UNIVERSIDAD NACIONAL MICAELA BASTIDAS DE APURÍMAC

FACULTAD DE INGENIERÍA

ESCUELA ACADÉMICO PROFESIONAL DE INGENIERÍA INFORMÁTICA Y SISTEMAS

Tesis

Detección de enfermedades en el cultivo de papa mediante el uso de Machine Learning en Abancay, 2022

Presentado por:

Nelida Alicia Alvarez Vargas

Para optar el título de Ingeniero Informático y Sistemas

Abancay, Perú

2025

UNIVERSIDAD NACIONAL MICAELA BASTIDAS DE APURÍMAC

FACULTAD DE INGENIERÍA

ESCUELA ACADÉMICO PROFESIONAL DE INGENIERÍA INFORMÁTICA Y SISTEMAS

TESIS

Detección de enfermedades en el cultivo de papa mediante el uso de Machine Learning en Abancay, 2022

Presentado por **Nelida Alicia Alvarez Vargas**, para optar el título de Ingeniero Informático y Sistemas

Sustentado y aprobado el 05 de diciembre del 2024 ante el jurado evaluador:

Presidente: _____

Mag. Francisco Cari Incahuanaco

Primer miembro: _____

Mtro. Rafael Ricardo Quispe Merma

Segundo miembro: _____

Mtro. Virgilio Martinez Duran

Asesor(es): _____

Dr. Manuel Jesús Ibarra Cabrera

UNIVERSIDAD NACIONAL
MICAELA BASTIDAS
DE APURIMAC
Licenciada por SUNEDU

CONSTANCIA DE ORIGINALIDAD
Nº 228-2024

La Universidad Nacional Micaela Bastidas de Apurímac, a través de la Unidad de Investigación de la Facultad de Ingeniería declara que, la Tesis intitulada: Detección de enfermedades en el cultivo de papa mediante el uso de Machine Learning en Abancay, 2022, presentado por la Bach. Nelida Alicia Alvarez Vargas, Para optar el Título de **Ingeniero Informático y Sistemas;** ha sido sometido a un mecanismo de evaluación y verificación de similitud, a través del Software Turnitin, siendo el índice de similitud ACEPTABLE de **(14%)** por lo que, cumple con los criterios de originalidad establecidos por la Universidad.

Abancay, 26 de noviembre del 2024

UNIVERSIDAD NACIONAL MICAELA BASTIDAS DE APURIMAC
Dr. Lintol Contreras Salas
DIRECTOR(E) DE LA UNIDAD DE INVESTIGACIÓN
FACULTAD DE INGENIERÍA

C. c.
Archivo
REG. Nº 833

Agradecimiento

En primer lugar, deseo expresar mi profundo agradecimiento a Dios, quien me otorga la vida y colma de bendiciones mi camino junto a mis seres queridos.

Agradezco de manera especial a mis padres, quienes con su amor incondicional y sabias enseñanzas han sido mi guía en la vida.

A mi esposo, mi compañero de vida, agradezco su orientación y colaboración, brindándome su valioso conocimiento y experiencia.

Quiero extender mi reconocimiento al Ingeniero Manuel Ibarra Cabrera, mi asesor, por su invaluable guía y apoyo en el desarrollo de esta tesis. Su vasto conocimiento y experiencia han sido fundamentales para su realización.

Dedicatoria

A mis adorados hijos, Ethan Robert y Anne Elaine Amira, cuyas sonrisas y travesuras llenan mis días de motivación y alegría. A mi querido padre, cuyo legado de perseverancia y apoyo sigue guiando mis pasos hacia el logro de mis metas. A mi amada madre, siempre presente para brindarme su incondicional apoyo. Finalmente, a mi esposo, mi compañero de vida, cuyo amor y aliento son mi sostén en cada camino que emprendo.

Detección de enfermedades en el cultivo de papa mediante el uso de Machine Learning en Abancay, 2022

Línea de investigación: Ingeniería Informática, Industria y Sociedad.

ÍNDICE

ÍNDICE DE TABLAS

Pág.

ÍNDICE DE FIGURAS

Pág.

INTRODUCCIÓN

El Perú es uno de los principales productores de papa a nivel mundial y un centro clave de diversidad genética para este cultivo. La papa es un alimento fundamental en la dieta peruana y desempeña un papel crucial en la seguridad alimentaria y la economía del país. La producción en los departamentos de la sierra sur aporta el 47,7% de la producción nacional, con Apurímac contribuyendo con el 7% de la producción total de papa. (MIDAGRI, 2023). El Perú es un país con mayor diversidad de papas en el mundo, al contar con ocho especies nativas domesticadas y 2,301 variedades de las más de 4,000 que existen en Latinoamérica. Dentro de las principales variedades están: papa Amarilla, papa Huamantanga, papa Canchan, papa Peruanita, papa Yungay, entre otras. Los agricultores que viven en la zona rural de Apurímac, gran parte de ellos son familias que se sostienen económicamente de la actividad agrícola (MINISTERIO DE DESARROLLO AGRARIO Y RIEGO DEL PERÚ, 2020). La papa es uno de los tubérculos más consumidos en nuestro país por ser barata; por ello, las personas lo consumen a diario en su dieta alimenticia.

Un problema común en los cultivos de papa son las enfermedades que afectan a las plantas, lo que provoca una disminución significativa en la producción y representa una gran pérdida económica. Esta situación se convierte en una problemática grave, ya que los daños pueden ser totales o parciales, reduciendo tanto el rendimiento como la calidad del producto, y complicando la rentabilidad del cultivo. Por ello, los agricultores se ven obligados a buscar alternativas y soluciones para intentar mejorar su producción. (TORRES, 2002)

Por otro lado, la tecnología ofrece programas de reconocimiento de imágenes para objetos, personas y animales. En este contexto, en los últimos años se han desarrollado programas específicos para la detección de enfermedades en hojas de papa. Un ejemplo es el proyecto de investigación presentado por Cinthya C. H., titulado "Detección temprana del Potato *Yellow Vein* Virus en cultivos de *Solanum tuberosum L.* mediante la teledetección" (CARRIÓN, 2017). Asimismo, existe el artículo presentado por Md Ashiqur R. titulado "Predicting and Classifying Potato Leaf Disease using K-means Segmentation Techniques and Deep Learning Networks"(NISHAD, MITU Y JAHAN, 2022). Sin embargo, es importante señalar que estos proyectos fueron desarrollados en contextos y con tipos de papas y enfermedades diferentes.

Cabe mencionar, que Machine Learning es una rama de la inteligencia artificial, basado en algoritmos

que permite a las computadoras tener la capacidad de aprender a base de patrones, y a partir de éstos, podrá reorganizar sus datos y tener la habilidad de predecir o identificar las situaciones que ya aprendió.

Este proyecto, se centró en demostrar la hipótesis principal, al usar correctamente las técnicas: Faster R-CNN y YOLO V4, entonces se determina que YOLO V4 es más eficiente (accuracy) que Faster R-CNN para la detección de enfermedades en el cultivo de papa en Abancay, Apurímac, 2022. Para lo cual se parte de la descripción del problema, y a partir de ello se plantean los objetivos e hipótesis mencionados en esta investigación, asimismo, se detalla las metodologías utilizadas para lograr los resultados, su debida interpretación y discusión correspondiente, para finalmente llegar a las conclusiones y plantear recomendaciones, enfocados en que el agricultor pueda manejar su prevención y control con el fin de lograr un cultivo sano y de alta calidad.

RESUMEN

El Perú es un país con una gran diversidad de tipos de papa y es, uno de los principales cultivos agrícolas que sustenta la alimentación de las personas, enfrenta pérdidas en la producción de este cultivo clave debido a enfermedades como el tizón tardío y el pie negro entre otras; lo cual genera una gran pérdida en su producción y, conlleva a un déficit económico para el agricultor. La presente investigación se desarrolló con el objetivo de lograr la detección temprana de enfermedades de pudrición blanda y tizón tardío en el cultivo de la papa, a través del uso de técnicas de Machine Learning y para determinar la eficiencia de la clasificación se utilizaron los modelos de Faster R-CNN y YOLO V4. El procedimiento para esta investigación consistió en recolectar un total de 1011 imágenes de hojas de papa, tanto sanas como enfermas, en la localidad de Abancay, Apurímac. Estas imágenes se dividieron aleatoriamente para realizar el entrenamiento, la validación y las pruebas. Luego, se llevó a cabo el etiquetado de las imágenes utilizando la herramienta LabelImg y Roboflow. Posteriormente, se empleó la herramienta Google Colab con lenguaje de programación Python para realizar el entrenamiento, con cada uno de los modelos, seguido de las validaciones y, finalmente, realizar las pruebas. Los resultados obtenidos revelan que, Faster R-CNN demostró un rendimiento sólido en la detección de Pie Negro, alcanzando una precision, recall, F-value y accuracy del 100%. En el caso de la detección de Rancha, aunque los valores fueron ligeramente más bajos, aún mostró una precision del 98%, un recall del 79%, un F-value del 87% y un accuracy del 78%. Por otro lado, YOLO V4 también sobresalió en la detección de Pie Negro, logrando una precision, recall, F-value y accuracy del 100%. En cuanto a la detección de Rancha, los valores experimentaron una variación mínima, obteniendo un precision del 97%, un recall del 90%, un F-value del 93% y un accuracy del 87%. Finalmente podemos concluir que YOLO V4 destaca en la detección de ambas enfermedades en hojas de papa en la región de Abancay, Apurímac.

Palabras clave: *Enfermedades en el cultivo de la papa, machine learning, Faster R – CNN, YOLO V4*

ABSTRACT

Peru is a country with a great diversity of potato types and is one of the main agricultural crops that sustains people's food, faces losses in the production of this key crop due to diseases such as late blight and blackleg among others; which generates a great loss in its production and leads to an economic deficit for the farmer. The present research was developed with the objective of achieving early detection of soft rot and late blight diseases in the potato crop, through the use of Machine Learning techniques and to determine the efficiency of the classification, the Faster R-CNN and YOLO V4 models were used. The procedure for this research consisted of collecting a total of 1011 images of potato leaves, both healthy and diseased, in the town of Abancay, Apurimac. These images were randomly divided for training, validation and testing. The images were then labeled using the LabelImg and Roboflow tools. Subsequently, the Google Colab tool with Python programming language was used to perform the training, with each of the models, followed by the validations and, finally, to perform the tests. The results obtained reveal that Faster R-CNN demonstrated a solid performance in the detection of Blackfoot, reaching a precision, recall, F-value and accuracy of 100%. In the case of Rancha detection, although the values were slightly lower, it still showed an accuracy of 98%, a recall of 79%, an F-value of 87% and an accuracy of 78%. On the other hand, YOLO V4 also excelled in Blackfoot detection, achieving a precision, recall, F-value and accuracy of 100%. As for the detection of Rancha, the values experienced a minimal variation, obtaining an accuracy of 97%, a recall of 90%, an F-value of 93% and an accuracy of 87%. Finally, we can conclude that YOLO V4 stands out in the detection of both diseases in potato leaves in the region of Abancay, Apurimac.

Keywords**:** *Potato diseases, machine learning, Faster R - CNN, YOLO V4.*

CAPÍTULO I

PLANTEAMIENTO DEL PROBLEMA

1.1 Descripción del problema

La papa tiene nombre científico “Solanum Tuberosum”, es uno de los cuatro alimentos más consumidos e importantes en el mundo, los otros tres son el trigo, el maíz y el arroz. . (RONNIE-GAKEGNE Y MARTINEZ-COCA, 2019)

El Perú es uno de los países con mayor diversidad de papas en el mundo, con ocho especies nativas domesticadas y 2,301 variedades de las más de 4,000 existentes en Latinoamérica. También alberga 91 de las 200 especies silvestres del continente. (MINISTERIO DE DESARROLLO AGRARIO Y RIEGO DEL PERÚ 2020). Según ADAMA Perú (2021), Perú es el principal productor de papa en América Latina y ocupa el puesto 11 a nivel mundial. Durante la campaña 2019-2020, el área cultivada de papa en el país fue de más de 346,000 hectáreas, con una producción de 4.5 millones de toneladas, según el Sistema de Información de Cultivos del Ministerio de Agricultura.

La sierra del Perú, y en especial Apurímac, es una de las zonas en que las personas se dedican a la agricultura y producen diferentes tipos de papa, En 2022, la producción de papa se concentró principalmente en los departamentos de la sierra sur, incluyendo Apurímac, Arequipa, Ayacucho, Cusco, Moquegua, Puno y Tacna. Esta región aportó el 47,7% de la producción nacional, con un total de 2,9 millones de toneladas. (MIDAGRI, 2023)

En el departamento de Apurímac, un estudio determinó que el 95% de las familias en las principales provincias consume papa, mientras que solo el 5% no lo hace, lo que indica una alta demanda del producto que debe ser atendida. (APURÍMAC 2011) En la figura 1 podemos observar un cuadro a detalle de producción de papa, en Apurímac contribuye con el 7% de la producción nacional de papa, equivalente a 423,241 toneladas, destacando su importancia en el suministro de este alimento básico a nivel nacional. (MIDAGRI, 2023)

PERÚ: PRODUCCIÓN DE PAPA SEGÚN ZONA GEOGRÁFICA, 2021-2022

Zona de producción	2021 (t)	2022 (t)	Variación porcentual (%)	Estructura porcentual 2022 (%)
SIERRA SUR	2 470 367	2 865 585	16,0	47,7
Apurímac	402 118	423 241	5,3	7,0
Arequipa	300 631	325 911	8,4	5,4
Ayacucho	353 155	579 176	64,0	9,6
Cusco	440 153	520 810	18,3	8,7
Moquegua	7 232	7 119	-1,6	0,1
Puno	957 130	999 027	4,4	16,6
Tacna	9 949	10 301	3,5	0,2

FUENTE: MIDAGRI, 2023

Figura 1 — Producción de papa en Apurímac

Uno de los principales desafíos en la producción de papa son los problemas fitosanitarios, que afectan tanto a las plantas como a los tubérculos. Estas enfermedades pueden causar pérdidas significativas en el rendimiento y la calidad del producto final, con daños que van desde parciales hasta totales, comprometiendo así la rentabilidad del cultivo. En la región de la sierra sur del Perú, incluido el departamento de Apurímac, las enfermedades más perjudiciales son: Erwinia (Pie negro o pudrición blanda), sarna común, costra negra, pudrición seca y pudrición húmeda, tizón tardío, tizón temprano y virosis. (MÉNDEZ Y GAETE, 2010)

Por otro lado, existen técnicas de machine learning para clasificar imágenes, así por ejemplo se tiene investigaciones como “Detección de una enfermedad de la papa (tizón temprano) utilizando inteligencia artificial (Detection of a Potato Disease (Early Blight) Using Artificial Intelligence)” (AFZAAL y otros, 2021), “Detección automática de tizón en plantas de papa (*Solanum tuberosum L.*) y tomate (*Solanum lycopersicum, L.* 1753) mediante aprendizaje profundo (Automatic blight disease detection in potato (Solanum tuberosum L.) and tomato (Solanum lycopersicum, L. 1753) plants using deep learning)” (ANIM-AYEKO, SCHILLACI Y LIPANI, 2023), como también investigaciones orientadas a la detección de enfermedades de cáncer a la piel (GONZÁLEZ-CRUZ, y otros, 2020), investigaciones que detectan enfermedades en hojas de tomate (FIGUEREDO Y BALLESTEROS, 2016), entre otros; sin embargo, existen pocos estudios orientados a la detección de enfermedades en las hojas de la papa.

Para cultivar la papa, se necesita trabajar en la tierra, preparar abonos para que pueda tener los nutrientes necesarios y de esa manera tener una buena cosecha. El problema es que el cultivo de la papa es afectado por diferentes enfermedades (SORIA, ROJAS Y ORTUÑO, 2021), que de alguna forma ocasionan daño económico, porque a medida que los cultivos de papa van

creciendo se presentan patógenos que afectan su rendimiento y su calidad (LUNA QUECAÑO, y otros, 2020), algunos de estos son hongos, insectos, bacterias, nematodos y virus (RUEDA-PUENTE, y oros, 2014) que causan daños principalmente a las hojas de la papa, provocando que sus hojas se marchiten, provocando pudriciones o malformaciones en el mismo tubérculo en desarrollo (RONNIE-GAKEGNE Y MARTINEZ-COCA, 2019). La Figura 2 muestra las enfermedades que afectan a la hoja de la papa y por consiguiente la calidad en la producción.

Figura 2 —
Enfermedades de la papa

1.2 Enunciado del problema

1.2.1 Problema general

¿En qué medida el uso de las técnicas de Machine Learning detecta las enfermedades en el cultivo de la papa en Abancay Apurímac, 2022?

1.2.2 Problemas específicos

- ¿En qué medida el uso del modelo Faster R-CNN detecta las enfermedades en el cultivo de la papa en Abancay Apurímac, 2022?
- ¿En qué medida el uso del modelo YOLO V4 detecta las enfermedades en el cultivo de la papa en Abancay Apurímac, 2022?

1.3 Justificación de la investigación

Este estudio surge ante la falta de información para detectar a tiempo las enfermedades de Tizón tardío o Rancha y Erwinia (Pie Negro o Pudrición Blanda) que se presentan en el cultivo de la papa, sobre todo dificultades para lograr identificarlos y poder combatirlas a tiempo, provocando que tenga disminución en los cultivos lo cual conlleva a pérdidas económicas.

Esta investigación, basada en tecnología de Machine Learning, facilita la colaboración entre personas y máquinas, permitiendo que las computadoras, gracias a su capacidad para aprender de manera similar a los humanos, puedan detectar enfermedades en los cultivos de papa de manera efectiva, puede aprender a identificarlos a base de imágenes, permitiendo que su

diagnóstico sea mucho más rápido y tenga una mayor precisión, lo que consentirá que pueda combatirlas anticipadamente, ya que al lograr

identificar estos patógenos, se obtendrá el método adecuado para ser exterminados o combatirlos.

CAPÍTULO II
OBJETIVOS E HIPÓTESIS

2.1 Objetivos de la investigación

2.1.1 Objetivo general

Determinar en qué medida el uso de las técnicas de machine Learning detecta las enfermedades en el cultivo de la papa en Abancay Apurímac, 2022.

2.1.2 Objetivos específicos

- Determinar en qué medida el uso de las técnicas de Faster R-CNN detecta las enfermedades en el cultivo de la papa en Abancay Apurímac 2022.
- Determinar en qué medida el uso de las técnicas de YOLO V4 detecta las enfermedades en el cultivo de la papa en Abancay Apurímac 2022.

2.2 Hipótesis de la investigación

2.2.1 Hipótesis general

Al usar correctamente las técnicas: Faster R-CNN y YOLO V4, entonces se determina que YOLO V4 es más eficiente (accuracy) que Faster R-CNN para la detección de enfermedades en el cultivo de papa en Abancay, Apurímac, 2022.

2.2.2 Hipótesis específicas

- El uso de la técnica de Faster R-CNN es menos eficiente (accuracy) en la detección de enfermedades en el cultivo de la papa, en Abancay, Apurímac, 2022.
- El uso de la técnica de YOLO V4 es más eficiente (accuracy) en la detección de enfermedades en el cultivo de la papa, en Abancay, Apurímac, 2022.

2.3 Operacionalización de variables

Tabla 1 — Operacionalización de variables

VARIABLE	DIMENSIÓN	INDICADOR	ÍDICE/ESCALA
Variable 1: Machine Learning	Faster R-CNN	Precisión (Precisión)	Escala de 0 a 1
		Exhaustividad (Recall)	Escala de 0 a 1
		Valor-F (F-value)	Escala de 0 a 1
		Exactitud (Accuracy)	Escala de 0 a 1
	YOLO V4	Precisión (Precisión)	Escala de 0 a 1
		Exhaustividad (Recall)	Escala de 0 a 1
		Valor-F (F-value)	Escala de 0 a 1
		Exactitud (Accuracy)	Escala de 0 a 1
Variable 2: Enfermedades del cultivo de papa	Tizón Tardío o Rancha	Cantidad Real	Número Absoluto
		Cantidad de Detección	Número Absoluto
	Erwinia (Pie Negro o Pudrición Blanda)	Cantidad Real	Número Absoluto
		Cantidad de Detección	Número Absoluto

CAPÍTULO III
MARCO TEÓRICO REFERENCIAL

3.1. Antecedentes

3.1.1 Antecedentes internacionales

a) FUENTES PLAZA Fabián Nicolás (2021), realizó un proyecto de tesis "Visión por Computadora para el manejo de plagas y enfermedades en cultivos de papa", el autor describe la importancia de la papa como alimento, para combatir estas enfermedades y tomar decisiones de manera correcta, propone la alternativa del uso de aprendizaje profundo con modelos de redes neuronales convolucionales (CNN), de tal forma generar clasificadores de alta precisión entre distintos tipos de cultivos de papa, con los cuales se obtuvo como resultado un modelo capaz de diferenciar las hojas entre la clase sana, del tizón temprano y tizón tardío con una precisión del 98,44%, y para el caso de los tubérculos con una precisión del 96,88%, permitiendo abordar el problema y tener resultados visibles.

b) En artículo de investigación de OPPENHEIM Dor (2019) , titulado "Using deep learning for image-based potato tuber disease detection", aparte de señalar que indica una gran variedad de enfermedades, indica que también el clima y la tierra afectan la calidad y el rendimiento de los tubérculos. El autor señala que es necesario examinar los tubérculos de manera minuciosa, sin embargo, esas estructuras no siempre están presentes, indicando que la detección y clasificación manual de tubérculos enfermos es difícil, costosa y lleva mucho tiempo, mientras que la inspección desde una computadora será más eficiente y rentable. Por lo tanto, propone aprovechar los avances de visión por la computadora y el reconocimiento de objetos, una red neuronal convolucional profunda, utilizando el modelo CNN, los resultados que se obtuvieron al estar completamente entrenados aborda entre el 83% y 96% de los datos, los resultados sugiere la combinación de un algoritmo de ventana deslizante o una red de detección de objetos, como R-CNN más rápida, de tal manera puedan facilitar la detección de las enfermedades de los tubérculos de manera más completa.

c) También, tenemos el artículo de investigación "Detection of a Potato Disease (Early Blight) Using Artificial Intelligence" de AFZAAL Hassanm (2021), el autor señala que el tema de identificación de enfermedades de las plantas es un temas investigado desde hace mucho tiempo, en este artículo el autor se enfoca en uno de los varios tipos de enfermedades que existen, Tizón temprano, su estrategia para identificarlo y tratarlo, propuso el método de detección de enfermedades mediante el uso de una red neuronal. Su algoritmo aplicó el análisis de componentes principales a un conjunto de imágenes de baja resolución, seguido de la alimentación de las imágenes procesadas a una red neuronal para la detección de enfermedades de las plantas, utilizó una técnica de cuantificación de enfermedades basada en el color no supervisada, que dividió la imagen en varias clases seguidas del entrenamiento de cada clase mediante una técnica de aprendizaje supervisado probabilístico. La presente investigación se centró en comparar tres edades de CNNs (GoogleNet-antigua, VGGNet-mediana edad, y EfficientNET-nueva) para identificar la enfermedad del Tizón en sus diferentes etapas de crecimiento, también calcularon sus tiempos de secuela de las CNN y evaluar su capacidad en el desarrollo de un pulverizador inteligente para la toma de decisiones al momento de aplicar los fungicidas en base a sus necesidades, lo que añade novedad e innovación al estudio realizado.

d) En el artículo de investigación "Potato Plant Leaves Disease Detection and Classification using Machine Learning Methodologies" de los autores ADITI Singh y HARJEET Kaur (2021), consideran que la agricultura es uno de los sectores escenciales para la supervivencia de la humanidad, así como la importancia de la digitalización facilita la diversidad de tareas que se presentan en el día a día. En esta investigación, los autores presentaron una metodología para la detección y clasificación de enfermedades que afectan a las plantas de papa. Tomaron consideración en el conjunto de datos accesible, estándar y confiable conocido popularmente como Plant Village Dataset. En el proceso de segmentación de imágenes, implementaron la metodología K-means. Para la extracción de características, los autores aplicaron el concepto de Gray Level Co-occurrence Matrix (Matriz de Co-ocurrencia de Niveles de Gris), y con fines de clasificación, se emplearon la metodología de máquina de soporte vectorial multiclase. Con la metodología que fue propuesta en la investigación se logró un alcance de precisión del 95,99%.

e) En el trabajo de graduación presentado por AMORES ROMERO Kevin Steven y TRELLES MUÑOZ Kaiser Geovanny (2022), con el título "Implementación de un sistema para detectar la enfermedad de la Sigatoka Negra en una plantación de banano empleando técnicas de visión artificial.", Los autores se enfocaron en la detección de la enfermedad de la Sigatoka Negra en una plantación de banano, reconociendo la importancia de contar con una herramienta visual para identificar los estados iniciales de la enfermedad. Para abordar esta necesidad, implementaron un sistema utilizando técnicas de visión artificial. Su proyecto se centró en el uso de una red neuronal YOLOv4, la cual fue entrenada mediante el etiquetado manual de imágenes. Para facilitar este proceso, utilizaron Google Colab como una herramienta en la nube, lo que les permitió ejecutar códigos en línea de manera eficiente. Los autores lograron una precisión superior al 85%, estableciendo así la incidencia de la enfermedad en cada lote de plantaciones de banano.

3.1.2 Antecedentes nacionales

a) CARRIÓN HERRERA Cinthya María (2017) presentó un proyecto de tesis "Detección temprana del Potato Yellow Vein Virus en cultivos de Solanum tuberosum L. mediante la teledetección", el autor se enfoca en el *Potato yellow vein virus* (PYVV*)* que es un virus que reduce la producción de la papa, propone la técnica de la teledetección mediante el uso del espectroradiómetro, por el cual puede evaluar el estado nutricional y fitosanitario de las plantas, detectando las incidencias de plagas y enfermedades. En su proceso de investigación, se seleccionaron 5 tipos de papa (Única, Clon W.A. Canchan INIA, Amarillis y Costanera), con los que se realizaron el proceso de experimentación al inducir el PYVV; resultados en los que se identificó la variedad más susceptible frente a este virus, la Canchan INIA, presentó mayor porcentaje de reducción en su rendimiento con un 36,63% y el Clon W.A. fue la variedad menos afectada con una reducción de su rendimiento en un 6,67%.

b) El proyecto de tesis presentado por GALAN ZAPATA Jefferson Luis (2021), con el título "Sistema Inteligente de reconocimiento de imágenes para apoyar el diagnóstico de plagas y enfermedades en el cultivo de arroz en el departamento de Lambayeque en el año 2019", este proyecto es enfocado en el reconocimiento de enfermedades y plagas del arroz, indicado que también es un alimento primordial,

como la papa. El autor propone implementar un sistema inteligente de reconocimiento de imágenes para lograr el diagnóstico de plagas y enfermedades, mediante redes neuronales convolucionales. Para su desarrollo de software se hizo uso de las metodologías CommonKads y RUP, realizando sus pruebas de calidad. Como resultado se logró establecer el modelo que brinda un reconocimiento de imágenes para identificar las peculiaridades de las plagas y enfermedades, de este modo considerar las que inciden en los cultivos.

c) CÓRDOVA PÉREZ Claudia Sofía (2021) presentó una tesis de investigación titulada "Aplicación de aprendizaje profundo para la detección y clasificación automática de insectos agrícolas en trampas pegantes". En su estudio, destaca los desafíos que enfrenta la horticultura en diversas regiones de Perú debido a la alta incidencia de plagas de insectos. Para abordar este problema, se utilizan trampas pegantes para atraer y capturar insectos. Sin embargo, el proceso manual de identificación de estos insectos puede verse afectado por factores como la habilidad individual y la fatiga, afectando la precisión de la información recopilada. La autora propone el uso de modelos de detección preentrenados, específicamente Faster R-CNN y YOLOv4, aplicándoles aprendizaje por transferencia para adaptarlos a la detección de tres tipos de plagas de insectos: mosca blanca, mosca minadora y pulgón verde. Los resultados obtenidos muestran que el modelo Faster R-CNN alcanzó un 94,06%, mientras que el modelo YOLOv4 obtuvo un 95,82%. La conclusión es que el rendimiento de ambos detectores es aceptable para la detección automática de plagas en trampas pegantes.

3.1.3 Antecedentes locales

Se identificaron algunos trabajos locales similares y relacionados con machine learning, así, por ejemplo:

a) MORENO MAYHUIRE Joel Saul (2020) desarrollo un proyecto de Tesis titulada: “Eficiencia de un Ssistema de Control de Calidad mediante Procesamiento Digital de Imágenes en la Clasificación de la Tunta en la Planta de Producción de Kishuara – Andahuaylas”, en esta investigación, el autor destaca la importancia de la tunta como un producto destacado en los platos típicos del Perú, ganando interés no solo en el altiplano peruano sino también en la planta de producción de la tunta en Kishuara – Andahuaylas. Indica que la tunta, con sus características distintivas de

tamaño, forma, color y sabor, ha enriquecido nuestra cultura culinaria, ocupando una posición relevante tanto a nivel nacional como internacional en la gastronomía. El autor llevó a cabo un estudio de eficiencia de clasificación utilizando algoritmos de machine learning de tipo supervisado para el entrenamiento de datos y la construcción de modelos de clasificación, con el fin de evaluar posteriormente su rendimiento. El autor consideró 800 unidades de tuntas de diversas categorías, conformando un banco de imágenes que fueron analizadas individualmente mediante un algoritmo de procesamiento digital de imágenes. Este algoritmo le permitió la extracción de características relacionadas con el tamaño, forma y color, contribuyendo a la construcción del corpus de conocimiento. El corpus fue entrenado con diversos algoritmos de aprendizaje automático supervisado, como K-Nearest Neighbors, Decision Trees, Gaussian Naive Bayes, Multinomial Naive Bayes y Support Vector Machines, generando así modelos de clasificación. Según los resultados obtenidos, el autor determinó que el modelo generado con el algoritmo Support Vector Machines exhibió el mejor rendimiento en la clasificación de tuntas, con una Accuracy del 93,13%, un Recall del 93,48%, y un F1-Score del 93,18%.

Debido a la eficiencia demostrada por el modelo de clasificación basado en el algoritmo Support Vector Machines, el autor lo utilizó para la construcción de un sistema clasificador de tuntas, integrando algoritmos de procesamiento digital de imágenes. Los resultados fueron destacables, logrando una Accuracy del 97,5%, un Recall del 97,5%, y un F1-Score del 97,51%.

3.2 Marco teórico

3.2.1 Machine Learning

Machine Learning, se considera como la destreza de programación de computadoras con el fin de aprender de los datos. Los datos son considerados como un conjunto de capacitación o modelos. Se considera que usar Machine Learning es completamente práctico, ya que el programa podrá aprender a base de los patrones que generan todos los datos de cualquier aplicación, una web de búsqueda, antispam de correos electrónicos, etc. (RUSSELL 2018)

a) Tipos de sistemas de Machine Learning

Machine Leaning supervisado

En este tipo de aprendizaje, agrupa tareas de clasificación, que son los datos con los que se alimenta al algoritmo, con una solución y son referidos con etiquetas. Un ejemplo, es la predicción del valor de un inmueble, los algoritmos aprenderán de los

datos basados en los precios, ubicación y tamaño, estos son llamados predictores o regresión. (RUSSELL, 2018)

Algoritmos Supervisados más importantes:

- K-nearest neighbors (KNN, vecinos más cercanos K)
- Red Neuronal
- Máquinas de soporte de vectores
- Regresión logística
- Arboles de decisiones y bosques aleat

Machine Learning no supervisado

Es considerado como los datos sin etiquetas, a base de los datos que existen, usará un algoritmo de asociación y dividirá cada grupo en subgrupos.

Existen algoritmos relevantes como los algoritmos de visualización; Se necesitará proporcionar muchos datos y datos sin etiquetas como salida, y como resultado se obtendrá visualización 2D o 3D como salida. Su meta es lograr que la salida sea tan sencilla de tal manera que no se extravíe ninguna información.

Algoritmos No Supervisados más importantes:

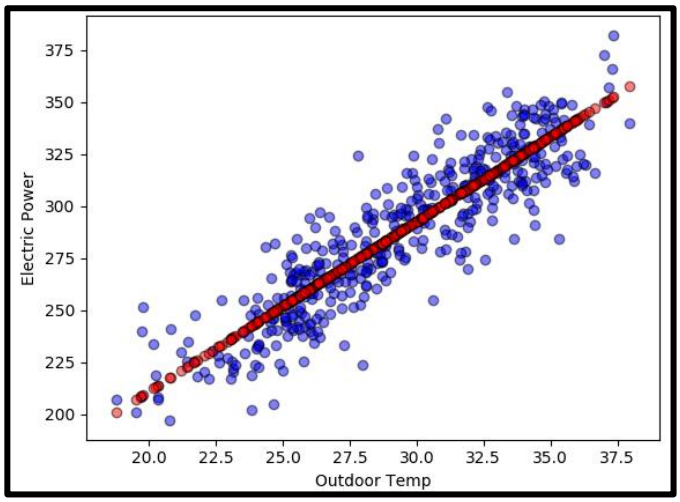
- Agrupamiento (Clustering): Medios k, análisis de agrupamiento jerárquico
- Machine Learning de Asociación de Regla: Eclat y A priori
- Visualización y Reducción de Dimensionalidad: Núcleo PCA, distribuido de t PCA. (RUSSELL, 2018)

b) Modelos de Machine Learning

Se consideran tres modelos según su algoritmo:

Modelos lineales

Destaca desde modelos muy conocidos y usados en la regresión lineal. Tratan de encontrar una línea que se “ajuste” bien a la nube de puntos que se disponen. Este modelo tiene el problema de “overfit”, es decir, que se encuentra muy preciso, y existe el riesgo que los datos puedan actualizarse o llegar más nuevos. No son relativamente simples, y no ofrecen buenos resultados para procesos más complejos, (SANDOVAL, 2018). Podemos ver Figura 3.



FUENTE: SANDOVAL, 2018

Figura 3 — Test y regresión lineal

Los algoritmos con los que puede trabajar:

a) Perceptrón: Es un modelo de neurona simple, su aprendizaje se basa en la corrección de error, con la competencia para captar e identificar patrones. A su vez, se fundamenta en una serie de sensores o entradas desde donde recepciona los datos que podrá discriminar o identificar, para ello se cuenta con la neurona de salida que da a conocer el resultado, a base de 0 y 1. (RUIZ, 2018)

Funciona de la siguiente forma:

1. Iniciamos los pesos sinápticos con valores aleatorios en el intervalo [1,1]. Ir al paso 2 con k=1.
2. Iteración k-ésima:

   a. Calcular la salida y(k) según la función comentada previamente
3. Corregir pesos sinápticos:

   a. Si la salida de $y(k) \neq z(k)$ (la salida no es la deseada), entonces modificamos pesos aplicando la regla de aprendizaje del perceptrón simple comentada previamente
4. Fin.

   a. Si los pesos sinápticos no se han modificados en las últimas p iteraciones entonces la red se ha estabilizado

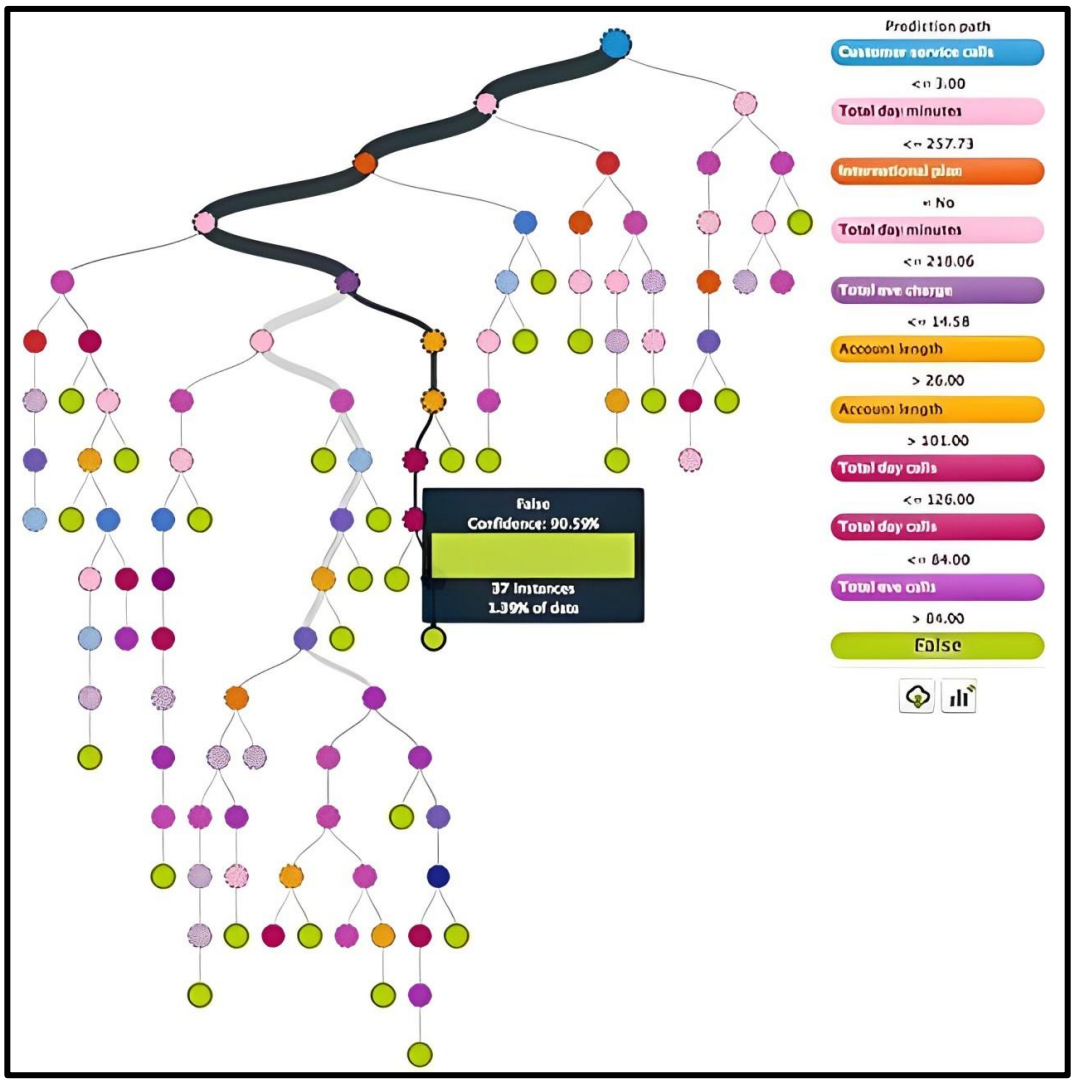
   b. En otro caso ir al paso 2 con k = k+1 (RUIZ, 2018).

b) Regresión logística: Es un patrón para clasificación, más no para regresión, se considera un algoritmo muy sencillo para ser implementado; no obstante, tiene complejidad a la hora de coincidir si los datos que se están tratando no son separables linealmente. Para entender cómo funciona este método es relevante

comprender la ratio de probabilidad, haciendo referencia a la probabilidad con la que se da un hecho. Éste ratio se puede definir como sigue: $\frac{p}{1-p}$ donde p es la probabilidad para que ocurra un evento positivo, señalando la certeza con la que se produce un evento. (RUIZ, 2018)

Modelos de árbol

Estos son los que tienen mayor exactitud, estabilidad y simplicidad para interpretar, Son modelos más precisos, estables y sencillos de interpretar, esencialmente porque da lugar a más reglas decisivas que se representan como un árbol. De este modo, destaca los árboles de decisión y los de random forest (una media de árboles de decisión), debido a su precisión, elaboración y su capacidad predictiva, sin embargo, desperdicia rendimiento, (SANDOVAL, 2018) ver figura 4



FUENTE: SANDOVAL, 2018

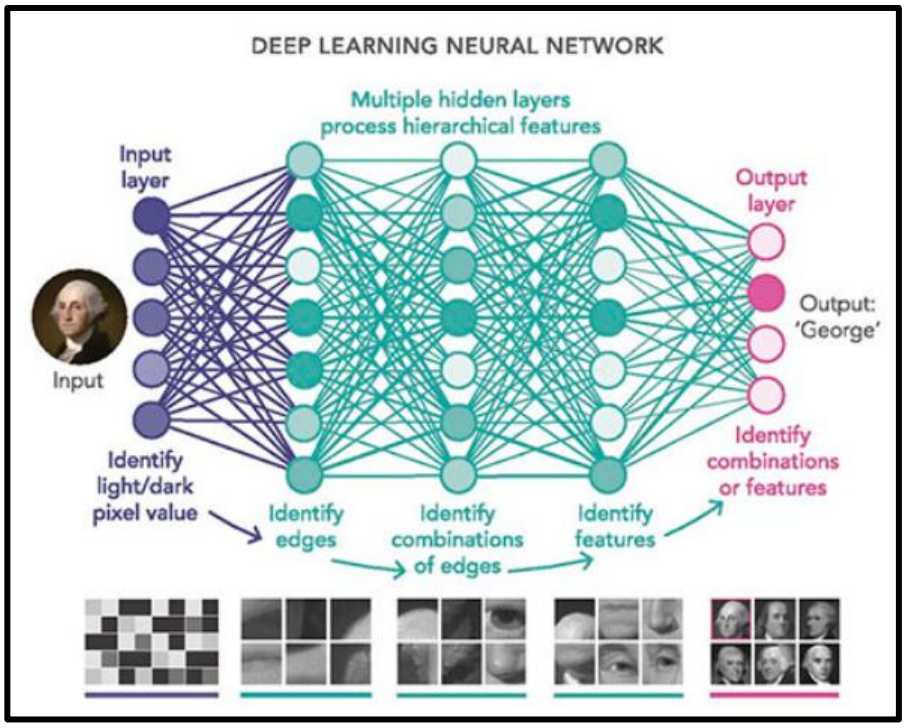
Figura 4 — Modelo de árbol

El algoritmo que se puede usar para este modelo:

a) Decision tree: Refiere una clase de algoritmo cuya visualización de certeza está fundamentado por sus pautas de clasificación muy sencillas que facilitan la comprensión. Su proceso de fallo es de la siguiente manera: cada nodo intermedio del árbol se considera un atributo o característica de datos a tratar, se considera la raíz del nodo el atributo más relevante y cada hoja corresponde a una clase o etiqueta. (LÓPEZ BRIEGA, 2018)

Redes neuronales

Estas replican de alguna manera el comportamiento del cerebro, como las redes neuronales, estas son redes artificiales de neuronas, las cuales cuentan con un sinfín de neuronas que están entrelazados en una red para permutar información. En tal sentido, es uno de los modelos más utilizados, ya que representa las habilidades cognitivas de razonamiento que se Esta réplica del funcionamiento del cerebro es uno de los modelos más usados, ya que presenta sus capacidades cognitivas de razonamientos que se obtienen mediante adiestramiento o ejercicio (SANDOVAL, 2018), ver Figura 5.



FUENTE: QASIM Jaffery, 2023

Figura 5 — Redes neuronales

El tipo de algoritmos que se puede usar para este modelo:

a) Multi Layer Perceptron: El perceptrón multicapa, una evolución del perceptrón simple, aborda las limitaciones en la clasificación de datos no linealmente separables. Su arquitectura consiste en varias capas de neuronas: una capa de entrada (sensores o características del conjunto de datos), una capa de salida (tipos a discriminar) y una o más capas intermedias ocultas (unidades de proceso sin conexión directa al exterior). Esta estructura forma una red de alimentación hacia adelante. (RUIZ, 2018)

Medidas de desempeño del modelo

Con esta matriz se puede calcular el desempeño del algoritmo en términos de precisión, exhaustividad, exactitud, valor F. Podemos revisar a detalle en la Tabla 2.

- **FP: False Positive (Falso positivo):** Es un error en el que el modelo identifica incorrectamente un caso negativo como positivo.
  Ejemplo: En una prueba de detección de enfermedades, si el modelo predice que una persona está enferma (positiva), pero en realidad está saludable (negativa), esto es un falso positivo.
- **FN: False Negative (Falso negativo):** Es un error cuando el modelo predice incorrectamente un caso positivo como negativo.
  Ejemplo: En la misma prueba de detección de enfermedades, si el modelo predice que una persona está saludable (negativa), pero en realidad está enferma (positiva), esto es un falso negativo.
- **TP: True Positive (Verdadero positivo):** Ocurre cuando el modelo identifica correctamente un caso que realmente pertenece a la clase positiva.
  Ejemplo: Si el modelo predice que una persona está enferma y efectivamente lo está, esto es un verdadero positivo.
- **TN: True Negative (Verdadero negativo):** Ocurre cuando el modelo predice correctamente un caso negativo.
  Ejemplo: Si el modelo predice que una persona está sana y efectivamente está sana, esto es un verdadero negativo. (ALPAYDIN, 2014)

Tabla 2 — Matriz de confusión

Predicción	Positivo	Negativo
Positivo	True Positive (Verdadero Positivo)	False Negative (Falso Negativo)
Negativo	False Positive (Falso Positivo)	True Negative (Verdadero Negativo)
FUENTE: ALPAYDIN, 2014		

Precision (Precisión):

La precisión se considera a la relación entre observaciones positivas indicadas como correctamente y el total. (GIL, CRUZ Y PERDOMO, 2022)

$$\boldsymbol{Precision} = \frac{True\ positives}{True\ positives\ +\ False\ positives}$$

Recall (Exhaustividad)

Recall se considera a la proporción de observaciones positivas indicadas como correctamente a todas las observaciones en la clase real. (GIL, CRUZ Y PERDOMO, 2022)

$$\boldsymbol{Recall} = \frac{True\ positives}{True\ positives\ +\ False\ negatives}$$

F-Value

Es el promedio ponderado de precisión y recuperación. De tal manera, se considera tanto los falsos positivos como los falsos negativos. F1 suele ser más ventajoso que la precisión, fundamentalmente si asume una división de clases desigual. La precisión actúa eficientemente si los falsos positivos y los falsos negativos tienen un costo similar. Si el costo de los falsos positivos y los falsos negativos es muy diferente, es mejor mirar tanto Precision como Recall. (GIL, CRUZ Y PERDOMO, 2022)

$$\boldsymbol{F-Value} = 2 * \frac{Precision * Recall}{Precision\ +\ Recall}$$

Accuracy

Es la medida de rendimiento más intuitiva y es una relación entre la observación indicada como correcta y el total de las observaciones. Si ésta, tiene una alta precisión, por lo tanto, nuestro modelo se valida como la mejor, solo si la precisión es una gran medida, siempre y cuando, tiene conjuntos de datos simétricos donde los valores de falsos positivos y falsos negativos son casi iguales. (GIL, CRUZ Y PERDOMO, 2022)

$$\boldsymbol{Accuracy} = \frac{True\ positives + True\ negatives}{True\ positives + False\ positives + False\ negatives + True\ negatives}$$

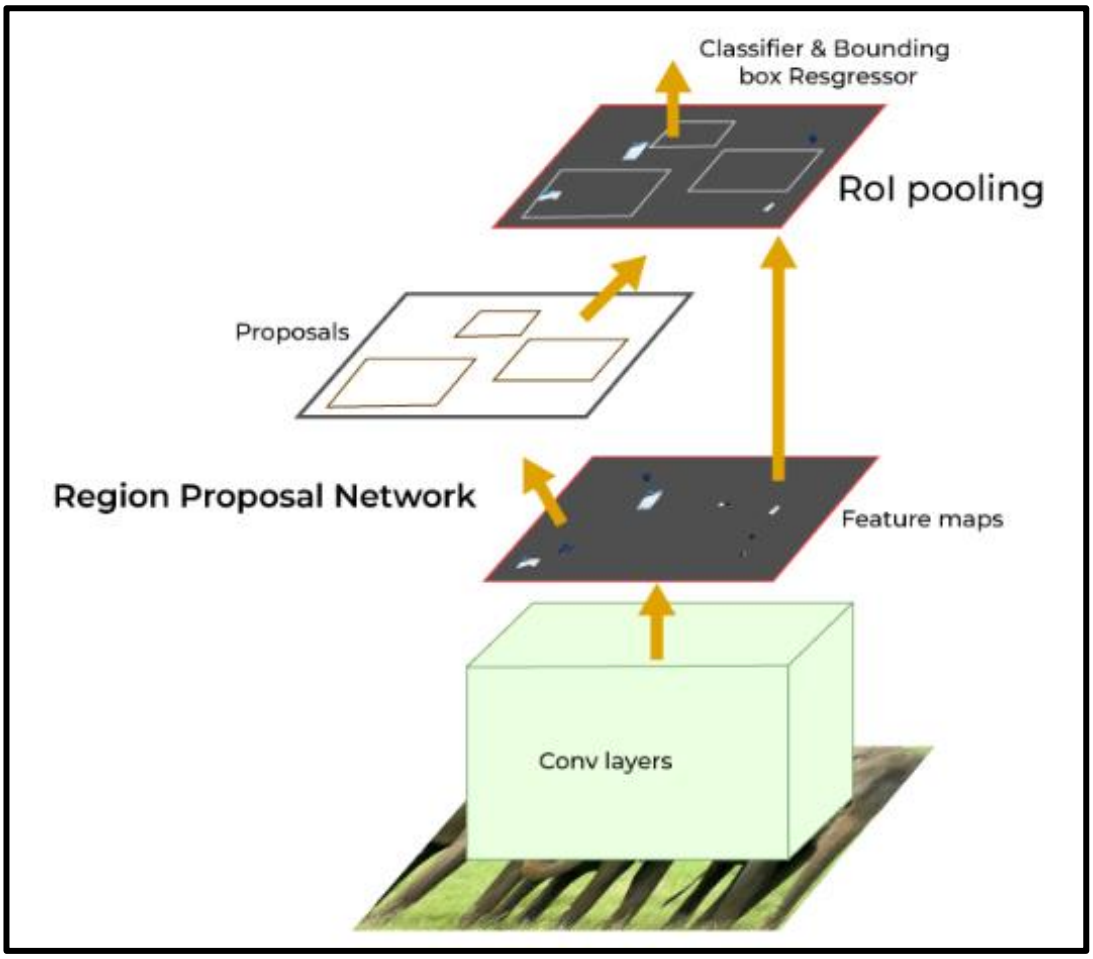
3.2.2.1. Faster R-CNN

Es un sistema de detección de objetos, que se encuentra compuesto por dos módulos, el primero, es una red insondable convolucional que propone regiones, y el segundo, es el detector Fast R-CNN, utilizando regiones planteadas. El sistema completo es una red única y unificada para consentir la detección de objetos.

Fast R-CNN logra disminuir la carga computacional, con CNN disminuye el tiempo de detección que presenta la capa, permitiendo buenos resultados en tiempo real. (REN y otros, 2017)

Podemos ver en la figura 6 la arquitectura de Faster R-CNN que consta de dos componentes:

1. Red de Propuestas de Regiones (RPN)
2. Detector Fast R-CNN



FUENTE: GEEKSFORGEEKS, 2023

Figura 6 — Arquitectura Faster R-CNN

Backbone de la Red Neuronal Convolucional (CNN) - Convolutional Neural Network (CNN) Backbone

Es la capa inicial en la arquitectura de Faster R-CNN, y se encarga de extraer mapas de características de la imagen de entrada. Estos mapas, generados por el backbone (como por ejemplo ResNet o VGG), capturan diferentes niveles de información visual que son utilizados posteriormente por la Red de Propuestas de Regiones (RPN) y el detector Fast R-CNN.

El propósito principal del backbone es extraer características importantes de la imagen a través de múltiples capas convolucionales que aplican diversos kernels para identificar detalles jerárquicos. Las capas iniciales capturan características de bajo nivel, como bordes y texturas, mientras que las capas más profundas extraen características semánticas avanzadas, como partes de objetos y formas. Al usar las mismas características jerárquicas tanto para la RPN como para el detector Fast R-

CNN, se logra una notable reducción en el tiempo de computación y el uso de memoria, optimizando así la eficiencia del modelo. (GEEKSFORGEEKS, 2023)

Region Proposal Network (RPN)

Region Proposal Network o traducido Red de Propuestas de Regiones (RPN), es un componente clave de Faster R-CNN que genera posibles regiones de interés en imágenes. A diferencia de los modelos anteriores que usaban el algoritmo de búsqueda selectiva en CPU, la RPN utiliza una red convolucional, lo que reduce significativamente el tiempo de procesamiento a 10 ms por imagen y mejora la representación de características al compartir capas con las etapas de detección. (GEEKSFORGEEKS, 2023)

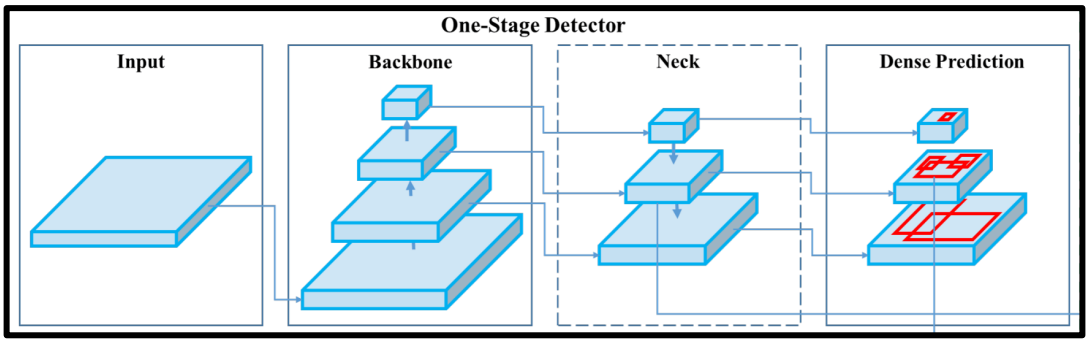
Fast R-CNN detector

El detector Fast R-CNN es responsable de la detección de objetos dentro de las regiones propuestas, clasificando cada región y ajustando las cajas delimitadoras para mejorar la precisión de la detección. Es una mejora significativa sobre los métodos anteriores, ya que permite una detección más rápida y precisa al procesar las regiones de interés de manera eficiente. (GEEKSFORGEEKS, 2023)

3.2.2.2.YOLO V4

Es un detector de objetos basado en CNN. En general, existen dos tipos de algoritmos de detección de objetos, de una o dos etapas. Los algoritmos de dos etapas, requieren de una etapa preliminar donde seleccionan regiones candidatas sobre las cuales detectará objetos. En YOLO V4, se basa en una sola etapa, descarta la búsqueda de regiones candidatas con la finalidad de lograr detecciones rápidas. (BOCHKOVSKIY, WANG Y LIAO, 2020)

El motivo principal de esta versión de Yolo V4 es optimizar el detector para realizar cálculos en paralelo. Está compuesto de 3 etapas que podemos observar en la figura 7:



FUENTE: ULTRALYTICS YOLO DOCS, 2024

Figura 7 — Arquitectura de la red YOLOv4

- **Backbone:** CSPDarknet53, una red que incrementa la capacidad de aprendizaje y, al integrar el módulo de agrupamiento de pirámide espacial, mejora el campo receptivo y facilita la identificación de características clave.
- **Neck:** Se utiliza el módulo de agrupación de pirámides espaciales y la agregación de rutas PANet. PANet se implementa en lugar de las redes piramidales de características usadas en la detección en YOLOv3.
- **Head:** YOLOv3. (ROZADA RANEROS, 2021)

3.2.2 Enfermedades de la papa

Las enfermedades con microrganismos no son visibles a la vista del ojo humano, pero atacan cualquier parte de la planta, entre ellos tenemos a los hongos, los virus y las bacterias. Pueden provocar la pérdida de la planta debido a que se han instalado dentro de la misma, de tal modo que las bacterias provocan pústulas, agallas y tumores; mientras que los virus, decoloraciones, manchas y purulencias de color blanco, gris, rojo, café o negro; a su vez, deformaciones o encrespamiento de las hojas. (UNIVERSIDAD NACIONAL DE LA PLATA, 2020)

A) Tizón tardío o rancha

Esta enfermedad es conocido con diferentes nombres, principalmente como añublo o rancha. Esta daña a las hojas, tallos y tubérculos de la planta misma. Mostrándose sintomatologías distintas en sus órganos:

- **En hojas**: Inicialmente, aparecen como máculas minúsculas de color verde pálido u oscuro. En condiciones óptimas de temperatura (12-15°C) y 100% de humedad relativa, estas máculas, que se forman en los bordes y ápices de los foliolos, se desarrollan rápidamente, causando grandes lesiones necróticas de color marrón a negro con un halo amarillento. (TORRES, 2002) ver Figura 8.

Figura 8 — Desarrollo inicial de los foliolos

- **En tallos:** La sintomatología se manifiesta como lesiones oscuras y persistentes, ubicadas generalmente en el tercio medio o superior de la planta, y que pueden alcanzar hasta 10 cm de longitud. Estas lesiones son propensas a romperse fácilmente debido al viento o al roce con fuerzas externas (TORRES, 2002), ver figura 9.

Figura 9 — Lesión necrótica en los tallos

- **En tubérculos:** En el fragmento exterior de los tubérculos contaminados se divisan depresiones muy ligeras y desiguales, cuya dimensión es inconstante y dura. Al realizar un leve raído, debajo de la piel afectada se puede observar que el tejido es

de tono marrón, con aspecto granuloso que se aproxima desde los contornos hacia la médula (Torres, 2002), ver figura 10.

FUENTE: TORRES, 2002

Figura 10 — Sección transversal del tubérculo de la papa

B) Erwinia (Pie negro o pudrición blanda)

Esta enfermedad es ocasionada por bacterias como *Erwinia carotovora spp. Carotovora (Ecc) y Erwinia carotovora spp. atroséptica (Eca),* la sintomatología de este padecimiento se presenta en cualquiera de las etapas de la evolución del cultivo, principalmente inicia en el mismo tubérculo el cual se extiende hacia arriba siguiendo el tallo, por lo que se puede visualizar en la base del tallo como pudrición de color negro. (MÉNDEZ Y GAETE, 2010)

- **En hojas:** Sus hojas tienden a enrollarse con dirección hacia arriba, y tienden a marchitarse y morir.
- **En tallos:** Cuando llegan a ser afectados se muestran de color amarillento (clorótico), llegando a marchitarse y morir, ver Figura 11.

Figura 11 — Tallos afectados por la Erwinia

- **En tubérculos:** Como ya se comentó, la enfermedad inicia en el tubérculo, con la pudrición, ver Figura 12.

FUENTE: TORRES, 2002

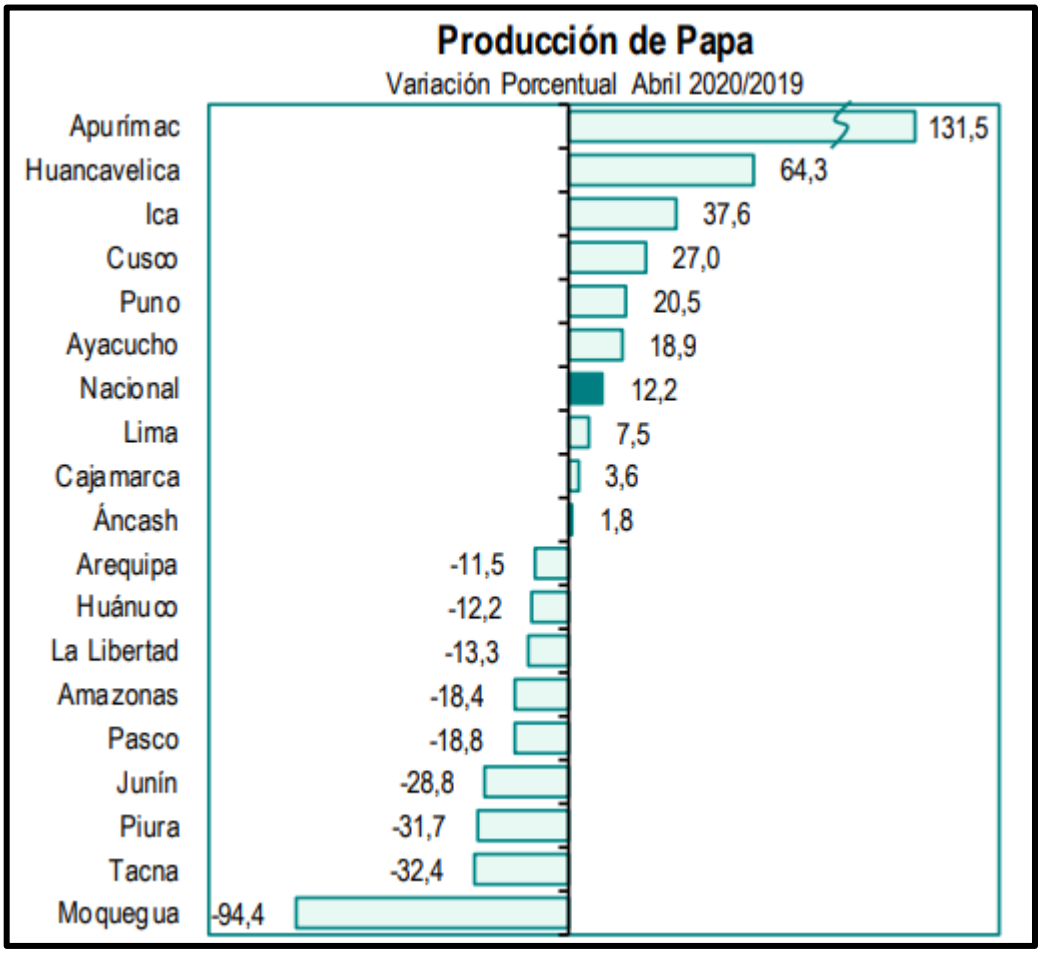
Figura 12 — Tubérculo con la enfermedad de la erwinia

3.2.1.1. Producción de la papa

En abril de 2020, la producción de la papa totalizó 1 millón 133 mil 33 toneladas y creció en 12,2% determinado por las mayores superficies cosechadas (INEI y otros, 2020).

Los departamentos que contribuyeron con este resultado fueron Apurímac (131,5%), Huancavelica (64,3%), Cusco (27,0%), Puno (20,5%) y Ayacucho (18,9%), los cuales participaron con el 70,9% de la producción nacional de este tubérculo.

Igualmente, se expandió en Ica (37,6%), Lima (7,5%), Cajamarca (3,6%) y Áncash (1,8%). Sin embargo, disminuyó en La Libertad (-13,3%) y Huánuco (-12,2%), por la presencia de la rancha que afectó dicho cultivo. Así también en Moquegua (-94,4%), Tacna (-32,4%), Piura (-31,7%), Junín (-28,8%), Pasco (-18,8%), Amazonas (-18,4%) y Arequipa (-11,5%) (INEI y otros, 2020), ver Figura 13.



FUENTE: INEI y otros, 2020

Figura 13 — Producción de la papa - valoración porcentual 2019/2020

3.1 Marco conceptual

a) **Accuracy (Exactitud):** Es una métrica que evalúa la proporción de predicciones correctas hechas por un modelo en relación con el total de predicciones realizadas. (Gil, Cruz y Perdomo, 2022)

b) **Análisis predictivo:** Es una metodología que utiliza datos históricos, algoritmos estadísticos y modelos de Machine Learning para identificar patrones y predecir eventos futuros, aplicándose en áreas como la salud, finanzas, marketing y agricultura. Este enfoque permite anticipar comportamientos y apoyar la toma de decisiones basadas en la experiencia pasada. (ESPINO, 2017)

c) **Dataset (Conjunto de datos):** Es una colección organizada de datos que se emplea para el análisis, la capacitación de modelos de Machine Learning y la evaluación de algoritmos. En el ámbito del aprendizaje automático, estos conjuntos de datos suelen contener ejemplos etiquetados que reflejan tanto las entradas como las salidas esperadas para un modelo. Son esenciales para el proceso de entrenamiento y validación, ya que proporcionan a los algoritmos la información necesaria para identificar patrones y relaciones dentro de los datos. (SZELISKI, 2021)

d) **Deep learning o aprendizaje profundo:** Es considerado un subcampo del Machine Learning, emplea redes neuronales artificiales organizadas de manera jerárquica, similar a la estructura del cerebro humano, con nodos conectados entre sí como una telaraña. Al igual que otros algoritmos de aprendizaje, entrena a los ordenadores para que puedan aprender y hacer deducciones de manera autónoma. (CENTENO, 2019)

e) **Enfermedad**: Se considera a la deformación o malfuncionamiento de células o tejidos vegetales, producido ya se por agentes internos o externos, bióticos o abióticos que afecta cada parte de la planta, y que ocasiona malos síntomas y puede terminar en daños permanentes. (MÉNDEZ Y GAETE, 2010)

f) **Entrenamiento de modelo:** Es el proceso por el cual un modelo de Machine Learning aprende a identificar y detectar objetos en imágenes a partir de datos de entrada. El sistema examina y descompone las imágenes para extraer características relevantes, ajustando sus parámetros en múltiples iteraciones para reconocer patrones. El objetivo es lograr que el modelo realice detecciones automáticas y precisas sin intervención humana, permitiendo un reconocimiento autónomo de objetos. (ROZADA, 2021)

g) **Faster R-CNN:** "Faster Region-Based Convolutional Neural Network" (Red Neuronal Convolucional Basada en Regiones Más Rápida), es un modelo de aprendizaje profundo para detectar objetos en imágenes y videos, mejorado respecto a R-CNN y Fast R-CNN, con mayor eficiencia y velocidad sin perder precisión. (GEEKSFORGEEKS, 2023)

h) **Google colab:** Es un entorno colaborativo basado en Python que permite almacenar borradores y trabajar con datos guardados en Drive, facilitando el intercambio de trabajos con tu equipo. Además, es un servicio gratuito de Google que promueve la investigación en Aprendizaje Automático e Inteligencia Artificial. (THIAGO, 2020)

i) **Inteligencia artificial:** Es una rama de la informática enfocada en crear sistemas capaces de llevar a cabo tareas que suelen requerir inteligencia humana, como aprender, resolver problemas y tomar decisiones. (RUSSELL Y NORVIG, 2004)

j) **LabelImg**: Es una herramienta visual utilizada para la anotación y etiquetado de áreas específicas en imágenes. Este procedimiento consiste en asignar etiquetas o categorías a datos

en bruto, como imágenes, para prepararlos adecuadamente para el entrenamiento de modelos de Machine Learning. (CÓRDOVA, 2021)

k) **LOSS:** La función de pérdida mide la discrepancia entre las predicciones del modelo y los valores reales, evaluando el grado de desviación de las predicciones respecto a las etiquetas correctas. Su objetivo es reducir este valor para mejorar la precisión del clasificador. Un valor de pérdida cercano a 0 indica una alta precisión, ya que las predicciones son correctas y el modelo penaliza efectivamente las clasificaciones erróneas. Minimizar la función de pérdida contribuye a maximizar la precisión del clasificador y a mejorar su capacidad predictiva. (RUIZ, 2018)

l) **mAP (Mean Average Precision):** s una métrica comúnmente empleada para evaluar el rendimiento de modelos en detección de objetos y recuperación de información. Proporciona una evaluación integral de la precisión del modelo al combinar tanto la precisión como el recall en diferentes niveles de umbral. (REDMON y otros, 2016)

m) **Python:** Es un lenguaje de programación de alto nivel, interpretado y general, conocido por su simplicidad y legibilidad. Creado por Guido van Rossum y lanzado en 1991, Python se destaca por su sintaxis clara, que favorece la legibilidad del código. Como lenguaje orientado a objetos con tipado dinámico, facilita la modularidad y reutilización del código, respaldado por un extenso conjunto de bibliotecas estándar. Estas características convierten a Python en una herramienta versátil y efectiva para diversas aplicaciones. (MARZAL Y GRACIA, 2014)

n) **Redes neuronales convolucionales (CNN):** Es un modelo multicapa inspirado en la corteza visual de los animales, eficaz para el procesamiento de imágenes. Las capas iniciales detectan características básicas como bordes, mientras que las capas profundas combinan estas características para formar representaciones complejas. Su principal ventaja es la especialización de cada sección de la red, lo que reduce la necesidad de capas ocultas y acelera el entrenamiento. Además, su capacidad para mantener la invarianza a la traslación de patrones las hace muy efectivas para detectar patrones en imágenes. (CENTENO, 2019)

o) **Roboflow**: Es una plataforma robusta y accesible que facilita el entrenamiento de modelos de visión por computadora de manera rápida y eficiente. Gracias a su interfaz intuitiva y a su habilidad para destacar aspectos clave, se presenta como una opción excelente para quienes están iniciándose en el campo de la inteligencia artificial. (ROBOFLOW, 2022)

p) **RPN:** (Region Proposal Network) es un componente de Faster R-CNN que genera regiones candidatas donde podrían estar los objetos en una imagen. Lo hace evaluando cuadros predefinidos (anchors) en un mapa de características y prediciendo si contienen un objeto o son fondo, además de ajustar sus coordenadas. Es rápida, precisa y se integra directamente

con la red base, eliminando la necesidad de métodos externos para detectar regiones. (REN y otros. 2015)

q) **TensorFlow**: Es una plataforma de código abierto para aprendizaje automático que proporciona múltiples niveles de abstracción. Su ecosistema completo y versátil de herramientas y bibliotecas facilita la implementación de aplicaciones de aprendizaje automático, simplificando el proceso mediante la API de alto nivel de Keras. (CÓRDOVA, 2021)

r) **Visión por computadora:** El objetivo es crear métodos matemáticos para interpretar la estructura tridimensional de objetos en imágenes, imitando la percepción visual humana. A pesar de los avances en reconstrucción 3D y segmentación, los sistemas aún no pueden igualar la precisión y comprensión humana. Esto se debe a la complejidad del problema, que exige modelos avanzados que integren radiometría, óptica y gráficos por computadora, enfrentando desafíos mayores que en otras áreas como la modelización del tracto vocal. (SZELISKI, 2021)

s) **YOLO:** "You Only Look Once" (Solo Miras una Vez) es un modelo de aprendizaje profundo diseñado para detectar objetos en tiempo real. Su nombre destaca su enfoque eficiente, ya que realiza todo el proceso de detección en una única pasada sobre la imagen, a diferencia de otros métodos que analizan las regiones de interés de forma separada. (REN y otros, 2017)

CAPÍTULO IV
METODOLOGÍA

4.1 Tipo y nivel de investigación

a) Tipo de investigación

En presente proyecto se basa en la "investigación aplicada", porque basado en el uso del machine learning a través del reconocimiento de imágenes se pretende resolver un problema que de alguna manera afecta la vida social y económica. (HERNÁNDEZ, FERNÁNDEZ Y BAPTISTA, 2014)

b) Nivel de investigación

El nivel de investigación que será manejado será el "nivel descriptivo" (HERNÁNDEZ, FERNÁNDEZ Y, BAPTISTA, 2014), porque la computadora fue capaz de aprender a través de cada detalle y características de las imágenes, usando las técnicas del machine learning, para que de esta manera sea capaz de reconocer cada imagen y midiendo la eficiencia de sus resultados.

4.2 Diseño de la investigación

Será un "diseño no experimental", porque en esta investigación no se manipularon deliberadamente datos o variables, basados en la observación y como se dan en su contexto natural es decir su comportamiento. (HERNÁNDEZ, FERNÁNDEZ Y BAPTISTA, 2014)

4.3 Descripción ética de la investigación

No corresponde.

4.4 Población y muestra

a) Población

Se consideró como población: total de 1011 fotografías de enfermedades de la planta tomadas en diversos puntos de la ciudad de Abancay, incluyendo el sector de Ccanabamba, anexo del distrito de Tamburco; la comunidad de Kishuar, perteneciente al distrito de Huanipaca; y la comunidad de Concacha, ubicada en el distrito de Curahuasi, provincia de Abancay, región Apurímac.

b) Muestra

La muestra se basó en las imágenes de buena resolución de plantas de papa, tanto sanas como afectadas por enfermedades: El 90% de fotografías serán usadas para el entrenamiento de aprendizaje del modelo, el 6% de fotografías para realizar el proceso de validación y el otro 4% de fotos para su realizar las pruebas.

4.5 Procedimiento

El proceso de la experimentación se realizó en el siguiente orden:

1. **Etapa I:** Aprobación de tesis.
2. **Etapa II:** Evaluación del modelo de Machine Learning.
3. **Etapa III:** Codificación de programa de reconocimiento de imágenes.
4. **Etapa IV:** Obtener las imágenes para tratar los datos, las imágenes serán conseguidas de la siguiente forma: se tomarán distintas fotos de la misma planta de la papa (hojas, tallos).
5. **Etapa V:** Entrenamiento para el aprendizaje del programa, se usará el 90% de las imágenes para su entrenamiento y el 6% para el proceso de validación.
6. **Etapa VI:** Proceso de pruebas de las imágenes, se usará el 4% de las imágenes.
7. **Etapa VII:** Análisis de los resultados obtenidos por cada uno de los modelos.
8. **Etapa VIII:** Desarrollo del Informe Final, detallado por los resultados obtenidos en la investigación.

4.6 Técnica e instrumentos

Técnica

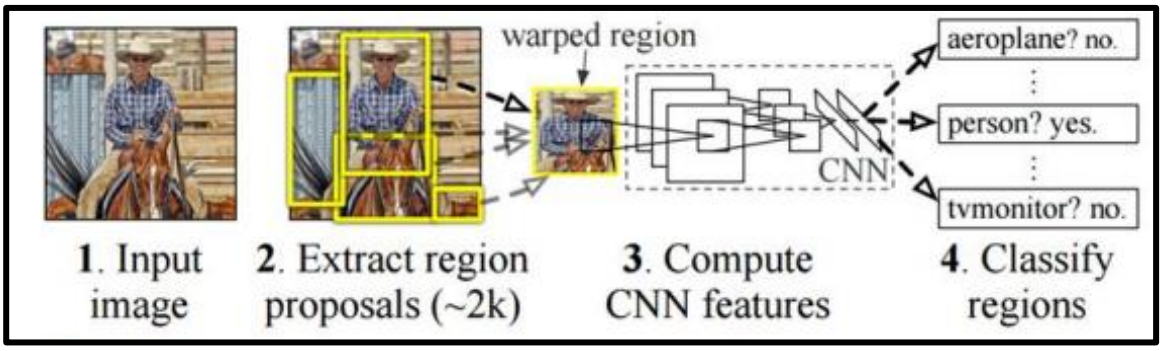
Se empleó la técnica de "observación" en dos modalidades:

- ✓ **Observación directa:** Se realizó observando y analizando directamente el comportamiento de las enfermedades en el cultivo de papa mediante fotografías, sin intermediarios.
- ✓ **Observación estructurada:** Se llevó a cabo siguiendo un procedimiento sistemático y controlado, utilizando cámaras para capturar los datos que posteriormente fueron procesados con modelos de Machine Learning para su entrenamiento, validación y pruebas. (CAMPOS Y LULE, 2012)

Instrumento

El instrumento utilizado fue la "ficha de observación", para el entrenamiento y validaciones, es el R-CNN o Red Convolucional basada en regiones (GIRSHICK y otros, 2014). Está basado en el aprendizaje supervisado y su proceso de aprendizaje se basa en trazar una serie de áreas dentro

de una imagen, que seguidamente serán analizadas y procurará reconocer cada una de ellas, para que de esa manera pueda identificar los detalles y el patrón de cada área, ver figura 14.



FUENTE: GIRSHICK y otros, 2014

Figura 14 — Ejemplo del funcionamiento Red RCNN

4.7 Estadístico de investigación

Valores a considerar:

- **FP:** False Positive (Falso Positivo)
- **FN:** False Negative (Falso Negativo)
- **TP:** True Positive (Verdadero Positivo)
- **TN:** True Negative (Verdadero Negativo)

Fórmulas para medir el desempeño del modelo:

- $\mathbf{Precision\ (Precisión)} = \frac{TP}{TP + FP}$
- $\mathbf{Recall\ (Exhaustividad)} = \frac{TP}{TP + FN}$
- $\mathbf{F - Value} = 2 * \frac{Precision * Recall}{Precision + Recall}$
-
- $\mathbf{Accuracy\ (Precisión)} = \frac{TP + TN}{TP + FP + FN + TN}$

CAPÍTULO V
RESULTADOS Y DISCUSIÓN

5.1 Análisis de resultados

A continuación, se detalla los resultados obtenidos en la presente investigación:

5.1.1 Resultado del procesamiento de datos

Para obtener los resultados, se llevó a cabo el entrenamiento de cada modelo utilizando imágenes de muestra específicas para cada una de las enfermedades. Tanto el modelo Faster R-CNN como el modelo YOLO V4 fueron entrenados con fotografías que representan la enfermedad de Tizón Tardío o Rancha, así como con imágenes correspondientes a la enfermedad de Erwinia (también conocida como Pie Negro o pudrición blanda). Como principal herramienta, se empleó Google Colab para facilitar el almacenamiento de borradores y trabajar con las fotografías almacenadas en Google Drive.

En los Anexos 01 y 02 se muestran las imágenes de las plantas de la papa afectadas por ambas enfermedades. Estas imágenes están disponibles en Google Drive, con la cuenta de usuario Nelida Alvarez Vargas y para ser más directos se encuentran en el link: https://drive.google.com/drive/folders/1nM7qtZRkcrkq6Yv_Uk1i9X4CYwHucgF-?usp=sharing , en este link, podemos encontrar 2 carpetas: TensorFlow (Faster R-CNN) y YOLO V4.

TensorFlow (Faster R-CNN):

a) Entre los archivos encontraremos unos archivos con la extensión ". ipynb":

- **model_train_rcnn.ipynb :** Se encuentra el código con el que se entrenó con las imágenes con el modelo Faster R-CNN.

  En el Anexo 06, en la Figura 44, tenemos el código fuente, o podemos acceder directamente por el link de acceso a Google Colab: https://colab.research.google.com/drive/1pfXjwZ7n21j9xct23BcXrqFLQlxVVxXv
- **model_prediction_rcnn.ipynb** : Se encuentra el código donde se realizó las predicciones con las imágenes con el modelo Faster R-CNN ya entrenado.

En el Anexo 06, en la Figura 45, tenemos el código fuente, o podemos acceder directamente por el link de acceso a Google Colab: https://colab.research.google.com/drive/1wm7OMjwQ_XWAc_rk2uJCEms6WMhyXrZD

b) También encontraremos carpetas con las fotografías:

- **images_train**: Se encuentran las fotografías con las que se realizó el entrenamiento con el modelo Faster R-CNN.
- **images_prediction_evaluated**: Se encuentran las fotografías evaluadas por el modelo Faster R-CNN.

YOLO V4:

a) Entre los archivos encontraremos unos archivos con la extensión ". ipynb":

- **model_train_yolo.ipynb** : Se encuentra el código con el que se entrenó con las fotografías con el modelo Yolo v4.

  En el Anexo 07, en la Figura 46, tenemos el código fuente, o podemos acceder directamente por el link de acceso a Google Colab: https://colab.research.google.com/drive/1FCD_7ssrGU9AWWU6FOoDHX-7dAJodE4g
- **model_prediction_yolo.ipynb** : Se encuentra el código donde se realizó las predicciones con las fotografías con el modelo Yolo v4 ya entrenado.

  En el Anexo 07, en la Figura 47, tenemos el código fuente, o podemos acceder directamente por el link de acceso a Google Colab: https://colab.research.google.com/drive/1OUFs43bDYhPg03cITyxDaTqVwIckT4DX

b) También encontraremos carpetas con las fotografías:

- **images_train**: Se encuentran las fotografías con las que se realizó el entrenamiento.
- **images_evaluated**: Se encuentran las fotografías evaluadas por el modelo Yolo v4.

5.2 Contrastación de datos y entrenamiento

5.2.1 Recopilación de datos

Para llevar a cabo el procesamiento de datos, se utilizaron 1011 fotografías tomadas en distintos puntos de Abancay, como Ccanabamba (distrito de Tamburco), Kishuar (Huanipaca) y Concacha (Curahuasi), en la región Apurímac. De este total, el 90% (909 imágenes) se destinó al

entrenamiento del modelo, abarcando tanto imágenes de plantas con enfermedades específicas como rancha y pie negro, como de plantas saludables. El 6% (58 imágenes) se reservó para validar el rendimiento del modelo, mientras que el 4% (46 imágenes) se usó para pruebas.

Esta distribución sigue las recomendaciones de Yoshua Bengio en “Practical Recommendations for Gradient-Based Training of Deep Architectures” (2012), que sugiere utilizar la mayor cantidad posible de datos para el entrenamiento como el 90%, especialmente con conjuntos limitados. Para modelos profundos como YOLOv4 y Faster R-CNN, esta estrategia optimiza el aprendizaje del modelo y mejora su capacidad de generalización. Además, Bengio recomienda usar un conjunto de validación del 6% para ajustar los hiperparámetros y evitar el sobreajuste, sin comprometer demasiado el entrenamiento. Ron Kohavi, en “A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection” (1995), destaca la importancia de un conjunto de prueba independiente para evaluar el rendimiento del modelo de manera imparcial. Aunque el 4% para pruebas es pequeño, proporciona una evaluación adecuada de la performance del modelo. Podemos visualizar el gráfico 1, con el detalle de distribución.

En el Anexo 05, en la tabla 17, se proporciona la "Ficha de Observación del Entrenamiento", la cual detalla el proceso de entrenamiento de los modelos utilizados. Esto incluye información sobre los tipos de modelos empleados, el número de fotografías utilizadas para el entrenamiento, así como detalle del número de etiquetas. En la tabla 18, encontramos la "Ficha de Observación de Prueba de Modelos", donde se presenta en detalle el proceso de pruebas de los modelos entrenados. Esta ficha proporciona información sobre el número de predicciones realizadas por los modelos y compara estos resultados con los valores reales etiquetados en el conjunto de datos de detecciones.

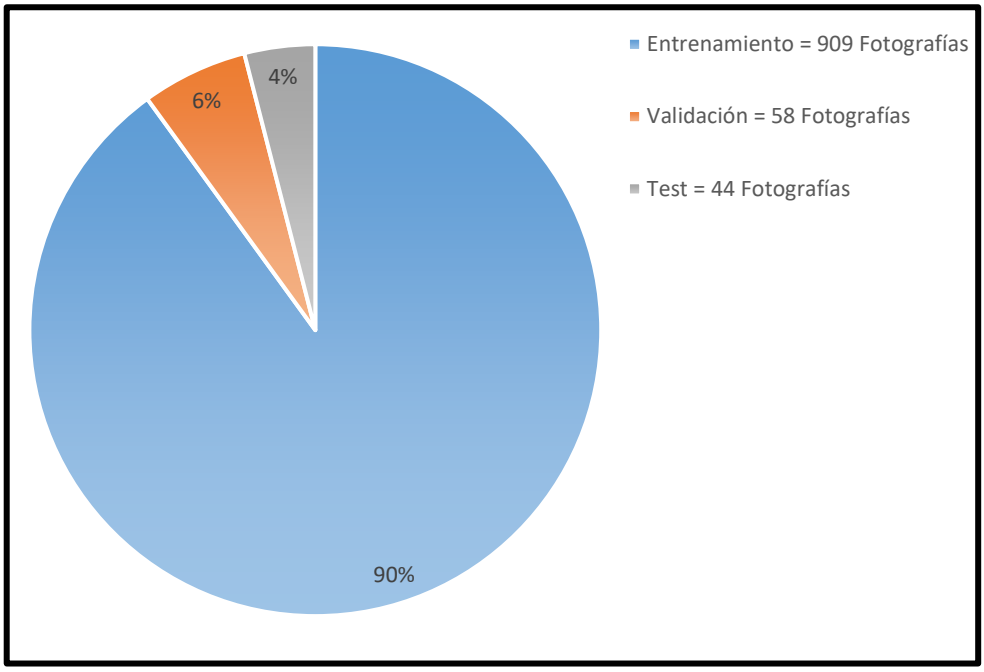


Figura 15 — Detalle de distribución de fotos

a) Tizón tardío o rancha

Para lograr obtener los resultados, se consideró un total de 657 fotografías, de las cuales se 588 fotos para su entrenamiento, en las cuales están etiquetadas la enfermedad del Tizón Tardío o Rancha como también existen fotos de hojas que están "sanas" de tal manera el modelo pueda identificar mejor la enfermedad, se tomaron 47 fotos para realizar la validación y 22 fotos para que realice las pruebas, es decir que a este punto el modelo ya puede detectar la enfermedad de manera automática. En la figura 16 se muestra un ejemplo de detección de Rancha utilizando el modelo Faster R-CNN, mientras que en la figura 17, se presenta otro ejemplo con el modelo YOLO v4.

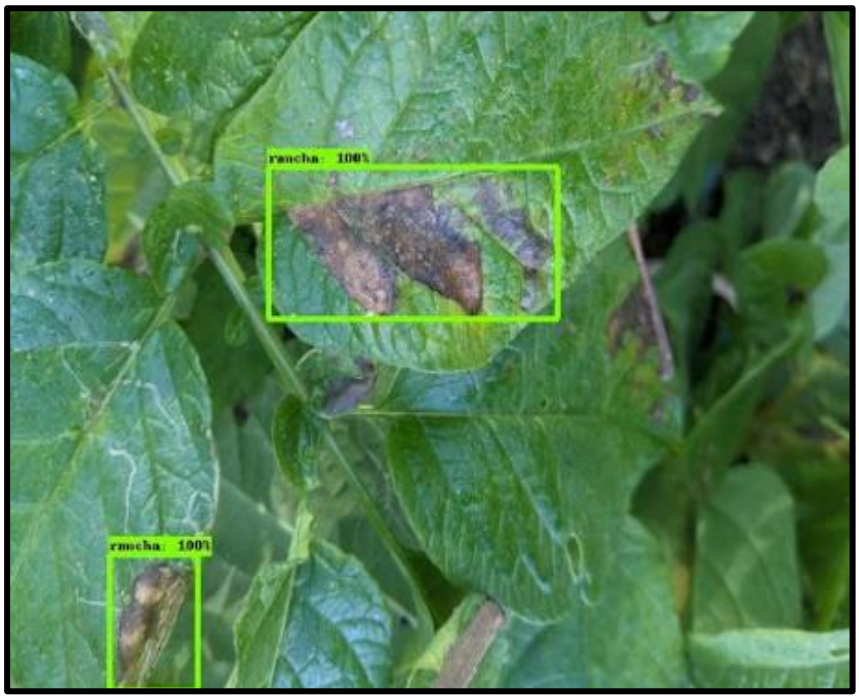


Figura 16 — Hoja de papa con Rancha evaluada por Faster R-CNN

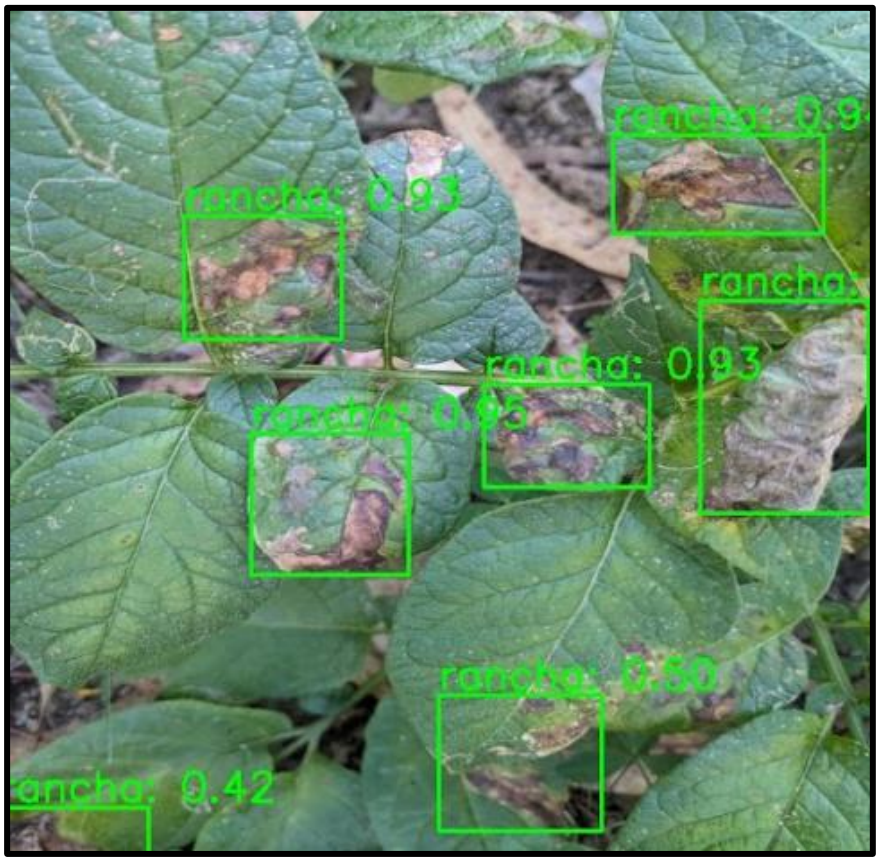


Figura 17 — Hoja de papa con Rancha evaluada por YOLO V4

b) Erwinia (Pie negro o pudrición blanda)

Para obtener los resultados, también se consideró el total de 354 fotografías, de las cuales se 321 fotos para su entrenamiento, en las cuales están etiquetadas la enfermedad de la Erwinia o Pie negro como también existen fotos de hojas que están “sanas” de tal manera el modelo pueda identificar mejor la enfermedad, se tomaron 11 fotos para realizar la validación y 22 fotos para que realice las pruebas, es decir que el modelo ya puede detectar la enfermedad de manera automática. En la figura 18 se muestra un ejemplo de detección de Pie negro utilizando el modelo Faster R-CNN, mientras que en la figura 19, se presenta otro ejemplo con el modelo YOLO v4.

Figura 18 — Tallo de papa con Pie Negro evaluada por Faster R-CNN

Figura 19 — Tallo de papa con Pie Negro evaluada por YOLO V4

5.2.2 Estadística de entrenamiento

5.2.2.1. Evolución del LOSS (Pérdidas) durante el entrenamiento del modelo Faster R-CNN

En la figura 20 nos muestra la tendencia del valor del LOSS (pérdidas) durante el entrenamiento del modelo Faster R-CNN. El objetivo del entrenamiento es minimizar este valor, ya que indica la discrepancia entre las predicciones del modelo y las etiquetas reales de los objetos en las imágenes. A medida que avanza el entrenamiento, se espera que el valor del LOSS (pérdidas) disminuya gradualmente, lo que indica que el modelo está mejorando su capacidad para detectar objetos con mayor precisión. Por lo tanto, una curva descendente en el gráfico refleja un progreso positivo en el aprendizaje del modelo.

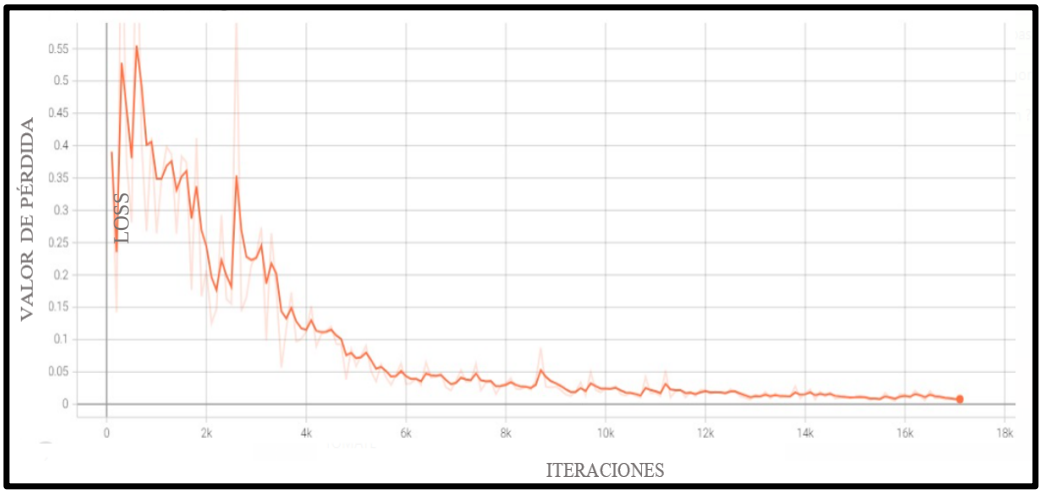


Figura 20 — Tendencia del LOSS de entrenamiento del modelo Faster R-CNN

Para más detalle de la figura 20 que proporciona una visualización de la evolución del LOSS durante el proceso de entrenamiento del modelo. A continuación, se explican los elementos clave que se destacan en el gráfico:

- **Curva del LOSS**: Muestra cómo las pérdidas disminuyen progresivamente a medida que avanza el entrenamiento. Una curva descendente indica que el modelo está aprendiendo y ajustándose mejor a los datos.
- **Iteraciones**: Reflejan el progreso del entrenamiento en términos de pasos o ciclos de actualización de los parámetros del modelo.
- **Eje Y (LOSS):** Representa el valor del *total_loss* en cada iteración. Un valor más bajo sugiere un mejor rendimiento del modelo, ya que significa que el modelo está haciendo predicciones más precisas.
- **Eje X (Iteraciones):** Muestra el número acumulado de pasos realizados durante el entrenamiento. Un mayor número de iteraciones generalmente proporciona más oportunidades para que el modelo aprenda y mejore.

5.2.2.2. Evolución del LOSS (Pérdidas) durante el entrenamiento del modelo YOLO V4

En la figura 21 muestra cómo varía el valor del LOSS durante el proceso de entrenamiento del modelo YOLO V4. El LOSS es una medida de la discrepancia entre las predicciones del modelo y las etiquetas reales de los objetos en las imágenes. Durante el entrenamiento, el objetivo es minimizar este valor para mejorar la precisión del modelo en la detección de objetos. Por lo tanto, una tendencia descendente en el gráfico indica que el modelo está aprendiendo y ajustándose a los datos de entrenamiento de manera efectiva. Este descenso en el LOSS refleja una mejora en la capacidad del modelo para identificar y localizar objetos con mayor precisión a lo largo del tiempo de entrenamiento.

En este caso también nos muestra el progreso del mAP (Mean Average Precision) es el promedio de las precisiones promedio para cada clase en un conjunto de datos. En este gráfico, el descenso del LOSS indica que el modelo está ajustándose y aprendiendo de manera efectiva durante el entrenamiento. Por otro lado, el seguimiento del mAP nos proporciona una medida más holística del rendimiento del modelo a medida que varía el umbral de decisión para la detección de objetos. En términos simples, un mAP más alto indica un mejor rendimiento del modelo, lo que significa que el modelo es más preciso y confiable en la detección de objetos de diferentes clases. Por lo tanto, la combinación de la disminución del LOSS y el aumento del mAP a lo largo del tiempo de entrenamiento

indica que el modelo YOLO está mejorando su capacidad de detección de objetos y está alcanzando un rendimiento más óptimo con el tiempo.

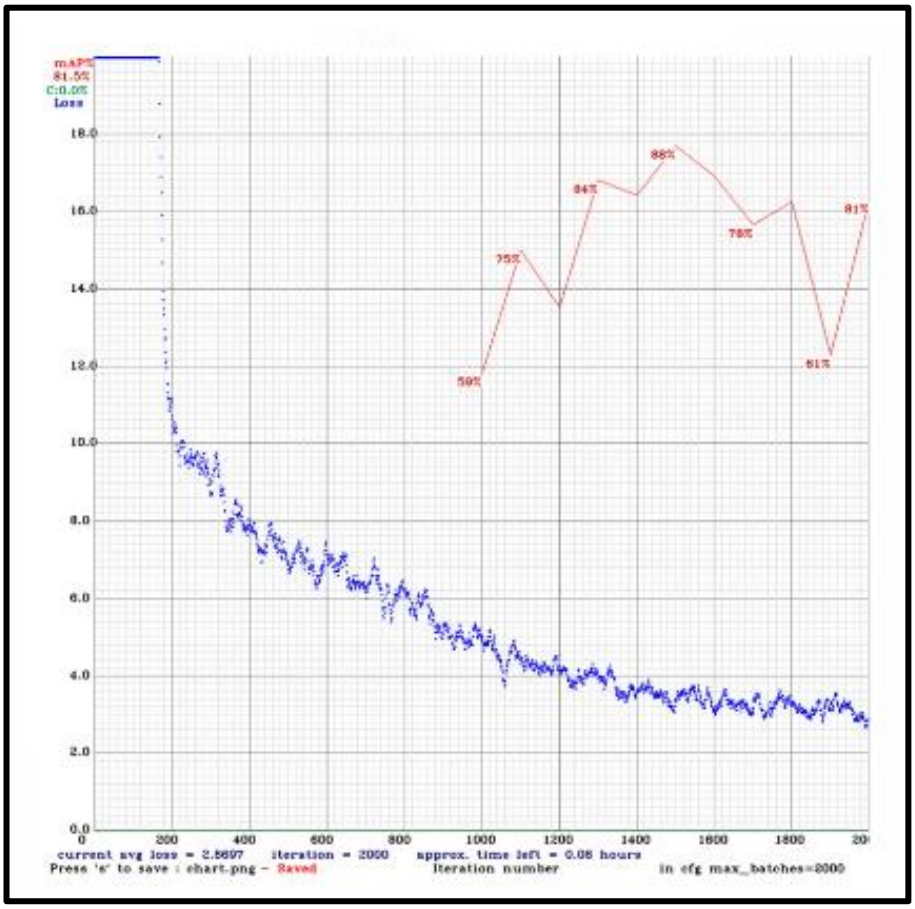


Figura 21 — Tendencia del LOSS y mAP de entrenamiento del modelo YOLO V4

La figura 21 muestra la evolución del LOSS y del mAP a lo largo del proceso de entrenamiento del modelo. A continuación, se detallan los elementos clave:

- **Curva azul (LOSS)**: Representa cómo se reduce el valor de la pérdida durante el entrenamiento. Una tendencia descendente en esta curva indica que el modelo está mejorando su capacidad para ajustar los datos y minimizar los errores.
- **Curva roja (mAP)**: Refleja la evolución del Mean Average Precision (mAP), una métrica clave que mide la precisión general del modelo. Un aumento en esta curva sugiere que el modelo está logrando un mejor rendimiento en términos de detección y predicción precisas.
- **Iteraciones**: Corresponden al número de veces que se actualizan los pesos del modelo a lo largo del proceso de entrenamiento.
- **Eje Y (LOSS y mAP):** Muestra los valores del LOSS y del mAP obtenidos en cada iteración. La coexistencia de ambas curvas permite observar cómo la reducción de la pérdida se relaciona con la mejora en la precisión.

- **Eje X (Iteraciones)**: Indica la cantidad de iteraciones realizadas durante el entrenamiento, permitiendo seguir el progreso temporal del modelo.

5.2.3 Contrastación de hipótesis

5.2.3.1 Contrastación de la hipótesis 1:

El uso de la técnica de Faster R-CNN es menos eficiente (acurracy) en la detección de enfermedades en el cultivo de la papa, en Abancay, Apurímac, 2022.

Tizón tardío o rancha:

a) Especificación de datos

La verificación de datos se realiza utilizando el 4% de las fotografías que ya han sido destinadas para la prueba, lo que equivale a un total de 44 imágenes. En la siguiente tabla 3, en la primera columna titulada "N°", cada foto se identificará con un número de referencia, por ejemplo, F01, F02, etc. En la segunda columna, titulada "Cantidad Real", se proporcionará para cada imagen la cantidad real de la enfermedad etiquetada como "*Rancha*", contabilizada manualmente y en la tercera columna, titulada "Cantidad Detecciones", se indicará el número de predicciones detectadas por el modelo Faster R-CNN.

Tabla 3 — Detalle de predicciones de Rancha con Faster R-CNN

FASTER R-CNN "Rancha"		
N°	**Cantidad Real**	**Cantidad Detecciones**
F01	0	0
F02	0	0
F03	2	1
F04	5	5
F05	0	0
F06	0	0
F07	14	11
F08	0	0
F09	0	0
F10	4	1
F11	2	2
F12	1	0
F13	0	0
F14	0	0
F15	2	1
F16	6	6
F17	0	0

F18	12	8
F19	0	0
F20	2	2
F21	0	0
F22	0	0
F23	0	0
F24	0	0
F25	0	0
F26	9	10
F27	3	3
F28	2	2
F29	0	0
F30	2	2
F31	0	0
F32	6	5
F33	0	0
F34	0	0
F35	22	18
F36	0	0
F37	2	2
F38	4	1
F39	5	3
F40	1	1
F41	0	0
F42	2	3
F43	0	0
F44	2	2
TOTAL	110	89

b) Estadístico

Matriz de confusión

Valores a considerar:

- **FP:** False Positive (Falso Positivo)
- **FN:** False Negative (Falso Negativo)
- **TP:** True Positive (Verdadero Positivo)
- **TN:** True Negative (Verdadero Negativo)

Tabla 4 — Matriz de confusión para rancha con faster R-CNN

Matriz de confusión para Rancha	
FP	2
FN	23
TP	87
TN	0

Esclarecimiento de Valores:

En cuanto al conteo de valores por imagen, es crucial destacar que se realiza una suma total de 112 debido a la presencia de 2 Falsos Positivos. Esto implica que el modelo identificó erróneamente 2 enfermedades que en realidad no estaban presentes. Por consiguiente, es necesario tener en cuenta la siguiente fórmula:

$$\boldsymbol{TOTAL = (TP + FP + FN + TN) - FP}$$

Estadística de Predicciones

Según las fórmulas:

- $\text{Precision (Precisión)} = \frac{TP}{TP + FP}$
- $\text{Recall (Exhaustividad)} = \frac{TP}{TP + FN}$
- $\text{Accuracy (Precisión)} = \frac{TP + TN}{TP + FP + FN + TN}$
- $\text{F} - \text{Value} = 2 * \frac{Precision * Recall}{Precision + Recall}$

Tabla 5 — Cuadro estadístico para rancha con faster R-CNN

Estadística para "Rancha"			
Precision	**Recall**	**F-value**	**Acurracy**
0,98	0,79	0,87	0,78

c) Análisis de estadística:

Analizando los valores proporcionados para la precision, recall, accuracy y el F-value ofrece una perspectiva integral sobre el rendimiento del modelo:

- ✓ **Precision: 0,98**

La alta precisión indica que el modelo tiende a hacer pocas predicciones positivas incorrectas en comparación con el total de predicciones positivas. Es especialmente útil cuando se busca minimizar los falsos positivos.

- ✓ **Recall: 0,79**

Aunque la exhaustividad es menor que la precisión, sigue siendo aceptable. Indica que el modelo captura una proporción considerable de todas las instancias positivas. Sin embargo, podría haber ocasiones en las que no logre identificar algunos casos positivos.

- ✓ **Accuracy: 0,78**

La exactitud mide la proporción total de predicciones correctas. En este caso, la exactitud es más baja que la precisión, sugiriendo que el modelo podría tener un número significativo de falsos negativos, lo que afecta su capacidad para clasificar correctamente todas las instancias.

- ✓ **F-value: 0,87**

El F-value combina la precisión y la exhaustividad en una métrica única. Un valor de 0,87 sugiere un buen equilibrio entre ambas métricas. Es especialmente útil cuando se busca una métrica que considere tanto los falsos positivos como los falsos negativos.

Análisis General:

El acurracy 78% indica que el modelo tiene un rendimiento razonable en términos generales, pero no es excelente. Dado que la precisión es alta, el modelo es confiable cuando predice que la enfermedad está presente. No obstante, el menor valor de Recall y el F1-score sugieren que el modelo podría beneficiarse de ajustes para mejorar la detección de más casos reales de Rancha, lo cual es crítico en contextos médicos o agrícolas donde es vital detectar la mayoría de los casos.

Erwinia (Pie negro o pudrición blanda):

a) Especificación de datos

La verificación de datos se lleva a cabo utilizando el 4% de las fotografías que ya han sido destinadas para la prueba, lo que se traduce en un total de 44

imágenes. En la siguiente tabla 6, en la primera columna titulada "N°", cada foto se identificará con un número de referencia, por ejemplo, F01, F02, etc. En la segunda columna, titulada "Cantidad Real", se proporcionará para cada imagen la cantidad real de la enfermedad etiquetada como "*Pie negro*", contabilizada manualmente. En la tercera columna, titulada "Cantidad Detecciones", se indicará el número de predicciones detectadas por el modelo Faster R-CNN.

Tabla 6 — Detalle de predicciones de pie negro con faster R-CNN

FASTER R-CNN "Pie Negro"		
N°	**Cantidad Real**	**Cantidad Detecciones**
F01	1	1
F02	1	1
F03	0	0
F04	0	0
F05	1	1
F06	1	1
F07	0	0
F08	0	0
F09	2	2
F10	0	0
F11	0	0
F12	0	0
F13	1	1
F14	1	1
F15	0	0
F16	0	0
F17	2	2
F18	0	0
F19	1	1
F20	0	0
F21	0	0
F22	1	1
F23	1	1
F24	1	1
F25	1	0
F26	0	0
F27	0	0
F28	0	0
F29	1	1

F30	0	0
F31	0	0
F32	0	0
F33	2	2
F34	0	0
F35	0	0
F36	2	2
F37	0	0
F38	0	0
F39	0	0
F40	0	0
F41	2	2
F42	0	0
F43	2	2
F44	0	0
TOTAL	24	23

b) Estadístico

Matriz de confusión

Valores a considerar:

- **FP:** False Positive (Falso Positivo)
- **FN:** False Negative (Falso Negativo)
- **TP:** True Positive (Verdadero Positivo)
- **TN:** True Negative (Verdadero Negativo)

Tabla 7 — Matriz de confusión para pie negro con faster R-CNN

Matriz de confusión para Pie Negro	
FP	0
FN	1
TP	23
TN	0

Esclarecimiento de valores:

En relación con los valores totales para cada cantidad de etiquetas reales y predicciones, es importante destacar que el modelo se entrenó de manera efectiva, logrando identificar un número mayor de unidades de la enfermedad de las que fueron etiquetadas inicialmente. Estas predicciones adicionales se

han comprobado como correctas y se clasificaron como "Verdaderos Positivos".

Estadística de predicciones

Según las fórmulas:

- Precision (Precisión) $= \frac{TP}{TP + FP}$
- Recall (Exhaustividad) $= \frac{TP}{TP + FN}$
- Accuracy (Precisión) $= \frac{TP + TN}{TP + FP + FN + TN}$
- $\mathrm{F} - \mathrm{Value} = 2 * \frac{Precision * Recall}{Precision + Recall}$

Tabla 8 — Cuadro estadístico para pie negro con faster R-CNN

Estadística para "Pie negro"			
Precision	**Recall**	**F-value**	**Acurracy**
1,00	0,96	0,98	0,96

c) Análisis de estadística:

Analizando los valores proporcionados para la precision, recall, accuracy y el F-value los valores sugieren un rendimiento excepcionalmente bueno del modelo. Detallamos cada métrica:

- ✓ **Precision: 1,0**

La precisión perfecta nos indica que todas las instancias clasificadas como positivas son realmente positivas. No hay falsos positivos. Esto es altamente deseable en muchas aplicaciones, especialmente cuando se busca evitar errores de clasificación positivos.

- ✓ **Recall: 0,96**

La alta exhaustividad nos indica que el modelo está capturando la gran mayoría de las instancias positivas. Solo se están perdiendo un pequeño porcentaje de casos positivos, lo que es excelente para garantizar que la mayoría de las instancias relevantes están siendo detectadas.

- ✓ **Accuracy: 0,96**

La alta exactitud sugiere que el modelo clasifica correctamente la gran mayoría de las instancias, independientemente de su clase. Es una métrica global que

refleja un buen rendimiento general del modelo.

- ✓ **F-value: 0,98**

El F-value, que combina precisión y exhaustividad, también muestra un rendimiento excepcionalmente alto. Un valor de 0,98 sugiere un equilibrio sobresaliente entre la capacidad del modelo para evitar falsos positivos y falsos negativos.

Análisis general:

El acurracy 96% del modelo en la detección de Pie negro es muy elevada, lo que indica un rendimiento sobresaliente en términos de predicciones correctas. Con una precisión perfecta y una cobertura muy alta, el modelo es altamente confiable para identificar la enfermedad correctamente y minimizar errores. El alto f-value refuerza su balance entre precisión y recall, siendo muy eficiente para aplicaciones que requieren una detección precisa de Pie negro.

5.2.3.2 Contrastación de la hipótesis 2:

El uso de la técnica de YOLO V4 es más eficiente (acurracy) en la detección de enfermedades en el cultivo de la papa, en Abancay, Apurímac, 2022.

Tizón tardío o rancha:

a) Especificación de datos

La verificación de datos se efectúa utilizando el 4% de las fotografías previamente destinadas para la prueba, totalizando 44 imágenes. En la tabla 9, en la primera columna titulada "N°", cada foto se identificará con un número de referencia, por ejemplo, F01, F02, etc. En la segunda columna, titulada "Cantidad Real", se proporcionará para cada imagen la cantidad real de la enfermedad etiquetada como "*Rancha*", contabilizada manualmente. En la tercera columna, titulada "Cantidad Detecciones", se indicará el número de predicciones detectadas por el modelo YOLO V4.

Tabla 9 — Detalle de predicciones de rancha con YOLO V4

YOLO V4 "Rancha"		
N°	**Cantidad Real**	**Cantidad Detecciones**
F01	0	0
F02	0	0
F03	2	0
F04	5	5
F05	0	0
F06	0	0
F07	14	19
F08	0	0
F09	0	0
F10	4	0
F11	2	5
F12	1	0
F13	0	0
F14	0	0
F15	2	2
F16	6	6
F17	0	0
F18	12	12
F19	0	0
F20	2	3
F21	0	0
F22	0	0
F23	0	0
F24	0	0
F25	0	0
F26	9	9
F27	3	6
F28	2	2
F29	0	0
F30	2	2
F31	0	0
F32	6	7
F33	0	0
F34	0	0
F35	22	28
F36	0	0
F37	2	2
F38	4	1
F39	5	2
F40	1	1
F41	0	0

F42	2	3
F43	0	0
F44	2	2
TOTAL	110	117

b) Estadístico

Matriz de confusión

Valores a considerar:

- **FP:** False Positive (Falso Positivo)
- **FN:** False Negative (Falso Negativo)
- **TP:** True Positive (Verdadero Positivo)
- **TN:** True Negative (Verdadero Negativo)

Tabla 10 — Matriz de confusión para rancha con YOLO V4

Matriz de confusión para Rancha	
FP	4
FN	13
TP	113
TN	0

Esclarecimiento de valores:

En cuanto a los valores totales relacionados con la cantidad de etiquetas reales y predicciones, es fundamental señalar que la cantidad de predicciones es superior a la cantidad de etiquetas, ya que el modelo desglosó una unidad en varias partes. Por ejemplo, si una fotografía muestra una hoja con un gran tamaño de la enfermedad que inicialmente fue etiquetada con 1 o 2 etiquetas, el modelo la identificó en 5 partes, todas las cuales son acertadas. En consecuencia, estas predicciones adicionales se consideran como "Verdaderos Positivos".

c) Estadística de predicciones

Según las fórmulas:

- Precision (Precisión) $= \frac{TP}{TP + FP}$

- Recall (Exhaustividad) $= \frac{TP}{TP + FN}$

- Accuracy (Precisión) $= \frac{TP + TN}{TP + FP + FN + TN}$
- $\mathrm{F} - \mathrm{Value} = 2 * \frac{Precision * Recall}{Precision + Recall}$

Tabla 11 — Cuadro estadístico para rancha con YOLO V4

Estadística para "Rancha"			
Precision	**Recall**	**F-value**	**Acurracy**
0,97	0,90	0,93	0,87

d) Análisis de estadística:

Analizando los valores proporcionados para la precision, recall, accuracy y el F-value, detallamos cada métrica:

✓ **Precision: 0,97**

La precisión de 0,97 indica que el modelo tiene una alta capacidad para predecir positivos y evitar falsos positivos. En otras palabras, cuando predice que algo es positivo, tiene una alta probabilidad de estar en lo correcto.

✓ **Recall: 0,90**

Con una exhaustividad del 0,90, el modelo captura una proporción significativa de todas las instancias positivas, pero deja escapar alrededor del 10%. Esto podría ser aceptable dependiendo del contexto, pero es importante considerar que la pérdida de algunos positivos es crítica para la aplicación.

✓ **Accuracy: 0,87**

La exactitud del 0,87 indica que el modelo clasifica correctamente el 87% de todas las instancias. La exactitud es una métrica global y, en este caso, refleja un rendimiento bastante sólido en la clasificación general.

✓ **F-value: 0,93**

Un F-value de 0,93 nos indica un buen equilibrio entre precisión y exhaustividad. Esto sugiere que el modelo tiene un rendimiento balanceado y es capaz de minimizar tanto los falsos positivos como los falsos negativos.

Análisis General:

El acurracy 87% indica que el modelo tiene un buen rendimiento en términos generales. La alta precisión y el buen valor de recall muestran que el modelo es confiable tanto para identificar correctamente la enfermedad de Rancha como para minimizar errores de predicción. Aunque la exactitud podría mejorarse, el modelo

es adecuado para aplicaciones que requieren una detección razonablemente precisa de la enfermedad, y el equilibrio entre precisión y recall es sólido.

Ewinia (Pie negro o pudrición blanda):

a) Especificación de datos

Para la prueba de los datos se utilizó el 4% de las fotografías, es decir, un total de 44. En la tabla 12, en la primera columna titulada "N°", cada foto se identificará con un número de referencia, por ejemplo, F01, F02, etc. En la segunda columna, titulada "Cantidad Real", se proporcionará para cada imagen la cantidad real de la enfermedad etiquetada como "*Pie negro*", contabilizada manualmente. En la tercera columna, titulada "Cantidad Detecciones", se indicará el número de predicciones detectadas por el modelo YOLO V4.

Tabla 12 — Detalle de predicciones de pie negro con YOLO

YOLO V4 "Pie negro"		
N°	**Cantidad Real**	**Cantidad Detecciones**
F01	1	1
F02	1	1
F03	0	0
F04	0	0
F05	1	1
F06	1	1
F07	0	0
F08	0	0
F09	1	2
F10	0	0
F11	0	0
F12	0	0
F13	1	1
F14	1	1
F15	0	0
F16	0	0
F17	2	2
F18	0	0
F19	1	2
F20	0	0
F21	0	0
F22	1	1
F23	1	1
F24	1	1
F25	1	1

F26	0	0
F27	0	0
F28	0	0
F29	1	2
F30	0	0
F31	0	0
F32	0	0
F33	2	2
F34	0	0
F35	0	0
F36	2	2
F37	0	0
F38	0	0
F39	0	0
F40	0	0
F41	2	2
F42	0	0
F43	1	2
F44	0	0
TOTAL	22	26

b) Estadístico

Matriz de confusión

Valores a considerar:

- **FP:** False Positive (Falso Positivo)
- **FN:** False Negative (Falso Negativo)
- **TP:** True Positive (Verdadero Positivo)
- **TN:** True Negative (Verdadero Negativo)

Tabla 13 — Matriz de confusión para pie negro con YOLO V4

Matriz de confusión para Pie negro	
FP	0
FN	0
TP	26
TN	0

Esclarecimiento de valores:

En relación con los totales de cada cantidad de etiquetas reales y predicciones, es destacable resaltar que el modelo fue entrenado con eficacia, logrando identificar un mayor número de instancias de la enfermedad que las

originalmente etiquetadas. Estas predicciones adicionales han sido confirmadas como precisas y han sido clasificadas como "Verdaderos Positivos".

Estadística de Predicciones

Según las fórmulas:

- Precision (Precisión) $= \frac{TP}{TP + FP}$
- Recall (Exhaustividad) $= \frac{TP}{TP + FN}$
- Accuracy (Precisión) $= \frac{TP + TN}{TP + FP + FN + TN}$
- F − Value $= 2 * \frac{Precision * Recall}{Precision + Recall}$

Tabla 14 — Cuadro estadístico para pie negro con YOLO V4

Estadística para "Pie Negro"			
Precision	**Recall**	**F-value**	**Acurracy**
1,00	1,00	1,00	1,00

c) Análisis de estadística:

Analizando los valores proporcionados para la precision, recall, accuracy y el F-value, indican un rendimiento perfecto del modelo en todas las métricas evaluadas. Detalle por cada métrica:

✓ **Precision: 1,0**

Una precisión perfecta de 1,0 significa que todas las instancias clasificadas como positivas son realmente positivas. No hay falsos positivos. Esto es ideal y sugiere que el modelo no cometió errores de clasificación de positivos.

✓ **Recall: 1,0**

Una exhaustividad perfecta de 1,0 indica que el modelo captura todas las instancias positivas. No hay falsos negativos. Esto es altamente deseable y sugiere que el modelo no omite ninguna instancia positiva.

✓ **Accuracy: 1,0**

La exactitud perfecta de 1,0 sugiere que el modelo clasifica correctamente todas las instancias, tanto positivas como negativas. No hay errores de clasificación en el conjunto de datos evaluado.

- ✓ **F-value: 1,0**

Un F-value de 1,0 indica un equilibrio perfecto entre precisión y exhaustividad. Esto significa que el modelo logra identificar todas las instancias positivas sin cometer errores de clasificación positivos.

Análisis general:

El acurracy 100% el modelo tiene un rendimiento ideal en la detección de Pie Negro que indica predicciones completamente precisas. La combinación de precisión, recall y f-value perfectos sugiere que el modelo es infalible en la detección de esta enfermedad, lo que lo hace extremadamente confiable para aplicaciones donde se requiere una precisión absoluta.

5.2.3.3 Contrastación de hipótesis general:

Al usar correctamente las técnicas: Faster R-CNN y YOLO V4, entonces se determina que YOLO V4 es más eficiente (accuracy) que Faster R-CNN para la detección de enfermedades en el cultivo de papa en Abancay, Apurímac, 2022.

a) **Especificación de datos**

Con base en los objetivos específicos, se puede proceder a comparar el rendimiento de los modelos Faster R-CNN y YOLO V4 en la detección y clasificación de enfermedades como Rancha y Pie negro. Este análisis permitirá evaluar la eficiencia de cada modelo en función de métricas clave, con especial énfasis en el Accuracy (Exactitud), para determinar cuál de ellos se desempeña mejor en la identificación precisa de estas enfermedades.

En la Tabla 15, se presenta la comparación del rendimiento de los modelos para la detección de la enfermedad Rancha, evaluando diversas métricas clave, donde se destaca el Accuracy (Exactitud) como la métrica principal para medir la eficacia general de ambos modelos en la tarea específica de detección y clasificación.

b) Estadístico tizón tardío o rancha:

Tabla 15 — Estadística de comparación de modelos para rancha

Modelo Machine Learning	Precision	Recall	F-value	Acurracy
Faster R-CNN	0,98	0,79	0,87	0,78
YOLO V4	0,97	0,90	0,93	0,87

Para una visualización más clara de la comparación de modelos para Rancha, podemos analizar el gráfico 4.

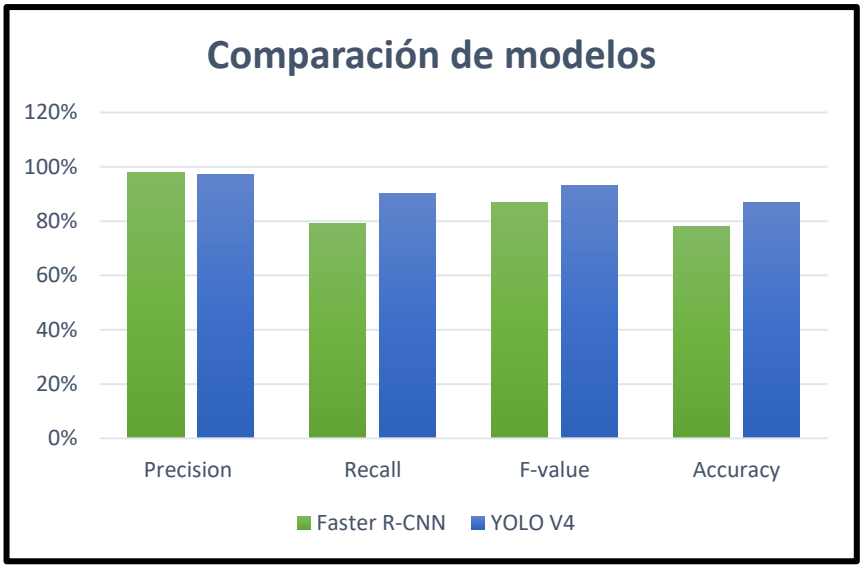


Figura 22 — Comparación de modelos para rancha

Análisis de estadística comparativa

Para la detección de la enfermedad de la Rancha, YOLO V4 supera a Faster R-CNN en exactitud, con una diferencia de 9 puntos porcentuales. Esto indica que, en términos generales, YOLO V4 ofrece un mejor rendimiento en la tarea de detección y clasificación. Su mayor exactitud sugiere una mayor eficiencia en la identificación de la enfermedad, respaldada por sus métricas superiores en precisión, recall y F1-score. Por lo tanto, YOLO V4 se presenta como una opción más robusta y confiable para aplicaciones que exigen alta precisión en la detección.

c) Estadístico erwinia (Pie negro o pudrición blanda)

En la Tabla 16, se presenta la comparación del rendimiento de los modelos para la detección de la enfermedad Pie negro, evaluando diversas métricas clave, donde se destaca el Accuracy (Exactitud) como la métrica principal para medir la eficiencia general de ambos modelos en la tarea específica de detección y clasificación.

Tabla 16 — Estadística de comparación de modelos para pie negro

Modelo Machine Learning	Precision	Recall	F-value	Acurracy
Faster R-CNN	1,00	0,96	0,96	0,98
YOLO V4	1,00	1,00	1,00	1,00

Para una visualización más clara de la comparación de modelos para Pie Negro, podemos analizar el gráfico 5.

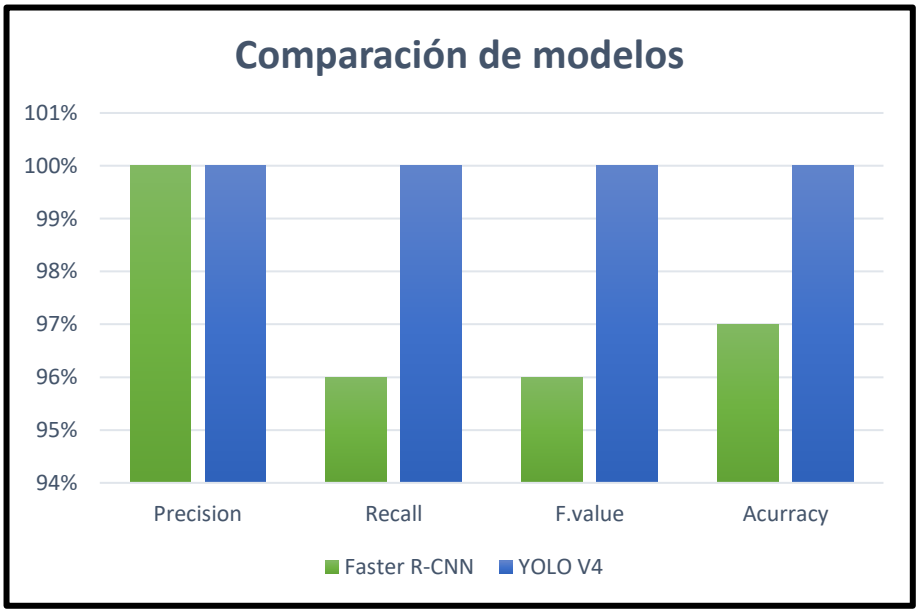


Figura 23 — Comparación de modelos para pie negro

Análisis de estadística comparativa

Para la detección de la enfermedad del Pie negro, YOLO V4 supera a Faster R-CNN en exactitud por 2 puntos porcentuales, lo que indica un rendimiento superior en términos generales de detección y clasificación. Con una exactitud del 100%, YOLO V4 no solo ofrece una precisión perfecta en todas las predicciones, sino que también destaca por su rendimiento impecable en

precisión, recall y F1-score. Este nivel de exactitud refleja una eficacia total en la tarea de detección y clasificación, haciendo de YOLO V4 la opción más robusta y confiable para aplicaciones que requieren una detección sin errores.

d) Análisis general

En conclusión, para la detección de enfermedades en el cultivo de la papa en Abancay, Apurímac, en 2022, YOLO V4 demuestra un rendimiento superior en comparación con Faster R-CNN en ambos contextos de detección de enfermedades, destacándose por su mayor exactitud, precisión perfecta, y equilibrio en recall y F-value. Esto lo convierte en una herramienta más eficiente y confiable para la detección de enfermedades en cultivos, ofreciendo una capacidad superior para identificar y clasificar enfermedades con alta precisión y sin errores.

5.3 Discusión de resultados

A partir de los resultados obtenidos en relación con el objetivo general, que busca "Determinar cuál de las técnicas de Machine Learning es más eficiente en la detección de enfermedades en el cultivo de la papa en Abancay, Apurímac, en 2022", se concluye que YOLO V4 es considerado el modelo más eficiente con una Precision de 97% en detección de la Rancha y con 100% en todos sus indicadores en detección del Pie Negro. Sin embargo, es relevante destacar que Faster R-CNN también demostró ser altamente competente en esta tarea con 98% de detección de Rancha y 100%, con mínima variación en los demás indicadores en detección del Pie Negro. A pesar de esto, Faster R-CNN no logró superar a YOLO V4 en términos generales.

Esta conclusión se alinea con investigaciones previas, se encuentra respaldo para la eficiencia de YOLO V4 en la investigación llevada a cabo por AMORES ROMERO Kevin Steven y TRELLES MUÑOZ Kaiser Geovanny (2022), Estos autores se enfocaron en el uso de la red neuronal YOLOv4, alcanzando una precisión superior al 85% y estableciendo la incidencia de la enfermedad en cada lote de plantaciones de banano.

En resumen, los resultados obtenidos, respaldados por investigaciones previas, respaldan la eficiencia tanto de YOLO V4 como de Faster R-CNN en la detección de enfermedades en el cultivo de la papa, proporcionando una base sólida para futuras aplicaciones y desarrollos en este ámbito.

Asimismo, la investigación realizada por CÓRDOVA PÉREZ Claudia Sofía (2021), la autora propone el uso de modelos de detección preentrenados, específicamente Faster R-CNN y YOLOv4. Los resultados obtenidos indican que el modelo Faster R-CNN alcanzó una precisión

del 94,06%, mientras que el modelo YOLOv4 logró un 95,82%. Se observó que YOLOv4 superó a Faster R-CNN, aunque por un margen mínimo.

Es importante resaltar que, al igual que con cualquier modelo, estas métricas pueden variar en función del contexto o del tipo de tarea.

CAPÍTULO VI
CONCLUSIONES Y RECOMENDACIONES

6.1. Conclusiones

En este trabajo de investigación se concluye lo siguiente:

- Faster R-CNN demostró buenos resultados en la detección de la enfermedad Pie Negro, alcanzando una Precisión (Precisión) del 100%, una Exhaustividad (Recall) del 96%, un Valor-F (F-value) del 96% y una Exactitud (Accuracy) del 98%. En cuanto a la detección de la enfermedad Rancha, aunque los valores fueron ligeramente más bajos, se observó una Precisión (Precisión) del 98%, una Exhaustividad (Recall) del 79%, un Valor-F (F-value) del 87% y una Exactitud (Accuracy) del 78%.
- Por otro lado, YOLO V4 destacó por sus excelentes resultados en la detección de la enfermedad Pie Negro, logrando una Precisión (Precisión) del 100%, una Exhaustividad (Recall) del 100%, un Valor-F (F-value) del 100% y una Exactitud (Accuracy) del 100%. En cuanto a la detección de la enfermedad Rancha, los valores experimentaron una variación mínima, obteniendo una Precisión (Precisión) del 97%, una Exhaustividad (Recall) del 90%, un Valor-F (F-value) del 93% y una Exactitud (Accuracy) del 87%.

En términos generales, se puede concluir que YOLO V4 supera a Faster R-CNN en términos de exactitud (Accuracy) en ambas tareas de detección. Mientras que Faster R-CNN ofrece un rendimiento notable, especialmente en la detección de Pie Negro, YOLO V4 demuestra una precisión y eficacia superiores, con una exactitud (Accuracy) perfecta en la detección de ambas enfermedades. Esto sugiere que YOLO V4 es el modelo más robusto y confiable para aplicaciones que requieren una alta precisión en la detección y clasificación de enfermedades en el cultivo de papa.

6.2. Recomendaciones

Con respecto a resultado obtenido sobre el alto rendimiento de YOLO V4 sobre Faster R-CNN. Se debería considerar la variación en las métricas que pueden estar influenciadas por el contexto de implementación. Se debería considerar factores como las condiciones climáticas, la variabilidad en las muestras de datos, y otras variables específicas del entorno agrícola.

Dado que YOLO V4 ha demostrado una exactitud perfecta (100%) en la detección de Pie Negro y una exactitud del 87% en la detección de Rancha, se recomienda su adopción para aplicaciones que requieren una detección extremadamente precisa y fiable. Su capacidad para ofrecer resultados perfectos en la detección de Pie Negro lo convierte en la opción ideal para situaciones donde la precisión es crucial.

Faster R-CNN ha mostrado un desempeño sobresaliente en la detección de Pie Negro, con una exactitud del 98%. Por lo tanto, puede ser una opción viable para aplicaciones donde esta enfermedad es la principal preocupación. Aunque su rendimiento en la detección de Rancha es ligeramente inferior, sigue siendo bastante competitivo.

También se recomienda mantener una evaluación continua de ambos modelos y actualizar los datos de entrenamiento conforme se obtengan nuevas muestras puede mejorar aún más la precisión y la eficacia. Realizar pruebas periódicas con datos actualizados ayudará a mantener la relevancia y la eficacia de los modelos en escenarios cambiantes.

Se sugiere la posibilidad de extender la investigación a otros cultivos locales, explorando la versatilidad y aplicabilidad de los modelos para abordar enfermedades en diferentes contextos agrícolas de la región.

REFERENCIAS BIBLIOGRÁFICAS

ADAMA. 2021. Plagas y enfermedades en el cultivo de papa.

AFZAAL Hassanm, FAROOQUE Aitazaz A., SCHUMANN Arnold W. HUSSAIN Nazar, MCKENZIE-GOPSILL Andrew, ESAU Travis, ABBAS Farhat. 2021. Detection of a potato disease (Early blight) using artificial intelligence Remote Sensing. MDPI AG, Vol. 13, núm. 3, p. 1-17. ISSN 20724292. DOI 10.3390/rs13030411.

ALPAYDIN, Ethem. 2014. Introduction to Machine Learning. Massachuse. Estados Unidos de América: ISBN 9780262028189.

AMORES Kevin. y TRELLES Kaiser. 2022. "Implementación de un sistema para detectar la enfermedad de la Sigatoka Negra en una plantación de banano empleando técnicas de visión artificial." Escuela De Ingeniería Electrónica. p. 1-21.

ANIM-AYEKO Alberta Odamea., SCHILLACI Calogero y LIPANI, Aldo. 2023. Automatic blight disease detection in potato (Solanum tuberosum L.) and tomato (Solanum lycopersicum, L. 1753) plants using deep learning. Smart Agricultural Technology. Elsevier B.V., Vol. 4, núm. January, p. 100178. ISSN 27723755. DOI 10.1016/j.atech.2023.100178.

APURÍMAC, G.R. de., 2011. "Incremento de la productividad de papa asociacion de productores de papas nativas andino de sacclaya".

BENGIO Yoshua. 2012. Practical Recommendations for Gradient-Based Training of Deep Architectures.

BOCHKOVSKIY Alexey, WANG Chien-Yao y LIAO Hong-Yuan Mark. 2020. YOLOv4: Optimal Speed and Accuracy of Object Detection.

CAMPOS Y COVARRUBIAS, G. y LULE MARTÍ-NEZ, N.E., 2012. La Observación, Un Método Para El Estudio De La Realidad. Vol. 7, núm. 13, p. 45-60. DOI 10.37646/xihmai.v7i13.202.

CARRIÓN HERRERA Cinthya María Carrión, 2017. "Detección temprana del Potato Yellow Vein Virusen cultivos de Solanum tuberosumL. mediante la teledetección". Lima:

CENTENO FRANCO Alba. 2019. Deep Learning. España, Sevilla:

CÓRDOVA PÉREZ Claudia Sofía. 2021. Aplicación de aprendizaje profundo para la detección y clasificación automática de insectos agrícolas en trampas pegantes. Pontificia Universidad Católica del Perú.

ESPINO TIMÓN Carlos. 2017. "Análisis predictivo: técnicas y modelos utilizados y aplicaciones del mismo - herramientas Open Source que permiten su uso.

FIGUEREDO Andres y BALLESTEROS Javier. 2016. Identificación del estado de madurez de las frutas con redes neuronales artificiales, una revisión Fruit ripeness identification with artificial neural networks-A review. A: Revista Ciencia y Agricultura (Rev. Cien. Agri. Vol. 13, núm. 1, p. 117-132.

FUENTES PLAZA Fabián Nicolás. 2021. Visión por computadora para el manejo de plagas y enfermedades en cultivos de papa. Concepción:

GALAN ZAPATA Jefferson Luis. 2021. Sistema inteligente de reconocimiento de imágenes para apoyar el diagnóstico de plagas y enfermedades en el cultivo de arroz en el departamento de lambayeque en el año 2019. Chiclayo.

GEEKSFORGEEKS., 2023. Faster R-CNN | ML. A: 23 Agosto.

GIL Ricardo, CRUZ Edwin y PERDOMO Oscar. 2022. Modelos de Machine Learning para clasificar la cartera en un fondo de pensiones. A: . p. 1-50.

GIRSHICK Ross, DONAHUE Jeff, DARRELL Trevor, MALIK Jitendra. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation.

GONZÁLEZ-CRUZ, C. y otros, 2020. Machine learning in melanoma diagnosis. Limitations about to be overcome. A: Actas Dermo-Sifiliográficas (English Edition). Elsevier, Vol. 111, núm. 4, p. 313-316.

HERNÁNDEZ SAMPIERI Roberto, FERNÁNDEZ COLLADO Carlos y BAPTISTA

LUCIO Pilar. 2014. Metodologia Investigacion.

INEI. 2020. PERÚ: Panorama Económico Departamental.

KOHAVI Ron. 1995. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. p. 1137-1143.

LÓPEZ BRIEGA Raúl.E.. 2018. Introducción al Machine Learning. IAAR. Argentina.

LUNA QUECAÑO Juan Carlos, ZAPANA PARI Juan Gregorio, CUTIPA LIMACHE Alberto Magno, FLORIDA ROFNER Nelino. 2020. Efecto de la micorriza (Glomus Intrarradices), en el rendimiento de dos variedades de papa (Solanum Tuberosum L.) en el Altiplano de Puno-Perú. A: Revista de Investigaciones Altoandinas. Universidad Nacional del Altiplano, Vol. 22, núm. 1, p. 58-67.

MARZAL VARÓ Andrés y GRACIA LUENGO Isabel. 2014. Introducción a la programación con Python. ISBN 9788469258699. DOI 10.6035/sapientia93.

MÉNDEZ Patricio y GAETE Nelba. 2010. Las principales enfermedades que afectan al cultivo de papa. A: INIA Carrillanca p. 7-34.

MIDAGRI., 2023. Observatorio de siembras y perspectivas de producción de Papa. A:

Articulo. p. 1-52.

MINISTERIO DE DESARROLLO AGRARIO y RIEGO DEL PERÚ, 2020. Generalidades del cultivo de papa en el Perú.

MORENO MAYHUIRE Joel Saul. 2020. Eficiencia De Un Sistema De Control De Calidad Mediante Procesamiento Digital De Imágenes En La Clasificación De La Tunta En La Planta De Producción

De Kishuara - Andahuaylas. A: Proceedings - 14th Latin American Conference on Learning Technologies, LACLO 2019. Vol. 7, núm. 2, p. 69-79. ISSN 0301-7036.

NISHAD Md Ashiqur Rahaman, MITU Meherabin Akter y JAHAN Nusrat. 2022. Predicting and Classifying Potato Leaf Disease using K-means Segmentation Techniques and Deep Learning Networks. A: Procedia Computer Science. Elsevier B.V., Vol. 212, núm. C, p. 220-229. ISSN 18770509. DOI 10.1016/j.procs.2022.11.006.

OPPENHEIM Dor, SHANI Guy, ERLICH Orly, TSROR Leah. 2019. Using deep learning for image-based potato tuber disease detection. A: Phytopathology. American Phytopathological Society, Vol. 109, núm. 6, p. 1083-1087. ISSN 0031949X. DOI 10.1094/PHYTO-08-18-0288-R.

QASIM Jaffery. 2023. Deep Neural Network.

REDMON Joseph, DIVVALA Santosh, GIRSHICK Ross, FARHADI Ali. 2016. You only look once: Unified, real-time object detection. A: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. Vol. 2016-Decem, p. 779-788. ISSN 10636919. DOI 10.1109/CVPR.2016.91.

REN Shaoqing, HE Kaiming, GIRSHICK Ross, SUN Jian. 2015. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.

REN Shaoqing, HE Kaiming, GIRSHICK Ross, SUN Jian. 2017. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. A: IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol. 39, núm. 6, p. 1137-1149. ISSN 01628828. DOI 10.1109/TPAMI.2016.2577031.

ROBOFLOW., 2022. Roboflow.

RONNIE-GAKEGNE Edwin y MARTINEZ-COCA Benedicto. 2019. Eficacia de dos biofungicidas para el manejo en campo del Tizón temprano (Alternaria solani Sorauer) de la papa (Solanum tuberosum L.). A: Revista de Protección Vegetal. 1986 Centro Nacional de Sanidad Agropecuaria, Vol. 34, núm. 1.

ROZADA RANEROS Saúl. 2021. Estudio de la arquitectura YOLO para la detección de objetos mediante deep learning. Universidad de Valladolid.

RUEDA-PUENTE Edgar Omar, HERNÁNDEZ-MONTIEL Luis G, HOLGUÍN-PEÑA R J, FRANCISO H, LÓPEZ ELÍAS Jesús, HUEZ LOPEZ Marco A, JIMÉNEZ LEÓN

José, BORBOA FLORES Jesús, ORTEGA-GARCÍA Jesús. 2014. Ralstonia solanacearum: Una enfermedad bacteriana de importancia cuarentenaria en el cultivo de Solanum tuberosum L. A: Rev. INVURNUS. Vol. 9, núm. 1, p. 24-36.

RUIZ Juan Zamorano. 2018. Comparativa y análisis de algoritmos de aprendizaje automático para la predicción del tipo predominante de cubierta arbórea. Madrid:

RUSSELL Rudolph. 2018. Machine Learning Guía Paso a Paso Para Implementar Algoritmos De Machine Learning Con Python.

RUSSELL Stuart y NORVIG Peter. 2004. Inteligencia artificial. Un enfoque moderno. México: Pearson Educación, S.A. ISBN 978-84-205-4003-0.

SANDOVAL Lilian. 2018. Algoritmos de aprendizaje automático para análisis y predicción de datos. A: ITCA FEPADE Revista Tecnológica. Santa Tecla, El Salvador: Vol. 11, p. 36-40. ISSN 2072-568X.

SINGH Aditi y KAUR Harjeet. 2021. Potato plant leaves disease detection and classification using machine learning methodologies. A: IOP Conference Series: Materials Science and Engineering. Vol. 1022, núm. 1. ISSN 1757899X. DOI 10.1088/1757-899X/1022/1/012121.

SORIA Elocia, ROJAS Fátima y ORTUÑO Noel. 2021. Inducción de tolerancia a Globodera spp. con microorganismos promotores de crecimiento en el cultivo de papa (Solanum tuberosum subsp. tuberosum). A: Revista Latinoamericana de la Papa. Vol. 25, núm. 1, p. 3- 20.

SZELISKI Richard. 2021. Computer Vision: Algorithms and Applications, 2nd Edition [en línia]. ISBN 978-3-527-41365-2.

G. SANTOS Thiago. 2020. Google Colab: ¿qué es y cómo usarlo?.

TORRES Hebert. 2002. Manual de las enfermedades más importantes de la papa en el Perú. Centro Internacional de la Papa. ISBN 9290602120.

ULTRALYTICS YOLO DOCS., 2024. YOLOv4: Detección de Objetos de Alta Velocidad y Precisión.

UNIVERSIDAD NACIONAL DE LA PLATA., 2020. Sanidad vegetal. Buenos Aires. Argentina.

ANEXOS

Anexo 01 — Fotografías de pie negro con YOLO V4

Figura 24 — Tallo de papa con pie negro con YOLO V4 F01

Figura 25 — Tallo de papa con pie negro con YOLO V4 F02

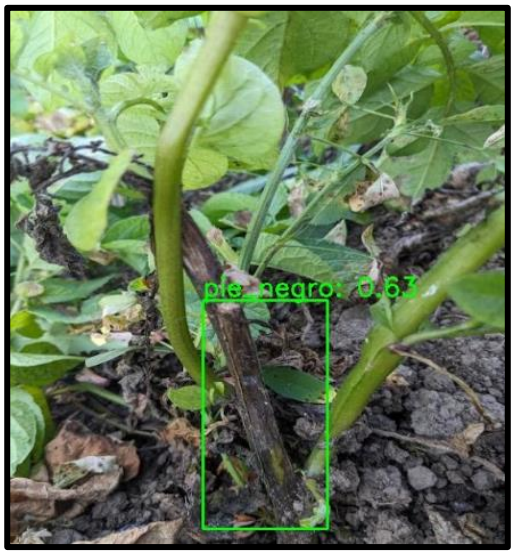


Figura 26 — Tallo de papa con pie negro con YOLO V4 F03

Figura 27 — Tallo de papa con pie negro con YOLO V4 F04

Figura 28 — Tallo de papa con pie negro con YOLO V4 F05

Figura 29 — Tallo de papa con pie negro con YOLO V4 F06

Anexo 02 — Fotografías de rancha con YOLO V4

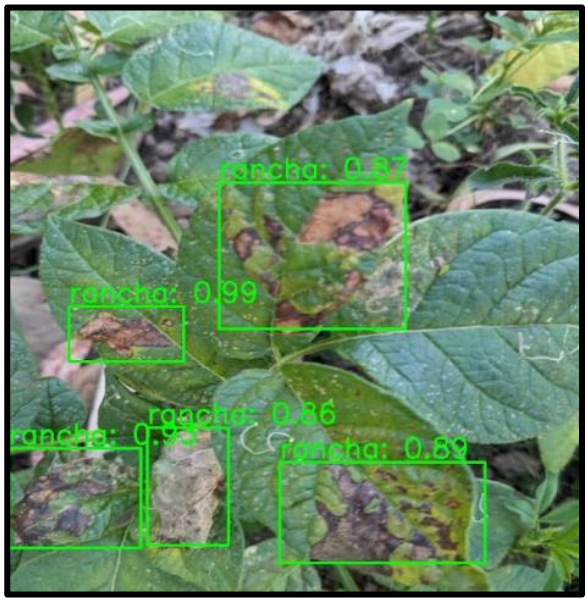


Figura 30 — Tallo de papa con rancha con YOLO V4 F01

Figura 31 — Tallo de papa con rancha con YOLO V4 F02

Figura 32 — Tallo de papa con rancha con YOLO V4 F03

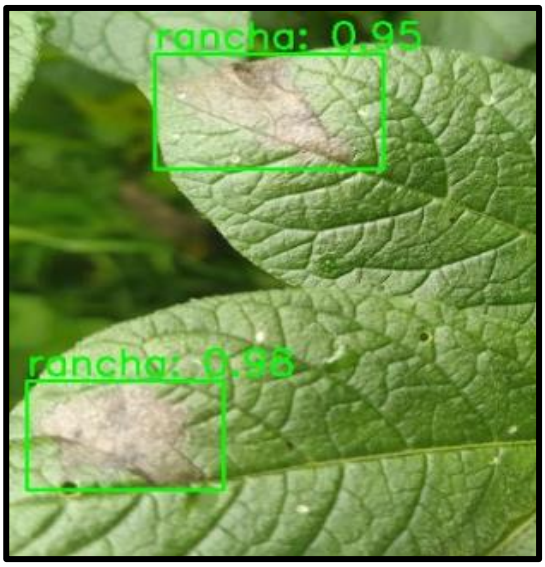


Figura 33 — Tallo de papa con rancha con YOLO V4 F04

Figura 34 — Tallo de papa con rancha con YOLO V4 F05

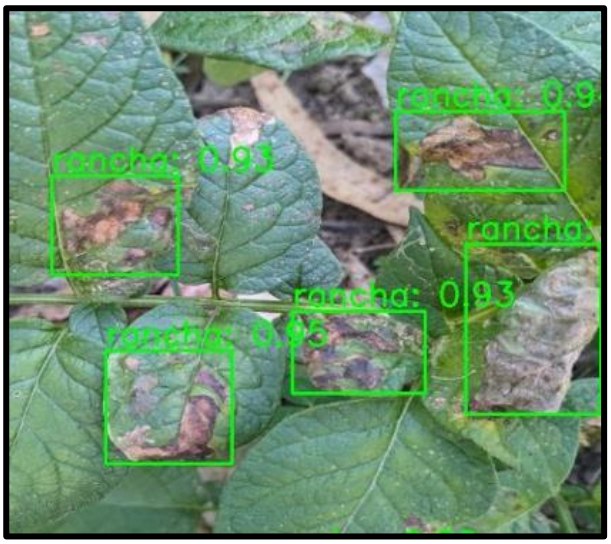


Figura 35 — Tallo de papa con rancha con YOLO V4 F06

Anexo 03 — Fotografías de pie negro con faster R-CNN

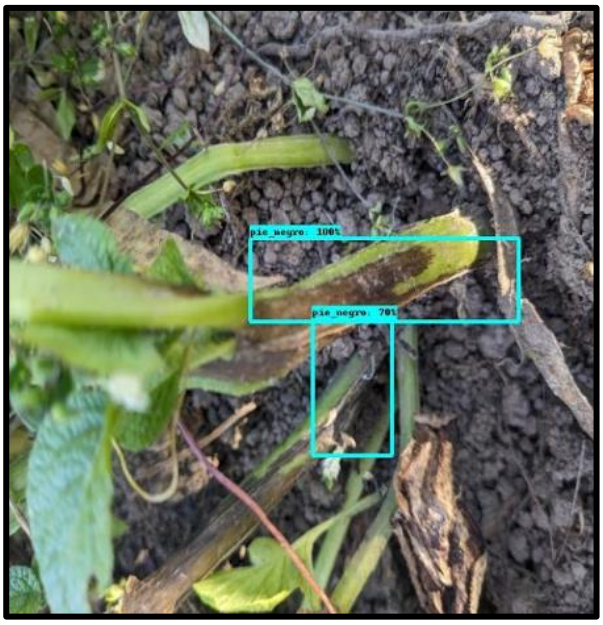


Figura 36 — Tallo de papa con pie negro con faster R-CNN F01

Figura 37 — Tallo de papa con pie negro con faster R-CNN F02

Figura 38 — Tallo de papa con pie negro con faster R-CNN F03

Figura 39 — Tallo de papa con pie negro con faster R-CNN F04

Anexo 04 — Fotografías de rancha con faster R-CNN

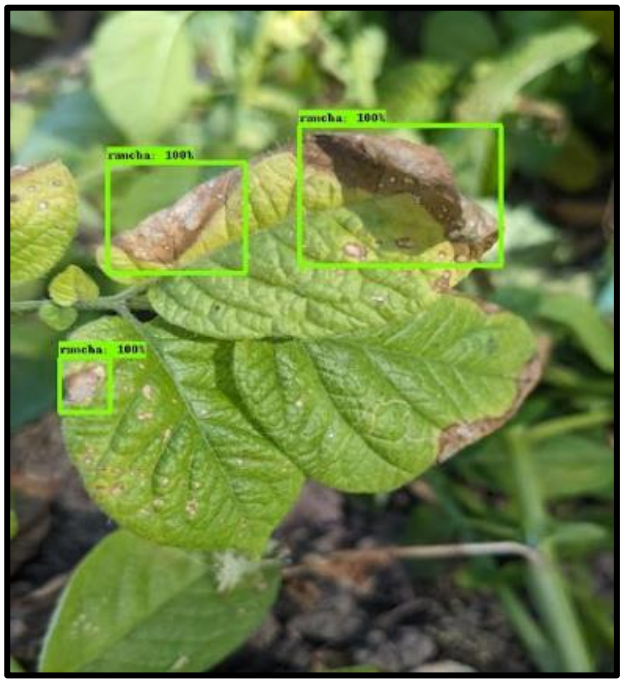


Figura 40 — Tallo de papa con rancha con faster R-CNN F01

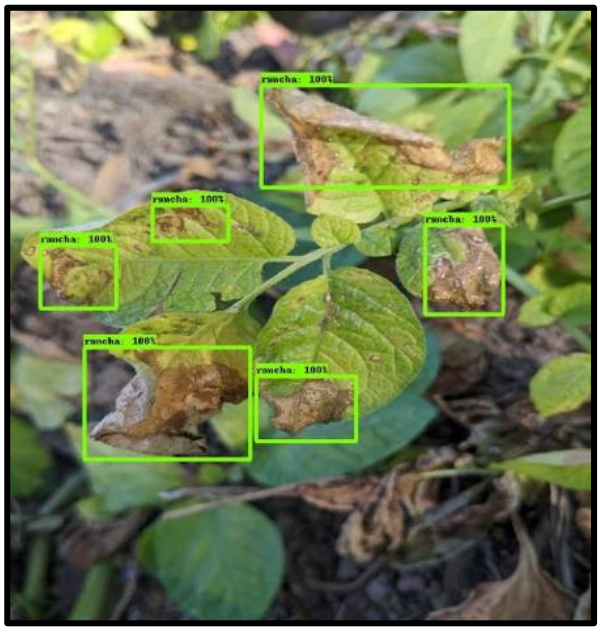


Figura 41 — Tallo de papa con rancha con faster R-CNN F02

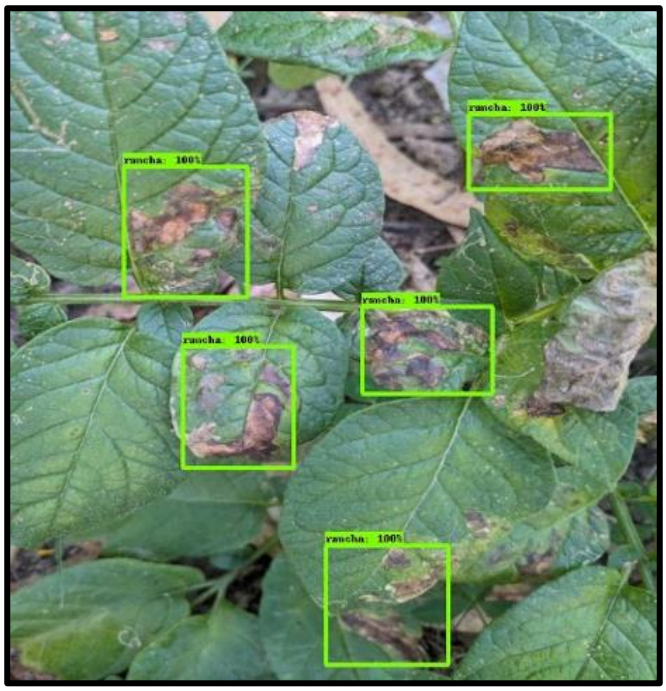


Figura 42 — Tallo de papa con rancha con faster R-CNN F03

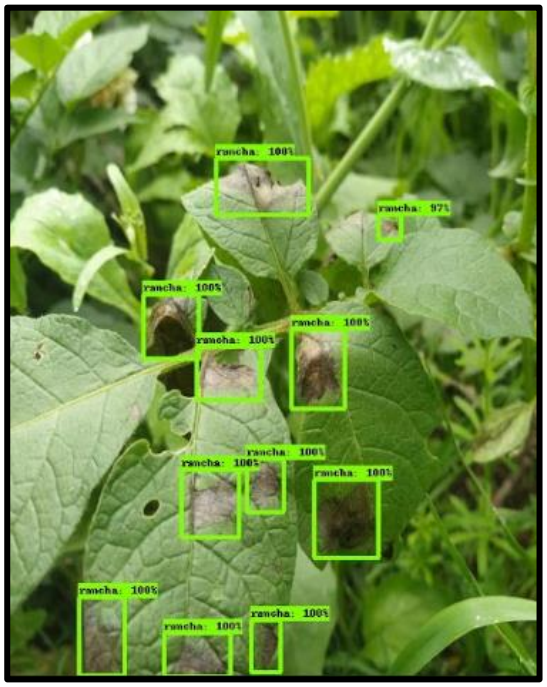


Figura 43 — Tallo de papa con rancha con faster R-CNN F04

Anexo 05 — Fichas de observación

Tabla 17 — Ficha de observación para el entrenamiento de modelos

Ficha de observación "Entrenamiento de modelos"				
Tipo de modelo:		• YOLO V4 • Faster R-CNN		
Datos de entrenamiento:		• **N° de fotografías:** 909 fotografías • **Etiquetado de datos:** Las fotografías fueron previamente etiquetadas con el tipo de enfermedad correspondiente para facilitar el aprendizaje supervisado de ambos modelos.		
categoría	imagen_jpg	archivo_txt	**label**	cantidad_ etiquetas
train	PXL_20230812_135206074_jpg.rf.44a2811dc83f77c27f2d34cf44ab93d5.jpg	PXL_20230812_135206074_jpg.rf.44a2811dc83f77c27f2d34cf44ab93d5.txt	**pie_negro**	1
train	IMG_20230401_084722_jpg.rf.bab27d22a60cf71635f41dea02064829.jpg	IMG_20230401_084722_jpg.rf.bab27d22a60cf71635f41dea02064829.txt	**rancha**	2
train	PXL_20230812_135200846_jpg.rf.086b0228f270132ef77a6fb5abe8e4cb.jpg	PXL_20230812_135200846_jpg.rf.086b0228f270132ef77a6fb5abe8e4cb.txt	**pie_negro**	1
train	IMG_20230318_065156_jpg.rf.fbe28f70e9d29a31b14cc72d9a21fa59.jpg	IMG_20230318_065156_jpg.rf.fbe28f70e9d29a31b14cc72d9a21fa59.txt	**rancha**	1
train	PXL_20230812_134634403-MP_jpg.rf.a70235e8d8a92a818b8022d7145ccc97.jpg	PXL_20230812_134634403-MP_jpg.rf.a70235e8d8a92a818b8022d7145ccc97.txt	**rancha**	5
train	IMG_20230401_084815_jpg.rf.01464bd4d335ea1e9bf70b8d74fc4cfa.jpg	IMG_20230401_084815_jpg.rf.01464bd4d335ea1e9bf70b8d74fc4cfa.txt	**rancha**	4
train	PXL_20230812_135207055_jpg.rf.6761c4146952c456b9d864faf51468aa.jpg	PXL_20230812_135207055_jpg.rf.6761c4146952c456b9d864faf51468aa.txt	**pie_negro**	1
train	PXL_20230812_134839528_jpg.rf.4e089265cb5b15531b069c099307398a.jpg	PXL_20230812_134839528_jpg.rf.4e089265cb5b15531b069c099307398a.txt	**pie_negro**	1
train	IMG_20230401_085742_jpg.rf.8ba2e1a036d32bf6544c9f3920b5e19c.jpg	IMG_20230401_085742_jpg.rf.8ba2e1a036d32bf6544c9f3920b5e19c.txt	**rancha**	7
train	IMG_20230401_085602_jpg.rf.56f98de09fb9ce1acc1866b0c9e3f9ae.jpg	IMG_20230401_085602_jpg.rf.56f98de09fb9ce1acc1866b0c9e3f9ae.txt	**rancha**	7
train	PXL_20230812_135158991_jpg.rf.3b70f2e3d2d7f113c7c6525b7efa1a01.jpg	PXL_20230812_135158991_jpg.rf.3b70f2e3d2d7f113c7c6525b7efa1a01.txt	**pie_negro**	1
train	IMG_20230401_085453_jpg.rf.7429ac64f30bc87116c61f00f4f05e83.jpg	IMG_20230401_085453_jpg.rf.7429ac64f30bc87116c61f00f4f05e83.txt	**rancha**	4
train	IMG_20230401_085453_jpg.rf.779bd61020929bbb09941725d5b69ec2.jpg	IMG_20230401_085453_jpg.rf.779bd61020929bbb09941725d5b69ec2.txt	**rancha**	4
. . .				

train	IMG_20230401_085331_jpg.rf.930dc9f5964d7fc56c715c360c5a79d8.jpg	IMG_20230401_085331_jpg.rf.930dc9f5964d7fc56c715c360c5a79d8.txt	**rancha**	5
train	PXL_20230812_134505053_jpg.rf.c21fe37a643aa534816f6ffa91e5e5fa.jpg	PXL_20230812_134505053_jpg.rf.c21fe37a643aa534816f6ffa91e5e5fa.txt	**pie_negro**	1

Tabla 18: — Ficha de observación de prueba de modelos

Ficha de observación "Prueba de modelos"					
Tipo de modelo:		• YOLO V4 • Faster R-CNN			
Datos de prueba:		• **N° de fotografías:** 44 fotografías • **Prueba de datos:** Se obtienen el número de detecciones por cada modelo y comparación de los números reales de etiquetado.			
categoría	**imagen_jpg**	**Label**	**cantidad_etiquetas**	**Numero_de_detecciones_RCNN**	**Numero_de_detecciones_YOLO**
test	PXL_20230812_134242209_jpg.rf.62205137b62854576e2d9625f3fbcb72.jpg	pie_negro	**1**	**1**	**1**
test	PXL_20230812_135614459_jpg.rf.56a9ba239b390dbf4eb72243e25c7132.jpg	pie_negro	**1**	**1**	**1**
test	IMG_20230318_070105_jpg.rf.b9196111d11445a5ab1de598f752bd53.jpg	Rancha	**2**	**1**	
test	PXL_20230812_134635671_jpg.rf.cc929d4202a87e129375f6f63ae15075.jpg	Rancha	**5**	**5**	**5**
test	PXL_20230812_135902150_jpg.rf.6b85941290fa103a97aa36456a2bd94e.jpg	pie_negro	**1**	**1**	**1**
test	PXL_20230812_134851859-MP_jpg.rf.285050e64e85181dfde719ba2dba2e51.jpg	pie_negro	**1**	**1**	**1**
test	IMG_20230401_085428_jpg.rf.208ae6cbfd095d969edf9b7f1db88ce1.jpg	rancha	**14**	**11**	**19**
test	PXL_20230812_133521449_jpg.rf.e1d8df5d47a4422b5826dbf62aaba8bd.jpg		**0**		
test	PXL_20230812_135917956_jpg.rf.c07f1eb165bf710be41bd7c62325c4bc.jpg	pie_negro	**1**	**2**	**2**
test	IMG_20230318_070111_jpg.rf.5499a51fc10c04684ee35cab097fca36.jpg	rancha	**4**	**1**	
test	IMG_20230401_084808_jpg.rf.5f4014382f02da59c95b8d3a2ff352e9.jpg	rancha	**2**	**2**	**5**

test	IMG_20230318_065054_jpg.rf.703542b8ed9fe6306074dd29b20a7845.jpg	rancha	**1**		
test	PXL_20230812_135204449_jpg.rf.dbf856ac668ae712eba620efb54046e0.jpg	pie_negro	**1**	**1**	**1**
test	PXL_20230812_135607360_jpg.rf.ba2c7c78d20024ba6cde5a296167f827.jpg	pie_negro	**1**	**1**	**1**
test	IMG_20230401_085152_jpg.rf.730a091ef763d3b0956c7b17b572c874.jpg	rancha	**2**	**1**	**2**
test	PXL_20230812_135251660-MP_jpg.rf.1ef28f7c4f6b8a28b32dbbd2fe5843bf.jpg	rancha	**6**	**6**	**6**
test	PXL_20230812_135824378_jpg.rf.2a651251594f0488d9551615e4ae7dcb.jpg	pie_negro	**2**	**4**	**2**
test	IMG_20230401_085252_jpg.rf.0a5cb9a4f4c46efe7a186c22f8e27976.jpg	rancha	**12**	**8**	**12**
test	PXL_20230812_135911312-MP_jpg.rf.33c07a612ff882133aa31e74c305b6cd.jpg	pie_negro	**1**	**1**	**2**
test	PXL_20230812_133418448_jpg.rf.ab1f3af44d571523509e1b4e570e8db5.jpg	rancha	**2**	**2**	**3**
test	PXL_20230812_133925477_jpg.rf.99ceed1cd46e60e4811575577c4848f6.jpg		**0**		
test	PXL_20230812_134315318_jpg.rf.67c1c5c9e7ed897194b217205991ec18.jpg	pie_negro	**1**	**1**	**1**
test	PXL_20230812_134838827_jpg.rf.1a544c5333c5f012cf02d47898deb132.jpg	pie_negro	**1**	**1**	**1**
test	PXL_20230812_134843008_jpg.rf.a45261c495caba5bbeb6b9de587fbf36.jpg	pie_negro	**1**	**1**	**1**
test	PXL_20230812_134159234_jpg.rf.6511a41da62cf6fc72cc2e9616180766.jpg	pie_negro	**1**		**1**
test	IMG_20230401_084957_jpg.rf.7690b8531b1bdcf789ac8adf50902720.jpg	rancha	**9**	**10**	**9**
test	PXL_20230812_135256615_jpg.rf.051dc3bfe715e6f21e419449c144f9e0.jpg	rancha	**3**	**3**	**6**
test	PXL_20230812_134352790_jpg.rf.a4c60e9a5906f95260825d167d8d67a1.jpg	rancha	**2**	**2**	**2**

test	PXL_20230812_135910487_jpg.rf.127be9f386930e9e70419e740443d1e3.jpg	pie_negro	**1**	**1**	**2**
test	IMG_20230401_084714_jpg.rf.9efe53097bc80bde56823b7a5d49cedb.jpg	rancha	**2**	**2**	**2**
test	PXL_20230812_135224240_jpg.rf.948abc3968cc3f92a9739de595c919e2.jpg		**0**		
test	PXL_20230812_134633738_jpg.rf.af52a12a4dad3efdf2ad02ea2682285a.jpg	rancha	**6**	**5**	**7**
test	PXL_20230812_135825661_jpg.rf.78f211a3e629429116d2eed1b55068e8.jpg	pie_negro	**2**	**4**	**2**
test	PXL_20230812_134340180_jpg.rf.c2fd708b06049bf02a7eea8d75cf1bf1.jpg		**0**		
test	IMG_20230401_085233_jpg.rf.dc0a08ac00be4ecad15195ac36b442cb.jpg	rancha	**22**	**18**	**28**
test	PXL_20230812_135152805_jpg.rf.636be2691d49e2f0b77a0e23172d9806.jpg	pie_negro	**2**	**2**	**2**
test	IMG_20230401_084724_jpg.rf.1e85ea6c1a2e786c07db9724603843d5.jpg	rancha	**2**	**2**	**2**
test	IMG_20230318_065737_jpg.rf.db5b86d394e83d2d11b0392c12d1188b.jpg	rancha	**4**	**1**	**1**
test	IMG_20230401_085205_jpg.rf.cd8d034c3aacf8af8e8af50308f8dded.jpg	rancha	**5**	**3**	**2**
test	IMG_20230318_070028_jpg.rf.bdb50861cef36be43506e77f1ca63f25.jpg	rancha	**1**	**1**	**1**
test	PXL_20230812_135822868-MP_jpg.rf.2b57bdfecbc8cf9118c1fc2eea05f849.jpg	pie_negro	**2**	**2**	**2**
test	IMG_20230401_084603_jpg.rf.92d89aeae655cfa579a295a75b8097a4.jpg	rancha	**2**	**3**	**3**
test	PXL_20230812_135826384_jpg.rf.89f54877e57b68ac4e33cfaf30d30d13.jpg	pie_negro	**1**	**4**	**2**
test	IMG_20230318_065519_jpg.rf.6703dcda6afa1b36c3894b02ef36bbb5.jpg	rancha	**2**	**2**	**2**

Anexo 06 — Códigos fuente del modelo faster R-CNN

```
# -*- coding: utf-8 -*-
"""model_train_rcnn.ipynb

Automatically generated by Colab.

Original file is located at
    https://colab.research.google.com/drive/1pfXjwZ7n21j9xct23BcXrqFLQlxVVxXv

# Dependencies
"""

# Use last runtine colab (No longer active (Feb2024))
# Connect to your GPU and go to tools -> command palette -> search for use
fallback runtime version

"""## CUDA"""

!wget
https://developer.download.nvidia.com/compute/cuda/11.8.0/local_installers/cud
a-repo-ubuntu1804-11-8-local_11.8.0-520.61.05-1_amd64.deb

!dpkg -i cuda-repo-ubuntu1804-11-8-local_11.8.0-520.61.05-1_amd64.deb

!ls /var/cuda-repo-ubuntu1804-11-8-local | grep .pub

!apt-key add /var/cuda-repo-ubuntu1804-11-8-local/7D65C20C.pub

!apt-get update

!apt-get install cuda-11-8

!ls /usr/local/cuda-11.8

!export CUDA_PATH=/usr/local/cuda-11.8/

!nvcc --version

"""## Tensorflow"""

!pip install tensorflow[and-cuda]==2.13.0

!python3 -c "import tensorflow as tf;
print(tf.reduce_sum(tf.random.normal([1000, 1000])))"

!python3 -c "import tensorflow as tf;
print(tf.config.list_physical_devices('GPU'))"

!git clone --depth 1 https://github.com/tensorflow/models
```

```
"""## Utils"""

#AUX: Solve problem version file

def copy_and_update_version():
  with open('/content/models/research/object_detection/packages/tf2/setup.py')
as f:
    content_list = f.readlines()

  content_list.insert(20, "'tensorflow==2.13.0',")

  with open('/content/models/research/setup.py', 'w+') as f:
    f.writelines(content_list)
  print('Updated!')

"""## Setup model"""

# Commented out IPython magic to ensure Python compatibility.
# %%bash
# cd models/research/
# pwd
# protoc object_detection/protos/*.proto --python_out=.

copy_and_update_version()

# Commented out IPython magic to ensure Python compatibility.
# %%capture
# %%bash
# cd models/research/
# pip install .

"""### Validate version & GPU"""

pip show tensorflow

import tensorflow as tf
tf.test.gpu_device_name()

"""## Test Model"""

!python
/content/models/research/object_detection/builders/model_builder_tf2_test.py

"""# Connect to Google Drive

## Mount
"""

base_path = '/content/drive/'
```

```
potatoes_project_path = f'{base_path}/My Drive/Educacion/Tesis Unamba/Modelo-Papa/Tensorflow/'

from google.colab import drive
drive.mount(base_path)
!ls -l "{potatoes_project_path}"

"""## Copy images & config files"""

!cp "{potatoes_project_path}images_train/train.zip" .
!cp "{potatoes_project_path}images_train/test.zip" .
!cp -r "{potatoes_project_path}config" .

# Commented out IPython magic to ensure Python compatibility.
# %%capture
# !unzip -o ./train.zip -d data/
# !unzip -o ./test.zip -d data/

!mkdir ./scripts
!cp "{potatoes_project_path}utils/generate_tfrecord.py" ./scripts/

"""# Pre-train

## Generate text files with images data to train model
"""

!python ./scripts/generate_tfrecord.py -x ./data/train -l ./config/label_map.pbtxt -o ./config/train.record
!python ./scripts/generate_tfrecord.py -x ./data/test -l ./config/label_map.pbtxt -o ./config/test.record

"""## Download Model"""

!wget http://download.tensorflow.org/models/object_detection/tf2/20200711/faster_rcnn_resnet101_v1_640x640_coco17_tpu-8.tar.gz

"""## Unzip model"""

!mkdir pre-trained-models
!tar -zxf  faster_rcnn_resnet101_v1_640x640_coco17_tpu-8.tar.gz --directory ./pre-trained-models/

#Copy main TF
!cp /content/models/research/object_detection/model_main_tf2.py .

"""# Train"""

model_dir = 'config/models/faster_rcnn_resnet101_v1'
```

```
pipeline_config_path =
'config/models/faster_rcnn_resnet101_v1/pipeline.config'

!python /content/models/research/object_detection/model_main_tf2.py \
  --model_dir={model_dir}\
  --pipeline_config_path={pipeline_config_path} \
  --alsologtostderr

"""## Evaluating"""

# Commented out IPython magic to ensure Python compatibility.
# %load_ext tensorboard
# %tensorboard --logdir {model_dir}

potatoes_project_path

#Backup model trained
!pwd
!cp -r {model_dir} "{potatoes_project_path}/backup"

#Evaluating model
!python model_main_tf2.py --model_dir={model_dir} --
pipeline_config_path={model_dir}/pipeline.config --checkpoint_dir={model_dir}

"""# Export Model"""

!cp /content/models/research/object_detection/exporter_main_v2.py .

!python exporter_main_v2.py --input_type image_tensor --pipeline_config_path
{pipeline_config_path} --trained_checkpoint_dir {model_dir} --output_directory
exported-models/faster_rcnn_resnet101_v1

!mkdir -p "{potatoes_project_path}exported-models/faster_rcnn_resnet101_v1"

# Commented out IPython magic to ensure Python compatibility.
#SAVE EXPORTED MODEL
# %cd /content
!pwd
!cp -r exported-models/faster_rcnn_resnet101_v1
"{potatoes_project_path}exported-models/faster_rcnn_resnet101_v1"
```

Figura 44 — Código fuente para entrenar el modelo faster R-CNN

```
# -*- coding: utf-8 -*-
"""model_prediction_rcnn.ipynb

Automatically generated by Colab.

Original file is located at
    https://colab.research.google.com/drive/1wm7OMjwQ_XWAc_rk2uJCEms6WMhyXrZD

# Dependencias

## Tensorflow
"""

# Commented out IPython magic to ensure Python compatibility.
# %%capture
# !pip install tensorflow[and-cuda]==2.13.0

!git clone --depth 1 https://github.com/tensorflow/models

"""## Utilitarios"""

#AUX: Solve problem version file

def copy_and_update_version():
  with open('/content/models/research/object_detection/packages/tf2/setup.py')
as f:
    content_list = f.readlines()

  content_list.insert(20, "'tensorflow==2.13.0',")

  with open('/content/models/research/setup.py', 'w+') as f:
    f.writelines(content_list)
  print('Updated!')

"""## Configurando el modelo"""

# Commented out IPython magic to ensure Python compatibility.
# %%bash
# cd models/research/
# pwd
# protoc object_detection/protos/*.proto --python_out=.

copy_and_update_version()

# Commented out IPython magic to ensure Python compatibility.
# %%capture
# %%bash
# cd models/research/
```

```
# pip install .

"""### Validando versión & GPU"""

pip show tensorflow

import tensorflow as tf
tf.test.gpu_device_name()

"""# Conectando a Google Drive

## Montar
"""

base_path = '/content/drive/'
potatoes_project_path = f'{base_path}/My Drive/Educacion/Tesis Unamba/Modelo-Papa/Tensorflow/'

from google.colab import drive
drive.mount(base_path)
!ls -l "{potatoes_project_path}"

"""
## Copiando imagenes y archivos de configuración"""

!cp -r "{potatoes_project_path}images_prediction" .
!cp -r "{potatoes_project_path}exported-models/faster_rcnn_resnet101_v1/faster_rcnn_resnet101_v1" .
!cp -r "{potatoes_project_path}config" .

"""# Predicción"""

# Commented out IPython magic to ensure Python compatibility.
# %matplotlib inline

# Commented out IPython magic to ensure Python compatibility.
# %cd /content/models/research/

import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'    # Suppress TensorFlow logging (1)
import pathlib
import tensorflow as tf

import time
from object_detection.utils import label_map_util
from object_detection.utils import visualization_utils as viz_utils

tf.get_logger().setLevel('ERROR')           # Suppress TensorFlow logging (2)

# Enable GPU dynamic memory allocation
```

```
gpus = tf.config.experimental.list_physical_devices('GPU')
for gpu in gpus:
    tf.config.experimental.set_memory_growth(gpu, True)

PATH_TO_TEST_IMAGES_DIR = pathlib.Path('/content/images_prediction/')
IMAGE_PATHS = sorted(list(PATH_TO_TEST_IMAGES_DIR.glob("*.jpg")))
# IMAGE_PATHS

MODEL_PATH = '/content/faster_rcnn_resnet101_v1'
LABEL_PATH = '/content/config/label_map.pbtxt'

#Load Model

PATH_TO_SAVED_MODEL = MODEL_PATH + "/saved_model"

print('Loading model...', end='')
start_time = time.time()

# Load saved model and build the detection function
detect_fn = tf.saved_model.load(PATH_TO_SAVED_MODEL)

end_time = time.time()
elapsed_time = end_time - start_time
print('Done! Took {} seconds'.format(elapsed_time))

category_index =
label_map_util.create_category_index_from_labelmap(LABEL_PATH,
                                                                    use_displa
y_name=True)
category_index

!mkdir /content/images_prediction_evaluated

!mkdir -p "{potatoes_project_path}images_prediction_evaluated"
!mkdir -p "{potatoes_project_path}images_prediction_evaluated_csv"

from collections import defaultdict

def getClassesDetectedByImage(
    image,
    boxes,
    classes,
    scores,
    category_index,
    min_score_thresh=.5
    ):
  classesDetected = []

  for i in range(boxes.shape[0]):
```

```
    if scores is None or scores[i] > min_score_thresh:
      box = tuple(boxes[i].tolist())

      if scores is None:
        raise ValueError("scores should not be None")
      else:
        class_name = category_index[classes[i]]['name']
        display_str = str(class_name)
        # if not skip_scores:
        #   if not display_str:
        #     display_str = '{}%'.format(round(100*scores[i]))
        #   else:
        #     display_str = '{}: {}%'.format(display_str,
round(100*scores[i]))

        classesDetected.append((image, display_str))

  counts = defaultdict(int)
  for _, label in classesDetected:
      counts[(image, label)] += 1

  result = [(image_name, label, count) for (image_name, label), count in
counts.items()]

  return result

import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
import matplotlib.image as mpimg  # Importar matplotlib.image para guardar la
imagen
import warnings
warnings.filterwarnings('ignore')   # Suprimir advertencias de Matplotlib

def load_image_into_numpy_array(path):
    return np.array(Image.open(path))

info_data = []

for image_path in IMAGE_PATHS:

    print('Running inference for {}... '.format(image_path), end='')

    image_np = load_image_into_numpy_array(image_path)

    input_tensor = tf.convert_to_tensor(image_np)
    input_tensor = input_tensor[tf.newaxis, ...]

    detections = detect_fn(input_tensor)
```

```
    num_detections = int(detections.pop('num_detections'))
    detections = {key: value[0, :num_detections].numpy()
                   for key, value in detections.items()}
    detections['num_detections'] = num_detections

    # Las clases de detección deben ser enteros.
    detections['detection_classes'] =
detections['detection_classes'].astype(np.int64)

    image_np_with_detections = image_np.copy()

    viz_utils.visualize_boxes_and_labels_on_image_array(
          image_np_with_detections,
          detections['detection_boxes'],
          detections['detection_classes'],
          detections['detection_scores'],
          category_index,
          use_normalized_coordinates=True,
          max_boxes_to_draw=200,
          min_score_thresh=.30,
          agnostic_mode=False)

    info_data += getClassesDetectedByImage(
          image_path.name,
          detections['detection_boxes'],
          detections['detection_classes'],
          detections['detection_scores'],
          category_index,
          min_score_thresh=.30)

    output_image_path = str(image_path).replace('images_prediction',
'images_prediction_evaluated')
    mpimg.imsave(output_image_path, image_np_with_detections)  # Guardar la
imagen modificada

    print('Done')

"""## Copia de imagenes evaluadas por el modelo"""

!cp -r /content/images_prediction_evaluated
"{potatoes_project_path}images_prediction_evaluated/"

import pandas as pd
info = pd.DataFrame(info_data, columns=['Image', 'Deteccion',
'Numero_de_detecciones'])

"""## Resumen"""

info
```

```
info.to_csv(f"{potatoes_project_path}images_prediction_evaluated_csv/rcnn_resu
lts.csv")
```

Figura 45 — Código fuente para prueba del modelo faster R-CNN

Anexo 07 — Códigos fuente del modelo YOLO V4

```
# -*- coding: utf-8 -*-
"""model_train_yolo.ipynb

Automatically generated by Colab.

Original file is located at
    https://colab.research.google.com/drive/1FCD_7ssrGU9AWWU6FOoDHX-7dAJodE4g

# Dependencias

## OpenCV
"""

pip install opencv-python==4.5.4.58

"""## Yolo V4 - darknet"""

!git clone https://github.com/AlexeyAB/darknet

# Commented out IPython magic to ensure Python compatibility.
# %cd darknet
!sed -i 's/OPENCV=0/OPENCV=1/' Makefile
!sed -i 's/GPU=0/GPU=1/' Makefile
!sed -i 's/CUDNN=0/CUDNN=1/' Makefile
!sed -i 's/CUDNN_HALF=0/CUDNN_HALF=1/' Makefile

# Commented out IPython magic to ensure Python compatibility.
# %%capture
# !make

"""# Utilitarios

"""

# Commented out IPython magic to ensure Python compatibility.
def imShow(path):
  import cv2
  import matplotlib.pyplot as plt
#   %matplotlib inline

  image = cv2.imread(path)
  height, width = image.shape[:2]
  resized_image = cv2.resize(image,(3*width, 3*height), interpolation =
cv2.INTER_CUBIC)

  fig = plt.gcf()
  fig.set_size_inches(18, 10)
  plt.axis("off")
```

```
  plt.imshow(cv2.cvtColor(resized_image, cv2.COLOR_BGR2RGB))
  plt.show()

# use this to upload files
def upload():
  from google.colab import files
  uploaded = files.upload()
  for name, data in uploaded.items():
    with open(name, 'wb') as f:
      f.write(data)
      print ('saved file', name)

# use this to download a file
def download(path):
  from google.colab import files
  files.download(path)

"""# Conectando a Google Drive

## Montado
"""

base_path = '/content/drive/'
potatoes_project_path = f'{base_path}/My Drive/Educacion/Tesis Unamba/Modelo-
Papa/Yolo/'

from google.colab import drive
drive.mount(base_path)
!ls -l "{potatoes_project_path}"

"""## Cargando imagenes & archivos de configuración"""

!cp "{potatoes_project_path}images_train/train.zip" ../
!cp "{potatoes_project_path}images_train/test.zip" ../

# Commented out IPython magic to ensure Python compatibility.
# %%capture
# !unzip -o ../train.zip -d data/
# !unzip -o ../test.zip -d data/

!cp "{potatoes_project_path}config/yolov4-obj.cfg" ./cfg
!cp "{potatoes_project_path}config/obj.names" ./data
!cp "{potatoes_project_path}config/obj.data"  ./data
!cp "{potatoes_project_path}utils/generate_train.py" ./
!cp "{potatoes_project_path}utils/generate_test.py" ./

"""# Pre - Entrenamiento

## Generando archivos de informacion de las imagenes para el entrenamiento
"""
```

```
!python generate_train.py
!python generate_test.py
!ls data/

"""# Entrenamiento"""

# Commented out IPython magic to ensure Python compatibility.
# %%capture
# !wget
https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v3_optimal/
yolov4.conv.137

"""## Entrenamiento inicial del modelo
En esta sección se lleva a cabo el entrenamiento del modelo. En esta fase es
importante considerar los siguientes puntos:


*   Se busca que el valor de pérdida **x, y avg loss ** tienda a cero.
*   Cada 100 pasos de entrenamiento se genera un archivo en la carpeta
/backup. Estos archivos se deben de estar descargando manualmente durante el
entrenamiento.
* En la iteración 1000 se puede esperar tener un valor menor a 8, de no ser
así, puede haber mejoras al set de datos
* Se debe de asegurar de haber actualizado los valores en la carpeta /config
* Los archivos del modelo entrenado se sobre escriben por lo cual es
importante estarlos descargando y catalogando de manera periodica.



"""

!./darknet detector train data/obj.data cfg/yolov4-obj.cfg yolov4.conv.137 -
dont_show -map

"""## Entrenamiento incremental del modelo
Esta sección es específica a entrenar sobre un modelo que ya fue entrenado
anteriormente. Se requiere contar con el archivo **.weights** que se guardó en
el último entrenamiento del modelo.
"""

# !./darknet detector train data/obj.data cfg/yolov4-obj.cfg backup/yolov4-
obj_v4_1400_l1.weights -dont_show

"""## Copia de seguridad del modelo entrenados
El archivo final son todos aquellos con terminación .weigts
"""

!cp -r backup "{potatoes_project_path}backup"
```

```
"""## Analisis del modelo entrenado"""

!./darknet detector map data/obj.data cfg/yolov4-obj.cfg backup/yolov4-
obj_best.weights

imShow('/content/darknet/chart.png')

imShow('/content/darknet/chart_yolov4-obj.png')
```

Figura 46 — Código fuente para entrenar el modelo YOLO V4

```
# -*- coding: utf-8 -*-
"""model_prediction_yolo.ipynb

Automatically generated by Colab.

Original file is located at
    https://colab.research.google.com/drive/1OUFs43bDYhPg03cITyxDaTqVwIckT4DX

# Dependencias

## OpenCV
"""

pip install opencv-python==4.5.4.58

"""# Conección a Google Drive

## Montar
"""

base_path = '/content/drive/'
potatoes_project_path = f'{base_path}/My Drive/Educacion/Tesis Unamba/Modelo-Papa/Yolo/'
trained_file = 'yolov4-obj_best.weights'
prediction_path = 'images_prediction'

from google.colab import drive
drive.mount(base_path)
!ls -l "{potatoes_project_path}"

!cp -r "{potatoes_project_path}{prediction_path}" .
!cp -r "{potatoes_project_path}config" .
!cp -r "{potatoes_project_path}backup" .
!mkdir -p "{potatoes_project_path}images_evaluated" .
!mkdir -p "{potatoes_project_path}images_evaluated_csv" .
!mkdir output

"""# Predicción utilizando el modelo"""

import cv2
import os
import pandas as pd
from google.colab.patches import cv2_imshow

"""## Cargar el modelo entrenado"""

with open('config/obj.names', 'r') as f:
    classes = f.read().splitlines()
```

```
net = cv2.dnn.readNetFromDarknet('config/yolov4-obj.cfg',
f'backup/backup/{trained_file}')

model = cv2.dnn_DetectionModel(net)
model.setInputParams(scale=1 / 255, size=(416, 416), swapRB=True)

info = pd.DataFrame(columns=['Imagen','Deteccion','Numero_de_detecciones'])
print('init')
sc = []
for archivo in os.listdir(f"./{prediction_path}"):
  img = cv2.imread(f"{prediction_path}/{archivo}")
  classIds, scores, boxes = model.detect(img, confThreshold=0.1,
nmsThreshold=0.1)
  className = None
  for (classId, score, box) in zip(classIds, scores, boxes):
    sc.append(score)
    cv2.rectangle(img, (box[0], box[1]), (box[0] + box[2], box[1] + box[3]),
                  color=(0, 255, 0), thickness=2)
    className = classes[classId]
    text = '%s: %.2f' % (className, score)
    cv2.putText(img, text, (box[0], box[1] - 5), cv2.FONT_HERSHEY_SIMPLEX, 1,
                color=(0, 255, 0), thickness=2)
    cv2.imwrite("output/{}".format(archivo),img)

  info.loc[len(info.index)] = [archivo, className, len(classIds)]

"""* Backup de las imagenes dentro de google drive"""

!cp -r /content/output "{potatoes_project_path}images_evaluated/"

"""* Resumen"""

info

"""* Exportando resultados en CSV"""

info.to_csv(f"{potatoes_project_path}images_evaluated_csv/yolo_results.csv")
```

Figura 47 — Código fuente para prueba del modelo YOLO V4