



Identificación de tizón de la soja mediante el uso de redes convolucionales

Sischik, Martin; Sanabria, Jorge; Colmán, Alice;

martinstolaruk@proton.me; jorge.sanabria@uc.edu.py; alice.colman@uc.edu.py

Facultad de Ciencias y Tecnología

Universidad Católica Ntra. Sra. de la Asunción – Campus Itapúa

Resumen

La investigación desarrolla un sistema de detección del tizón foliar (*Cercospora kikuchii*) en hojas de soja, empleando procesamiento de imágenes y redes convolucionales como herramienta de apoyo a agricultores en la identificación temprana de la enfermedad. El proyecto propone una solución tecnológica innovadora y eficaz para el monitoreo fitosanitario. El objetivo principal fue diseñar un modelo de red neuronal que permita reconocer síntomas, estableciéndose como objetivos específicos la recopilación y preprocesamiento de imágenes de hojas infectadas y no infectadas, el entrenamiento del modelo con datos y validación de resultados frente a evaluaciones agronómicas convencionales. La metodología se desarrolló en tres etapas: captura de imágenes, procesamiento, segmentación y entrenamiento del modelo. El sistema resultante fue integrado a una interfaz gráfica que permite la carga de imágenes para su análisis, facilitando su uso por parte de los técnicos. Los resultados arrojaron una precisión del 94% con 6 % de falsos positivos y 2 % de falsos negativos. Se observó que factores como la calidad de imagen y la iluminación influyen en la clasificación, recomendándose ampliar la diversidad del conjunto de datos para futuras mejoras. En conclusión, la propuesta representa una alternativa confiable para el diagnóstico fitosanitario, fomentando el uso de inteligencia artificial en prácticas de agricultura de precisión.

Palabras clave: Tizón foliar, redes neuronales convolucionales, procesamiento de imágenes, inteligencia artificial.



Introducción

La agricultura es uno de los motores productivos más grandes del mundo, especialmente en el Paraguay, siendo la soja uno de las principales producciones agrarias y Paraguay su tercer principal exportador mundial, esto conlleva a una gran dependencia de dicho cultivo para el progreso.

El cultivo de soja es el más predominante en el país, los cultivos deben estar en constante observación debido a la aparición repentina de enfermedades que afectan negativamente a la producción local y por ende a la economía de todo el país ya que el sector a agricultor es responsable del 22% del PIB del mismo.

Aunque este sector sea tan importante para el desarrollo sigue estando atrasado en temas de utilización de las nuevas tecnologías disponibles a nuestro alcance para poder asistir en el mejoramiento del sector.

Entre todas las enfermedades que la soja puede verse afectada una de las más comunes es el “tizón de la hoja” causado por el hongo *Cercospora Kikuchii* una enfermedad que afecta tanto a la planta de la soja como a sus semillas causando la comúnmente llamada mancha purpura de la semilla.

La detección más temprana posible de esta y otras enfermedades en la soja juega un papel crucial en la lucha contra la misma, ya que afecta directamente a la producción de una de las materias primas más importantes del Paraguay.

El presente trabajo se dedica a la creación de una herramienta que pueda determinar si una planta de soja está enferma con el “tizón de la hoja” utilizando técnicas de inteligencia artificial combinadas con visión por computadora utilizando redes neuronales convolucionales que puedan reconocer la forma de las hojas e identificar las distintas tonalidades que adquieren las mismas estando bajo los efectos de la enfermedad.



Objetivos

Objetivo General

- Identificación de Tizón de la soja en hojas mediante el uso de redes convolucionales

Objetivos Específicos

- Determinar las características visuales del "Tizón de la hoja" en la soja a través de resultados de investigaciones existentes y observación directa.
- Emplear modelos de redes neuronales convolucionales para entrenar el algoritmo de detección de la enfermedad.

Métodos y Materiales

En esta sección se describen los principales recursos y técnicas empleadas para la implementación del sistema, abarcando desde el entorno de desarrollo y el hardware utilizado, hasta las bibliotecas y herramientas de software que facilitan el procesamiento de imágenes, el entrenamiento del modelo y la evaluación de los resultados.

Software

Para el desarrollo del prototipo se optó por una combinación integral de herramientas de software que garantizan un alto nivel de robustez, flexibilidad y eficiencia en todas las etapas del proyecto.

La selección se realizó de manera cuidadosa para cubrir desde la escritura, depuración del código hasta la gestión de referencias bibliográficas y la implementación de complejos modelos de aprendizaje profundo. Entre los principales programas y entornos utilizados se encuentran:

- Visual Studio Code

Se eligió este entorno de desarrollo integrado (IDE) por su gran versatilidad, su amplio soporte de extensiones y su naturaleza de código abierto, lo que facilita la personalización según las necesidades específicas del proyecto. Visual Studio Code es compatible con múltiples sistemas operativos (Windows, Linux y macOS), lo que permite mantener un entorno de trabajo uniforme y coherente en diferentes plataformas. Además, su integración con sistemas de control de versiones como Git, a través de extensiones especializadas como GitLens, permite una gestión efectiva del código fuente, facilitando la colaboración en equipo y la trazabilidad de los cambios realizados. Su avanzada capacidad de depuración y



las herramientas integradas para el análisis del rendimiento del código aceleran la detección y corrección de errores, lo que resulta en un proceso de desarrollo mucho más ágil. Asimismo, su compatibilidad con el Subsistema de Windows para Linux (WSL) es crucial, ya que permite aprovechar al máximo las capacidades de aceleración por GPU y ejecutar scripts en un entorno Linux, lo cual es esencial para el entrenamiento de modelos de deep learning que requieren de grandes recursos de cómputo.

- Python

Se elige Python debido a su gran versatilidad y robustez. Su sintaxis clara y legible permite desarrollar y mantener código de forma sencilla, lo que acelera el prototipado y la iteración de soluciones. Además, su amplia comunidad y ecosistema de bibliotecas de terceros facilitan la integración de herramientas para análisis de datos, procesamiento de imágenes y desarrollo de algoritmos de inteligencia artificial. Esto se traduce en un entorno de trabajo que soporta tanto proyectos simples como aplicaciones complejas, garantizando compatibilidad multiplataforma y una curva de aprendizaje accesible que favorece la escalabilidad y la adaptabilidad de los proyectos.

- Tensorflow y Keras

Se han seleccionado estas dos bibliotecas especializadas en aprendizaje profundo debido a su amplia adopción en la comunidad de inteligencia artificial y su robusto ecosistema. Desarrollado por Google Brain, es un framework de código abierto diseñado para trabajar con grandes volúmenes de datos y aprovechar la aceleración por GPU mediante CUDA, lo que es fundamental para entrenar modelos de redes neuronales complejas en un tiempo razonable. Keras, que funciona como una API de alto nivel integrada en TensorFlow, simplifica enormemente el diseño y la experimentación con arquitecturas de redes neuronales convolucionales y otros modelos de deep learning. Gracias a su interfaz intuitiva, es posible construir y modificar modelos de forma rápida, permitiendo a los desarrolladores experimentar con diversas configuraciones de capas, funciones de activación, técnicas de regularización y callbacks personalizados. Esto agiliza la fase de prototipado y ajuste de parámetros, mejorando la precisión y la capacidad de generalización del modelo. Además, la extensa documentación y la activa comunidad de usuarios facilitan la resolución de problemas y la incorporación de nuevas funcionalidades que puedan surgir durante el desarrollo.



Hardware

El rendimiento y la eficiencia del entrenamiento de modelos de deep learning dependen en gran medida del hardware utilizado. Para este proyecto se emplearon dos configuraciones principales que, según la disponibilidad de recursos, permitieron realizar pruebas en distintos entornos:

- Configuración 1

Procesador: Intel Core i7-9850H

Memoria RAM: 32 GB

Tarjeta Gráfica: NVIDIA Quadro T2000

Sistema operativo: Windows 11 con Subsistema de Windows para Linux (WSL) configurado con Ubuntu 22.04.

Esta configuración se utilizó para aprovechar la capacidad de procesamiento paralelo de la GPU y para gestionar cargas intensivas de datos durante el entrenamiento del modelo. La alta cantidad de memoria RAM permitió manejar grandes conjuntos de datos y realizar operaciones de preprocesamiento sin cuellos de botella, mientras que la integración con WSL facilitó la utilización de herramientas y librerías orientadas a Linux.

- Configuración 2

Procesador: AMD Ryzen 5 3600

Memoria RAM: 16 GB

Sistema operativo: Windows 10

Esta configuración se usó en fases de desarrollo preliminares y pruebas de concepto, permitiendo validar la viabilidad del modelo en equipos de gama media. Aunque con menor capacidad de memoria y procesamiento gráfico, esta configuración demostró ser adecuada para tareas de preprocesamiento y entrenamiento inicial, permitiendo la iteración rápida y la identificación de parámetros clave.

Librerías

El proyecto se basa en el uso intensivo de librerías de Python, instaladas mediante el gestor de paquetes pip, que permiten desde la manipulación de datos hasta la construcción y evaluación del modelo de aprendizaje profundo.



- **Numpy**
(NumPy, 2022) lo define como una biblioteca de Python que proporciona un objeto de arreglo multidimensional y diversos objetos derivados (como arreglos enmascarados y matrices), así también, incluye una variedad de rutinas para realizar operaciones rápidas en arreglos como operaciones matemáticas, lógicas, manipulación de formas entre otros. Numpy se selecciona por su capacidad para manejar de manera eficiente datos en forma de arreglos y matrices multidimensionales. Su implementación en C permite ejecutar operaciones matemáticas y lógicas a alta velocidad, lo que es crucial para el procesamiento intensivo de datos y el manejo de grandes volúmenes de información. Esta eficiencia se complementa con una sintaxis intuitiva que simplifica la manipulación y transformación de datos, convirtiéndola en una herramienta indispensable para algoritmos de visión por computadora y análisis numérico en proyectos de inteligencia artificial.
- **Matplotlib**
(Matplotlib, s.f.) lo define como una biblioteca para generar visualizaciones estáticas, animados e interactivas en Python.
Se opta por Matplotlib porque ofrece una plataforma poderosa para la creación de gráficos y visualizaciones de datos. La posibilidad de generar representaciones estáticas, animadas e interactivas permite explorar y comunicar resultados de manera clara y detallada.
- **OpenCV**
(OpenCV, 2023) lo define como una librería de visión artificial y aprendizaje automático de código abierto, también conocida como Open Computer Vision. Cuenta con 2.500 algoritmos optimizados que abarcan tanto métodos clásicos como de última generación en visión por computadora y aprendizaje automático. Estos algoritmos permiten realizar diversas tareas, como detectar y reconocer rostros, identificar objetos, clasificar acciones humanas en videos, rastrear movimientos de cámaras, seguir objetos en movimiento, y extraer modelos 3D de objetos, entre otros. OpenCV ofrece interfaces para varios lenguajes de programación, incluidos C++, Python, Java y MATLAB. Además, es multiplataforma, lo que significa que es compatible con sistemas operativos como Windows, Linux, Android y Mac OS.



- Tensorflow

(TensorFlow, s.f.) es una biblioteca de código abierto desarrollada por Google Brain para el desarrollo y entrenamiento de modelos de aprendizaje automático y redes neuronales profundas.

Está escrita en C++ y Python, y permite la ejecución eficiente en CPU, GPU y TPU.

Proporciona un ecosistema de herramientas como TensorFlow Extended (TFX) para la implementación en producción y TensorFlow Lite para dispositivos móviles y embebidos. Además, permite la creación de modelos mediante la API de bajo nivel o a través de Keras, una interfaz de alto nivel integrada en TensorFlow para facilitar el desarrollo de redes neuronales.

Se selecciona por su capacidad para construir, entrenar y desplegar modelos de aprendizaje profundo de manera escalable. Su soporte para múltiples dispositivos (CPU, GPU y TPU) permite manejar grandes volúmenes de datos y acelerar el proceso de entrenamiento de modelos complejos.

Asimismo, el ecosistema que lo rodea —incluyendo herramientas para producción y aplicaciones móviles— facilita la transición del prototipo al producto final, asegurando que las soluciones desarrolladas puedan operarse en entornos reales con alta eficiencia y rendimiento.

- Keras

(Keras, s.f.) lo describe como una librería de código abierto que posee componentes de redes neuronales escrita en Python. Es capaz de ejecutarse sobre TensorFlow, Theano, PlaidML y otros.

El autor principal del mismo es Francois Chollet, ingeniero de Google que también escribió Xception, un modelo de red neuronal profunda.

Tiene bibliotecas de componentes de aprendizaje automático que incluyen objetivos, funciones de activación, optimizadores y soporte para redes neuronales recurrentes y convolucionales.

El objetivo de la red neuronal convolucional es que, a partir del entrenamiento de un conjunto de datos compuesto de imágenes de dígitos hechos a mano, la red neuronal sea capaz de identificar al dígito del conjunto de datos.

Keras es elegido debido a su interfaz de alto nivel, que simplifica enormemente el diseño y la experimentación con modelos de redes neuronales. Su facilidad de uso



permite a los desarrolladores concentrarse en la optimización y el ajuste de parámetros sin perder tiempo en detalles de implementación compleja. Esta abstracción y flexibilidad hacen que el desarrollo de modelos, tanto convolucionales como recurrentes, sea más rápido y accesible, lo cual es esencial para proyectos que requieren iteraciones rápidas y prototipos funcionales en el ámbito de la inteligencia artificial.

- Tkinter

Para (Python, 2023) es la interfaz estándar de Python para el kit de herramientas GUI de Tcl/Tk. El paquete tkinter es una fina capa orientada a objetos encima de Tcl/Tk. Es un conjunto de envoltorios que implementan los widgets Tk como clases de Python. Las principales virtudes son que es rápido y que generalmente viene incluido con Python.

- Subsistema de Linux para Windows (WSL) - Ubuntu 22.04

(WSL, s.f.) es una característica de Windows que permite ejecutar un entorno de Linux dentro de Windows sin necesidad de una máquina virtual o arranque dual. Proporciona compatibilidad con la mayoría de herramientas de línea de comandos, utilidades y aplicaciones de Linux.

Ubuntu 22.04, disponible en WSL, es una de las distribuciones más utilizadas, ofreciendo un entorno completo de desarrollo con acceso al sistema de archivos de Windows y compatibilidad con herramientas como Bash, Git y Python. WSL admite dos versiones: WSL 1, con mejor integración con el sistema de archivos de Windows, y WSL 2, que ofrece un kernel de Linux real con mayor rendimiento y compatibilidad con Docker y otras herramientas avanzadas. El uso de WSL, especialmente con Ubuntu 22.04, se justifica por la combinación de la comodidad del entorno Windows con la potencia y flexibilidad de Linux. Este subsistema permite ejecutar herramientas y scripts nativos de Linux sin abandonar el entorno Windows, facilitando la integración de utilidades de línea de comandos y aplicaciones de desarrollo avanzadas.

La capacidad de trabajar en un entorno híbrido optimiza el flujo de trabajo, garantizando compatibilidad, eficiencia y la posibilidad de aprovechar herramientas propias de ambos sistemas operativos en un único entorno.



- **Visualkeras**

(Gavrikov, 2021) lo determina como una biblioteca de Python de código abierto que facilita la visualización de la arquitectura de redes neuronales de Keras. Esta herramienta permite crear diagramas intuitivos de redes neuronales, lo que hace más sencillo comprender y depurar los modelos. En estos diagramas, las formas y colores indican el tipo de capa o proceso. A continuación, se describen las leyendas predeterminadas adoptadas por la biblioteca:

Bloques amarillos: representan las capas convolucionales.

Bloques rosas: representan las capas de submuestreo.

Bloques verdes: representan la técnica de regularización.

Bloques celestes: representan el proceso de aplanamiento.

Bloques azules: representan las capas totalmente conectadas.

En la siguiente figura se muestra un ejemplo de un modelo de red neuronal convolucional visualizado con Visualkeras, donde se puede apreciar la leyenda predeterminada de la biblioteca.

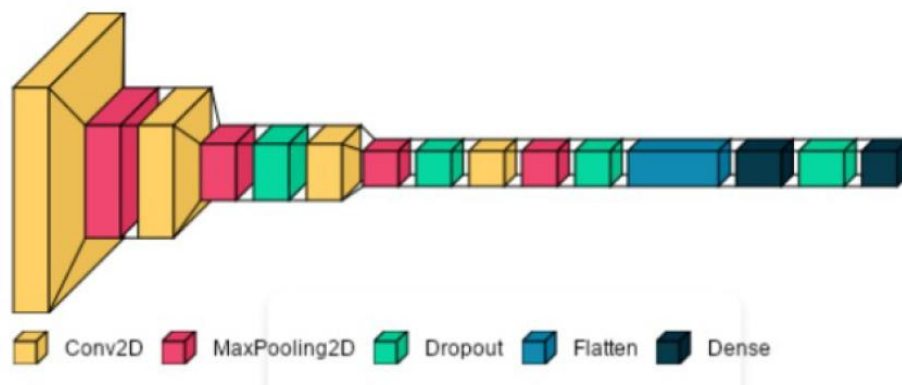


Figura 1: Ejemplo de la visualización de un modelo con Visualkeras.
(<https://www.analyticsvidhya.com/blog/2022/03/visualize-deep-learning-models-using-visualkeras/>)



Visuallkeras se elige porque simplifica la visualización de la arquitectura de redes neuronales, permitiendo representar de forma gráfica y codificada por colores cada capa y proceso. Esta representación intuitiva facilita la identificación de errores, la comprensión de la estructura del modelo y la comunicación de los diseños entre desarrolladores y equipos técnicos. La claridad en la visualización contribuye a una mejor depuración y optimización del modelo, haciendo que el proceso de ajuste y validación sea más accesible y transparente.

- Bases de datos de imágenes

Para este proyecto se utilizó un conjunto de datos obtenido de datadryad.org, una base de datos abierta que proporciona imágenes de alta calidad para análisis científico. El conjunto original incluía 1,632 imágenes de hojas de soja negativas y 1,598 imágenes de hojas positivas. Tras un proceso de selección y clasificación manual para garantizar la calidad y relevancia de las imágenes, se obtuvieron 2000 imágenes de hojas para entrenamiento, 1000 negativas y 1000 positivas y 598 imágenes de hojas para validación 300 positivas y 298 negativas. Las imágenes fueron preprocesadas para eliminar el fondo, optimizando así el entrenamiento del modelo de aprendizaje profundo.

Resultados y Discusión

- Determinar las características visuales del “Tizón de la hoja” en la soja a través de resultados de investigaciones existentes y observación directa.

El desempeño alcanzado por el modelo es resultado de una cadena metodológica rigurosamente diseñada, que combinó fundamentos científicos con decisiones técnicas basadas en buenas prácticas del aprendizaje profundo. A lo largo del desarrollo del sistema, se establecieron tres pilares fundamentales para la obtención de resultados consistentes:

Preprocesamiento exhaustivo de datos

La base de datos original fue sometida a una serie de etapas de depuración y mejora. Se eliminaron imágenes corruptas, se validaron manualmente las etiquetas y se aplicaron técnicas de aumento de datos (data augmentation) como rotaciones, desplazamientos, zooms, volteo horizontal, variaciones de brillo y eliminación de fondo. Esto permitió ampliar la diversidad del conjunto de entrenamiento sin necesidad de adquirir nuevas



imágenes, mejorando así la capacidad del modelo para generalizar frente a nuevas condiciones visuales.

Diseño de arquitectura CNN adecuada y parametrización progresiva

A través de un proceso iterativo de prueba y error, se ajustaron hiperparámetros como el número de épocas, la tasa de aprendizaje, la función de activación y la estructura de capas, siempre con una mirada puesta en la relación entre el rendimiento y la capacidad de generalización. Se incorporaron mecanismos de regularización y callbacks que ayudaron a mitigar riesgos como el sobreajuste (overfitting), lo que contribuyó a estabilizar los resultados en las últimas fases de entrenamiento.

Separación anticipada de un conjunto de pruebas final

Como parte de una medida clave para preservar la objetividad de los resultados, se apartó desde el inicio del proyecto un conjunto de 100 imágenes originales, completamente nuevas y sin exposición previa al modelo. Estas imágenes fueron seleccionadas de forma aleatoria pero controlada, asegurando una representación equitativa de hojas sanas y enfermas. Este conjunto fue utilizado exclusivamente en la etapa final de validación y no fue empleado en ningún momento del proceso de entrenamiento ni de ajuste del modelo. Esta práctica, ampliamente recomendada en el ámbito de la inteligencia artificial aplicada, asegura que el rendimiento reportado refleje la verdadera capacidad del modelo para enfrentarse a datos no vistos, tal como ocurriría en un entorno real de producción.

- Emplear modelos de redes neuronales convolucionales para entrenar el algoritmo de detección de la enfermedad.

El algoritmo desarrollado fue entrenado desde cero, diseñado específicamente para abordar la tarea de clasificación de hojas negativas y positivas para tizón de hoja.

Se construyó una arquitectura personalizada que incorpora capas ajustadas a los requerimientos del problema. Durante las primeras fases de entrenamiento, se realizaron ajustes estratégicos para maximizar su capacidad de aprendizaje y prevenir problemas como el sobre ajuste. El desempeño del modelo fue validado por el ingeniero agrónomo Carlos Prendeski quien permitió evaluar su capacidad para generalizar correctamente. Además, se monitoreó su progreso durante el entrenamiento, utilizando gráficos y métricas clave, para identificar oportunidades de mejora y garantizar su efectividad en la tarea específica.



Alice Nathalia Colman Barboza, también egresada de Ingeniería Informática de la UC Itapúa, presento como proyecto final la detección de roya asiática (*Phakopsora pachyrhizi*) en cultivos de soja mediante visión por computador. Este trabajo se desarrolló en colaboración con el Instituto Paraguayo de Tecnología Agrícola (IPTA) en Capitán Miranda, Itapúa. La herramienta desarrollada constaba de cinco módulos: preprocesamiento, segmentación, identificación de manchas, extracción de características y clasificación. Las imágenes, capturadas con un dispositivo móvil en un entorno controlado, se dividieron aleatoriamente, utilizando un 70% para entrenamiento y un 30% para pruebas. Los resultados finales arrojaron una precisión de 0.023470, exactitud de 0.922622, sensibilidad de 0.124670 y especificidad de 0.927575. (Colman Barboza, 2018).

Por su parte, María Mercedes Greco González, egresada de Ingeniería Informática de la UC Itapúa, desarrollo como trabajo final un software de visión por computador enfocado en la identificación de la enfermedad "mancha ojo de rana" (*Cercospora sojina*) en cultivos de soja. Su propuesta consistió en crear una herramienta que facilitara la visualización de esta patología, frecuente en dichos cultivos, mediante el uso de imágenes. El proceso metodológico incluyó la segmentación de las hojas en el espacio de color RGB y la identificación de las manchas en LAB, culminando con la clasificación a través del algoritmo KNN. (Greco González, 2019).

La investigación desarrollada por Souza (inteligencia artificial utilizada na determinacao longitudinal da cultura da soja) aplico técnicas de inteligencia artificial (IA) para la evaluación longitudinal de la distribución de plantas de soja en el campo. A través de procesamiento de imágenes y modelos de aprendizaje automático como el algoritmo Random Forest, el estudio logro estimar de manera precisa la densidad y disposición de las plantas a lo largo de las filas de siembra.

En una tesis doctoral el alumno Martin Montalvo Martínez de la Facultad de Informática - Universidad Complutense de Madrid, (Técnicas de visión artificial para la segmentación y detección de líneas de cultivo en imágenes agrícolas) presenta dos Sistemas Expertos Automáticos (SEA), basados en técnicas de visión por computador, cuya principal finalidad es la detección de líneas de cultivo y rodales de malas hierba en campos de maíz bajo condiciones ambientales de iluminación adversas y altamente cambiantes, además de abordar sendas problemáticas específicas que se presentan en estos campos de cultivo.



El sistema desarrollado alcanza una precisión general del 94%, lo cual representa un rendimiento altamente satisfactorio considerando la complejidad inherente al problema de clasificación visual en entornos naturales, donde intervienen múltiples factores que pueden alterar las características visuales de una hoja, como iluminación, variabilidad genética, presencia de otras patologías, o condiciones ambientales adversas.

El modelo evidencio además una tasa de falsos positivos del 6% y una tasa de falsos negativos del 2%. Estos valores indican que el sistema, además de ser preciso, es conservador en su clasificación, con un bajo nivel de riesgo de pasar por alto hojas realmente enfermas, lo cual es crucial en aplicaciones de campo donde una omisión puede derivar en una diseminación de la enfermedad a gran escala. Esta configuración es coherente con los objetivos del proyecto, ya que en un contexto agrícola, especialmente en zonas de producción intensiva, es preferible un sistema con mayor sensibilidad (aunque se produzcan algunos falsos positivos), a uno que minimice estos últimos a costa de perder casos reales de infección.

Conclusiones

La presente investigación ha permitido demostrar la viabilidad técnica y practica de aplicar redes neuronales convolucionales al diagnóstico temprano de enfermedades en cultivos agrícolas, en particular el Tizón de la soja. A lo largo del desarrollo del proyecto, se abordó de manera integral el problema desde una perspectiva científica, técnica y operativa, consolidando un sistema que integra procesamiento digital de imágenes, aprendizaje profundo e interacción con el usuario en un entorno intuitivo.

El sistema desarrollado alcanza un nivel de precisión elevado en la detección de la enfermedad, superando el 90% en pruebas finales con datos no utilizados en fases de entrenamiento o validación interna. Esta cifra no solo evidencia el rendimiento del modelo en términos estadísticos, sino también su capacidad de generalización, aspecto esencial en aplicaciones reales donde la variabilidad de condiciones ambientales, visuales y fisiológicas puede afectar el desempeño de herramientas automatizadas.

Durante el proceso se hizo especial énfasis en la calidad del conjunto de datos, aplicando un preprocesamiento meticuloso que incluyo la depuración de imágenes defectuosas, la normalización de parámetros visuales y la aplicación de técnicas de aumento de datos. Este enfoque fue clave para garantizar un modelo más robusto y tolerante a variaciones propias del entorno agrícola.



Asimismo, la implementación de una interfaz gráfica de usuario funcional y accesible representa un valor agregado importante, ya que facilita la adopción de esta tecnología por parte de actores no técnicos del sector agrícola. Esta interfaz permite analizar imágenes de manera rápida y sencilla, emitir alertas visuales cuando se detectan riesgos sanitarios y generar reportes exportables, integrando así el sistema en un flujo de trabajo compatible con las necesidades del usuario final.

El uso de un conjunto de imágenes originales, apartadas desde el inicio para validación final, fue una medida metodológica clave para asegurar la objetividad de los resultados. Esta práctica refuerza la confianza en la solución propuesta, validando su efectividad en condiciones similares a las que enfrentaría en su uso real.

En síntesis, el sistema desarrollado no solo representa un aporte técnico al campo de la inteligencia artificial aplicada a la agricultura, sino también una herramienta concreta con potencial de impacto en la sanidad vegetal, la productividad del cultivo y la eficiencia en la toma de decisiones agronómicas.

Bibliografía

- Facultad de Ciencias y Tecnología. (2019). Software desarrollado por egresada de la UCI identifica enfermedad más común en la soja.
<https://www.uci.edu.py/uciweb/public/vernoticia49>
- Souza, V. (2021). Morfología y fisiología foliar en cultivos de soja: Adaptaciones y respuestas al ambiente. *Ciencia y Agroindustria*, 38(1), 55-73.
- Montalvo Martinez, Martin. (2015). Técnicas de visión artificial para la segmentación y detección de líneas de cultivo en imágenes agrícolas. Recuperado el 27 de noviembre de 2024, de
<https://hdl.handle.net/20.500.14352/25987>
- Bevers, Noah, Sikora, Edward J, Hardy, Nate . (2022). *Pictures of diseased soybean leaves by category captured in field and with controlled backgrounds: Auburn soybean disease image dataset (ASDID)* Dryad. Recuperado el 28 de septiembre de 2024, de
<https://datadryad.org/dataset/doi:10.5061/dryad.41ns1rnj3>
- Python. (2023). Numpy. <https://pypi.org/project/numpy/>



VIII CONGRESO EN
**CIENCIA, CULTURA
Y SOCIEDAD**

"La Universidad y la Inteligencia
Artificial al Servicio de la Sociedad"

XII SIMPOSIO IBEROAMERICANO
DE COOPERACIÓN PARA EL DESARROLLO
Y LA INTEGRACIÓN REGIONAL

"El Territorio Trinacional Misionero en el
Contexto de las Transformaciones Globales"

4 y 5 de setiembre 2025



- Matplotlib. (2023). <https://matplotlib.org/>
- Python. (2023). OpenCV. <https://opencv.org/>
- Tensorflow y Keras. (2023). Keras.
<https://www.tensorflow.org/guide/keras?hl=es-419>
- Python. (2023). Tkinter. <https://docs.python.org/es/3.13/library/tkinter.html>
- Ubuntu. (2023). WSL 2 Ubuntu 22.04. <https://ubuntu.com/desktop/wsl>
- Gavrikov, P. (2020). visualkeras. *GitHub*. Available online <https://github.com/paulgavrikov/visualkeras>.