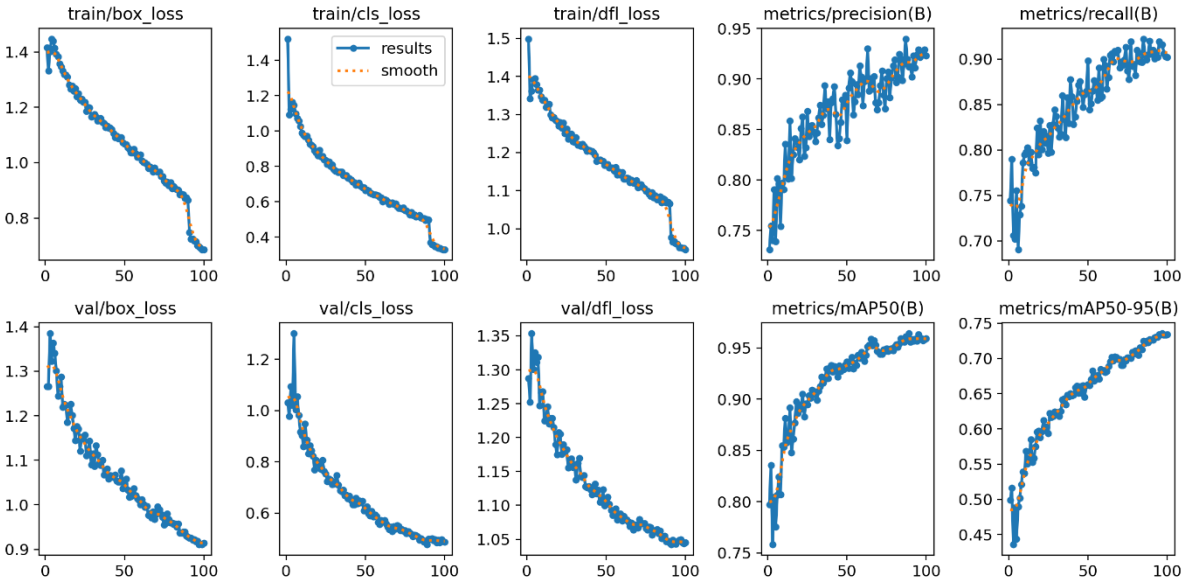


Épocas (Epochs)

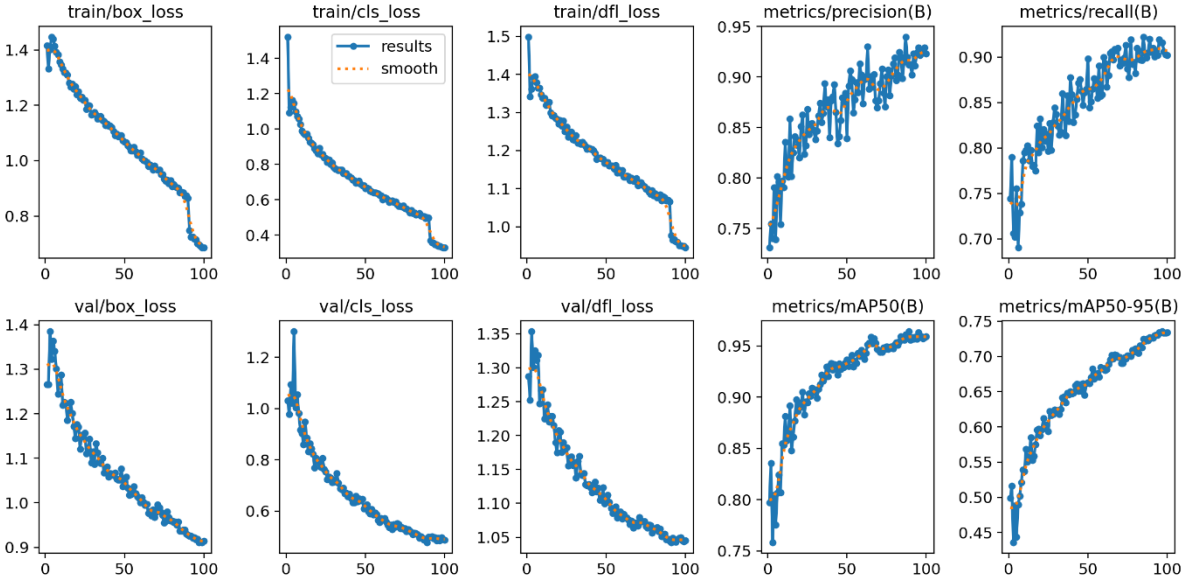
- **Epochs (50)**



- **Epochs (100)**



- **Epochs (200)**

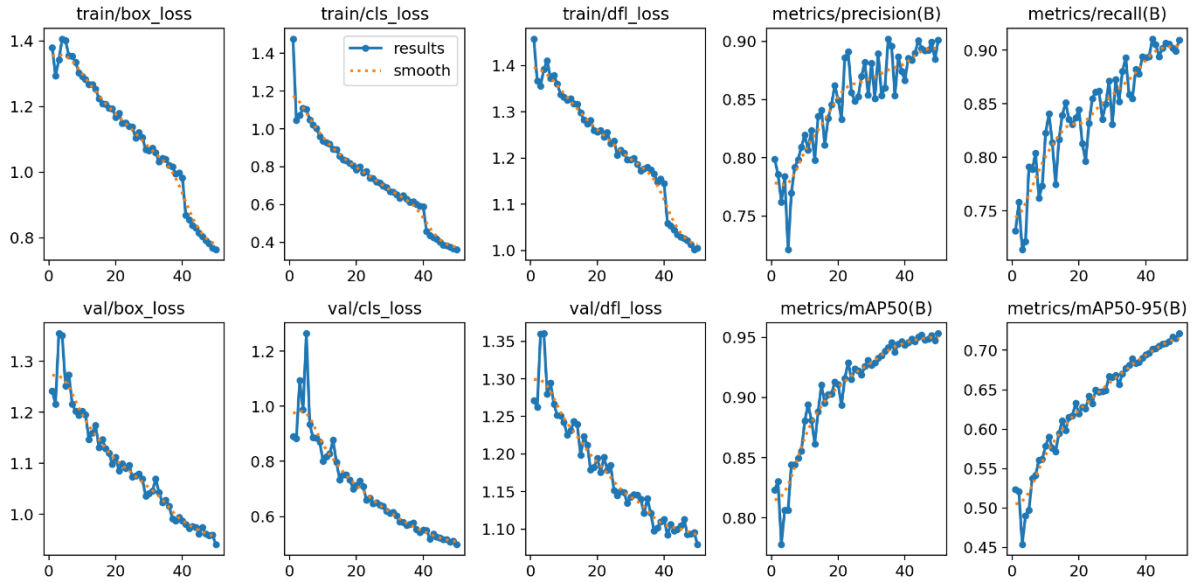


Anexo 5. Gráficas de rendimiento del modelo YOLOv8s durante el entrenamiento

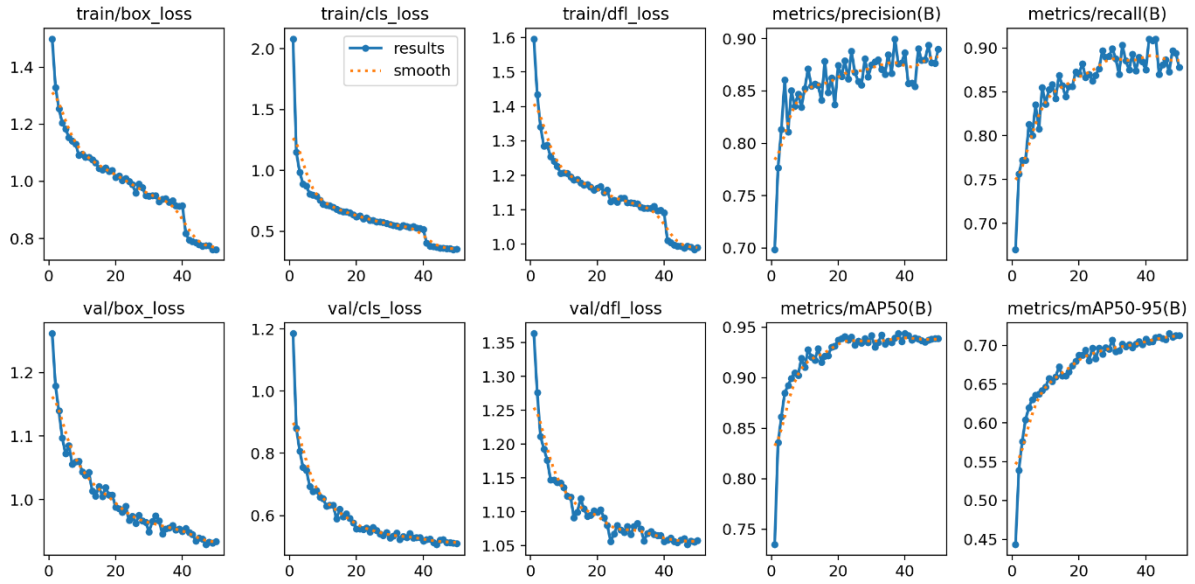
En esta sección se presenta una muestra de las gráficas de rendimiento al variar los hiperparámetros (Learning rate, Optimizer, Batch size, Weight decay, Epochs) en YOLOv8s durante el entrenamiento del modelo. Las gráficas completas están disponibles en Google Drive[34].

Tasa de aprendizaje (Learning rate)

- **Learning rate (0.01)**

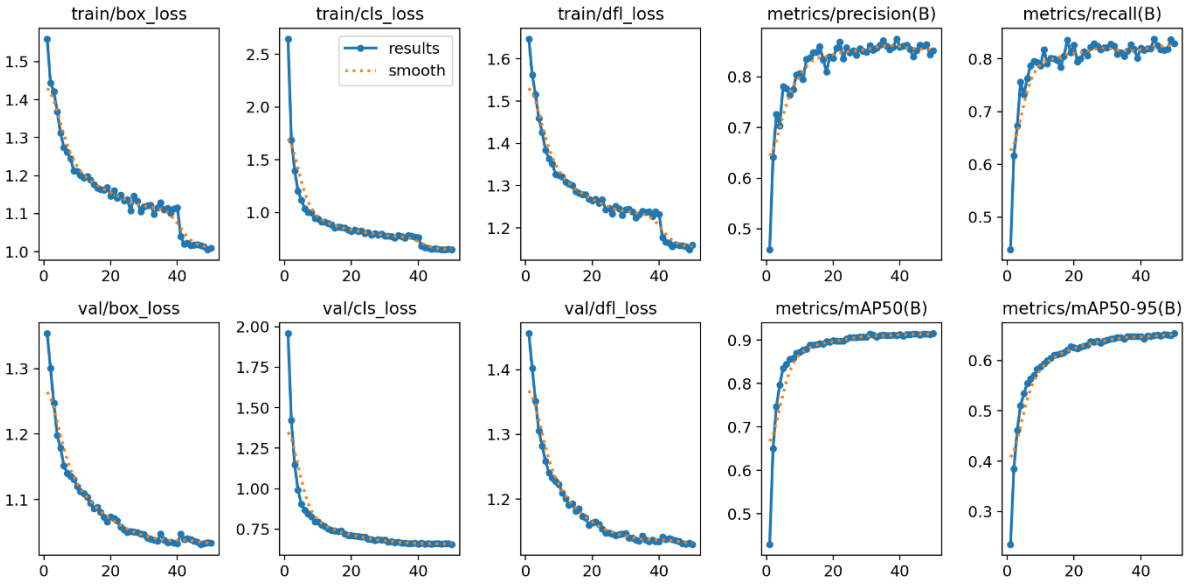


- **Learning rate (0.001)**



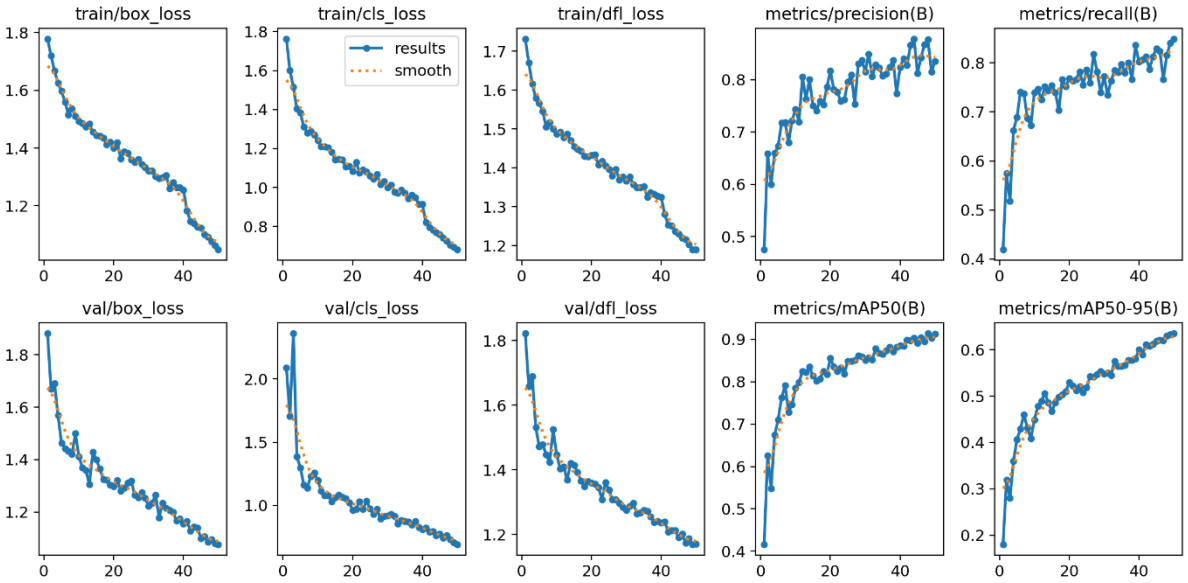
[34] Repositorio de las gráficas de YOLOv8s: https://drive.google.com/drive/folders/15ckmnezPGV7N3x4S4ofVgF3oLJERTJII?usp=sharing

- **Learning rate (0.0001)**

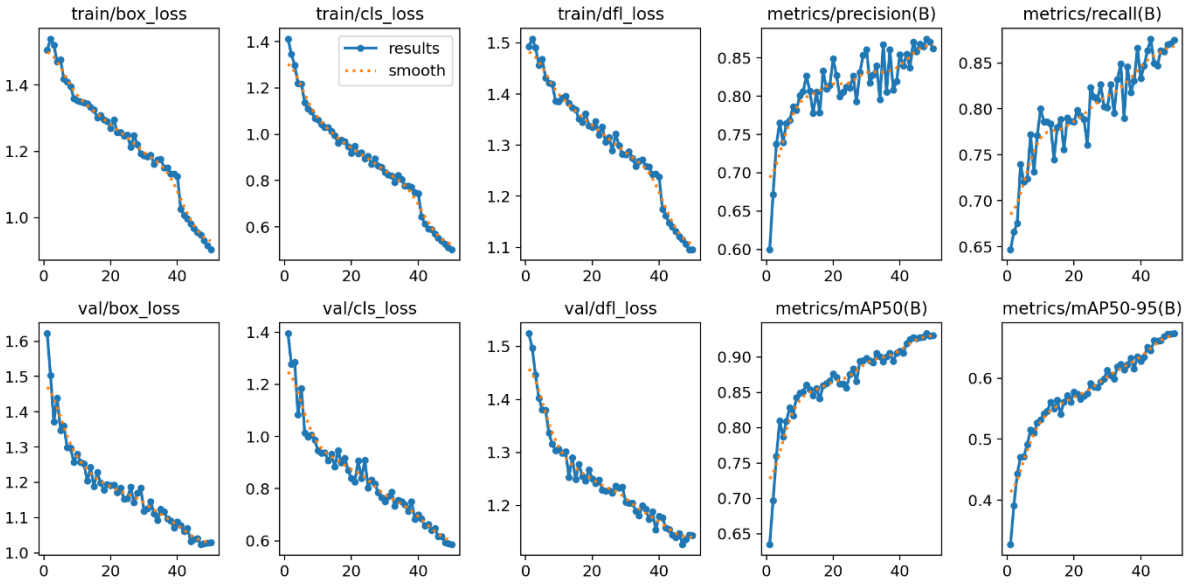


Optimizador (Optimzer)

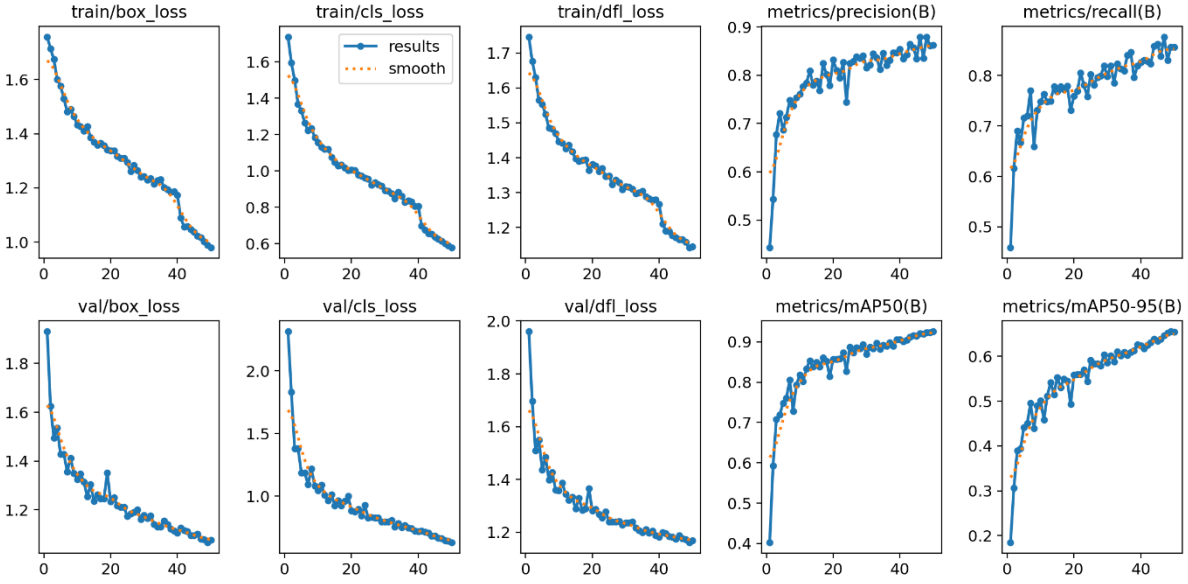
- **Optimizer (Adam)**



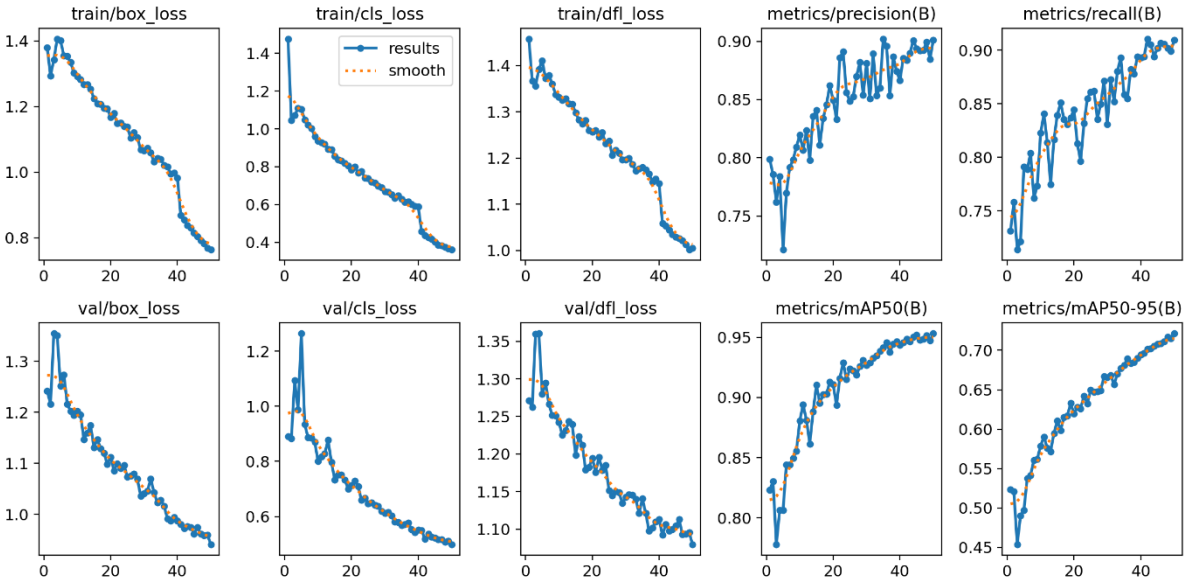
- **Optimizer (AdaMax)**



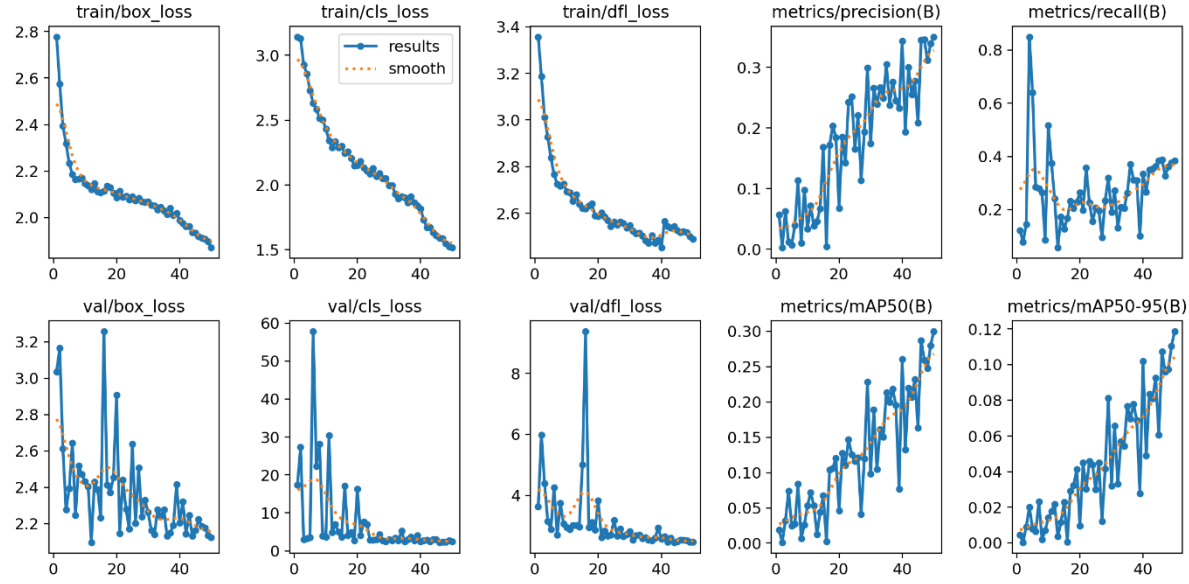
- **Optimizer (AdamW)**



- **Optimizer (SGD)**

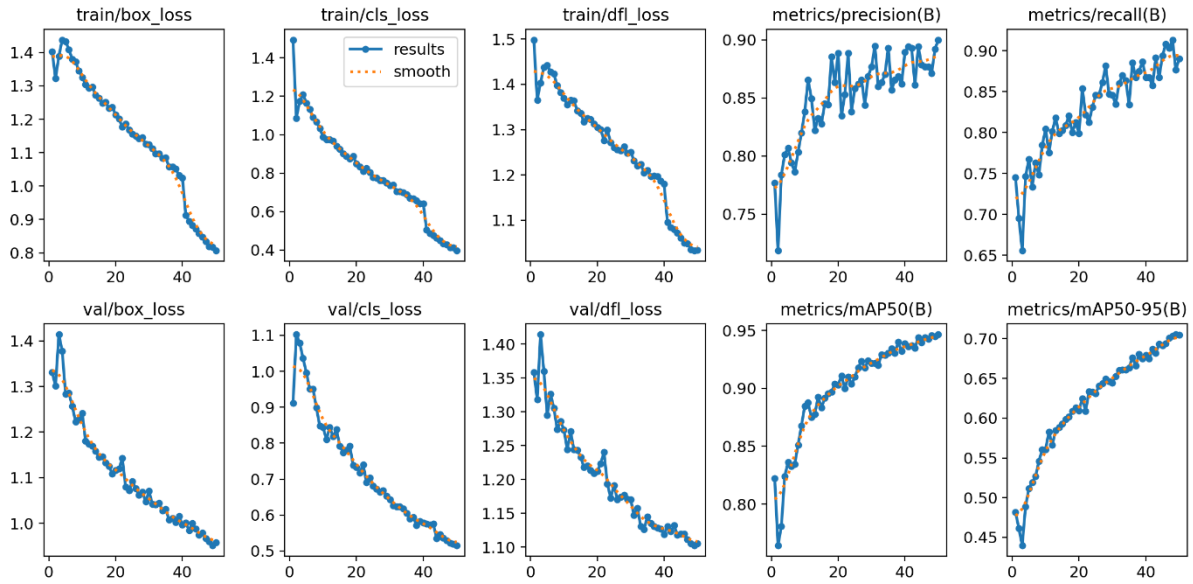


- **Optimizer (RMSProp)**

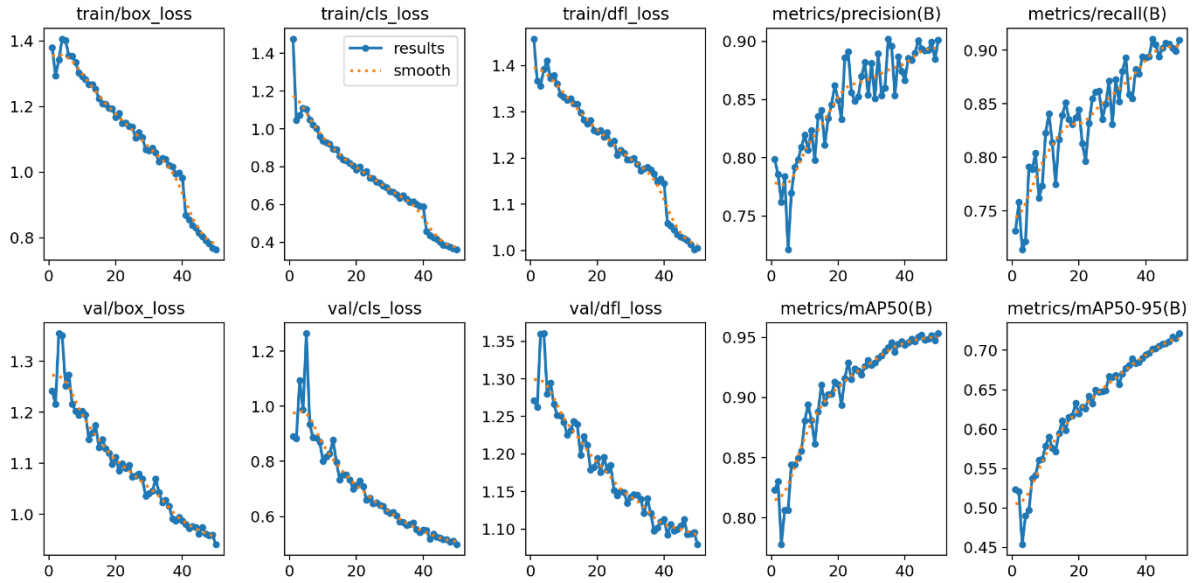


Tamaño de lote (Batch size)

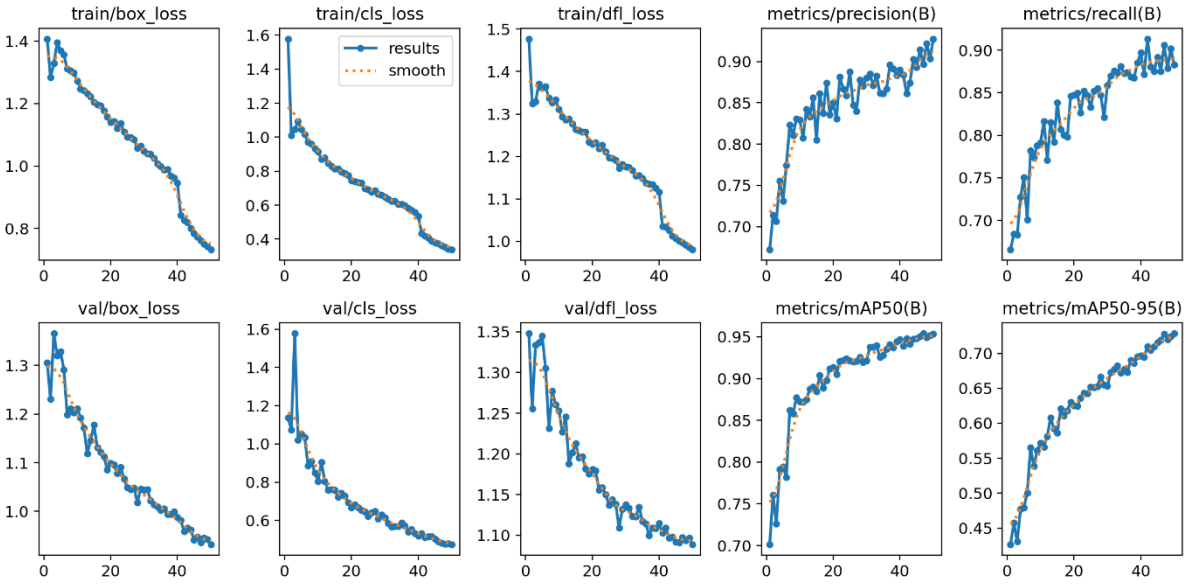
- **Batch size (4)**



- **Batch size (8)**

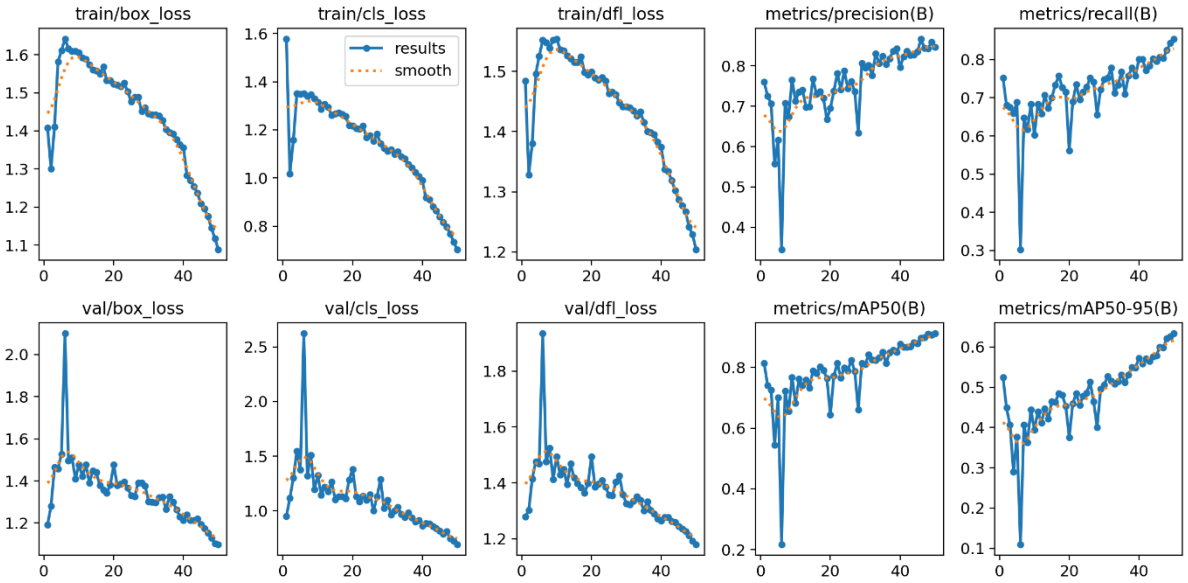


- **Batch size (16)**

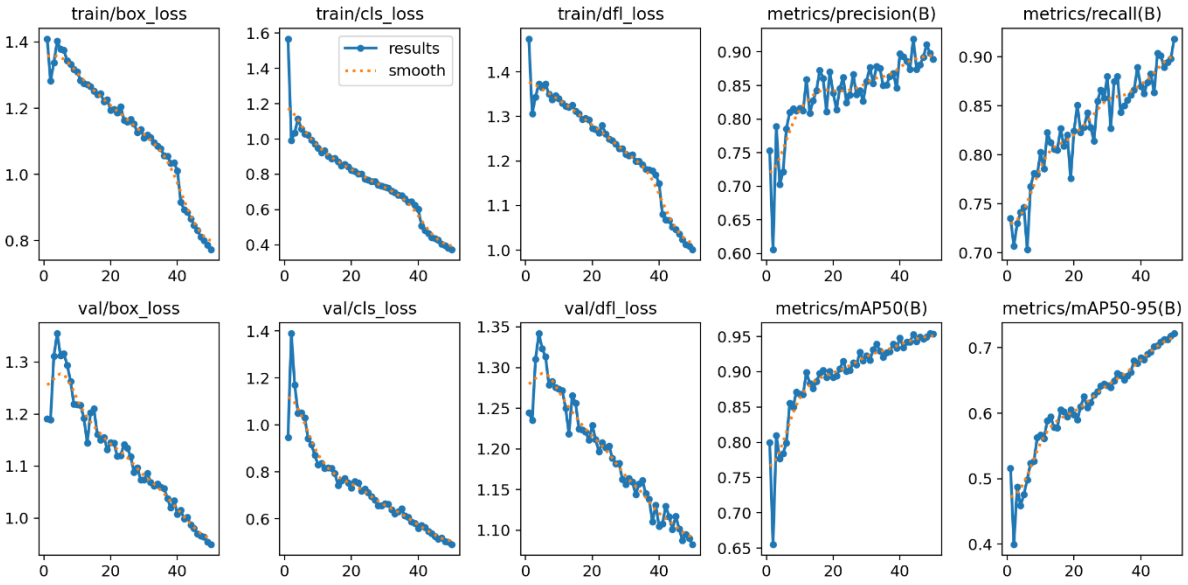


Decaimiento de peso (Weight decay)

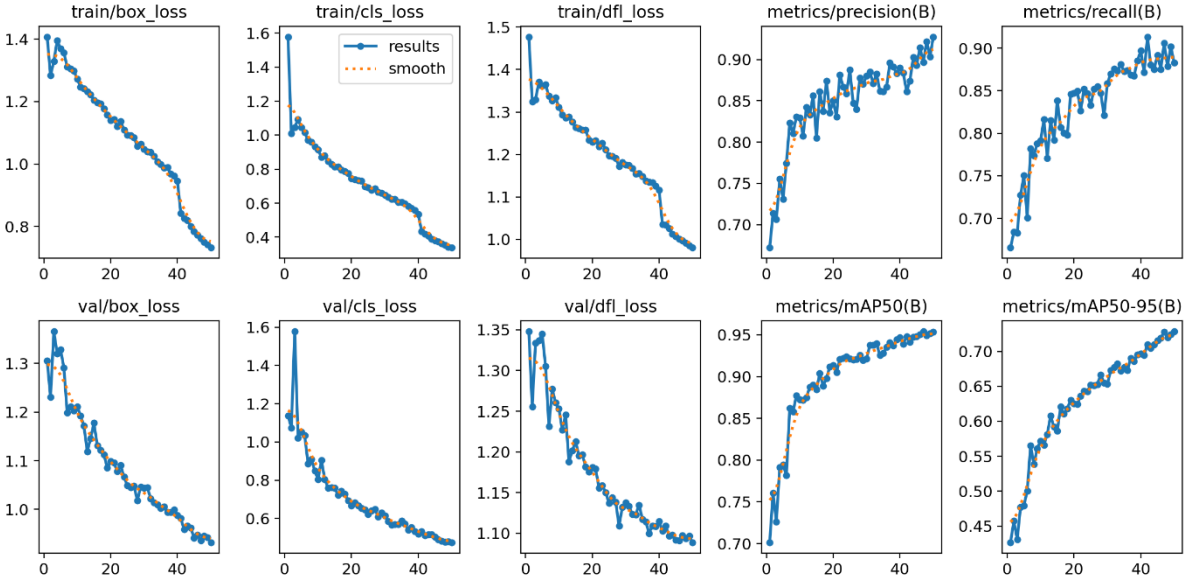
- **Weight decay (0.05)**



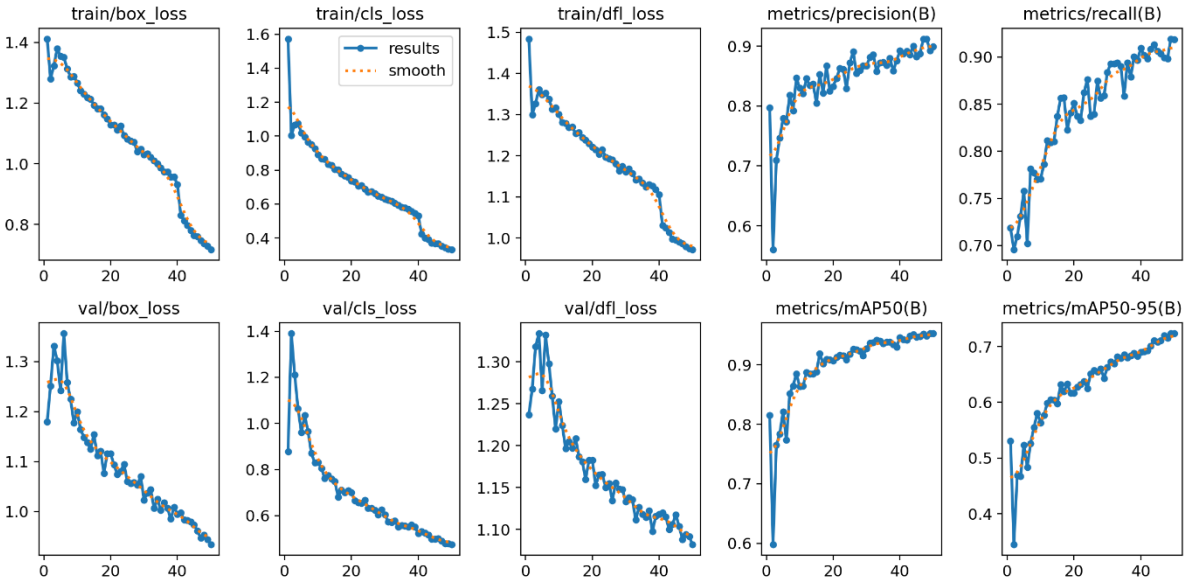
- **Weight decay (0.005)**



- **Weight decay (0.0005)**

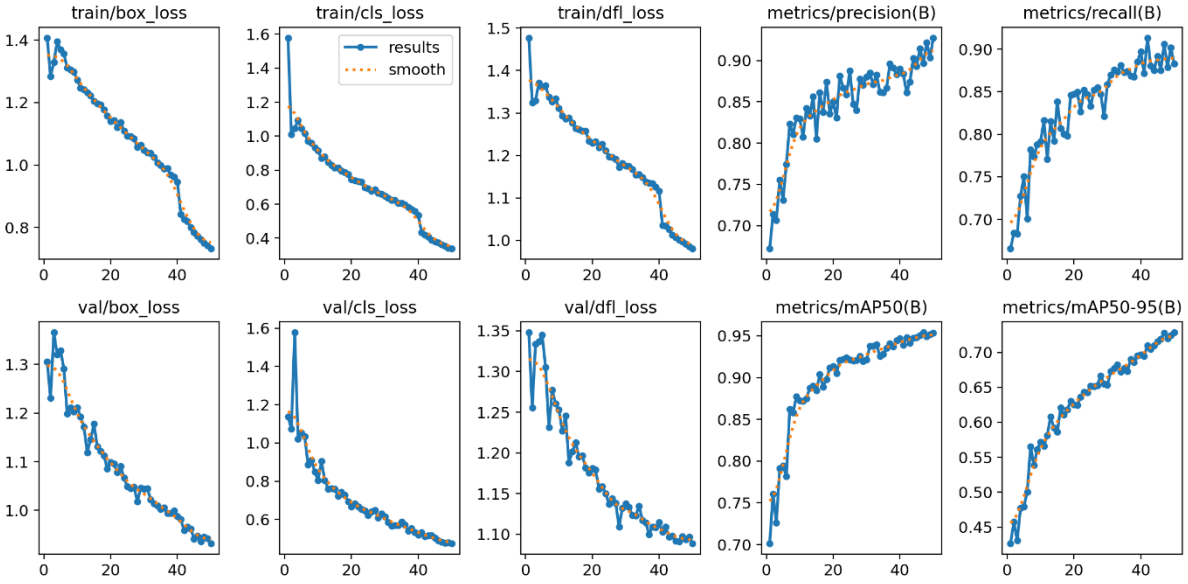


- **Weight decay (0.00005)**

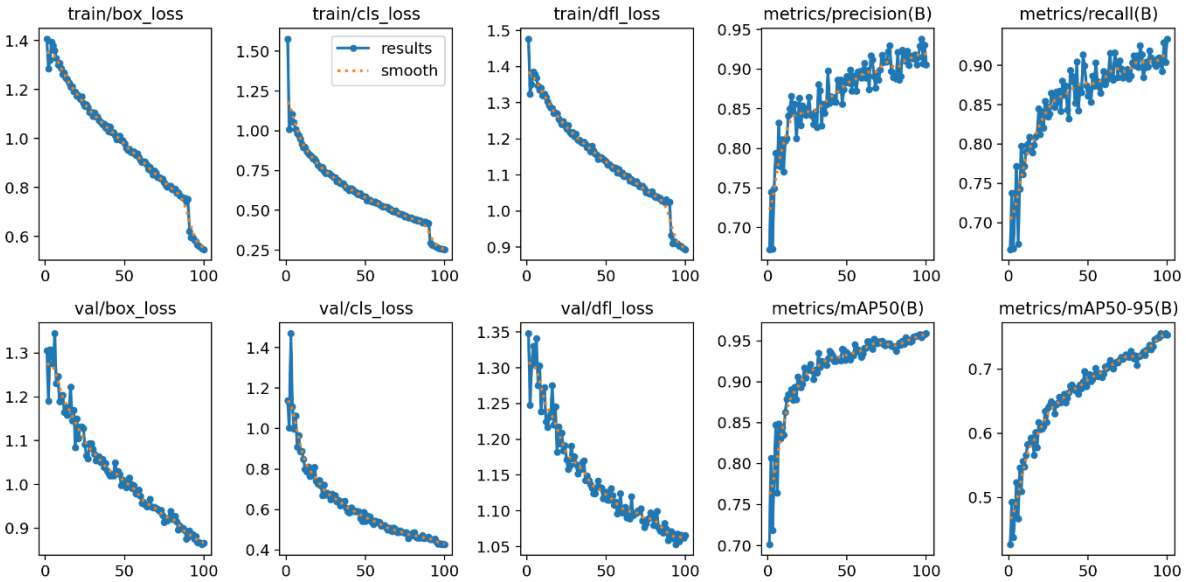


Épocas (Epochs)

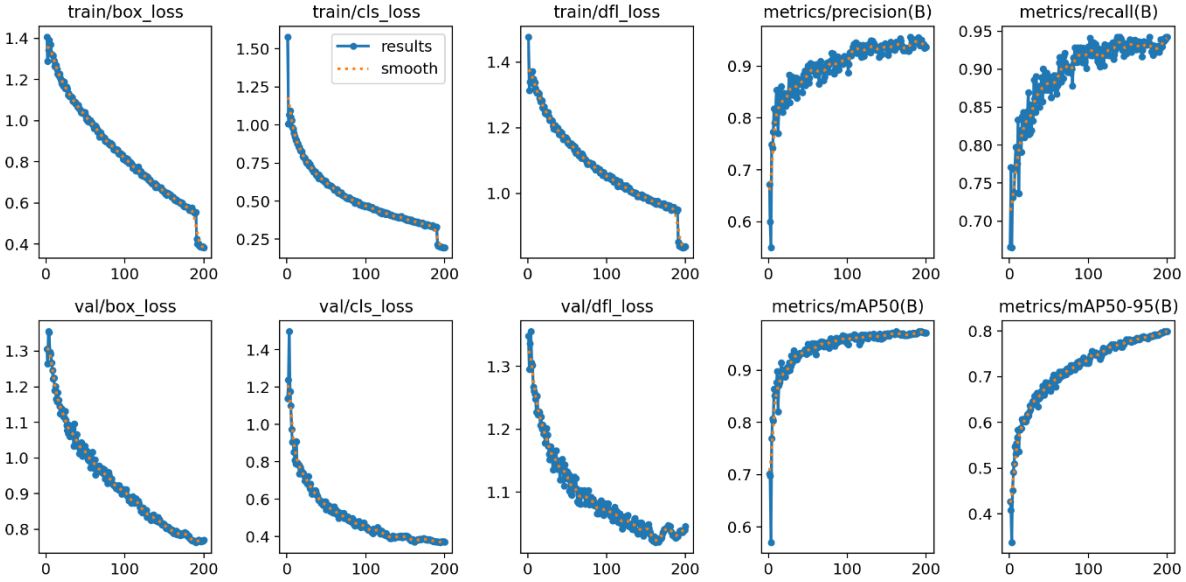
- **Epochs (50)**



- **Epochs (100)**



- **Epochs (200)**



Anexo 6. Resultados obtenidos en la validación del modelo en un entorno simulado

Imagen	Observación del experto	Áreas identifcadas por el experto	Áreas detectadas por el modelo	Error de detección	Incompleto
tizon_1	Enferma	1	1	0	0
tizon_2	Enferma	1	1	0	0
tizon_3	Enferma	4	5	1	0
tizon_4	Enferma	2	2	0	0
tizon_5	Enferma	4	4	0	0
tizon_6	Enferma	3	3	0	0
tizon_7	Enferma	1	1	0	0
tizon_8	Enferma	1	1	0	0
tizon_9	Enferma	3	2	0	1
tizon_10	Enferma	1	1	0	0
tizon_11	Enferma	1	1	0	0
tizon_12	Enferma	1	1	0	0
tizon_13	Enferma	2	2	0	0
tizon_14	Enferma	1	1	0	0
tizon_15	Enferma	1	1	0	0
tizon_16	Enferma	1	1	0	0
tizon_17	Enferma	1	1	0	0
tizon_18	Enferma	1	0	0	1
tizon_19	Enferma	3	4	1	0
tizon_20	Enferma	1	1	0	0
tizon_21	Enferma	1	1	0	0
tizon_22	Enferma	1	1	0	0
tizon_23	Enferma	1	1	0	0
tizon_24	Enferma	1	1	0	0
tizon_25	Enferma	1	1	0	0
tizon_26	Enferma	1	1	0	0
tizon_27	Enferma	1	1	0	0
tizon_28	Enferma	2	2	0	0
tizon_29	Enferma	2	2	0	0
tizon_30	Enferma	1	1	0	0
tizon_31	Enferma	1	1	0	0
tizon_32	Enferma	5	3	0	2
tizon_33	Enferma	1	1	0	0
tizon_34	Enferma	1	1	0	0
tizon_35	Enferma	3	3	0	0
tizon_36	Enferma	3	2	0	1
tizon_37	Enferma	3	2	0	1
tizon_38	Enferma	2	2	0	0
tizon_39	Enferma	1	1	0	0
tizon_40	Enferma	3	3	0	0
tizon_41	Enferma	2	2	0	0
tizon_42	Enferma	2	2	0	0
tizon_43	Enferma	2	1	0	1
tizon_44	Enferma	1	1	0	0

tizon_45	Enferma	1	1	0	0
tizon_46	Enferma	2	1	0	1
tizon_47	Enferma	2	2	0	0
tizon_48	Enferma	1	1	0	0
tizon_49	Enferma	1	1	0	0
tizon_50	Enferma	1	1	0	0
tizon_51	Enferma	1	1	0	0
tizon_52	Enferma	2	2	0	0
tizon_53	Enferma	1	1	0	0
tizon_54	Enferma	2	2	0	0
tizon_55	Enferma	1	1	0	0
	Total	91	85	2	8

Imagen	Observación del experto	Áreas identifcadas por el experto	Áreas detectadas por el modelo	Error de detección
sana_1	Sana	0	0	0
sana_2	Sana	0	0	0
sana_3	Sana	0	0	0
sana_4	Sana	0	0	0
sana_5	Sana	0	0	0
sana_6	Sana	0	0	0
sana_7	Sana	0	0	0
sana_8	Sana	0	0	0
sana_9	Sana	0	0	0
sana_10	Sana	0	0	0
sana_11	Sana	0	0	0
sana_12	Sana	0	0	0
sana_13	Sana	0	0	0
sana_14	Sana	0	0	0
sana_15	Sana	0	0	0
sana_16	Sana	0	0	0
sana_17	Sana	0	0	0
sana_18	Sana	0	0	0
sana_19	Sana	0	0	0
sana_20	Sana	0	0	0
sana_21	Sana	0	0	0
sana_22	Sana	0	0	0
sana_23	Sana	0	0	0
sana_24	Sana	0	1	1
sana_25	Sana	0	0	0
sana_26	Sana	0	0	0
sana_27	Sana	0	0	0
sana_28	Sana	0	0	0
sana_29	Sana	0	0	0
sana_30	Sana	0	0	0
sana_31	Sana	0	0	0
sana_32	Sana	0	0	0

sana_33	Sana	0	0	0	
sana_34	Sana	0	0	0	
sana_35	Sana	0	0	0	
sana_36	Sana	0	0	0	
sana_37	Sana	0	0	0	
sana_38	Sana	0	0	0	
sana_39	Sana	0	0	0	
sana_40	Sana	0	0	0	
sana_41	Sana	0	0	0	
sana_42	Sana	0	0	0	
sana_43	Sana	0	0	0	
sana_44	Sana	0	0	0	
sana_45	Sana	0	0	0	
sana_46	Sana	0	0	0	
sana_47	Sana	0	0	0	
sana_48	Sana	0	0	0	
sana_49	Sana	0	0	0	
sana_50	Sana	0	0	0	
sana_51	Sana	0	0	0	
sana_52	Sana	0	0	0	
sana_53	Sana	0	0	0	
sana_54	Sana	0	0	0	
sana_55	Sana	0	0	0	
	Total	0	1	1	

Anexo 7. Certificado de evaluación del prototipo Web por parte del experto

Ing. Angel Rolando Robles Carrión PhD.
Docente de la carrera de Agronomía de la Universidad Nacional de Loja

CERTIFICA:

Haber realizado pruebas de funcionamiento del prototipo web desarrollado en el Trabajo de Integración Curricular del estudiante **Gerardo Manuel Quizhpe Chocho**, a través del cual se logró identificar el tizón tardío en las hojas del cultivo de papa. Los resultados demostraron que el modelo implementado funcionó de manera satisfactoria. Por ello, considero que el TIC titulado **"Impacto de los hiperparámetros en arquitecturas YOLO para identificar el tizón tardío en las hojas del cultivo de papa (*Solanum tuberosum*)"** constituye una herramienta valiosa para la detección de dicha enfermedad.

Certifico lo anterior y faculto al peticionario hacer uso del presente documento para fines legales pertinentes.

Loja, 13 de enero del 2025

ANGEL ROLANDO ROBLES CARRION

Ing. Angel Rolando Robles Carrión
Docente de la carrera de Agronomía

Anexo 8. Certificado de traducción del resumen

Mgs. Mónica Jimbo Galarza

C E R T I F I C O:

Haber realizado la traducción de Español – Inglés del resumen del Trabajo de Integración Curricular previo a la obtención del título de Ingeniero en Ciencias de la Computación titulado **"Impacto de los hiperparámetros en arquitecturas YOLO para identificar el tizón tardío en las hojas del cultivo de papa (Solanum tuberosum)"** de autoría de Gerardo Manuel Quizhpe Chocho CI: 1106078833.

Se autoriza al interesado hacer uso de la misma para los trámites que crea conveniente.

Es todo cuanto puedo certificar en honor a la verdad.

Emitida en Loja, a los 28 días del mes de febrero 2025.

Firmado electrónicamente por
MONICA CECILIA
JIMBO GALARZA

Mgs. Mónica Jimbo Galarza

MAGÍSTER EN ENSEÑANZA DE INGLÉS COMO LENGUA EXTRANJERA

REGISTRO EN LA SENECYT N° 1021-2018-1999861