Caminante, no hay camino.
Se hace camino al andar.

Antonio Machado

Agradecimientos

A mi familia.

Índice general

Índice de figuras

Índice de tablas

Resumen

La agricultura de precisión, impulsada por tecnologías emergentes como la inteligencia artificial (IA), ha transformado la forma en que los agricultores gestionan sus cultivos, permitiéndoles tomar decisiones más informadas y optimizar el uso de recursos como el agua, los fertilizantes y la tierra. Este Trabajo de Fin de Máster se enmarca en este contexto, proponiendo una solución basada en redes neuronales convolucionales (CNN) para la identificación automática de enfermedades en hojas de tomate, una de las principales preocupaciones en la producción agrícola.

El modelo desarrollado utiliza MobileNet, una arquitectura ligera y eficiente, ideal para su implementación en dispositivos móviles, permitiendo su uso en campo con una alta precisión del $95{,}63\,\%$. El entrenamiento del modelo se llevó a cabo en Google Colaboratory, aprovechando técnicas avanzadas de aumento de datos para mejorar la capacidad de generalización del modelo, especialmente en clases con menos representaciones. Además, se utilizó Keras Tuner para optimizar los hiperparámetros clave, como la tasa de aprendizaje y el tamaño del lote, lo que resultó en un rendimiento óptimo durante el proceso de entrenamiento.

A lo largo del desarrollo, se observaron algunos desafíos en la clasificación de clases minoritarias, lo que subraya la necesidad de un balance adecuado de los datos. El modelo fue evaluado utilizando herramientas como TensorBoard, que permitieron monitorear de forma detallada la evolución del error y la precisión durante las distintas fases del entrenamiento.

El modelo final fue desplegado en una aplicación web interactiva a través de Streamlit, proporcionando una solución práctica para los agricultores. Esta herramienta permite cargar imágenes de hojas de tomate y obtener diagnósticos en tiempo real sobre posibles enfermedades, facilitando así la toma de decisiones rápidas y efectivas en la gestión de cultivos.

Este trabajo demuestra la efectividad de las redes neuronales en la agricultura de precisión y ofrece una solución escalable y accesible que puede mejorar significativamente la detección temprana de enfermedades, optimizando los recursos y reduciendo las pérdidas en la producción agrícola.

Abstract

Precision agriculture, driven by emerging technologies like artificial intelligence (AI), has revolutionized how farmers manage their crops, enabling them to make more informed decisions and optimize the use of resources such as water, fertilizers, and land. This Master's Thesis aligns with this context by proposing a solution based on convolutional neural networks (CNN) for the automatic identification of diseases in tomato leaves, a major concern in agricultural production.

The developed model leverages MobileNet, a lightweight and efficient architecture, ideal for deployment on mobile devices, making it suitable for field use with a high accuracy of 95,63 %. The model was trained using Google Colaboratory, utilizing advanced data augmentation techniques to improve the model's generalization ability, particularly for underrepresented classes. Additionally, Keras Tuner was used to optimize key hyperparameters such as learning rate and batch size, resulting in optimal performance during the training process.

Throughout the development, some challenges were encountered in classifying minority classes, highlighting the importance of proper data balancing. The model was evaluated using tools like TensorBoard, allowing for detailed monitoring of error and accuracy evolution across different training phases.

The final model was deployed through an interactive web application using Streamlit, providing a practical solution for farmers. This tool allows users to upload images of tomato leaves and receive real-time diagnoses of potential diseases, thus facilitating quick and effective decision-making in crop management.

This work demonstrates the effectiveness of neural networks in precision agriculture and offers a scalable, accessible solution that can significantly enhance early disease detection, optimize resources, and reduce losses in agricultural production.

Introducción 1

El crecimiento poblacional y los problemas asociados al cambio climático, tales como la reducción de la tierra cultivable y el difícil acceso a recurso hídrico, han generado gran preocupación en la población mundial en los últimos años. Estudios recientes de la Organización de las Naciones Unidas para la Alimentación y la Agricultura (FAO, 2017) han arrojado proyecciones alarmantes: la población mundial alcanzará los 10 mil millones de habitantes para el año 2050 y esto representará una demanda adicional de más del 50 % de los alimentos que se consumen hoy en día. Aunque las inversiones e innovaciones tecnológicas están impulsando la productividad, el crecimiento de los rendimientos se ha desacelerado a tasas demasiado bajas. Las pérdidas y el desperdicio de alimentos ocupan una proporción significativa de la producción agrícola, y reducirlos disminuiría la necesidad de aumentar la producción. Sin embargo, la necesaria aceleración del crecimiento de la productividad se ve obstaculizada por la degradación de los recursos naturales, la pérdida de biodiversidad y la propagación de plagas y enfermedades transfronterizas de plantas y animales, algunas de las cuales se están volviendo resistentes a los antimicrobianos.

Ante esta situación, el sector agroalimentario ha estado trabajando en mejorar los sistemas de producción de manera que sea posible incrementar la producción optimizando los procesos productivos. En el campo de la agricultura, desde hace varias décadas, se ha trabajado en la **agricultura de precisión**, método que busca producir más alimentos con menos recursos de agua y fertilizantes.

Debido a estos factores, resulta estratégico entonces que el sector tecnológico apoye al sector productivo en el desarrollo de herramientas que permitan a los agricultores alcanzar altos rendimientos para poder lidiar con los problemas proyectados a futuro. Una forma de lograr esto es a través de la implementación de técnicas de **inteligencia artificial** en los procesos de toma de decisiones.

En este contexto, la implementación de sistemas de apoyo a la toma de decisiones (DSS) que utilicen técnicas de inteligencia artificial (IA) en la agricultura de precisión emerge como una solución prometedora. Estos sistemas permiten optimizar el uso de recursos y mejorar la productividad agrícola mediante la automatización y el análisis avanzado de datos obtenidos de diversas fuentes, como sensores y redes de sensores inalámbricos (WSN). Además, la

capacidad de estos sistemas para ofrecer recomendaciones precisas y en tiempo real a los agricultores puede ayudar a mitigar los efectos adversos del cambio climático y garantizar la sostenibilidad de los sistemas agrícolas.

En este Trabajo Fin de Máster (TFM) no solo se ha centrado en identificar las técnicas más efectivas, si no también en la implementación de IA en la agricultura de precisión para un cultivo específico. También abordará la necesidad de desarrollar sistemas interpretables que no solo ofrezcan resultados precisos, sino que también proporcionen explicaciones claras y comprensibles a los usuarios finales.

1.1. Estructura

Este informe se ha organizado en cuatro bloques principales. A continuación, se describe brevemente en qué consiste cada uno y se nombra cuál será su contenido.

Primer bloque. Se describirá superficialmente en qué consiste este trabajo, cuál ha sido la motivación de su elección y su justificación. También se hará un breve estudio del estado del arte actual, nombrando y viendo en qué consisten algunos de los trabajos más recientes en este campo para poder ver desde donde partir y como proceder. Para finalizar veremos que objetivos se quieren lograr en este trabajo.

Segundo bloque. Es la parte correspondiente al marco teórico. En él se analizará y explicará las técnicas y modelos empleados en la implementación del proyecto.

Tercer bloque. Corresponde a la explicación general tanto del conjunto de datos usado como la implementación. Se explicará superficialmente el código, la tecnología, componentes y librerías se han utilizado para crear y entrenar los modelos.

Cuarto bloque. Se mostrará y analizarán los resultados obtenidos del modelo final y se redactaran soluciones finales, las diferentes equivocaciones cometidas y una recomendación para futuras ampliaciones e investigaciones relacionadas.

1.2. Justificación y elección del tema

La motivación de este proyecto viene dada principalmente por dos razones. La primera tiene que ver con el contexto profesional, ya que desde hace unos años empecé a trabajar en el departamento de analítica en el Grupo Hispatec, S.L. empresa dedicada a la digitalización del sector Agroalimentario. Durante mi trabajo he tenido la oportunidad de conocer el mundo agrícola así como su proyección futura digital. Todo esto hizo preguntarme las aplicaciones de la inteligencia artificial en todo este campo.

La segunda razón se debe al contexto local. La agricultura ha sido el principal motor económico en la provincia de Almería desde los años 60. Desde que tengo uso de razón siempre he estado influenciado por el mundo agrícola. Debido al gigantesco volumen de productos hortofrutícolas generados y a la enorme cantidad de exportaciones a Europa que se llevan a cabo desde esta región, Almería es conocida como la «Huerta de Europa» (Figura 3.1).

Figura 1.1: ***Imagen de satélite del Campo de Dalías y su «Mar de plástico» formado por invernaderos.***

Adicionalmente, la asignatura de Aprendizaje Profundo despertó en mi la necesidad de crear mis propias redes neuronales convolucionales. Es por ello que, como veremos más adelante, nos hemos decantado por este tipo de arquitectura para el desarrollo de un modelo de inteligencia artificial.

Todo estos factores han contribuido a la elección de este tema y ser capaz de aportar mi granito de arena en este nuevo campo.

1.3. Justificación del título

El título de este trabajo captura de manera clara la esencia del TFM, indicando un enfoque en la aplicación de tecnologías avanzadas para mejorar la eficiencia y sostenibilidad en la agricultura de precisión. Este título no solo refleja los componentes técnicos y científicos del trabajo, sino que también enfatiza la importancia de la sostenibilidad en la agricultura moderna.

Una parte clave de este título es el concepto de *gestión sostenible de cultivos.* Este término gira entorno a la sostenibilidad ambiental ya que dentro de las prácticas se busca que estas sean respetuosas con el medio ambiente incluyendo la reducción del uso de pesticidas y fertilizantes, la conservación del agua y la protección de la biodiversidad. Esta enfoque sostenible permitirá también al agricultor reducir costes y aumentar la resiliencia de sus sistemas de cultivos.

1.4. Estado del arte

Históricamente la revolución agrícola comenzó cuando los seres humanos comenzaron a practicar la agricultura en lugar de depender únicamente de la caza y la recolección. Este cambio permitió la creación de asentamientos permanentes y el desarrollo de las primeras civilizaciones. Incluyó la domesticación de plantas y animales, y el uso de herramientas básicas de cultivo. Posteriormente se introdujo técnicas como la rotación de cultivos de tres campos, que mejoró la fertilidad del suelo, y el uso de nuevos equipos agrícolas. Estas innovaciones llevaron a un aumento en la producción de alimentos y una mayor disponibilidad de alimentos para la población. Durante la **Revolución Verde** (década de 1940) se extendió la adopción de variedades de alto rendimiento de cultivos básicos como el trigo y el arroz, el uso extensivo de pesticidas y fertilizantes químicos, y la implementación de técnicas de riego avanzadas. Esta revolución resultó en aumentos significativos en la producción agrícola en muchas regiones del mundo, ayudando a reducir el hambre en algunos países.

Actualmente, la tecnología se usa principalmente para recolección de datos y transmisión, ha generado una fuerte relación entre el productor, los proveedores, los intermediarios y el consumidor final lo que permite hacer seguimiento a materias primas que permitan cultivar productos de calidad, transporte seguro de los alimentos y entrega de productos idóneos al consumidor, a través de GPS, así como otras tecnologías utilizadas que permite el control del sistema productivo y monitoreo de los rendimientos en tiempo real haciendo uso de los sensores remotos y tecnologías inalámbricas. Estos desarrollos unido con los drones permiten hacer monitoreo en tiempo real con vuelos no tripulados que además de reducir riesgos y costos, son eficaces y precisos para generar información que permita alimentar bases de datos para la toma de decisiones en campo.

Una vez que tenemos los datos, podemos plantearnos como extraer valor de ellos. Para ello, hemos investigado en diferentes áreas de informática, computación, ingeniería y agronomía usando palabras claves como IA en agricultura, agricultura de precisión, agricultura sostenible...

Hemos encontrado un total de 35 estudios relevantes que podemos clasificar los resultados en las siguientes subsecciones.

1.4.1. Técnicas de inteligencia artificial utilizadas

Categoría de la técnica de inteligencia artificial usada	Total estudios que la reportan
Machine learning	19
Logic based	14
Embodied intelligence	6
Search and optimization	4
Knowledge based	3
Probabilistic methods	5

Tabla 1.1: ***Técnicas de inteligencia artificial utilizadas.***

Tiene sentido que el aprendizaje automático junto con la lógica difusa sean las técnicas más utilizada ya que el análisis predictivo permite predecir el comportamiento del crecimiento de las plantas o de las condiciones de los invernaderos, para tomar medidas de control y optimizar el crecimiento y la producción; y en el segundo caso, la lógica difusa, tal y como vimos en la asignatura de aprendizaje aproximado, permite combinar variables de entrada para producir varias variables de salida.

1.4.2. Caracterización de las variables de entrada y salida en los sistema de IA

Tipo	Variable	Total
Clima	Temperatura	30
Clima	Humedad relativa	24
Clima	Radiación neta	15
Clima	Intensidad de la luz	11
Clima	Concentración de CO2	11
Suelo	Humedad del sustrato	11
Suelo	Temperatura	10
Suelo	Conductividad eléctrica	6
Suelo	Acidez (pH)	6
Planta	Crecimiento de cultivo	5
Planta	Temperatura de la hoja	4

Tabla 1.2: ***Caracterización de las variables de entrada en los sistema de IA***

Se puede observar que la mayoría de inputs a los sistemas de IA son variables climatológicas ya que se tratan de valores más fáciles de medir respecto a otras como puede ser la

temperatura de la hoja de las plantas.

En la siguiente tabla se detalla también las salidas reportadas:

Tipo de salida	Total
Visualización de los datos	19
Instrucciones a actuadores	18
Recomendaciones al agricultor	6
Otro tipo de salida	3
No definida	5

Tabla 1.3: ***Variables de salida en sistemas de IA.***

La visualización de datos está en primer lugar debido a la automatización de procesos y al uso de dispositivos móviles. Es importante destacar que son pocos los autores que muestran recomendaciones y/o alertas a los agricultores para que estos realicen alguna intervención en los procesos de cultivo.

1.4.3. Evaluación de los sistemas que utilizan IA

A continuación, se muestran los medios de evaluación utilizados en los estudios:

Tipo de evaluación	Total
Evaluación con cultivo	13
Evaluación en laboratorio	9
Evaluación de precisión	6
Evaluación de expertos	3
Evaluación de rendimiento	3
Otro tipo de evaluación	2
No definida	11

Tabla 1.4: ***.Evaluación de los sistemas que utilizan técnicas de IA***

Además, en los casos que se reportó la evaluación mediante un cultivo, se identificaron los tipos de cultivos: hortalizas (54,15 %), frutas (28,71 %), hidropónicos (8,57 %), flores (5,71 %) y ornamental (2,86 %)

Con todo esto, se recomienda explorar las técnicas reportadas por los autores para medir variables físicas de plantas e identificar la utilidad de tales variables en la obtención de resultados. Asimismo, se podría analizar de manera comparativa técnicas de aprendizaje de máquina para identificar aquella que mayor aporta mayor beneficio en la mejora del proceso de toma de decisiones y optimización de la producción para una agricultura más sostenible en el futuro.

1.5. Descripción de los objetivos

El presente trabajo consistirá en crear un modelo capaz de analizar una imagen para detectar una posible enfermedad en las hojas de las plantas de tomate. Además de identificar de que enfermedad se trata se dará algunos consejos al agricultor para contrarrestar la pérdida de rendimiento del cultivo así como la causa de dicha enfermedad. Nos hemos inclinado por este cultivo debido a que el tomate es el rey del cultivo ecológico en los invernaderos de Almería con un $34\,\%$ de la producción (Redactor (2023)), además de ser un cultivo con una gran cantidad de datos accesibles. Por otro lado, contrastáremos el modelo desarrollado con los presentes en la literatura actual.

Puesto que para alcanzar dicha meta hay varias formas de lograrlo, teniendo varios métodos, algoritmos y técnicas actuales que pueden solucionar el problema, se ha decidido centrarse solo en aquellas que para dicha tarea deben funcionar mejor, las basadas en aprendizaje profundo mediante el uso de redes neuronales. En particular, haremos uso de la red neuronal convolucional por su gran capacidad para procesar imágenes.

Sin embargo, debido a que dentro del campo en aprendizaje profundo (redes con varias capas ocultas) también hay muchos tipos de redes neuronales y arquitecturas diferentes, se seleccionarán solo aquellas que no sean muy costas computacionalmente y que se crea que puedan dar resultados notables. No se seleccionarán redes muy potentes, aunque tengan un potencial mucho mayor debido a la limitación del equipo técnico para el entrenado de los modelos. Más adelante, en la sección 4.1, comentaremos sobre las tecnologías utilizadas para el desarrollo de este trabajo.

Otros objetivos secundarios de este proyecto son:

1. Usar herramientas para el seguimiento de la evaluación del modelo.

2. Desarrollo de una interfaz sencilla para hacer uso del modelo.

Cronograma y metodología

2

Para que un proyecto no este destinado al fracaso es necesario tener una buena organización y planificación. Para conseguir las dos características anteriores debemos tener en cuenta los hitos de la asignatura *Trabajo Fin de Máster.* Dichos hitos consisten en aportaciones periódicas que el alumno deberá entregar, mediante el aula virtual, a su tutor durante el desarrollo del proyecto. En la siguiente lista veremos el objetivo que tiene cada hito así como su periodo de entrega:

- **Hito 1:** Consiste en entregar una borrador preliminar con la introducción, estado del arte y marco teórico. Se debe entregar el documento con 7 semanas de antelación a la fecha del depósito final y supone un $5\,\%$ de la nota final del TFM.

- **Hito 2:** Consiste en la entrega de un borrador preliminar con la entrega de los métodos aplicados, material involucrado y resultados preliminares. Se debe entregar el documento con 5 semanas de antelación a la fecha del depósito final y supone un $5\,\%$ de la nota final del TFM.

- **Hito 3** Consiste en la entrega del borrador con la versión definitiva de la memoria. Es importante en este hito haber mejorado y/o corregido los hito anteriores. Se debe entregar el documento con 3 semanas de antelación a la fecha del depósito final. Este hito, junto con el criterio del tutor, supondrá un $20\,\%$ de la nota final del TFM.

Todas las fechas anteriormente expuestas son flexibles y es trabajo del alumno acordar con su tutor otras fechas si se requiere. Por otro lado, debemos tener en cuenta que la fecha del depósito de este trabajo corresponde con el 14 de octubre de 2024.

Una vez que tenemos estas metas en cuenta, podemos empezar la planificación. Existen varias metodologías que nos ayudan a planificar y organizar nuestro trabajo. Una de las más frecuentes debido a su flexibilidad, es la metodología AGIL donde podemos encontrar marcos de trabajo como el SCRUM. Esta forma de trabajo permite realizar el trabajo en equipo en pequeñas partes a la vez, con experimentación continua y bucles de retroalimentación a lo largo de la vida del proyecto. Además, ayuda a los usuarios y a los equipos a generar valor de forma incremental y colaborativa. Dado que en este trabajo no se esta realizando en equipo, carece de sentido aplicar este tipo de metodologías. No obstante, se han usado algunas de

sus características para ayudar con la planificación.

Para mantener un orden y poder cumplir con las fechas de entrega, se realizó un diagrama de Gantt en el que se indicaron los grandes apartados del trabajo y cuanto tiempo se debería dedicar a estos. Para cada bloque, aunque esta vez sin fechas estrictas, dado que a priori no se sabía cuánto iba a durar cada sección del proyecto, se definieron los hitos a desarrollar. Durante la realización de este trabajo, el diagrama ha ido variando quedando finalmente tal y como muestra en la figura 5.5. Esta refleja cómo se distribuyen las tareas a lo largo del tiempo para gestionar adecuadamente la planificación, desarrollo y redacción del TFM.

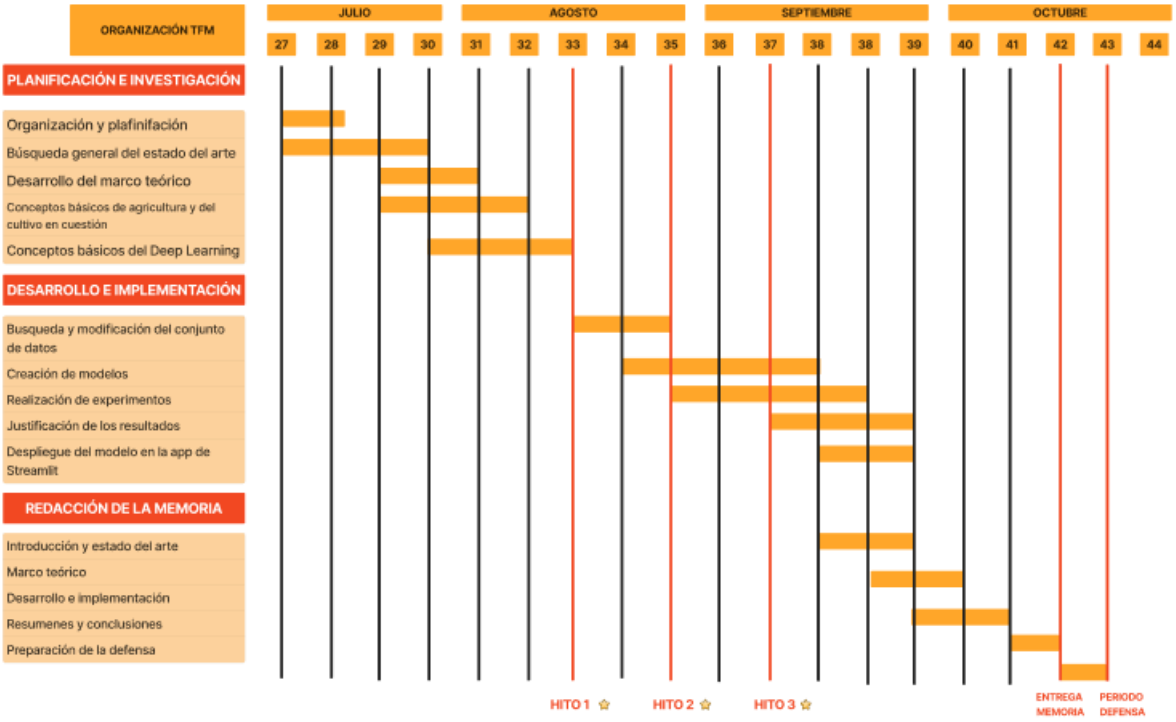

Figura 2.1: ***Diagrama de Grantt. Elaboración propia generada con Figma.***

En el diagrama anterior, las líneas rojas que denotan el periodo de entrega de los hitos comentados en la lista anterior, así como la entrega de la memoria y la defensa del trabajo realizado. Véase también que unos de los conceptos que mas tiempo ha requerido es la parte de la creación de los modelos y la realización de los experimentos. Esto se debe, como veremos más adelante, a las pruebas de arquitecturas y experimentos. En términos generales, el éxito de este tipo de planificaciones vendrá dado en gran medida por las técnicas aplicadas en Deep Learning y del grado de dificultad que puede llegar a tener el modelo para adaptarse a los datos.

Finalmente, para el control de versiones se ha utilizado GitHub, tanto para el código de los experimentos como para la memoria. En el caso de la memoria, al haber sido desarrollada en Overleaf, una plataforma en línea para LaTeX, se ha subido a GitHub después de cada

entrega correspondiente a los distintos hitos del proyecto.

Marco teórico

3

Este capítulo presenta los conceptos teóricos clave que sustentan el desarrollo de este proyecto. Se abordan temas relacionados con la agricultura de precisión, un que busca optimizar la productividad agrícola mediante el uso de tecnologías. Dentro de este marco, se explora la situación en España. Además, se discuten los métodos actuales de detección de enfermedades en las plantas, un aspecto crucial para la protección de los cultivos.

A continuación, se examina la inteligencia artificial (IA) y su impacto en la agricultura. Se destaca cómo las aplicaciones de la IA están revolucionando el sector agroalimentario, permitiendo una toma de decisiones más precisa y eficiente. Por último, se profundiza en las redes neuronales convolucionales, un tipo específico de IA que se utiliza ampliamente en el procesamiento de imágenes, con especial énfasis en su relevancia para la identificación de enfermedades en plantas.

3.1. Agricultura de precisión

Las técnicas agrícolas han evolucionado con el tiempo, y los enfoques tradicionales y modernos están a la vanguardia de las prácticas agrícolas. Comprender las diferencias entre estos dos métodos es fundamental para tomar decisiones informadas sobre la sostenibilidad, la productividad y el impacto ambiental. En la siguiente tabla comparativa, exploramos los contrastes clave entre la agricultura tradicional y la de precisión.

Según Delgado et al. (2011), la agricultura de precisión podría describirse como un conjunto de sistemas de apoyo a la decisión que buscan gestionar la variabilidad espacial y temporal, con el fin de maximizar el rendimiento, la calidad y el beneficio de los cultivos, así como mejorar la eficiencia de los insumos y los resultados ambientales minimizando el daño ambiental en cada unidad de tierra (tanto tierras de cultivo gestionadas, como tierras impactadas por tierras de cultivo).

Es necesario señalar que la agricultura de precisión es un proceso que requiere inversión, conocimiento e investigación a nivel de parcela, compra de drones, sensores, imágenes satelitales y aplicaciones. En este sentido, es necesario afirmar que es un proceso de toma de decisiones facilitado por herramientas tecnológicas, las cuales ayudan a tomar decisiones

	Agricultura tradicional	Agricultura de precisión
Gestión de la superfie	La gestión de cultivos y recursos se realiza de manera uniforme sin tener en cuenta las irregularidades del suelo.	Utiliza datos específicos del cultivo y del terreno para gestionar cada parcela de manera individual.
Aplicación de insumos	Los insumos se aplican de manera general sin atender a las necesidades especificas del cultivo y del terreno	Los insumos se aplican de manera localizada, basándose en las necesidades específicas de cada área del campo, lo que optimiza su uso.
Decisiones	Basada en la experiencia del agricultor	Basada en datos
Eficiencia y sostenibilidad	Tiende a ser menos eficiente en el uso de recursos y puede tener un mayor impacto ambiental debido al uso excesivo o insuficiente de insumos.	Mejora la eficiencia en el uso de recursos, reduce el impacto ambiental.

Tabla 3.1: ***Comparativa agricultura tradicional y de precisión.***

que mejoran el manejo de las relaciones cultivo/ambiente y a optimizar el uso de los recursos naturales y de los insumos, lo que a su vez mejora la rentabilidad.

3.1.1. Agricultura de precisión en España

Según Valero (2004), la adopción de la agricultura de precisión en España se encuentra en un momento de expectación en el que no está claro si los agricultores van a hacer uso generalizado de las tecnologías de esta nueva forma de gestionar el campo o habrá que esperar aún unos años para ver la revolución en nuestras fincas.

El verdadero desafío nunca ha sido la tecnología en sí, sino el modelo de negocio que hay detrás de ella. Las tecnologías siempre funcionan, pero solo en ciertos lugares, principalmente en áreas donde los agricultores tienen la capacidad económica para costearlas. La sofisticación de la agricultura de precisión hace que estas soluciones no sean fácilmente accesibles a precios asequibles.

Además, estas tecnologías no son tan simples de operar o mantener. Los agricultores necesitan formación especializada o deben depender de proveedores externos. Esto dificulta la adopción de la agricultura de precisión en ciertas regiones donde los recursos agrícolas son limitados y la subsistencia está en juego. La gran cuestión es cómo lograr que la agricultura de precisión se expanda a nivel nacional, lo que podría mejorar significativamente la disponibilidad de alimentos en el futuro.

3.1.2. Detección de las enfermedades en las plantas

Las enfermedades de las plantas causan grandes pérdidas productivas y económicas en la agricultura. Por ejemplo, la roya de la soja (una enfermedad fúngica de la soja) ha causado una importante pérdida económica y con solo eliminar el $20\,\%$ de la infección, los agricultores pueden beneficiarse con una ganancia aproximada de 11 millones de dólares (Roberts et al., 2006).

Al infectarse, una planta desarrolla síntomas que aparecen en diferentes partes de las plantas causando un impacto agronómico significativo (López et al., 2003). Muchas de estas enfermedades microbianas se han diseminado con el tiempo en un área más grande en arboledas y plantaciones a través de la introducción accidental de vectores o a través de materiales vegetales infectados. Otra ruta para la propagación de patógenos es a través de las plantas ornamentales que actúan como huéspedes. Estas plantas se venden con frecuencia a través de la distribución masiva antes de que se conozcan las infecciones. Un sistema de detección temprana de enfermedades puede ayudar a disminuir las pérdidas causadas por las enfermedades de las plantas y puede prevenir aún más la propagación de enfermedades.

La monitorización de las plantas sanas y la identificación de las enfermedades de las plantas es clave para prevenir la pérdida de cultivo y favorecer una agricultura sostenible. Antes de entrar en desarrollo con este tema, es importante dejar claro que al hacer mención a la identificación de enfermedades de las plantas nos referimos a la identificación de patrones visibles en las plantas.

La identificación manual de enfermedades en las plantas es difícil llevarla a cabo debido a que requiere una gran cantidad de trabajo, expertos en enfermedades agrícolas y un gran tiempo de procesamiento de imágenes.

3.2. Inteligencia artificial

La inteligencia artificial (IA) es una rama de las ciencias de computación que tiene como objetivo replicar la inteligencia humana en máquinas, es decir, copiar la capacidad de aprendizaje para adoptar conocimiento y poder usarlo para la resolución de problemas. La IA es omnipresente hoy en día, la podemos encontrar desde aplicaciones móviles de reconocimiento facial hasta automóviles con conducción autónoma. Actualmente, en el sector agroalimentario, los investigadores están usando tecnología basada en IA para afrontar problemas de productividad. Aunque otras industrias han experimentado aumentos considerables de productividad debido a estos sistemas, es difícil imaginar que la agricultura experimente una transformación digital. No obstante, la IA está llevando al futuro una de las industrias más antiguas.

Los avances en inteligencia artificial, procesamiento de imágenes y unidades de procesa-

miento gráfico (GPUs) pueden expandir y mejorar la práctica de la protección y el crecimiento precisos de las plantas. La mayoría de las enfermedades de las plantas generan varios síntomas en el espectro visible y, por lo tanto, los modelos de aprendizaje deben poseer buenas habilidades de observación para que uno pueda identificar los síntomas característicos de cualquier enfermedad.

3.2.1. Aplicaciones de la IA en el sector agrícola

Tal y como se comentó en la sección 1, la IA tiene un número sorprendente de aplicaciones agrícolas, por ello, la agricultura de precisión es posible. En la tabla 3.2 vemos algunas de ellas.

Aplicaciones	Descripción
Predicción del tiempo	Permite a los agricultores tomar decisiones informadas basadas en las condiciones meteorológicas futuras como la selección de fechas óptimas de siembre y cosecha.
Predecir enfermedades de las plantas	La IA puede identificar y eliminar malas hierbas, detectar e incluso predecir enfermedades de las plantas y recomendar medidas eficientes de control de plagas.
Información detallada sobre el cultivo	Los daños a los cultivos causados por diversos desastres son el desafío más difícil para los agricultores. La mayoría de las veces, los agricultores pierden sus cosechas debido a la falta de información adecuada. En tales casos, el reconocimiento de imágenes habilitado para IA será útil ya que los informes también ayudarán a mejorar la producción.
Proporcionar una guía adecuada sobre la gestión del agua	Utilizando algoritmos de aprendizaje automático y fotos de satélites y drones, podemos analizar la sostenibilidad de los cultivos, regular la nutrición y anticipar las condiciones climáticas con el fin de maximizar el rendimiento agrícola.

Tabla 3.2: ***Aplicaciones de la IA en la agricultura.***

3.3. Redes neuronales convolucionales

Tal y como se comentó en la sección 1.5, en este trabajo nos proponemos diseñar una **red neuronal convolucional** para extraer las características de la planta y ayudar al agricultor a identificar rápidamente enfermedades en las plantas de tomate y proporcionar una recomendación.

Las redes convolucionales (o CNN, del inglés) tienen amplias aplicaciones en el campo de la clasificación de imágenes, detección de objetos, reconocimiento de voz, sistemas de recomendación...

Las redes convolucionales (o CNN, del inglés) es un tipo de red neuronal especializado en para procesar datos con una topología de matriz (imágenes). El nombre de red neuronal convolucional indica que dicha red emplea la operación lineal de convolución en lugar de una simple multiplicación matricial (operación que usa las redes neuronales estándar). Dicha operación viene dada por la siguiente fórmula:

$$c(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a)$$

donde a mide cada instante que se aplica la convolución, x es la matriz de datos de entrada multidimensional y w es el **kernel.** A continuación mostraremos los elementos más influyentes de una CNN, para garantizar una arquitectura de red que nos vayan a dar buenos resultado.

3.3.1. Kernel

El kernel es una matriz cuadrada, por lo general, de dimensión impar que se desliza por diferentes parte de la imagen para realizar la convolución y en función de la posición va generando diferentes valores con el fin de crear un mapa de características (o de activación). Una propiedad importante del kernel es que diferentes kernels pueden extraer características diferentes en la imagen, así como la textura, los bordes, patrones específicos, etc.

Los valores del kernel van cambiando a lo largo del proceso de entrenamiento mediante el algoritmo de retropropagación, esto hace que durante cada iteración se vayan ajustando automáticamente, según alguna función que definiremos más adelante, para disminuir el error.

En la siguiente figura podemos ver la actuación de un kernel:

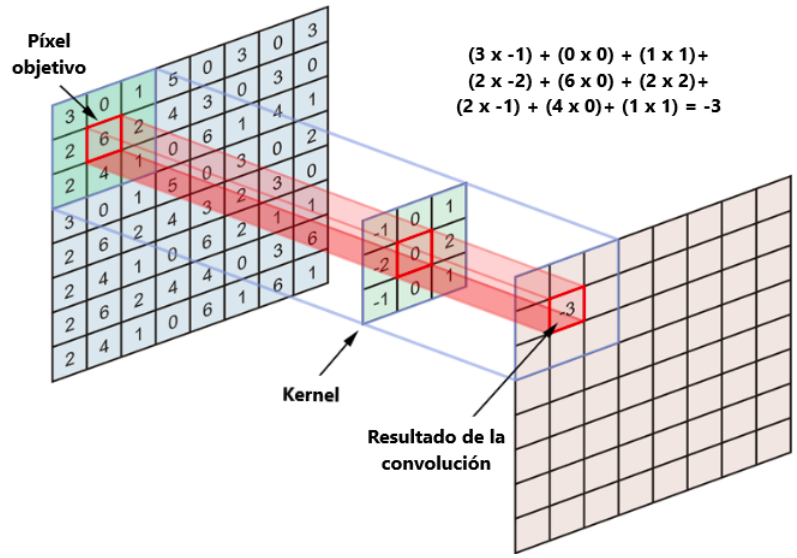

Figura 3.1: ***Funcionamiento del kernel. Elaboración propia.***

3.3.2. Pooling

Esta función permite a la red disminuir las dimensiones espaciales del mapa de activación que devuelve el kernel después de haber recorrido la imagen por completo. Las funciones mas populares son el *Max pooling*, encargada de tomar la activación máxima dentro de una región del mapa de activación (vease figura 4.1), y el *Average pooling*, encargada de retornar la media de una región rectangular del mapa de activación (suaviza la imagen).

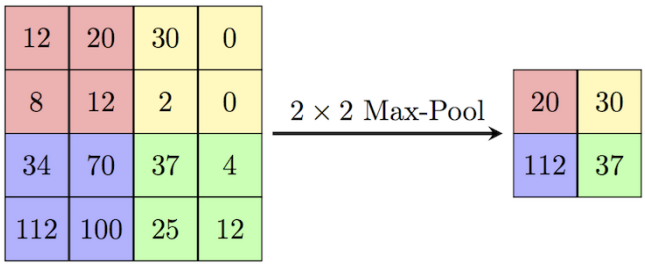

Figura 3.2: ***Representación gráfica de la función max pooling (con kernel*** $2x2$***). Adaptado de Firelord Phoenix bajo licencia CC BY-SA.***

En todos los casos, la función de pooling es bastante interesante por dos factores. Por un lado, ayuda a reducir la complejidad de la red mejorar los tiempos de entrenamiento y, por otro lado, ayuda a hacer que la representación sean invariables cuando se producen pequeñas translaciones en la entrada a los datos. La invarianza en una propiedades muy interesante cuando nos interesa saber si hay presencia de características relevantes más que saber donde están.

3.3.3. Capa de batch normalization

Antes de definir la capa de batch normalization, necesitamos saber que es un bacth. Durante el proceso de entrenamiento, el conjunto de muestras se dividen en subconjuntos dando lugar a lotes de muestras que la red preprocesará y consumirá para su aprendizaje. Cada uno de estos lotes es un batch. Es importante recalcar que el error final que la red comete se mide como la media de los errores de cada batch.

Ahora bien, la capa de batch normalization tiene como finalidad normalizar cada uno de los batch, es decir, escalando los datos a una distribución $N(0, 1)$. Con esta transformación se consigue que el modelo sea más estable debido a que los valores quedan cercanos a 0 implicando que todos los datos del modelo se consideren igual de importantes debido a que están en la misma escala. Es el modelo quién con el aprendizaje de los pesos quien decide posteriormente cuales son los datos relevantes.

3.3.4. Capas de activación

Las capas de activación no son más que simples funciones matemáticas que se aplican al valor obtenido por cada elemento de la matriz obtenida. Estas funciones permiten obtener no

linealidad en los datos, ya que sin esta linealidad la red sería simplemente una combinación lineal de los datos de entrada, cosa que no nos interesa.

Veamos algunas de las funciones más interesante que nos hemos encontrado durante el desarrollo de la sección 1.4:

3.3.4.1. ReLU (Rectified Linear Unit)

$$ReLU(0,x) = max(0,x)$$

- Es la más común dentro de las CNN
- Simple y eficiente, activa solo los valores positivos

3.3.4.2. Sigmoide

$$\sigma(x) = \frac{1}{1-e^{-x}}$$

- Salidas entre 0 y 1, útil para clasificación binaria.

3.3.4.3. Softmax

$$Softmax(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

- Se suele utilizar en la última capa de la red para clasificación multiclase.
- Convierte los valores de las salidas anteriores en probabilidades (la suma de todos ellos suman 1)

3.3.5. Dropout

La idea de esta capa es muy simple: durante el entrenamiento, la capa de dropout se encarga de desactivar aleatoriamente algunas neuronas de la red con fin de prevenir el sobreajuste del modelo. En la propia capa se indica la probabilidad de que una neurona sea desactivada, oscilando comúnmente entre el $20\,\%$ y el $50\,\%$

Una vez mencionados y explicadas algunas de las capas que nos podemos encontrar y/o utilizar, pasamos a definir transfer learning.

3.4. Transfer learning

El transfer learning es una técnica que consiste en reutilizar los pesos de un modelo entrenado para ser aplicado en una nueva red para abordar un problema similar. Por ejemplo,

una red entrenada para un problema de detectar gatos, puede ser reutilizada para un nuevo problema para detectar otros animales felinos. Esta técnica es habitual usarla cuando el conjunto de datos no es de gran tamaño. El flujo de trabajo con esta técnica se define a continuación:

1. Importar la red a utilizar.

2. Congelar las primeras capas para evitar pérdida de información durante el reentrenamiento futuro. Estas capas se centran en captar las características mas generales del problema.

3. Añadir nuevas capas extras en la cabeza de la red después de las capas congeladas. Esto permitirá a la red extraer características mas especificas del problema objetivo.

4. Entrenar las nuevas capas con el nuevo conjunto de datos.

Para importar las redes neuronales ya preentrenadas vamos a hacer uso **Keras**. Esta es una librería de código abierto que facilita el acceso y la construcción de todo tipo de redes neuronales. Keras esta desarrollada en Python y es conocida por su simpleza y eficiencia. Una importate caracteristica de esta libreria es que ofrece un rendimiento y una escalabilidad de nivel industrial. Es utilizada por grandes organizaciones como la NASA, YouTube o Waymo.

3.5. Callbacks

Los **callbacks** son objetos que puede realizar acciones en varias etapas del entrenamiento. En este trabajo hemos utilizado principalmente los siguientes:

- **Early Stopping.** Se encarga de monitorizar el rendimiento del modelo durante el entrenamiento. Si el rendimiento según una métrica preconfigurada no mejora pasado un número determinado de épocas, el entrenamiento se detiene de forma anticipada. También puede ser útil para evitar el sobreajuste. Una opción interesante que tiene este callback es que se puede restaurar los pesos correspondiente a la época en la que obtuvo mejores resultados.

- **ModelCheckpoint.** A diferencia de guardar el modelo solo al final del entrenamiento, Keras nos ofrece este callback para guardar los modelos (pesos y arquitectura) o solamente los pesos en puntos intermedios si se observa alguna mejora. Es especialmente útil para entrenamientos que requieren de una gran cantidad de tiempo y evitar interrupciones.

- **TensorBoard.** Es una herramienta de visualización que se integra con TensorFlow y Keras, y que permite realizar un seguimiento detallado del entrenamiento de modelos. Ayuda a entender mejor cómo están funcionando sus modelos, realizar ajustes y mejorar el rendimiento del entrenamiento. Permite visualizar diferentes gráficas como la

evolución de la precisión, la pérdida, la distribución de los pesos y del gradiente en diferentes épocas.

Desarrollo e implementación

4

Para la implementación de este trabajo, se siguieron los siguientes pasos:

1. En primer lugar debemos hacer uso de alguna tecnología, es decir, elegir los programas del cual haremos uso para entrenar el modelo, investigar las librerías más óptimas para el correcto desarrollo del proyecto y en qué equipo de va a llevar a cabo la implementación.

2. Búsqueda y preparación del conjunto de datos para entrenar el modelo. Es importante este punto pues el modelo extrae conocimiento a partir del ground truth.

3. Justificación de la arquitectura del modelo.

4. Justificación de los parámetros e hiperparámetros.

4.1. Tecnologías para la implementación

La elección de herramientas tecnológicas es una parte fundamental en los proyectos de machine learning. Para el desarrollo de este trabajo, hemos utilizado **Google Colaboratory (Colab)**, un entorno basado en Jupyter Notebook que se ejecuta en la nube como un servicio de Google. Esta plataforma permite la ejecución de código Python directamente desde un navegador web. Una de sus principales ventajas es que ofrece acceso gratuito a recursos de computación como GPU, TPU y memoria RAM, facilitando el desarrollo y experimentación sin necesidad de infraestructura local.

Una de las limitaciones de Colab es que sigue el concepto de asignación dinámica de límites de uso, es decir, los recursos no están fijos ni garantizados ya que estos se ofrecen en función de la disponibilidad y demanda en un momento dado. La asignación de los recursos de Colab a los usuarios son favorables para aquellos que usan la herramienta de manera más interactiva frente aquellos que ejecutan tareas que ocupan gran tiempo de ejecución. Por otro lado, una misma sesión de puede estar activa hasta 12 horas, sin embargo, esto puede variar por la disponibilidad de recursos.

En el transcurso de este proyecto hemos usado la versión gratuita de Colab, aunque como veremos más adelante, hemos añadido técnicas para paliar la pérdida de información durante el entrenamiento del modelo por desconexión del entorno. Veamos a continuación una lista de los recursos asignados durante estas largas sesiones de código. Por un lado tenemos:

- CPU: **Intel(R) Xeon(R) CPU @ 2.20GHz**
- Ram del sistema: **12 GB**
- GPU: **Testa T4 (version 535.104.05). CUDA version: 12.2**
- RAM de la GPU: **15 GB**
- Almacenamiento en disco: **112 GB**

Por otro lado, las tecnologías y librerías de fundamentales para el proyecto son:

- Python 3.10.12
- tensorboard==2.17.0
- keras==3.4.1
- numpy==1.26.4
- streamlit==1.38.0

Adicionalmente, otras librerías que fueran importantes son: *pandas*, *matplotlib*, *os* y *seaborn*.Para observar el resto de librerias necesarias y su versión, acceder al archivo *requirements.txt* del repositorio de GitHub.

Para la última parte del proyecto, fue necesario el uso de **Visual Code Studio** para poner el modelo en producción, instalado en la máquina local cuyas características son:

- CPU: **Intel(R) Core(TM) i5-6200U @ 2.30GHz**
- Ram del sistema: 8 **GB (**$7,87$ **GB usable)**
- OS: **Windows 10 Home**
- RAM de la GPU: **NVIDIA GeFORCE 920M**
- Almacenamiento en disco: **112 GB**

4.2. Análisis del conjunto de datos

Para cualquier proyecto de machine learning, no solo es importante la cantidad de información que tenemos, si no también la calidad y el conocimiento. Debemos asegurarnos que poseemos un buen conjunto de datos debido a que este va a ser el motor de conocimiento

para el modelo. Es por ello que, en la literatura, a este conjunto se le conoce como **ground truth** ("verdad fundamental"). Este término es discutible debido a que los datos, en el caso de que el conjunto este etiquetado, están sesgados por la opinión de la persona experta encargada del proyecto o de las etiquetas.

Teniendo en cuenta todos estos puntos, durante el desarrollo de la sección 3.1.2 hemos dado con un repositorio de libre acceso bastante interesante de imágenes de 150 cultivos y cerca de 1800 enfermedades de plantas. El acceso lo podemos encontrar en PlantVillage, plataforma dedicada a la recopilación de imágenes de plantas sanas y enfermas para permitir que los enfoques de visión artificial ayuden a resolver el problema de las pérdidas de rendimiento en los cultivos debido a enfermedades. Todas las imágenes de la base de datos de PlantVillage han sido tomadas en estaciones de investigación experimentales (tanto públicas como privadas) en los EE.UU y seguimos recopilando imágenes, verificadas y etiquetadas por expertos en fitopatología.

Los técnicos recolectaban hojas quitándolas de la planta con cultivos infectados por la enfermedad. Luego, las colocaban sobre una hoja de papel que proporcionaba un fondo gris o negro. Todas las imágenes se tomaron al aire libre y se busca intencionalmente una variedad de condiciones, ya que el usuario final (agricultor con un sistema inteligente) tomará imágenes en una variedad de condiciones. Para la toma fotográfica se ha usado una cámara Sony DSC - Rx$100/13$ de $20,2$ megapíxeles. El total en el momento de la realización de este proyecto es de $56{,}034$ muestras. Todas tiene un tamaño de $256x256$ píxeles en formato RGB (3 canales).

4.2.1. Preparación del conjunto de datos

Para simplificar nuestro objetivo, vamos a restringir nuestros datos solo al cultivo de tomate. Veamos a continuación un listado de las enfermedades encontradas:

- Tomato Bacterial spot.
- Tomato Early bligh.
- Tomato Late blight.
- Tomato Leaf Mold.
- Tomato Septoria leaf spot.
- Tomato Spider mites Two spotted spider mite.
- Tomato Target Spot.
- Tomato Tomato mosaic virus.
- Tomato Healthy.

Además, en la figura 4.1 podemos observar un ejemplo cada enfermedad.

Una vez que tenemos los datos, hemos repartido las muestras en tres carpetas con el siguiente porcentaje de muestras: **entrenamiento** (60 %), **validación** (20 %) y **test** (20 %) en un directorio organizado por carpetas.

El objetivo del directorio de entrenamiento cobra especial importancia debido a que el modelo aprenderá y ajustará su aprendizaje respecto a esto datos y los comparará con las muestras de validación para tener feedback de como va progresando el aprendizaje.

Finalmente, el directorio de test será de gran utilizada para evaluar el rendimiento del modelo, es decir, simula como se comporta el modelo en situaciones reales ante muestras que no ha visto anteriormente durante el etapa de aprendizaje.

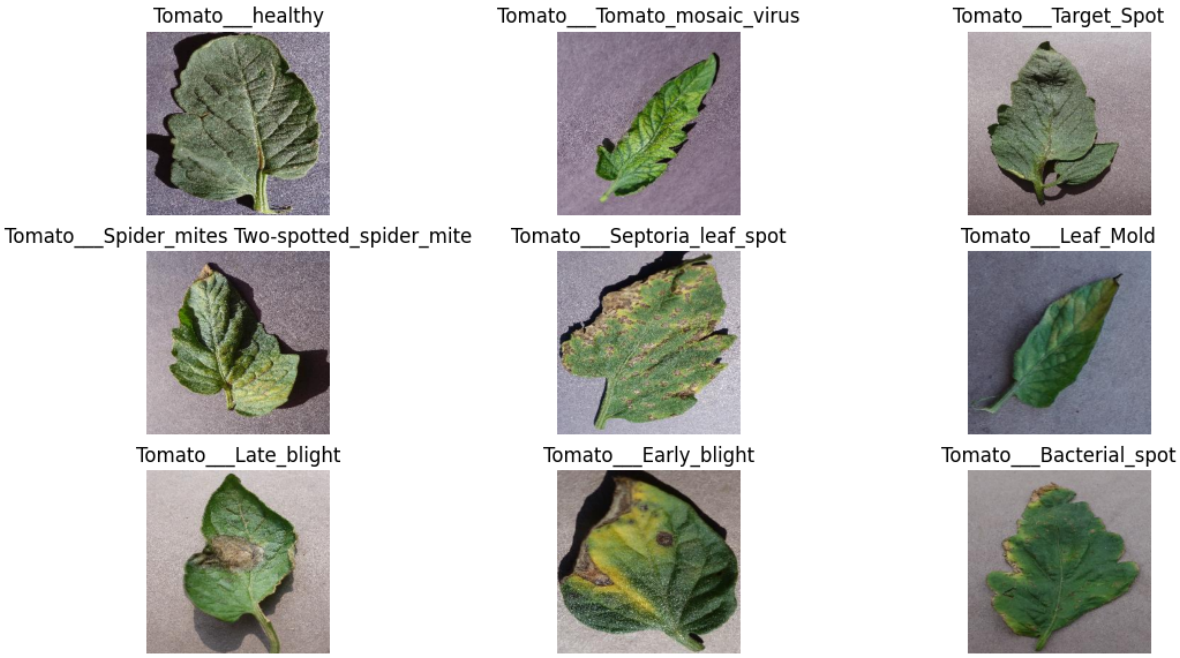

Figura 4.1: ***Enfermedades en la hoja de tomate. Fuente: PlantVillage/Tomato***

Por otro lado, en la figura 4.2 muestra un recuento de imágenes que tenemos de cada una de las clases en el directorio de entrenamiento

Obsérvese que el histograma anterior indica que las clases no están completamente balanceadas. Esto puede dar lugar a algo de imprecisión a la hora de predecir algunas clases ya que el modelo puede sesgarse hacia las clases mayoritarias y no aprender correctamente las clases minoritarias.

Para corregir el problema anterior, una técnica muy común es el **upsampling**. Esta técnica implica aumentar el número de muestras de las clases minoritarias a partir de sus muestras originales aplicando desplazamiento (horizontal y/o vertical), zoom, rotaciones... Tras esta técnica, el conjunto de entrenamiento quedaría tal y como se muestra en la siguiente figura 4.3

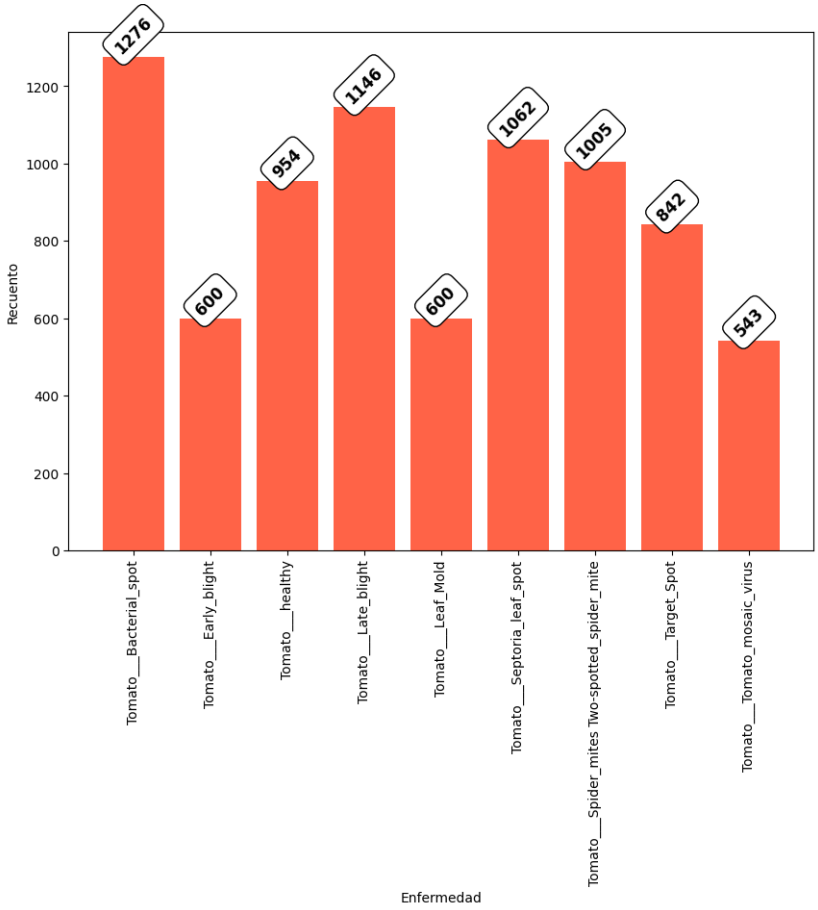

Figura 4.2: ***Histograma de imágenes por clase. Elaboración propia.***

4.3. Arquitectura del modelo

Una vez que tenemos el conjunto de datos final debemos elegir adecuadamente cual es la arquitectura del modelo atendiendo a nuestro problema. En esta fase es importante tener en cuenta todas las redes preentrenadas que nos ofrece el paquete Keras ya que, tal y como se mencionó en la sección 3.4, nos permite ahorrar horas de entrenamiento además de aportar muy buenos resultado.

4.3.1. MobileNet

MobileNet es una eficiente arquitectura de red neuronal convolucional diseñada para ser ejecutada en sistemas de bajos recursos lo que la hace interesante para ser utilizadas en dispositivos móviles, tablets o cualquier sistema con limitación de procesamiento y memoria. Sabemos que nuestras imágenes son de la forma $(256,256,3)$, es decir, existen 3 canales. Una CNN estándar aplica los kernels para cada uno de los canales mientras combina las características extraídas. La base teórica de esta arquitectura se fundamenta principalmente en la división del proceso anterior en dos etapas:

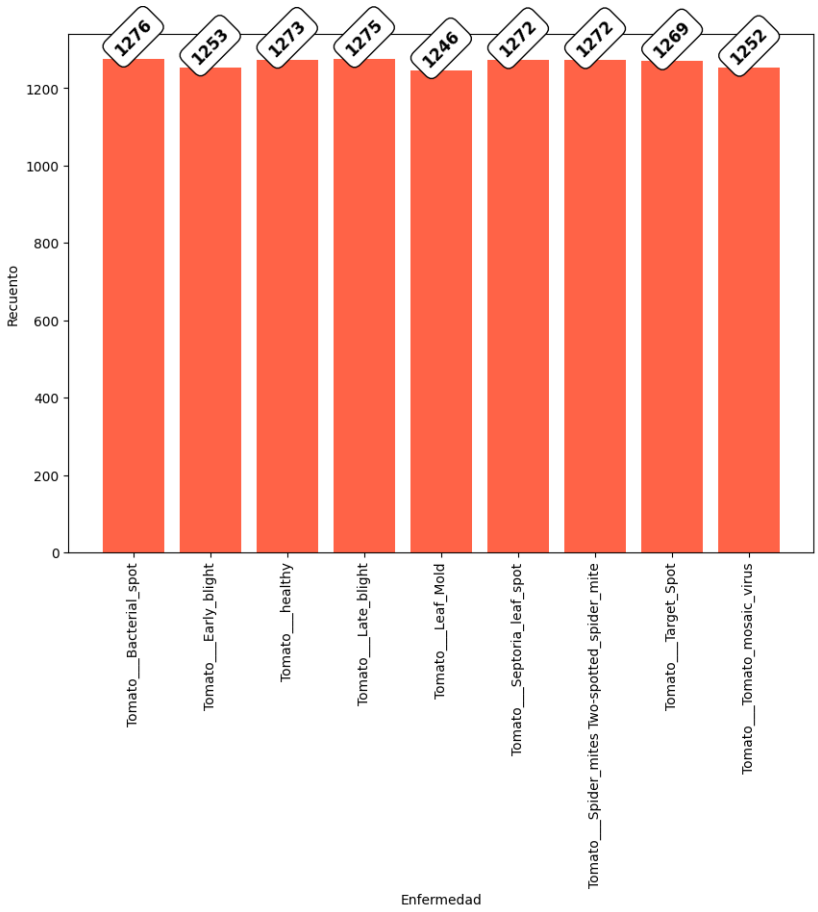

Figura 4.3: ***Histograma de imágenes por clase (upsampling). Elaboración propia.***

Convolución en profundidad: En esta etapa se aplica un filtro independiente a cada canal de la imagen de entrada. Esto reduce significativamente el número de operaciones de convolución, ya que no se combinan características en esta etapa.

Convolución punto a punto: En esta segunda etapa, se utiliza una convolución de $1x1$ (es decir, este kernel es una matriz de dimensión $1x1$) para combinar las características que se generaron en la primera etapa. Básicamente, esta operación toma las características filtradas de cada canal y las combina para producir las nuevas características, permitiendo que el modelo aprenda relaciones entre los diferentes canales de la imagen.

Como podemos observar en la tabla 4.4, la estructura de MobileNet se basa en convoluciones separables en profundidad excepto la primera capa, que es una convolución densa. Todas las capas van seguidas de una batchnormalization y una ReLU para eliminar la linealidad, con la excepción de la capa final totalmente conectada que no tiene no linealidad y se alimenta de una capa softmax para su clasificación. Es sencillo ver que después de cada convolución en profundidad (kernel $3x3$) se ha aplicado la convolución punto a punto $(1x1)$. Por último, observar que la capa de entrada tiene un tamaño de $(224, 224, 3)$ por lo que será necesario un preprocesamiento de nuestras imagenes antes de poner en marcha nuestro

Table 1. MobileNet Body Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Figura 4.4: ***Topología de MobileNet. Extraído de Howard et al. (2017).***

modelo.

MobileNet ha sido entrenado principalmente con el conjunto de datos **ImageNet**, que es una de las bases de datos más grandes y ampliamente utilizadas para tareas de visión por computadora. Contiene más de 14 millones de imágenes etiquetadas en más de 1000 categorías diferentes, que incluyen objetos, animales, paisajes y muchas otras clases. Este entrenamiento en ImageNet permite que MobileNet aprenda a detectar una gran variedad de características visuales generales, como formas, texturas, bordes y patrones que son útiles para muchas aplicaciones, incluyendo la clasificación de imágenes y la detección de objetos. Una vez que el modelo ha aprendido estas características generales, puede ser ajustado para tareas más específicas, en nuestro caso, clasificación de enfermedades en hojas de tomate. A continuación ajustaremos MobileNet según se habló en la sección 3.4.

4.4. Arquitectura de salida. Top model

Una vez que tenemos construido la arquitectura base de nuestro modelo, añadimos capas de personalizadas para ajustar la salida. En este trabajo se ha incluido las siguientes capas:

- Regularización: Hemos añadido una capa de *Global average pooling* $2D$ para reducir la dimensionalidad de las características que se han extraído de MobileNet y una *Dropout* con una probabilidad del $0,4$ que nos ayuda a generalizar el modelo y evitar el sobreajuste.

- Salida: Esta capa final de 9 neuronas nos permitirá clasificar entre las diferentes clases presentadas en este trabajo. La función softmax es una ideal para tareas de clasificación multiclase.

Capa (tipo)	Tamaño de salida	Número de parámetros
input_layer_9 (InputLayer)	$(None, 224, 224, 3)$	0
mobilenet_1.00_224 (Functional)	$(None, 7, 7, 1024)$*	3228864
global_average_pooling2d_1 (GlobalAveragePooling2D)	$(None, 1024)$	0
dropout_1 (Dropout)	$(None, 1024)$	0
dense_1 (Dense)	$(None, 9)$	9225

Tabla 4.1: ***Arquitectura final propuesta. Elaboración propia.***
* Nótese que esta salida es proporcionada por la capa *Avg Pool/s1* de la arquitectura base de la tabla 4.4.

En la tabla 4.1 podemos ver el resultado de nuestra arquitectura final.

4.5. Hiperparámetros del modelo

Los hiperparámetros son ajustes cruciales, como el número de capas, la tasa de aprendizaje, el tamaño de los lotes o la cantidad de neuronas, que pueden afectar significativamente el rendimiento del modelo. Para en este apartado se han tenido varias dificultades, ¿cómo podemos elegimos la configuración correcta del modelo para nuestro objetivo? ¿Cual es la combinación mas óptima? ¿Cómo podemos no arriesgaros a ejecutar un entrenamiento que no dará buenos resultados?

Tras estas cuestiones hemos recurrido a **Keras Tuner**, una biblioteca de optimización de hiperparámetros diseñada para ayudar a encontrar los mejores valores de hiperparámetros para los modelos de aprendizaje automático construidos con Keras facilitando la búsqueda automatizada de estos hiperparámetros.Hay subclases integradas de Tuner disponibles para algoritmos de ajuste bastante útiles como podría *RandomSearch*, *BayesianOptimization* e *Hyperband*.

En este trabajo nos hemos declinado por *Hyperband*, este algoritmo esta estrategia se basa en técnica de *Successive Halving* que, según Li et al. (2018), se encarga de seleccionar progresivamente las mejores configuraciones y asigna mas recursos a esas configuraciones a medida que avanza el proceso.

Tras lanzar la ejecución de Tuner, el modelo algoritmo ha considerado la configuración de la tabla 4.2 como la mejor:

En el caso del número de épocas no ha sido necesario pasarlo como hiperparámetro debido a la implementación de los callbakcs de la sección 3.5 listings xcolor

Hiperparámetros	Valor
Batch_size	64
Learning_rate	$0,001$
Dropout	$0,4$

Tabla 4.2: ***Configuración de hiperparámetros usando Keras tuner.***

Resultados y Discusión

5

En este capítulo se mostrarán las distintas pruebas que se han realizado para el Trabajo de Fin de Máster, también se describirán los resultados más relevantes obtenidos. Se han realizado múltiples pruebas con el propósito de ver el correcto funcionamiento de todos los desarrollos que se han realizado y veremos finalmente si las redes neuronales convolucionales sirven realmente para clasificar enfermedades en hojas de tomate.

Tras haber entrenado el modelo con los parámetros e hiperparámetros descritos en la sección anterior, hemos obtenido las siguientes resultados.

5.1. Evolución de la precesión del modelo

La gráfica de la evolución de la precisión del modelo tanto en el conjunto de entrenamiento como en el de validación se pueden observar en la figura 5.1.

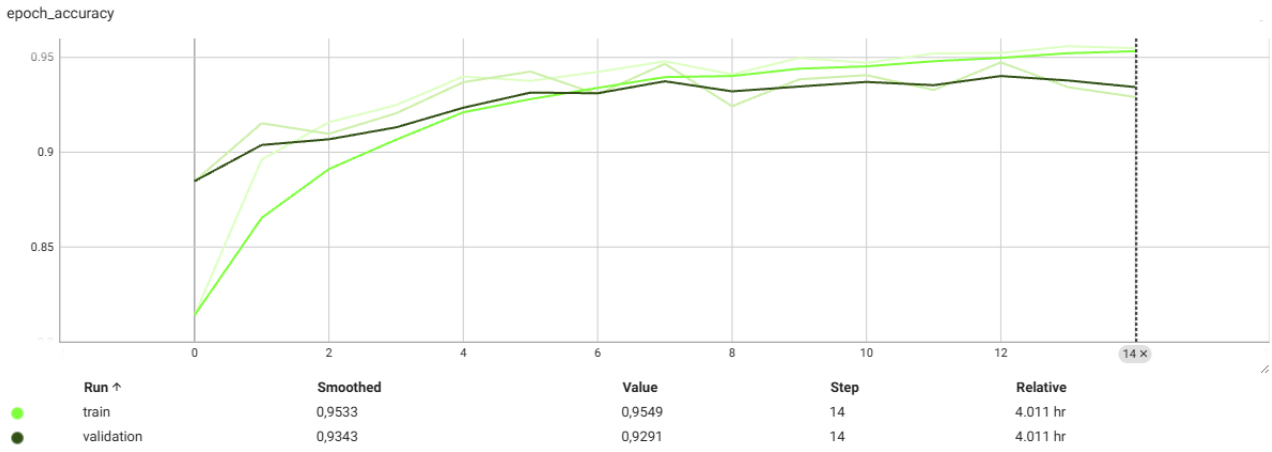

Figura 5.1: ***Gráfica de la evolución de la precisión. Imagen proporcionada por Tensorboard.***

Notase como la precisión del conjunto de entrenamiento empieza a aumentar rápidamente durante las primeras 5 épocas. Esto es normal en la fase inicial, ya que el modelo está

aprendiendo patrones básicos de los datos. Después de la época 6 la curva empieza a estabilizarse hasta alcanzar la época 14, donde se ha lanzado el EarlyStopping. Por último, observar que a partir de la época 8 podemos ver un leve sobreajuste del modelo en que, aun así, podemos afirmar que el modelo generaliza bien.

5.2. Evolución del error del modelo

La imagen 5.2 muestra la evolución de la pérdida del modelo durante el entrenamiento, con las curvas correspondientes al conjunto de entrenamiento y al conjunto de validación.

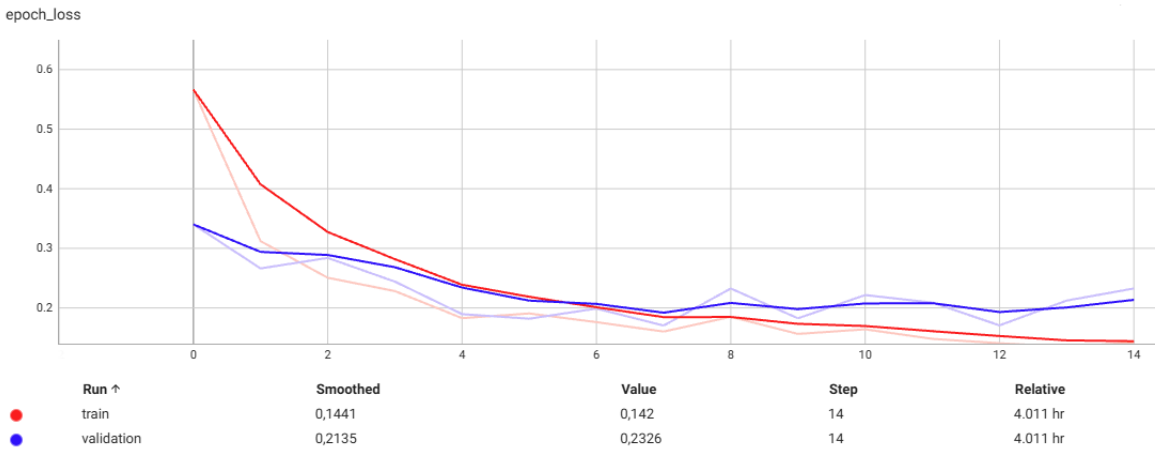

Figura 5.2: ***Gráfica de la evolución del error. Imagen proporcionada por Tensorboard.***

De la misma manera que las curvas de precisión, podemos observar que el modelo aprende rápidamente durante las primeras épocas, lo que es evidente por la reducción rápida de la pérdida tanto en el conjunto de entrenamiento como en el de validación.

5.3. Evaluación general del modelo

La evaluación general mide el rendimiento general del modelo con datos que nunca antes había visto, es decir, ahora es el conjunto de los datos de prueba quien realizará el papel principal para ver como se comportará el modelo en la realidad.

Podemos hacer uso de la función `model.evaluate()` que realiza predicciones internamente y luego calcula las pérdidas correspondientes basándose en las etiquetas verdaderas. Esta función de pérdida queda definida a la hora de compilar el modelo y durante todo el desarrollo de esta trabajo se ha usado la **sparse categorical crossentropy**. Esta función es útil cuando las etiquetas son representadas con valores enteros (y no en formato one-hot-encoding) para calcular la pérdida usando la entropía entre la clase correcta y las probabilidades predichas por el modelo para esa clase. Viene dada por la siguiente ecuación:

$$L = \frac{1}{N}\sum_{i=1}^{N} log(\hat{y}_i, y_i)$$

donde:

- N es el total de clases de nuestro conjunto de datos.
- y_i es la clase verdadera de la muestra i
- $\hat{y}_i, y_i$ es la probabilidad predicha para la clase verdadera y_i de la muestra i

```
# Evaluar el modelo
test_loss, test_acc = model.evaluate(test_generator, steps=len(
    test_generator))
print(f"Precision en los datos de prueba:{test_acc * 100:.2f}%")
```

```
27s 196ms/step - accuracy: 0.9602 - loss: 0.1248
Precision en los datos de prueba: 95.63%
```

5.4. Evaluación del modelo por clase

Para tener una completa evaluación de nuestro modelo, no basta con tener una evaluación general si no que es necesario también tener una evaluación por clases. Durante el desarrollo de este trabajo, se comenzó a entrenar el modelo sin aplicar el upsampling a nuestro conjunto de datos original, tal y como veíamos en la sección 4.2, este conjunto estaba desbalanceado. Sin embargo, los análisis vistos anteriormente era positivos llegando incluso al $89\,\%$ de acierto.

Entonces, ¿por qué fue necesario aplicar el upsampling? Pues bien, en términos generales, el modelo conseguía una precisión bastante buena pero al evaluar las precisiones por clase, los resultados no eran tan satisfactorios ya que el modelo se centraba en la clases con mayor representación en las muestras y, en consecuencia, había enfermedades que no era capaz de generalizar. A la hora de aplicar la función `model.evaluate()`, por estadística, tomaba una clase cuya representación en los datos era alta y conseguía una predicción correcta. De ahí viene la importancia de aplicar una evaluación individual a cada enfermedad de nuestro modelo. Para llevar a cabo esto, hemos utilizado la función de `classification_report()` de la librería `scikit-learn` que nos devuelve las siguientes métricas:

- **Precisión**

  Mide la proporción de casos positivos identificados correctamente por el modelo sobre el total de casos identificados como positivos.

$$Precision = \frac{TP}{TP + FP}$$

donde: TP (True Positives) indica los casos correctos que el modelo ha clasificado correctamente y FP (False Positives) indica los casos que el modelo ha clasificado incorrectamente como positivos.

Es importante considerar esta métrica cuando el **el costo de un falso positivo es alto.**

- **Recall** Mide la proporción de casos positivos identificados correctamente por el modelo sobre el total de casos positivos presentes en los datos.

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

donde FN (False Negatives) son casos positivos que el modelo ha clasificado incorrectamente como negativos. Es importante considerar esta métrica cuando **el costo de un falso negativo es alto.**

- **F1-Score**

Esta medida combina las dos métricas anteriores calculando la media armónica entre ambas para encontrar un equilibrio entre ambas.

$$F1 - Score = \frac{Precision * Recall}{Precision + Recall}$$

Los resultados de esta métrica oscila entre 0 y 1, siendo este último el mejor valor posible indicando que el modelo es capaz de identificar tanto los casos positivos como los negativos.

El F1-Score es especialmente útil cuando los datos están desequilibrados y hay más casos negativos que positivos o viceversa. En estos casos, la precisión y el recall por separado pueden ser métricas engañosas.

- **Support**

Representa el número total de instancias (muestras) reales de cada clase en el conjunto de datos.

Esto es crucial cuando interpretas las métricas de rendimiento, ya que las clases pequeñas pueden tener un impacto limitado en las métricas globales. Por ejemplo, si una clase tiene un support muy bajo, una precisión baja podría no ser tan significativa como en una clase con un support más alto.

En la tabla 5.3, podemos observar los valores por clase de cada una de las métrica expuestas anteriormente.

5.5. Matriz de confusión

Esta muestra la cantidad de casos que el modelo ha clasificado correctamente e incorrectamente en cada clase. La matriz de confusión es una matriz cuadrada no simétrica con

Evaluación por clase

Enfermedad	Precisión	Recall	F1-Score	Support
Tomato___Bacterial_spot	0.93	0.95	0.94	425
Tomato___Early_blight	0.76	0.80	0.78	200
Tomato___Late_blight	0.93	0.93	0.93	382
Tomato___Leaf_Mold	0.91	0.92	0.91	200
Tomato___Septoria_leaf_spot	0.93	0.90	0.92	354
Tomato___Spider_mites Two-spotted_spider_mite	0.92	0.93	0.93	335
Tomato___Target_Spot	0.86	0.86	0.86	281
Tomato___Tomato_mosaic_virus	0.99	0.95	0.97	181
Tomato___healthy	0.98	0.97	0.98	318

Figura 5.3: ***Evaluación del modelo por clase. Elaboración propia generada en Google Colab.***

el número de filas y columnas igual al número de clases en el problema de clasificación. Podemos ver los resultados obtenido en este trabajo en la figura 5.4

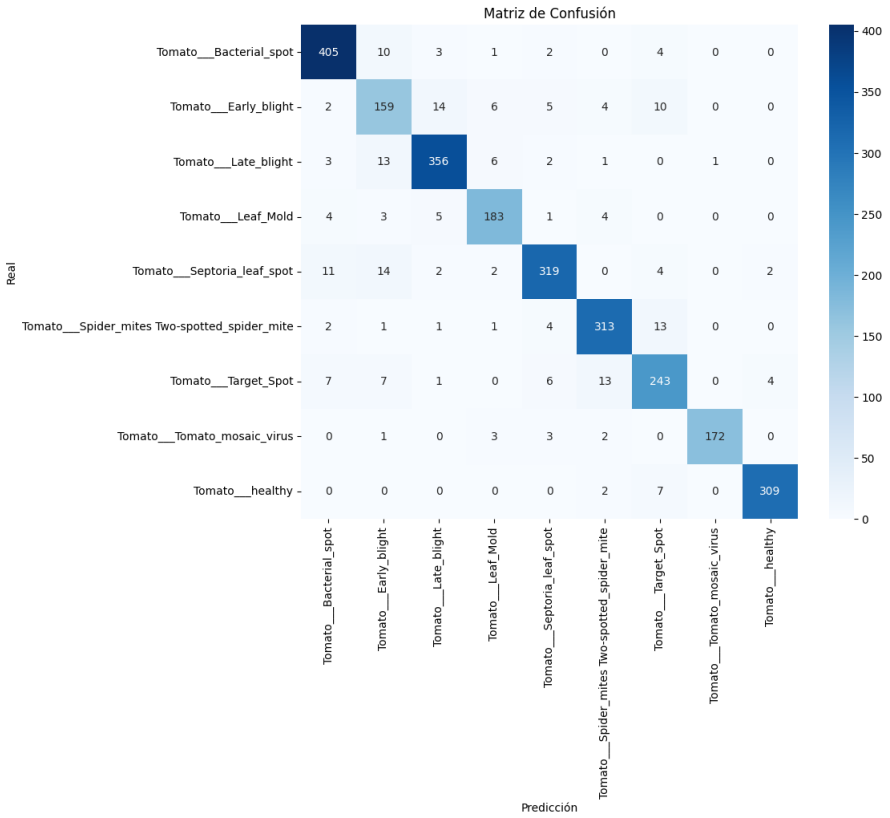

Figura 5.4: ***Matriz de confusión. Elaboración propia generada en Google Colab.***

5.6. ¿Qué esta observando realmente el modelo?

Cuando desarrollamos la arquitectura de una CNN nuestro principal objetivo es hacer que nuestra red, a medida que se vaya entrenando, preste atención a las zonas mas significativas

de la imagen para que pueda tener tomar una mejor decisión. En este apartado veremos mediante un mapa de calor (mapa de activación), como nuestro modelo observa una imagen de una hoja enferma de planta de tomate.

Para ello, utilizaremos una técnica llamada **Grad-CAM (Gradient Class Activation Map).** La idea que hay detrás es bastante sencilla: para averiguar la importancia de una determinada clase en nuestro modelo, simplemente tomamos su gradiente con respecto a la capa convolucional final y luego lo sopesamos con la salida de esta capa. Este es el esquema de uso de Grad-CAM:

1. Calcular la salida del modelo y la salida de alguna capa para la imagen en la que queremos calcular el mapa de activación.

2. Encontrar el índice de la clase que ha predicho el modelo dada la imagen.

3. Calcular el gradiente de la clase predicha con respecto a la última capa convolucional

4. Promediarlo y luego ponderarlo con la última capa convolucional (multiplicarlos).

5. Normalizar entre 0 y 1 para la visualización.

6. Convertir a RGB y superponerlo a la imagen original.

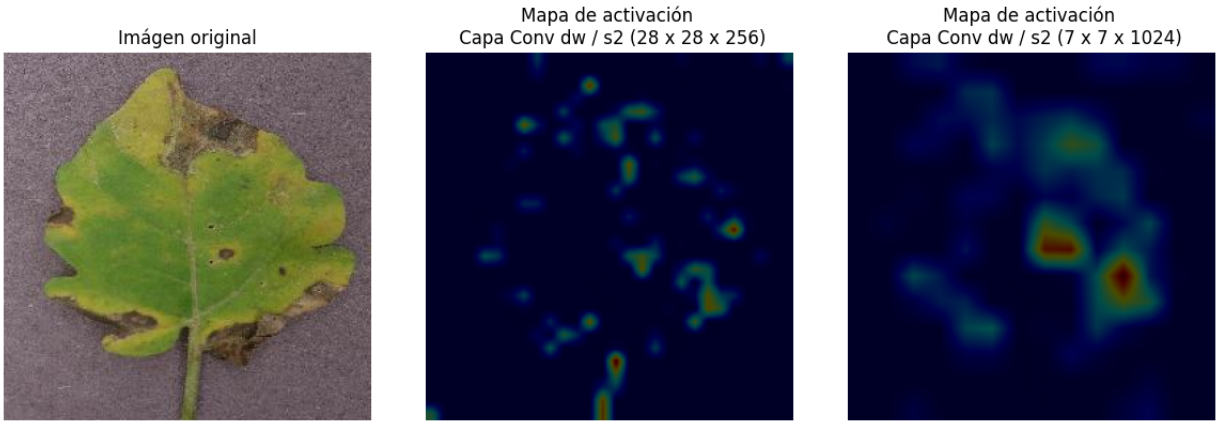

Figura 5.5: ***Mapas de activación usando Grad-CAM. Elaboración propia generada en Google Colab.***

En las imagenes de la figura 5.5 se aprecia en funcionamiento interno de una CNN.
Todo este proceso lo podemos ver en el fichero `Grad-CAM.py`

Despliegue del modelo

6

En este capítulo se aborda el despliegue del modelo desarrollado previamente en este trabajo, mediante la creación de una aplicación sencilla utilizando **Streamlit**.

Una vez probado el modelo y obtenido las conclusiones que verifican la fiabilidad de modelo, el siguiente objetivo es llevar el modelo desde un entorno de experimentación a un entorno práctico en el que usuarios puedan interactuar con la CNN de manera accesible y visual, facilitando su uso en situaciones del mundo real.

6.1. Streamlit

Streamlit es una herramienta de código abierto diseñada para crear aplicaciones web interactivas, enfocadas en proyectos de ciencia de datos y aprendizaje automático. Su facilidad de uso y su capacidad para integrar modelos de machine learning con interfaces amigables la convierten en una elección atractiva para implementar soluciones como la desarrollada en este proyecto. Además, Streamlit permite a los desarrolladores desplegar sus aplicaciones de forma rápida, con un enfoque en la interactividad y la visualización.

Una de las principales ventajas de Streamlit es que no requiere conocimientos profundos de desarrollo web. Permite transformar scripts de Python en aplicaciones web interactivas con mínimos ajustes, facilitando la creación de interfaces que permiten a los usuarios interactuar directamente con el modelo. Esto es especialmente útil en el contexto de este proyecto, ya que se busca que la herramienta sea accesible para usuarios que no necesariamente tienen formación técnica en programación o machine learning.

La aplicación desarrollada con Streamlit permite cargar imágenes de hojas de tomate y procesa estas imágenes utilizando el modelo de CNN entrenado. Los resultados se presentan de manera clara y comprensible, proporcionando al usuario información sobre la posible presencia de enfermedades en las plantas. Esta interfaz intuitiva facilita la adopción de la herramienta en entornos agrícolas, donde puede ser utilizada por agricultores y técnicos para el monitoreo y diagnóstico de cultivos.

6.2. Streamlit Cloud

Para facilitar aún más el acceso y la distribución de la aplicación, se utilizó **Streamlit Cloud**, una plataforma que permite alojar y compartir aplicaciones desarrolladas con Streamlit de manera sencilla y eficiente. Streamlit Cloud ofrece un entorno en línea donde las aplicaciones pueden ser desplegadas sin la necesidad de gestionar servidores o infraestructuras complejas, lo que agiliza el proceso de implementación y mantenimiento.

El uso de Streamlit Cloud presenta varias ventajas. En primer lugar, permite que la aplicación esté disponible en cualquier momento y lugar, accesible desde cualquier dispositivo con conexión a Internet. Esto es crucial para maximizar el impacto y la utilidad de la herramienta, ya que los usuarios pueden acceder a ella desde el campo o en situaciones donde no disponen de equipos especializados.

En segundo lugar, Streamlit Cloud simplifica el proceso de actualización y mejora de la aplicación. Al estar vinculada con un repositorio de código, cualquier modificación o mejora realizada en el código fuente se refleja automáticamente en la aplicación desplegada. Esto facilita la iteración y el desarrollo continuo, permitiendo incorporar nuevos hallazgos o ajustar el modelo en función de datos adicionales o retroalimentación de los usuarios.

Además, Streamlit Cloud gestiona de manera automática las dependencias y requerimientos del sistema, asegurando que la aplicación funcione correctamente sin necesidad de configuraciones manuales. Esto reduce la carga técnica asociada con el despliegue y permite enfocarse en la optimización del modelo y la experiencia del usuario.

En resumen, la combinación de Streamlit y Streamlit Cloud ha permitido trasladar el modelo desarrollado a un entorno práctico y accesible, ofreciendo una solución efectiva para la detección de enfermedades en plantas de tomate. La aplicación resultante es una herramienta poderosa que aprovecha las capacidades del aprendizaje automático, presentada de una forma que es fácilmente utilizable por profesionales en el campo agrícola, contribuyendo así a mejorar la productividad y la salud de los cultivos.

Conclusiones

7

En este trabajo, se ha desarrollado un sistema basado en redes neuronales convolucionales para la detección automática de enfermedades en las hojas de tomate utilizando imágenes. Los resultados obtenidos demuestran que es posible implementar con éxito un modelo de inteligencia artificial capaz de identificar diferentes tipos de enfermedades con una alta precisión, contribuyendo al avance en la agricultura de precisión.

Entre los principales logros se encuentran la implementación de una arquitectura eficiente de MobileNet y el uso de técnicas como el transfer learning y la normalización por lotes de imágenes para optimizar el rendimiento del modelo. Gracias a estas técnicas, se logró una precisión del $95{,}63\,\%$ en el conjunto de datos de prueba, superando las expectativas iniciales.

El análisis de los resultados revela que, aunque el modelo generaliza bien para la mayoría de las clases de enfermedades, todavía existen desafíos en cuanto a la clasificación de enfermedades menos representadas en el conjunto de datos. La aplicación de técnicas de upsampling mejoró considerablemente la capacidad del modelo para manejar clases desbalanceadas, aunque sigue habiendo margen de mejora en este aspecto.

En términos prácticos, el sistema desarrollado podría ser una herramienta útil para los agricultores, permitiendo una detección temprana y precisa de enfermedades en los cultivos, lo que facilitaría la toma de decisiones más informadas y eficientes en la gestión de los cultivos.

7.1. Cumplimiento de los objetivos

El objetivo principal de este trabajo, “crear una red neuronal convolucional para la detección de enfermedades en las hojas de tomate”, ha sido alcanzado. A lo largo del desarrollo del proyecto, se ha profundizado en la comprensión del problema, lo que ha llevado a la adopción de nuevas herramientas, como Keras Tuner y Grad-CAM, que se incorporaron durante el desarrollo del TFM. Además, los objetivos secundarios, mencionados en la sección 1.5, también se han logrado, gracias a la implementación de TensorBoard, que permitió monitorear detalladamente cada etapa del entrenamiento y extraer valiosas conclusiones reflejadas en este documento. Igualmente, el uso de Streamlit facilitó el despliegue del modelo mediante

una interfaz sencilla, accesible para el usuario final.

Esto nos permite afirmar que la red neuronal convolucional basada en MobileNet ha funcionado correctamente, y sus resultados pueden compararse de manera efectiva con otras arquitecturas reconocidas en la literatura, como *LeNet* (96,27 %), *VGGNet* (99,25 %), *ResNet50* (98,65 %) y *Xception* (98,13 %) (Kumar y Vani (2019)).

Respecto a las fechas de entrega de los hitos, el primer hito se entregó en la fecha correspondiente. Sin embargo, los siguientes hitos, se acordó con el tutor posponer la entrega una semana debido a las complicaciones que expondremos en la siguiente sección.

Resumiendo, este trabajo ha permitido desarrollar y mejorar mis habilidades de investigación, organización, planificación, análisis y, sobre todo, entendimiento del dominio trabajado, viendo varias de las técnicas y características que se usan. En definitiva, es un proyecto que ha servido para expandir el conocimiento global de la inteligencia artificial, así como aprender nuevas técnicas y modelos al mismo tiempo que se refrescaban y asentaban algunos de los contenidos teóricos vistos durante el máster.

7.2. Problemas y contratiempos

Se presentaron varios problemas y contratiempos a lo largo del desarrollo del proyecto.

El primero ocurrió durante la fase de implementación, relacionado con la compatibilidad entre las librerías y sus versiones. La versión de TensorFlow utilizada inicialmente no era compatible con la versión de Python instalada en ese momento. Este inconveniente se resolvió actualizando Python a una versión compatible.

En cuanto al conjunto de datos, como se explicó en la sección 4.2, se invirtieron tiempo y recursos en su separación y preparación para su uso óptimo. Sin embargo, debido a que el almacenamiento y procesamiento se realizaron en la nube, el proceso fue considerablemente lento. Además, errores en la programación provocaron que la partición del conjunto de datos no se realizara correctamente, lo que resultó en una mezcla de diferentes clases. En resumen, para futuros proyectos, aunque este tipo de problemas de programación pueda parecer sencillo, vale la pena invertir, o mejor dicho, ganar tiempo adicional asegurándose de que todo funcione correctamente con un subconjunto más pequeño de datos antes de proceder con el conjunto completo.

Otro desafío importante fue la incorrecta configuración de los parámetros del EarlyStopping. Durante varias fases experimentales, el modelo continuaba entrenándose innecesariamente, ya que no se observaban mejoras en la precisión, lo que resultó en una considerable pérdida de tiempo. Esto fue el principal motivo por el cual se retrasó la entrega del hito 2.

Otro problema al que tuve que enfrentarme fue la falta de planificación adecuada en la fase final del proyecto. Inicialmente, no se contempló el desarrollo de una aplicación, ya que su implementación dependía en gran medida del tiempo disponible tras alcanzar el objetivo principal. Esta situación provocó un retraso de una semana en la entrega del hito 3.

Limitaciones y Perspectivas de Futuro

8

Una vez puesto en marcha el modelo y tras realizar experimentos con él, se identificó una limitación importante relacionada con el tipo de imágenes que utiliza para realizar las predicciones. Si revisamos las imágenes con las que el modelo ha sido entrenado (figura 4.1), podemos ver que son de muy alta calidad, mostrando generalmente una sola hoja por imagen. Sin embargo, en la práctica cotidiana, las imágenes no se capturan en un entorno controlado de laboratorio, sino en condiciones reales de campo. Esto representa una limitación, ya que las condiciones en el campo pueden afectar significativamente el rendimiento del modelo [Ferentinos (2018)].

Algunos de los principales factores que pueden influir son:

- Las imágenes capturadas en el campo tienen fondos diferentes a los utilizados durante el entrenamiento.
- Las hojas pueden estar parcialmente sombreadas en las imágenes tomadas en campo.
- Las imágenes pueden incluir múltiples objetos además de la hoja, como dedos, manos, zapatos, o incluso hojas de otras plantas.
- Las hojas pueden ocupar solo una pequeña parte de la imagen o no estar centradas en el marco.

Como trabajo a futuro, con base en los resultados obtenidos y lo aprendido a lo largo de este proyecto a continuación se proponen varias mejoras e implementaciones para seguir investigando en la búsqueda del modelo.

Para ello, la primera recomendación es incluir mas variedades de imagenes para poder usarse en situaciones reales. Se debe recopilar una variedad mucho más amplia de datos de entrenamiento, de varias fuentes y/o bases de datos de diferentes áreas geográficas, condiciones de cultivo. El modelo propuesto mostró su alto potencial, por lo que es una cuestión de cantidad y calidad de los datos disponibles para mejorar el sistema, y hacerlo más amplio y más robusto en condiciones reales de cultivo.

Otro enfoque para resolver esta limitación consiste en omitir el fondo y los ruidos de la imagen capturada. Esto se puede lograrse mediante técnicas de Machine Learning, entre la

cuales destaca la **segmentación semántica**. El objetivo de este técnica es etiquetar cada píxel de una imagen con una clase correspondiente de lo que se está representando. La salida esperada en la segmentación semántica no son solo etiquetas y parámetros de cuadro delimitador. La salida en sí es una imagen de alta resolución (normalmente del mismo tamaño que la imagen de entrada) en la que cada píxel se clasifica en una clase determinada. A esta tarea se conoce comúnmente como **predicción densa**

En nuestro problema particular, antes de realizar la predicción de la enfermedad en la hoja, se deberá realizar un ejercicio de clasificación binaria a nivel de píxel, es decir, seleccionar mediante Deep Learning los píxeles que pertenecen a la hoja y reconstruir la imagen. De esta manera, omitiremos la información que no es relevante para la predicción haciendo que el modelo desarrollado en este trabajo sea útil en condiciones de campo.

Respecto a las arquitecturas de la segmentación semantica, al final de la asignatura Deep Learning y Redes Neuronales, se vio este tema enfocándose en la arquitectura **U-Net.** Recibe este nombre por la forma que tiene la arquitectura. Véase figura 8.1

Ronneberger et al. (2015) describe esta arquitecturas en dos partes. La primera parte es la ruta de contracción (*encoder*) que se utiliza para capturar el contexto de la imagen y no es más que red neuronal tradicional. La segunda parte consiste en la ruta de expansión (*decoder*) que se utiliza para permitir una localización precisa mediante convoluciones transpuestas. Además, tanto el decoder como el encoder tienen que tener exactamente las mismas dimensiones ya que están relacionadas mediante concatenación (*skip-connections*). Esta concatenación tiene la finalidad de recordar a la red la información original que podría haberse perdido durante el proceso.

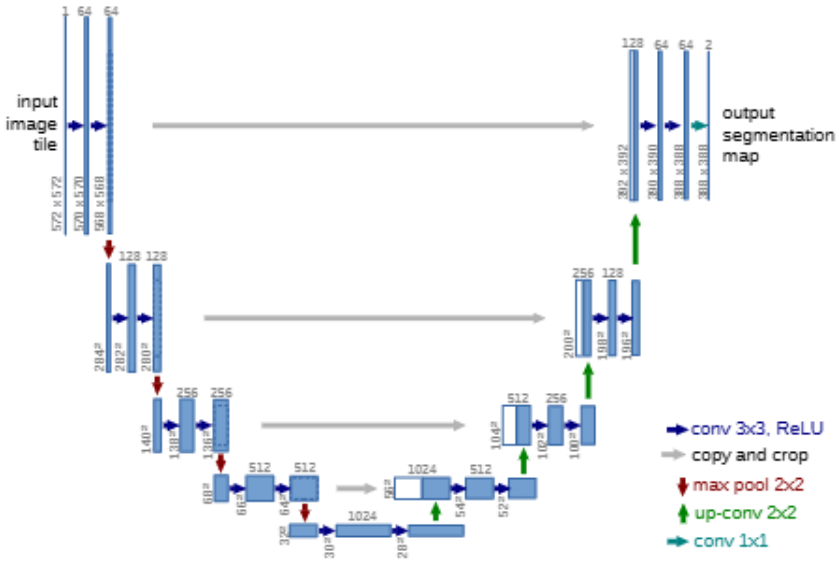

Figura 8.1: ***Arquitectura U-net con imagen de entrada (***572,572,3***). Imagen extraída de Ronneberger et al. (2015)***

Bibliografía

About Keras. https://keras.io/about/.

Adrián Colomer Granero, G. E. M. R. (2020). *REDES NEURONALES Y DEEP LEARNING*. Universidad Internacional de Valencia.

Autor Desconocido (2024). Los desafíos de la agricultura de precisión. Accedido: 20 de septiembre de 2024.

Bajpai, A., T. T. (2023). An efficient approach to detect and predict the tomato leaf disease using enhance segmentation neural network.

Camila Andrea Ramírez Borray, Carlos Eduardo Riveros Rey, J. C. T. D. P. A. V. A. Estado del arte: Aplicaciones de la agricultura 4.0. *Revista de Ciencias Agropecuarias*.

David P. Hughes, M. S. (2015). An open access repository of images on plant health to enable the development of mobile disease diagnostics. *Digital Epidemiology Lab, School of Life Sciences, School of Computer and Communication Sciences, EPFL, Switzerland*.

et al., D. (2011). Recent advances in precision (target) conservation. *Journal of Soil and Water Conservation, 66(6), 167A-170A*.

Ferentinos, K. P. (2018). Deep learning models for plant disease detection and diagnosis. *Computers and Electronics in Agriculture*, 145:311–318.

G., G. y J., A. P. (2019). Identification of plant leaf diseases using a nine-layer deep convolutional neural network. *Computers Electrical Engineering*, 76:323–338.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., y Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications.

Hughes, D. P. y Salathé , M. (2015). An open access repository of images on plant health to enable the development of mobile disease diagnostics through machine learning and crowdsourcing. *CoRR*, abs/1511.08060.

Hughes, D. P. y Salathé, M. (2015). An open access repository of images on plant health to enable the development of mobile disease diagnostics through machine learning and crowdsourcing. *CoRR*, abs/1511.08060.

José A. Brenes1, Alexandra Martínez1, C. Q.-L. M. J. (2020). Sistemas de apoyo a la toma de decisiones que usan inteligencia artificial en la agricultura de precisión: un mapeo sistemático de literatura. *ProQuest*.

Khirade, S. D. y Patil, A. (2015). Plant disease detection using image processing. In *2015 International Conference on Computing Communication Control and Automation*, pages 768–771.

Kozyrkov, C. (2023). Why ai and decision-making are two sides of the same coin. *Medium*.

Kumar, A. y Vani, M. (2019). Image based tomato leaf disease detection. In *2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pages 1–6.

Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., y Talwalkar, A. (2018). Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18(185):1–52.

Mohd Javaid, Abid Haleem, I. H. K.-R. S. (2023). Understanding the potential applications of artificial intelligence in agriculture sector. *Advanced Agrochem*, 2(1):15–30.

Pérez-Ortega, D. J., F. A. B.-A. . A. M. d. S. (2022). Variables que influyen en la aplicación de la agricultura de precisión en colombia: Revisión de estudios. [variables that influence the application of precision agriculture in colombia: review of studies]. *Revista Ciencia y Tecnología Agropecuaria, 23(1)*.

Redactor, P. L. (2023). El tomate, el rey del cultivo ecológico en los invernaderos de almería. *Diario de Almería*.

Ronneberger, O., Fischer, P., y Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. *ArXiv*, abs/1505.04597.

Sachin D. Khirade, A. B. P. (2015). Plant disease detection using image processing. In *International Conference on Computing Communication Control and Automation*.

Sankaran, S., Mishra, A., Ehsani, R., y Davis, C. (2010). A review of advanced techniques for detecting plant diseases. *Computers and Electronics in Agriculture*, 72(1):1–13.

Scrum.org (2022). Scrum. Accessed on october, 2024.

Transfer Learning in Keras.

Understanding Semantic Segmentation with UNET (2019). towardsdatascience.comhttps://towardsdatascience.com/understanding-semantic-segmentation-with-unet-6be4f42d4b47.

Valero, C. (2004). Situación de la agricultura de precisión en españa. *Informe CSIC*.

José A. Brenes1 (2020) Valero (2004) et al. (2011) Camila Andrea Ramírez Borray Hughes y Salathé (2015) Pérez-Ortega (2022) Redactor (2023) G. y J. (2019) Sankaran et al. (2010) Sachin D. Khirade (2015) Mohd Javaid (2023) Adrián Colomer Granero (2020) Transfer Learning in Keras About Keras Kozyrkov (2023) David P. Hughes (2015) Hughes y Salathé (2015) Howard et al. (2017) Li et al. (2018) Bajpai (2023) Scrum.org (2022) Khirade y Patil (2015) Understanding Semantic Segmentation with UNET (2019) Autor Desconocido (2024)