UNIVERSIDAD DE LA REPÚBLICA URUGUAY

FACULTAD DE INGENIERÍA

Detección y conteo de flores de manzano

Informe de Proyecto de Grado presentado por

Facundo Tessore

en cumplimiento parcial de los requerimientos para la graduación de la carrera de Ingeniería en Computación de Facultad de Ingeniería de la Universidad de la República

Supervisores

Gonzalo Tejera
Mercedes Marzoa

Montevideo, 28 de agosto de 2025

Resumen

El presente proyecto aborda la problemática de la detección de objetos pequeños mediante redes neuronales convolucionales, en particular la detección y conteo de flores de manzano en montes de frutales. El monitoreo del número de flores en árboles productores de frutos resulta fundamental en la agricultura de precisión, ya que permite estimar el potencial productivo del árbol, anticipar la carga frutal y optimizar decisiones agronómicas clave como el raleo, la fertilización y el manejo hídrico.

Las principales dificultades radican en el pequeño tamaño de las flores, su alta densidad, y la frecuente oclusión entre instancias. Si bien las arquitecturas de detección de objetos de última generación presentan un buen rendimiento en la identificación de objetos grandes y medianos, su desempeño en la detección de objetos pequeños dista de ser aceptable para tareas que requieren alta precisión en el conteo.

Con el objetivo de atacar esta problemática, se evaluó el impacto de aplicar el marco de trabajo *Slicing Aided Hyper Inference* (SAHI) en modelos entrenados para detección de flores. SAHI combina dos técnicas: el ajuste fino asistido mediante el particionado de las imágenes del conjunto de datos previo al entrenamiento, y la inferencia mediante particionado de imágenes, buscando mejorar la representación de objetos pequeños en el proceso de detección. Se entrenaron modelos de detección de objetos utilizando conjuntos de datos públicos que contienen imágenes de flores de manzano y perales en montes de frutales. Como arquitectura principal de detección se utilizó *YOLO (You Only Look Once)* en su versión v8.

Los resultados de los ensayos ponen de manifiesto mejoras significativas en el desempeño de los modelos tras la aplicación de estas técnicas. Se observaron incrementos entre 5 y 15 puntos porcentuales en las métricas de precisión media de los modelos posteriores a la aplicación del marco de trabajo SAHI. Estas mejoras se manifiestan principalmente en las instancias clasificadas como pequeñas, seguidas por aquellas de tamaño mediano, lo que evidencia una mejora en la sensibilidad del modelo frente a objetos de menor escala. Los resultados validan la efectividad de las estrategias aplicadas para mejorar la detección de flores de manzana, pudiendo ofrecer una mejora considerable en el rendimiento de una solución concreta aplicable en contextos agrícolas reales, que permita el conteo de flores mediante el procesamiento en diferido de videos capturados por robots terrestres.

Palabras clave: Detección de flores, Flores, Flores de manzana, Centros florales, Slicing Aided Hyper Inference, Agricultura de precisión, Redes convolucionales

Índice general

Capítulo 1

Introducción

El conteo de flores es un proceso crucial en la gestión agrícola, especialmente en cultivos como el manzano, donde el manejo adecuado de la floración impacta directamente en la productividad y la calidad de la cosecha. En particular, el conteo de flores permite implementar con mayor eficacia los métodos de raleo, un procedimiento esencial que busca eliminar el exceso de flores en desarrollo para mejorar el tamaño y calidad de los frutos finales. El raleo adecuado no solo optimiza la calidad del producto, sino que también reduce la competencia por recursos entre los frutos, incrementando así la uniformidad en el tamaño y forma de este.

Tradicionalmente, el conteo de flores o centros florales ha sido una tarea manual, tediosa y sujeta a errores humanos, afectando en consecuencia, la calidad y rentabilidad de la producción. En este contexto, la automatización mediante inteligencia artificial se presenta como una solución prometedora. Los sistemas de visión por computadora, combinados con algoritmos de aprendizaje automático, permiten el desarrollo de herramientas capaces de identificar objetos en imágenes de manera precisa y rápida, volviendo la detección de flores en manzanos mediante estas técnicas un área para investigar. Las dificultades que presenta la detección de flores de manzano en montes de frutales son varias; entre ellas se destacan la variabilidad en las condiciones de iluminación, la oclusión parcial de las flores por ramas u otras flores, la similitud cromática entre las flores y el fondo, y el tamaño reducido de las flores. En respuesta a estos desafíos, se han comenzado a aplicar técnicas avanzadas de visión por computadora, como redes neuronales convolucionales (del inglés, *convolutional neural networks*, CNN) y arquitecturas especializadas en segmentación y detección de objetos con un éxito moderado (Khaki y Wang, 2019); este proyecto busca explorar estrategias para mejorar la efectividad de los modelos de detección de flores de manzano.

El contenido del informe está estructurado de la siguiente manera, en el capítulo 2 se desarrolla el marco teórico que sustenta este trabajo, abordando los fundamentos de la visión por computadora, las redes neuronales convolucionales y el modelo YOLO como modelo representativo para la detección de objetos. El capítulo 3 presenta un relevamiento de la literatura existente, con el análisis

de trabajos previos relevantes que abordan la detección y el conteo de flores, y que sirven de referencia para este estudio. El capítulo 4 constituye el núcleo de este trabajo, en el cual se describen en detalle las estrategias propuestas para mejorar la detección de objetos pequeños, con especial énfasis en la aplicación del marco de trabajo SAHI. El capítulo 5 expone la experimentación realizada, donde se presentan los ensayos desarrollados, las técnicas implementadas y los resultados obtenidos. Finalmente, en el capítulo 6 se discuten las conclusiones alcanzadas y se proponen posibles líneas de trabajo futuro.

Capítulo 2

Marco Teórico

En este capítulo se presenta una revisión de los conceptos fundamentales y los avances recientes en los campos de la visión por computadora y el aprendizaje profundo, con énfasis en su aplicación a la detección y conteo de objetos pequeños en imágenes. El objetivo es proporcionar un marco conceptual que permita comprender las tecnologías empleadas en este proyecto, así como identificar las principales limitaciones de los enfoques tradicionales y modernos frente a tareas de complejidad visual, como la identificación de estructuras florales en montes de frutales. Para ello, se abordan en primer lugar los fundamentos de la visión por computadora, seguidos por una descripción de las redes neuronales y, en particular, de las redes neuronales convolucionales, que constituyen la base de las arquitecturas actuales de detección de objetos.

2.1. Visión por computadora

La visión por computadora es la base de la mayoría de los sistemas automáticos de detección y conteo de objetos. Esta disciplina combina algoritmos que permiten a las máquinas interpretar imágenes y videos, buscando simular las capacidades visuales del ser humano. Tradicionalmente, los enfoques basados en características han sido la norma en la detección de objetos, estos métodos clásicos incluyen técnicas como la segmentación de imágenes, la detección de bordes como el operador de Canny (Canny, 1986), la transformación de Hough para la detección de formas geométricas (Duda y Hart, 1972), y la extracción de características locales mediante descriptores como SIFT (Scale-Invariant Feature Transform) (Lowe, 2004), SURF (Speeded-Up Robust Features) (Bay y cols., 2006) o HOG (Histogram of Oriented Gradients) (Dalal y Triggs, 2005), técnicas que permiten representar regiones relevantes de una imagen de forma compacta y discriminativa (Szeliski, 2022).

Posteriormente, los clasificadores como SVM (Support Vector Machines) (Cortes y Vapnik, 1995) o Random Forests (Breiman, 2001) eran entrenados sobre estas representaciones para reconocer objetos. Aunque efectivos en ciertos

contextos, estos métodos suelen ser sensibles a cambios en las condiciones de iluminación, la perspectiva o la presencia de ruido visual, lo que limita su precisión y aplicabilidad en escenarios del mundo real (Lowe, 2004; Dalal y Triggs, 2005).

Con el avance de la computación y la disponibilidad de grandes volúmenes de datos, el enfoque dominante se ha desplazado hacia las redes neuronales profundas, particularmente las redes neuronales convolucionales. Estas redes han demostrado un rendimiento significativamente superior en tareas de clasificación, detección y segmentación de objetos, al aprender representaciones jerárquicas directamente de los datos (Rawat y Wang, 2017).

Este cambio de paradigma ha sido posible gracias al uso del aprendizaje automático supervisado, que consiste en entrenar modelos utilizando conjuntos de datos anotados, donde cada entrada está asociada a una salida esperada. En este proceso, el modelo aprende a mapear entradas a salidas mediante la minimización de una función de pérdida, que cuantifica el error de las predicciones y guía la actualización de los parámetros de la red durante el entrenamiento.

2.2. Redes neuronales convolucionales

Las redes neuronales son modelos computacionales inspirados en la estructura y funcionamiento del cerebro humano. Estos sistemas están compuestos por unidades de procesamiento interconectadas, llamadas "neuronas artificiales"(figura 2.1), que trabajan colectivamente para aprender patrones complejos a partir de datos (Goodfellow y cols., 2016). Su desarrollo ha evolucionado desde contribuciones fundamentales como el perceptrón de Rosenblatt (1958), hasta las arquitecturas profundas actuales.

El perceptrón es la forma más simple de una neurona artificial. Consiste en una unidad computacional con pesos ajustables w y un sesgo b. Dado un vector de entrada x, el perceptrón calcula una suma ponderada y aplica una función de activación para producir una salida binaria. Aunque revolucionario, el perceptrón presentaba limitaciones, como su incapacidad para resolver problemas no linealmente separables (ej. la función XOR), lo que motivó el desarrollo de redes multicapa.

La introducción de capas ocultas en los perceptrones multicapa permite modelar relaciones no lineales. Un avance clave para el desarrollo de las redes neuronales multicapa fue el algoritmo de retropropagación (Rumelhart y cols., 1986), que calcula gradientes para ajustar los pesos mediante la estrategia de descenso de gradiente estocástico (del inglés, *stochastic gradient descent*), un método iterativo que busca minimizar una función de error actualizando los pesos en dirección opuesta al gradiente, con el fin de aproximarse al mínimo de dicha función.

Las redes neuronales convolucionales surgieron como una solución especializada para el procesamiento de datos con estructura espacial, particularmente imágenes. Su arquitectura fue formalizada por LeCun y cols. (1989) con la creación de LeNet-5, una de las primeras CNN aplicadas con éxito al reconocimiento

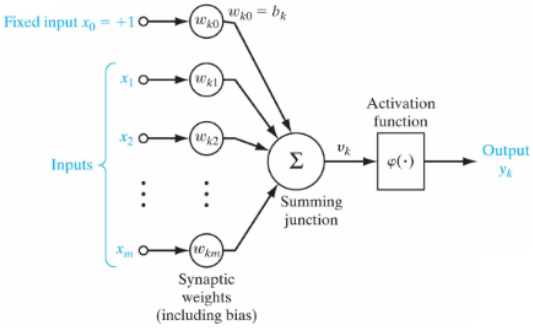

Figura 2.1: Modelo de neurona en una red neuronal (Haykin, 2008)

de dígitos escritos a mano. La innovación clave de las CNN radica en su capacidad para aprender características jerárquicas mediante operaciones locales y compartición de pesos, reduciendo significativamente los parámetros en comparación con redes totalmente conectadas.

Para comprender el funcionamiento de las redes neuronales convolucionales y el papel que desempeñan sus distintos componentes, resulta esencial tener una visión general de su arquitectura básica. Estas redes se organizan en una estructura jerárquica que combina diferentes tipos de capas, convolucionales, de pooling y completamente conectadas, las cuales trabajan de manera integrada para extraer progresivamente características relevantes de los datos de entrada y realizar tareas específicas, como la clasificación. La figura 2.2 presenta un esquema general de una red neuronal convolucional, que servirá como referencia para el análisis detallado de cada componente en las secciones siguientes.

2.2.1. Capas convolucionales

Las capas convolucionales son el componente principal de las CNN. Estas capas aplican filtros (también llamados kernels) que se desplazan sobre la imagen de entrada para detectar características locales como bordes, texturas o patrones más complejos. Cada filtro está diseñado para responder a características específicas, y a medida que la imagen pasa por múltiples capas convolucionales, la red es capaz de reconocer patrones cada vez más abstractos y complejos (Zeiler y Fergus, 2014). Una ventaja clave de las capas convolucionales es que reducen significativamente el número de parámetros en comparación con las redes totalmente conectadas. Esto se debe a que los filtros se aplican localmente y comparten pesos a lo largo de toda la imagen, lo que no solo hace que el modelo sea más eficiente computacionalmente, sino que también ayuda a prevenir el sobreajuste, es decir, cuando el modelo se adapta demasiado a los datos de entrenamiento y pierde capacidad de generalización y, por tanto, de desempeñarse correctamente con datos no vistos.

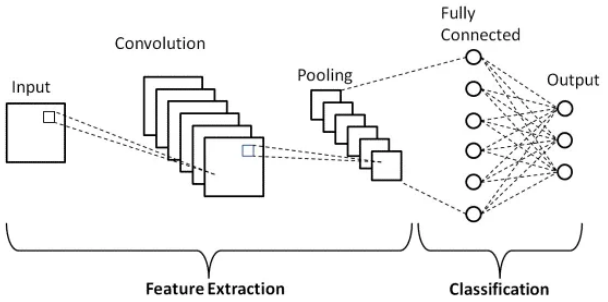

Figura 2.2: Esquema general de una red neuronal convolucional. La arquitectura se divide en dos etapas principales: extracción de características, donde las capas convolucionales y de pooling procesan la imagen de entrada para identificar patrones jerárquicos; y clasificación, donde las capas completamente conectadas transforman las características extraídas en una predicción final. (Datascientest, 2021)

2.2.2. Capas de pooling

Las capas de pooling se utilizan para reducir progresivamente la dimensionalidad espacial de los mapas de características (representaciones intermedias donde la red almacena patrones relevantes detectados en la imagen) generados por las capas convolucionales. Esto ayuda a disminuir la cantidad de cómputo requerido en las capas posteriores y hace que el modelo sea más robusto frente a pequeñas variaciones en la posición de las características en la imagen. El tipo más común de pooling es el max-pooling (figura 2.3 y 2.4), que selecciona el valor máximo dentro de una ventana deslizante. Esta operación preserva las características más destacadas mientras descarta información menos relevante (Zeiler y Fergus, 2014). Otra variante es el average-pooling, que calcula el promedio de los valores en la ventana, siendo útil para suavizar el ruido en los datos. Estas capas no tienen parámetros aprendibles y su función principal es reducir el tamaño espacial de las representaciones, manteniendo la información esencial.

2.2.3. Capas completamente conectadas

Las capas completamente conectadas generalmente se ubican al final de la arquitectura CNN y tienen como objetivo consolidar toda la información extraída por las capas anteriores para realizar tareas específicas, como la clasificación (asignación de la entrada a una categoría definida) o la regresión (estimación de un valor continuo). A diferencia de las capas convolucionales, donde las conexiones son locales, en estas capas cada neurona está conectada a todas las neuronas de la capa anterior. Estas capas reciben como entrada los mapas de características y los transforman en un vector unidimensional que representa las

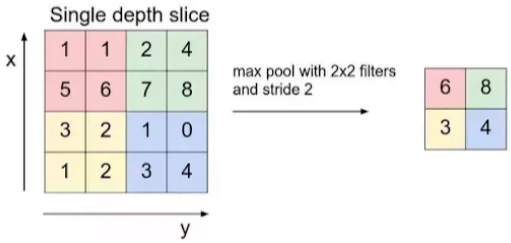

Figura 2.3: Procesamiento del max-pool 2x2 stride 2. La ventana de filtro se desplaza dos píxeles hacia la derecha (stride/paso = 2) y recupera en cada paso el "argmax" que corresponde al valor más alto de los 4 valores de píxeles. (Datascientest, 2021)

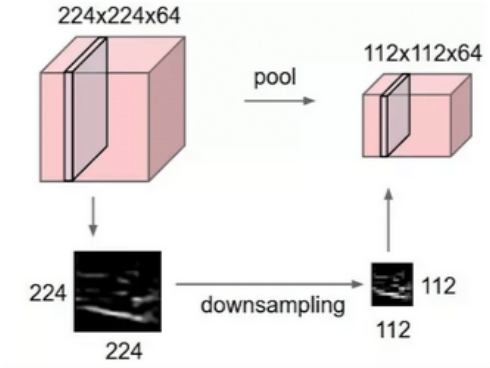

Figura 2.4: Ejemplo de aplicación de filtro max-pool 2x2 stride 2 a una imagen 224x224 pixeles. Logramos reducir el tamaño de la entrada para la red significativamente reduciendo el número de parámetros que manejan las capas posteriores. (Datascientest, 2021)

características de alto nivel de la entrada. Luego, este vector se procesa mediante una combinación lineal de pesos y sesgos, seguida de una función de activación adecuada para la tarea en cuestión (Zeiler y Fergus, 2014).

2.2.4. Entrenamiento de redes neuronales y funciones de activación

El entrenamiento de redes neuronales convolucionales constituye un proceso iterativo y sistemático que permite a estos modelos aprender patrones complejos a partir de datos. Este proceso se fundamenta en el algoritmo de retropropagación del error, desarrollado originalmente por Rumelhart y cols. (1986), que ha

demostrado ser particularmente efectivo para el entrenamiento de arquitecturas profundas. Durante el entrenamiento, la red procesa los datos de entrada en dos fases diferenciadas. En la fase de propagación hacia adelante, los datos fluyen a través de las distintas capas de la red, donde se aplican transformaciones lineales y no lineales mediante operaciones de convolución y funciones de activación. Posteriormente, en la fase de retropropagación, el error calculado en la capa de salida se distribuye hacia atrás a través de la red, permitiendo el ajuste progresivo de los pesos. La elección del optimizador resulta crucial para el éxito del entrenamiento. Mientras que el descenso de gradiente estocástico sigue siendo una opción válida, optimizadores más sofisticados como Adam, propuesto por Kingma y Ba (2014), han ganado popularidad por su capacidad para adaptar automáticamente las tasas de aprendizaje de cada parámetro, acelerando la convergencia y mejorando el rendimiento final del modelo.

Las funciones de activación juegan un papel fundamental en la capacidad expresiva de la red. La función ReLU (Nair y Hinton, 2010) se ha convertido en el estándar de facto para las capas intermedias de las CNN, debido a su simplicidad computacional y su efectividad para mitigar el problema de los gradientes que desaparecen (del inglés, *vanishing gradients*), un fenómeno que ocurre durante la etapa de entrenamiento, en el que los gradientes se hacen tan pequeños al propagarse por la red que afectan la capacidad de aprendizaje de las capas iniciales.

Para garantizar la capacidad de generalización del modelo y prevenir el sobreajuste, se pueden implementar diversas estrategias. Una de ellas es la desactivación aleatoria de nodos (del inglés, dropout), que ha demostrado ser útil en ciertas arquitecturas al forzar a la red a aprender representaciones más robustas y menos dependientes de características específicas de los datos (Srivastava y cols., 2014). Sin embargo, en redes convolucionales modernas, como las utilizadas en tareas de detección de objetos, es más común recurrir a técnicas como la normalización por lotes (del inglés, *batch normalization*), que normaliza las salidas de cada capa para mantenerlas dentro de un rango estable durante el entrenamiento, lo que acelera y estabiliza el proceso de aprendizaje (Ioffe y Szegedy, 2015). También se emplean estrategias como el aumento de datos o la penalización del crecimiento excesivo de los pesos mediante el decaimiento de pesos (del inglés, *weight decay*), una forma de regularización integrada en muchos optimizadores, que ayuda a mantener los parámetros bajo control y favorece una mejor generalización del modelo.

La evaluación continua del modelo durante el entrenamiento es esencial. Utilizar conjuntos de validación independientes permite ajustar los hiperparámetros de manera más precisa, mientras que el conjunto de verificación, reservado exclusivamente para la etapa final, ofrece una estimación más realista del rendimiento del modelo en condiciones del mundo real.

2.3. Redes de detección de objetos

Las redes de detección de objetos son un subcampo de la visión por computadora que se enfoca en identificar y localizar objetos dentro de imágenes o videos. Estas redes no solo clasifican las entidades presentes en la imagen, sino que también generan información sobre su localización, típicamente a través de un cuadro delimitador (del inglés, *bounding box*). Las técnicas modernas de detección de objetos utilizan redes neuronales convolucionales para aprender a extraer características jerárquicas de las imágenes, lo que les permite realizar detección de objetos de manera eficiente y precisa. A lo largo de los años, varias arquitecturas de redes neuronales se han propuesto para mejorar la precisión y la velocidad de la detección, desde métodos más clásicos hasta los enfoques más recientes basados en redes profundas.

Existen principalmente dos arquitecturas predominantes para la detección de objetos: los métodos basados en propuestas de regiones y los métodos de una sola etapa. Entre los primeros se encuentran R-CNN (Regions with CNN features) y sus variantes Fast R-CNN y Faster R-CNN (Girshick y cols., 2014), que emplean un esquema de dos etapas: primero generan posibles regiones de interés, y luego las clasifican y ajustan sus límites. Este enfoque tiende a lograr altos niveles de precisión, aunque generalmente a costa de una mayor complejidad computacional y tiempos de inferencia más largos que los métodos de una sola etapa, debido al procesamiento separado de múltiples regiones candidatas.

Por otro lado, las arquitecturas de una sola etapa, como YOLO (You Only Look Once) o SSD (Single Shot MultiBox Detector) (W. Liu y cols., 2016), integran en un solo paso la predicción de las clases y la localización de los objetos. Esta estrategia permite realizar detección de manera significativamente más rápida, manteniendo niveles de precisión competitivos, lo que resulta especialmente útil en aplicaciones en tiempo real.

2.3.1. You Only Look Once (YOLO)

You Only Look Once (YOLO) (Redmon y cols., 2016) es una de las arquitecturas más destacadas en el campo de la detección de objetos en tiempo real. A diferencia de los métodos basados en propuestas de regiones, YOLO adopta un enfoque de detección unificada, lo que significa que predice la clase y la localización de los objetos en una sola pasada a través de la red. Esta arquitectura se basa en la idea de dividir la imagen en una cuadrícula, donde cada celda de la cuadrícula predice los cuadros delimitadores y sus probabilidades de clase correspondientes.

YOLO convierte el problema de detección de objetos en un problema de regresión, donde la red produce una matriz de salida que incluye los valores de las coordenadas de los cuadros delimitadores y las probabilidades de clase. La red procesa toda la imagen de una sola vez, lo que le permite realizar detección en tiempo real con un alto rendimiento.

Existen múltiples variantes de la familia YOLO, entre las que se destacan versiones como YOLOv3, YOLOv5, YOLOv8 y YOLOv11, cada una de las cua-

les ha introducido mejoras progresivas orientadas a optimizar el rendimiento, la velocidad de inferencia y la eficiencia computacional de la arquitectura. En la actualidad, YOLOv8 constituye el representante más relevante de esta familia de modelos, ya que integra de manera consolidada los avances logrados en versiones anteriores e incorpora nuevas optimizaciones en su arquitectura. La figura 2.5 presenta la estructura general de YOLOv8, que mantiene los principios fundamentales de la serie YOLO.

Los componentes principales de la red son el tronco (del inglés, *backbone*), el cuello (del inglés, *neck*) y el cabezal (del inglés, *head*); el tronco es la parte de la red encargada de extraer las características de bajo nivel y alto nivel de la imagen. Se trata de una red convolucional profunda que toma la imagen de entrada y aprende representaciones jerárquicas a través de sus capas. Los detalles extraídos en esta fase son cruciales para la posterior localización y clasificación de objetos. El tronco se ha optimizado para ser más eficiente y rápido sin sacrificar la capacidad de extracción de características importantes. Este componente se basa en redes convolucionales avanzadas, como CSPDarknet (Wang y cols., 2020), que mejoran la eficiencia computacional al reducir la cantidad de parámetros necesarios sin perder calidad en las representaciones.

El cuello de la red, actúa como un conector entre el tronco y el cabezal. Su función principal es fusionar las características extraídas por el tronco a diferentes escalas y facilitar la detección de objetos a varias resoluciones. En YOLOv8, el cuello se basa en una estructura de PANet (Path Aggregation Network) (S. Liu y cols., 2018) que mejora la capacidad de la red para captar detalles tanto de objetos pequeños como grandes, mediante la agregación de características a diferentes niveles de la red. Este componente es esencial para mejorar la precisión de la detección, especialmente cuando se enfrentan a objetos de diferentes tamaños y en diferentes posiciones dentro de la imagen.

Finalmente, el cabezal es la parte de la red encargada de realizar la predicción final. El cabezal toma las características procesadas por el tronco y el cuello, y genera las salidas finales que incluyen las coordenadas de los cuadros delimitadores y las probabilidades de las clases de los objetos detectados. YOLOv8 utiliza un cabezal optimizado para proporcionar predicciones más precisas, mejorando la localización de los objetos y reduciendo los errores comunes en las versiones anteriores. Además, el diseño del cabezal permite realizar detecciones en tiempo real, manteniendo un alto rendimiento tanto en precisión como en velocidad.

YOLOv8 ofrece modelos preentrenados sobre el conjunto de datos COCO (Lin y cols., 2014), que contiene 80 clases de objetos comúnmente utilizados en tareas de detección. Estos modelos nos dan la posibilidad de aplicar la técnica de transferencia de pesos (del inglés, *weight transferring*) que consiste en inicializar el modelo con pesos previamente entrenados en un conjunto de datos grande y genérico, permitiendo que el modelo reutilice características de bajo nivel ya aprendidas, como bordes, texturas y formas, y se adapte más rápidamente a las características específicas del nuevo dominio. Las variantes disponibles de YOLOv8 están especificadas en la tabla 2.1, y varían principalmente en cuanto a la cantidad de parámetros y la complejidad computacional, lo que influye

directamente en la precisión, velocidad de inferencia y uso de memoria. Esta diversidad permite seleccionar el modelo más adecuado según las restricciones del sistema y los objetivos de la tarea.

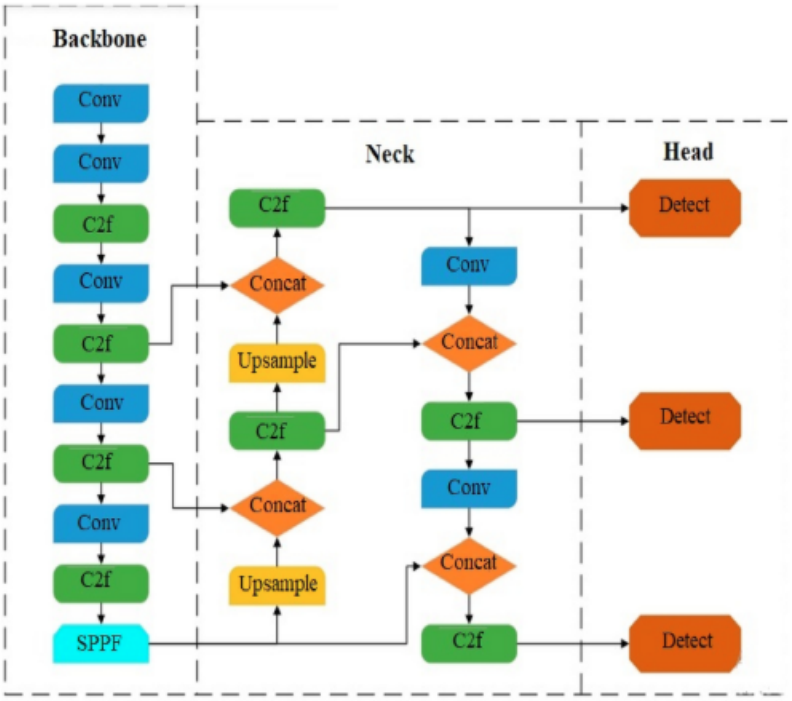

Figura 2.5: Arquitectura YOLOv8 (Elhanashi y cols., 2024). Se divide en tres bloques principales: **tronco** (*backbone*), **cuello** (*neck*) y **cabezal** (*head*). En el **cabezal**, las cajas azules representan capas convolucionales (*Conv*) que extraen características básicas de la imagen, mientras que las cajas verdes corresponden a módulos *C2f*, los cuales permiten una extracción más eficiente y profunda de características. La caja turquesa al final del tronco indica un módulo *SPPF* (Spatial Pyramid Pooling Fast), que mejora la captura de contexto a múltiples escalas. En el **cuello**, los diamantes anaranjados marcan operaciones de concatenación (del inglés, *Concat*) entre diferentes niveles de la red y las cajas amarillas indican operaciones de muestreo ascendente (del inglés, *Upsample*). Estas operaciones permiten fusionar información semántica de distintas resoluciones. En el **cabezal**, las cajas naranjas etiquetadas como *Detect* generan las predicciones finales, incluyendo las cajas delimitadoras, las clases detectadas y las puntuaciones de confianza asociadas.

Modelo	Tamaño (píxeles)	mAPval 50–95	Velocidad CPU ONNX (ms)	Parámetros (M)
YOLOv8n	640	37.3	80.4	3.2
YOLOv8s	640	44.9	128.4	11.2
YOLOv8m	640	50.2	234.7	25.9
YOLOv8l	640	52.9	375.2	43.7
YOLOv8x	640	53.9	479.1	68.2

Tabla 2.1: Comparación de variantes del modelo YOLOv8 entrenadas sobre el conjunto de datos COCO. Se presentan valores de precisión (mAPval 50–95), velocidad de inferencia en CPU (ONNX) y cantidad de parámetros. Fuente: (Ultralytics, 2025)

Capítulo 3

Revisión de antecedentes

En este capítulo se presenta el análisis de una selección de estudios relevantes en el área de detección, conteo y clasificación de flores en árboles. El documento de estado del arte, elaborado en el marco de este proyecto, se encuentra disponible en el repositorio referenciado en Tessore (2025a) y contiene un relevamiento más amplio de trabajos relacionados. La selección aquí presentada fue realizada en base al grado de correlación con los objetivos del presente proyecto, destacando las tecnologías aplicadas, las metodologías empleadas y los resultados obtenidos en cada caso.

3.1. Deep Learning based flower detection and counting in highly populated images

El estudio (Estrada y cols., 2024) busca comparar la efectividad de dos enfoques de visión por computadora para la detección y el conteo de flores de durazno, con el objetivo de asistir en la estimación temprana del rendimiento frutal, ya que al igual que en el caso de los manzanos, la cantidad de flores constituye un buen indicador del número potencial de frutos. Los materiales utilizados en el estudio (código fuente y conjunto de datos) no son de carácter público.

El trabajo propone evaluar y comparar el desempeño de dos enfoques principales para la estimación de la cantidad de flores en imágenes. Por un lado, se utilizan redes de detección basadas en YOLO, empleando las versiones más recientes disponibles hasta el momento (v5, v7 y v8) para detectar flores individualmente en las imágenes. Por otro lado, se explora el uso de mapas de densidad, donde en lugar de identificar cada flor de forma explícita, se entrena una red neuronal convolucional multi-columna para predecir un mapa de densidad que refleje la concentración de objetos en distintas regiones de la imagen. Este tipo de arquitectura ha demostrado ser eficaz en escenarios con alta variabilidad de escalas, ya que su diseño con ramas paralelas que operan con distintos tamaños de filtro permite capturar información contextual a múltiples niveles,

mejorando la estimación en escenas complejas.

El dataset utilizado para el estudio (figura 3.1) está compuesto por imágenes extraídas de vídeos grabados en duraznales de la región de Cataluña, España, utilizando una cámara de iPhone 5 con una resolución de 1280 × 720 píxeles. Se recolectaron un total de 600 imágenes, divididas en 400 para entrenamiento, 100 para validación y 100 para verificación.

Figura 3.1: Ejemplo imagen perteneciente al dataset utilizado en el trabajo de Estrada y cols. (2024)

Se especifica el uso de técnicas de preprocesamiento de datos para minimizar la interferencia del fondo (como árboles y el cielo), se utilizó un filtro basado en el espacio de color HSV, que destacó el color característico de las flores de durazno. Esto ayudó a mejorar la detección al eliminar algunos elementos no deseados del fondo. Se aplicó también corrección gamma a las imágenes para enfatizar el contraste entre las flores y los elementos del fondo.

Con el objetivo de evaluar la precisión de los modelos en la tarea de conteo de flores, se emplean métricas propias del ámbito de la regresión. En particular, se utilizan las siguientes: el Error Cuadrático Medio (RMSE, ecuación 3.1), el Error Absoluto Medio (MAE, ecuación 3.2) y el Error Porcentual Medio (%Error, ecuación 3.3). Estas métricas permiten cuantificar la discrepancia entre las cantidades predichas por el modelo y la verdad de referencia (del inglés, *ground truth*), proporcionando así una base objetiva para comparar el desempeño entre distintos enfoques.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2} \tag{3.1}$$

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i| \tag{3.2}$$

$$\%\text{Error} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{|\hat{y}_i - y_i|}{y_i}\right) \times 100 \tag{3.3}$$

donde:

- n es el número total de imágenes evaluadas,
- $\hat{y}_i$ representa el conteo predicho de flores para la imagen i,
- y_i es el conteo real (verdad de referencia).

YOLO model		Metrics					
Version	Size	Precision	Recall	mAP	MAE	MSE	%Err
v5	n	0.65	0.4	0.5	198	252.77	39.49
	s	0.66	0.4	0.51	215.4	277.8	48.4
	m	0.7	0.4	0.52	204.2	266.8	40.8
	l	0.71	0.42	0.54	208.5	273.6	41.5
	x	0.71	0.42	0.54	198.2	252.7	39.4
v7	tiny	0.6219	0.3616	0.4	173.5	240.3	36.8
	m	0.656	0.3805	0.282	161.7	229.4	34.2
	x	0.78	0.52	0.39	152.7	212.9	29.7
	d6	0.71	0.42	0.33	160.4	227.3	33.5
	e6	0.71	0.43	0.34	162.8	225.4	32.6
v8	n	0.752	0.442	0.61	170.85	244.5	35.83
	s	0.754	0.451	0.62	169.25	248.6	34.52
	m	0.756	0.442	0.6	173.32	245.83	35.82
	l	0.751	0.596	0.44	166.52	242.25	33.2
	x	0.779	0.456	0.615	160.39	221.02	31.45

Figura 3.2: Resultados de evaluación de modelos de detección. (Estrada y cols., 2024)

Los resultados, presentados en la figura 3.2 para modelos basados en YOLO y en la figura 3.3 para mapas de densidad, evidencian que, en escenarios con alta densidad de objetos como las flores de durazno, es más eficaz predecir el mapa de densidad que detectar cada instancia individual. El enfoque basado en mapas de densidad obtuvo un %Error de 9.98, un MAE de 19.13 y un RMSE de 69.69, posicionándose como la alternativa más precisa para el conteo. En contraste, el mejor desempeño entre los modelos de detección lo presentó YOLOv7x, con un %Error de 29.7, un MAE de 152.7 y un RMSE de 212.9, lo que representa una disminución significativa en la precisión frente al método de mapas de densidad.

Los autores concluyen que condiciones comunes en el entorno agrícola, como la alta densidad de flores, la oclusión, los fondos complejos y la similitud en

Hyperparameters						Metrics		
Model	F	GS	BS	LR	TR	MAE	RMSE	%Err
1	N	7	1	0.001	0.0005	38.63	59.8	10.95
2				0.005	0.0005	105.27	131.46	24.02
3				0.01	N/A	N/A	N/A	N/A
4			4	0.001	0.0005	109.59	143.84	25.48
5				0.005	0.0005	109.82	149.14	25.4
6				0.01	N/A	N/A	N/A	N/A
7		15	1	0.001	0.0005	83.75	106.61	19.63
8				0.005	0.000725	48.66	76.06	14.27
9				0.01	N/A	N/A	N/A	N/A
10			4	0.001	0.0005	86.71	118.11	22.25
11				0.005	0.0005	138.73	161.34	45.48
12				0.01	N/A	N/A	N/A	N/A
13	Y	7	1	0.001	0.0005	61.83	88.83	15.33
14				0.005	0.0005	88.52	119.64	21.16
15				0.01	N/A	N/A	N/A	N/A
16			4	0.001	0.0005	92.92	124.31	22.17
17				0.005	0.0005	81.77	113.82	20.86
18				0.01	N/A	N/A	N/A	N/A
19		15	1	0.001	0.0005	39.13	69.69	9.98
20				0.005	0.00054	77.41	109.86	19.1
21				0.01	N/A	N/A	N/A	N/A
22			4	0.001	0.0005	183.38	213.31	42.1
23				0.005	0.0005	109.72	152.7	26.22
24				0.01	N/A	N/A	N/A	N/A

Figura 3.3: Resultados de evaluación de modelos de mapas de densidad entrenados con diferentes configuraciones de hiperparámetros. Se presentan los valores de: **F** (uso del filtro de color en imágenes: N = no, Y = sí), **GS** (tamaño del filtro Gaussiano usado para generar el mapa de densidad), **BS** (tamaño del lote), **LR** (tasa de aprendizaje) y **TR** (tasa de umbral aplicado para eliminar regiones de baja densidad). Los modelos corresponden a una red neuronal convolucional multi-columna con arquitectura fija y variación de estos hiperparámetros. (Estrada y cols., 2024)

color y textura, dificultan el desempeño de métodos de detección como YOLO. A pesar de su eficacia en tareas generales de detección, estos modelos presentan limitaciones al identificar instancias individuales en escenas densas, lo que afecta la precisión del conteo (Estrada y cols., 2024).

3.2. Flower Detection and Counting Using CNN for Thinning Decisions in Apple Trees

La motivación principal del artículo (Kiktev y Kutyrev, 2023b) es mejorar el proceso de raleo de flores en manzanos mediante un método automatizado de monitoreo basado en aprendizaje automático. El objetivo del trabajo es superar los desafíos asociados a la detección de flores a través del entrenamiento de redes neuronales profundas.

Se optó por la arquitectura YOLOv8 debido a su rapidez y precisión para la detección de objetos en tiempo real. Esta versión introduce mejoras respecto a modelos anteriores, tanto en el tronco como en el sistema de detección

multiescala implementado en la cabeza de la red.

Para el entrenamiento, se capturaron 3000 imágenes con una cámara Sony Alpha ILCE-7M3 en resolución de 4000x2672 píxeles, clasificadas en dos categorías: capullo y floración. Un ejemplo representativo se muestra en la figura 3.4. Las imágenes fueron etiquetadas manualmente con cuadros delimitadores mediante la plataforma Roboflow (Nelson y cols., 2020), y luego se aplicaron técnicas de aumento de datos, como rotaciones, ajustes de matiz, adición de ruido, desenfoque, volteo y variaciones de brillo, con el fin de aumentar la robustez del modelo frente a cambios en la iluminación y orientación. Este proceso permitió duplicar el conjunto original a 6000 imágenes, distribuidas en conjuntos de entrenamiento (70 %), validación (20 %) y verificación (10 %). Cabe destacar que únicamente un subconjunto de 100 imágenes con resolución 1920x1080 del conjunto utilizado en este estudio se encuentra disponible públicamente en la plataforma Roboflow (Kiktev y Kutyrev, 2023a).

Figura 3.4: Imagen perteneciente al dataset utilizado en el trabajo Kiktev y Kutyrev (2023b) con las predicciones realizadas por la red entrenada.

Se empleó la técnica de aprendizaje por transferencia utilizando pesos preentrenados en el conjunto de datos COCO, lo cual permitió acelerar el entrenamiento y mejorar la capacidad de generalización del modelo. Según los parámetros de entrenamiento reportados, el modelo fue entrenado durante 124 épocas mediante descenso de gradiente, con una tasa de aprendizaje inicial de 0.01 y un tamaño de lote de 16. Para la evaluación del desempeño se utilizaron métricas como precisión, recall y precisión media promedio (mAP) (métricas definidas en

la sección 5.1). Los resultados por clase se presentan en la tabla 3.1.

El modelo alcanzó un AP@50 de 0.691 para la detección de la clase floración y de 0.448 para la clase capullo, situando el mAP@50 del modelo en 0.569.

Métrica	Clase capullo	Clase floración
Precisión	0.55	0.72
Recall	0.62	0.68
Precisión Media (AP)	0.448	0.691

Tabla 3.1: Métricas extraídas de los resultados obtenidos por el modelo entrenado para detectar y clasificar las flores de manzana. (Kiktev y Kutyrev, 2023b)

3.3. YO-AFD: an improved YOLOv8-based deep learning approach for rapid and accurate apple flower detection

El objetivo del estudio (Y. Zhao y cols., 2023) es mejorar la detección rápida y precisa de flores de manzana, lo cual es crucial para la predicción del período de floración y la estimación anticipada del rendimiento. Para ello, los autores proponen YO-AFD, una arquitectura mejorada basada en YOLOv8, optimizada para superar desafíos como iluminación variable, oclusiones y variabilidad morfológica de las flores.

El conjunto de datos utilizado en este estudio se encuentra disponible en Xu y cols. (2024) y fue recolectado en un huerto comercial de manzanos (variedad Granny Smith), ubicado en Wuquan, China. Las imágenes fueron capturadas durante el período de floración, los días 28 y 29 de marzo de 2021, bajo diversas condiciones de iluminación natural, incluyendo luz solar directa, contraluz y sombreado parcial. Para garantizar la diversidad en las condiciones de adquisición, las imágenes se tomaron en dos franjas horarias: de 9:00 a 11:30 h y de 14:30 a 17:00 h.

Se utilizó un dispositivo móvil iPhone 11 Pro Max equipado con una cámara de alta resolución (3024×4032 píxeles) para capturar un total de 2115 imágenes. Un ejemplo representativo de estas imágenes puede verse en la figura 3.5. Estas imágenes fueron posteriormente redimensionadas a 605×807 píxeles, a fin de equilibrar la conservación de detalles y la eficiencia computacional.

En total, se etiquetaron 43.031 instancias de flores de manzano, estas se dividen en dos clases flores y capullos. El conjunto de datos se dividió aleatoriamente en subconjuntos de entrenamiento (1270 imágenes), validación (425 imágenes) y verificación (420 imágenes). Los parámetros empleados para el entrenamiento son los indicados en la tabla 3.2

El modelo propuesto, denominado YO-AFD (YOLO-based Apple Flower Detector), se basa en la arquitectura YOLOv8n que corresponde a la variante más liviana de la serie YOLOv8, y ha sido diseñada con un fuerte énfasis en la

Figura 3.5: Imagen perteneciente al dataset utilizado en el estudio Y. Zhao y cols. (2023).

eficiencia computacional. Su arquitectura optimizada reduce significativamente el número de parámetros y operaciones, lo que permite su implementación en entornos con recursos limitados sin comprometer severamente el rendimiento. A pesar de su tamaño compacto, mantiene una capacidad de detección robusta.

El código fuente utilizado en el proyecto no se encuentra abierto al público. Se especifica el diseño de un nuevo módulo de atención denominado ISAT (Inverted Residual and Synergistic Attention Transformer), con el objetivo de mejorar la capacidad del modelo para enfocar selectivamente las regiones más relevantes de una imagen. Este tipo de mecanismos permite que la red aprenda a destacar automáticamente las características más útiles para la detección, mientras suprime el ruido o las regiones irrelevantes.

El módulo ISAT fue incorporado en el cuello de la red, que es responsable de combinar la información extraída en distintos niveles de la red. Específicamente, ISAT reemplaza el bloque estándar conocido como C2f, dando lugar a un nuevo componente denominado C2f-IS. Esta modificación mejora la forma en que se fusionan las características provenientes de diferentes escalas. En segundo lugar, se implementó una función de pérdida basada en Focaler Intersection over Union (FIoU), en reemplazo de la CIoU convencional. Esta función introduce un mapeo

lineal entre valores de IoU dentro de un rango específico, permitiendo que el modelo focalice el aprendizaje en muestras de dificultad intermedia. La función FIoU se define de la siguiente manera:

$$\text{FIoU} = \begin{cases} 0, & \text{si IoU} < d \\ \frac{\text{IoU}-d}{u-d}, & \text{si } d \leq \text{IoU} \leq u \\ 1, & \text{si IoU} > u \end{cases}$$

$$\mathcal{L}_{\text{FIoU}} = 1 - \text{FIoU}$$

Donde IoU representa el solapamiento entre la caja predicha y la caja real, y $[d, u] \subseteq [0, 1]$ define el intervalo donde la función aplica un descenso lineal. Este esquema permite asignar mayor peso a muestras con solapamientos moderados, fomentando el aprendizaje en casos de dificultad media y evitando que el modelo se enfoque únicamente en los ejemplos más fáciles o más extremos.

Tabla 3.2: Hiperparámetros utilizados durante el entrenamiento del modelo YOLO-AFD

Hiperparámetro	**Valor**
Learning rate	0.01
Batch size	16
Momentum	0.937
Weight decay	0.0005
Workers	8
Iteraciones	200 épocas

El modelo logró una precisión del 89.4 %, un recall de 87.9 % y un mAP@50 de 94.1 %, demostrando una alta capacidad de detección de flores.

Capítulo 4

Parte Central

Uno de los principales desafíos en visión por computadora es la detección de objetos pequeños. En particular, el trabajo relevado en la sección previa Estrada y cols. (2024) pone en manifiesto la dificultad de los modelos de detección de objetos para conseguir valores altos de precisión media cuando las instancias florales se solapan o resultan difíciles de distinguir. La alternativa presentada de uso de mapas de densidad generados por redes convolucionales para el conteo de las flores demuestra mejoras sustanciales en el rendimiento, obteniendo errores de conteo inferiores en comparación con los enfoques basados en detección de objetos. En la misma línea, el relevamiento realizado del trabajo Kiktev y Kutyrev (2023b) refuerza esta observación, reportando valores moderados de precisión media en la detección de flores de manzano con YOLOv8,

Por otra parte, el trabajo Y. Zhao y cols. (2023) se especifica una arquitectura YOLOv8 modificada, entrenada para detección de flores de manzano que logra una precisión media de 94.1 %, un valor muy por encima de los resultados de los otros estudios relevados; cabe destacar que este resultado se obtiene utilizando un conjunto de imágenes menos exigente, caracterizado por una menor cantidad de flores por imagen y una interferencia del entorno considerablemente menor.

En este contexto, considerando el potencial demostrado en el trabajo Y. Zhao y cols. (2023) por la red de detección de objetos entrenada para detectar flores, cuando éstas se presentan más visibles e identificables, resulta razonable la posibilidad de explorar técnicas e implementaciones que permitan a las redes de detección mejorar su desempeño en escenarios más complejos, caracterizados por mayor densidad de flores y oclusión.

Las limitaciones para lograr buenos rendimientos de las redes entrenadas para la detección de flores en imágenes con entornos complejos se deben, en gran parte, al reducido tamaño de las flores, lo que implica que cada instancia esté representada por una cantidad limitada de píxeles. Los criterios DORI (Detect, Observe, Recognize, Identify) (European Committee for Electro-technical Standardization, 2012) son un estándar ampliamente utilizado en videovigilancia y análisis de imágenes para establecer los requisitos mínimos de calidad necesarios para distintos niveles de interpretación visual. Este enfoque define la cantidad

mínima de píxeles que un objeto debe ocupar dentro de una imagen para ser correctamente procesado según el nivel deseado. Para alcanzar el nivel de detección, se considera que un objeto debe ocupar al menos el 10 % de la altura de la imagen, mientras que para el reconocimiento, este valor se eleva al 20 %. Estos umbrales permiten evaluar objetivamente la capacidad de un sistema de visión por computadora para interpretar escenas visuales en función de la resolución disponible y el tamaño relativo de los objetos de interés. El 10 % de la altura de una imagen HD es 108 píxeles, por lo que los objetos que ocupen al menos 108 píxeles de alto pueden ser detectados.

En el desarrollo del informe cuando clasificamos a los objetos en una imagen según su tamaño nos basamos en la evaluación de métricas COCO (Lin y cols., 2014). La clasificación de tamaño según las anotaciones (cuadros delimitadores) del objeto se basa en el cálculo de su área.

$$\text{Área} = (x_{\max} - x_{\min}) \cdot (y_{\max} - y_{\min}) \tag{4.1}$$

donde $(x_{\min}, y_{\min})$ y $(x_{\max}, y_{\max})$ representan las coordenadas del cuadro delimitador del objeto. A partir de este cálculo, se asigna una categoría de tamaño de acuerdo con los siguientes umbrales definidos por el protocolo COCO:

- **Pequeños (small)**: área menor a $32\,\text{px} \times 32\,\text{px}$ (es decir, menos de $1024\,\text{px}^2$ de área)
- **Medianos (medium)**: área entre $32\,\text{px} \times 32\,\text{px}$ y $96\,\text{px} \times 96\,\text{px}$ (de $1024\,\text{px}^2$ a $9216\,\text{px}^2$ de área)
- **Grandes (large)**: área mayor o igual a $96\,\text{px} \times 96\,\text{px}$ (es decir, $9216\,\text{px}^2$ o más)

Las redes neuronales convolucionales han demostrado ser altamente eficaces en la detección de objetos en imágenes, pero presentan limitaciones al trabajar con objetos de pequeño tamaño como se puede apreciar en la figura 4.1.

Modelos de detección como YOLOv3, EfficientDet y YOLOv4 demuestran que la precisión media promedio para objetos pequeños es considerablemente menor en comparación con objetos de mayor tamaño. Por ejemplo, para la arquitectura EfficientDet-D0, el AP@50 (métrica definida formalmente en la sección 5.1) para objetos pequeños es de apenas 12 %, mientras que para objetos grandes alcanza el 51 %, reflejando una diferencia de casi cinco veces (Bochkovskiy y cols., 2020).

Uno de los principales desafíos radica en la pérdida de detalle en las primeras etapas de las redes convolucionales. Esta pérdida ocurre principalmente como consecuencia de operaciones de pooling, que reducen progresivamente el tamaño de los mapas de características. Cuando los objetos de interés ocupan pocas celdas en dichos mapas, su representación espacial puede ser insuficiente, lo que dificulta su identificación. El entrenamiento de una red neuronal se basa en una función de pérdida que calcula la diferencia entre la predicción y la verdad de referencia; cuando los objetos de interés son muy pequeños, la red puede recibir una señal de entrenamiento más débil o ruidosa, debido a la escasa

Method	Backbone	Size	FPS	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
YOLOv4: Optimal Speed and Accuracy of Object Detection									
YOLOv4	CSPDarknet-53	416	96 (V)	41.2%	62.8%	44.3%	20.4%	44.4%	56.0%
YOLOv4	CSPDarknet-53	512	83 (V)	43.0%	64.9%	46.5%	24.3%	46.1%	55.2%
YOLOv4	CSPDarknet-53	608	62 (V)	**43.5%**	**65.7%**	47.3%	**26.7%**	46.7%	53.3%
EfficientDet: Scalable and Efficient Object Detection [77]									
EfficientDet-D0	Efficient-B0	512	62.5 (V)	33.8%	52.2%	35.8%	12.0%	38.3%	51.2%
EfficientDet-D1	Efficient-B1	640	50.0 (V)	39.6%	58.6%	42.3%	17.9%	44.3%	56.0%
EfficientDet-D2	Efficient-B2	768	41.7 (V)	43.0%	62.3%	46.2%	22.5%	**47.0%**	**58.4%**
EfficientDet-D3	Efficient-B3	896	23.8 (V)	45.8%	65.0%	49.3%	26.6%	49.4%	59.8%
Learning Spatial Fusion for Single-Shot Object Detection [48]									
YOLOv3 + ASFF*	Darknet-53	320	60 (V)	38.1%	57.4%	42.1%	16.1%	41.6%	53.6%
YOLOv3 + ASFF*	Darknet-53	416	54 (V)	40.6%	60.6%	45.1%	20.3%	44.2%	54.1%
YOLOv3 + ASFF*	Darknet-53	608×	45.5 (V)	42.4%	63.0%	**47.4%**	25.5%	45.7%	52.3%
YOLOv3 + ASFF*	Darknet-53	800×	29.4 (V)	43.9%	64.1%	49.2%	27.0%	46.6%	53.4%

Figura 4.1: Comparativa rendimiento de modelos para detección de objetos con distintos tamaños (pequeños, medianos y grandes). (Bochkovskiy y cols., 2020)

representación espacial en los mapas de características, lo que puede dificultar el aprendizaje efectivo de dichos objetos (Z. Zhao y cols., 2020).

Las principales líneas de investigación orientadas a mejorar el rendimiento de las redes neuronales convolucionales en la detección de objetos pequeños pueden dividirse en dos enfoques complementarios. Por un lado, se han propuesto modificaciones en la arquitectura de las redes, como se discute en la sección 3.3, con el objetivo de optimizar la extracción de características a diferentes escalas. Por otro lado, existen estrategias que abordan el problema desde el punto de vista del entrenamiento o la inferencia. Como podría ser la técnica de segmentación en mosaicos (del inglés, *tiling*), que consiste en dividir las imágenes originales en parches más pequeños para mejorar la visibilidad y detección de objetos diminutos durante el entrenamiento o la predicción.

En esta línea, ha ganado relevancia en los últimos años el marco de trabajo SAHI (Slicing Aided Hyper Inference) (Akyon y cols., 2022), el cual ha sido aplicado exitosamente en dominios donde la detección de objetos pequeños es crítica. Por ejemplo, ha demostrado ser eficaz en el ámbito médico, específicamente en la detección de huevos parasitarios en imágenes microscópicas (Tureckova y cols., 2022), así como en la detección de peatones y vehículos en imágenes aéreas de alta resolución (H. Zhang y cols., 2023).

Siguiendo esta línea de investigación, en este trabajo proponemos la aplicación del marco de trabajo SAHI para abordar el problema de detección de flores de manzano en escenarios complejos, con el objetivo de evaluar su eficacia y aplicabilidad en este dominio específico.

4.1. SAHI (Slicing Aided Hyper Inference)

Con el fin de optimizar el rendimiento de los modelos en la detección de flores de manzano, se propone el uso del marco de trabajo SAHI. Este introduce

dos técnicas, *Slicing Aided Fine Tuning* y *Slicing Aided Hyper Inference*, que permiten mejorar las detecciones sin modificar la arquitectura base del modelo (Akyon y cols., 2022).

4.1.1. Slicing Aided Fine Tuning (SF)

La técnica *Slicing Aided Fine Tuning* (SF) consiste en generar un conjunto de datos aumentado a partir de un dataset base mediante la creación de subimágenes o parches extraídos de las imágenes originales. El procedimiento implica una primera etapa de ajuste fino (del inglés, *fine-tuning*) sobre un modelo utilizando el dataset elegido para el caso de uso, seguida de una segunda etapa de ajuste fino sobre el dataset aumentado, que resulta de incorporar los parches al dataset base. Este enfoque permite incrementar la representación de los objetos pequeños durante el proceso de aprendizaje de la red, ya que al trabajar con parches de menor tamaño que la imagen original, los objetos pasan a representar una proporción mayor de la imagen total. Contribuyendo también a reducir la pérdida de información que suele producirse durante el proceso de redimensionamiento interno que realizan las redes de detección para adaptar las imágenes al tamaño requerido por la arquitectura, lo que permite preservar mejor la información visual de los objetos o incluso incrementarla si el tamaño de entrada de la red es mayor que el de los parches. En conjunto, esto favorece un aprendizaje más efectivo de objetos que, de otro modo, estarían subrepresentados o poco definidos en las representaciones internas de la red. La figura 4.2 ilustra el procedimiento aplicado.

Para la generación de parches se parte de un proceso iterativo en el cual se recorre cada imagen del dataset, dividiéndola en sub-imágenes que cubren la totalidad del área original mediante una cuadrícula definida por los siguientes parámetros:

- $\mathbf{H_p}$: altura de cada parche (en píxeles).
- $\mathbf{W_p}$: ancho de cada parche (en píxeles).
- **O**: superposición entre parches, expresada como un porcentaje en el rango de 0 a 100 %.

La superposición entre parches apunta a evitar pérdida de detecciones en los bordes debido a los cortes entre los parches. El valor que deben tomar estos parámetros dependerá del problema particular, tamaño de los objetos y calidad de la imagen.

4.1.2. Slicing Aided Hyper Inference (SAHI)

El método de inferencia SAHI, ilustrado en la figura 4.3, sigue un enfoque similar al de la técnica SF, pero se aplica en la fase de predicción en lugar de la etapa de entrenamiento. Al momento de la inferencia, se toman parches de la imagen en base a los parámetros definidos $\mathbf{H_p}$, $\mathbf{W_p}$, **O**, se realiza inferencia

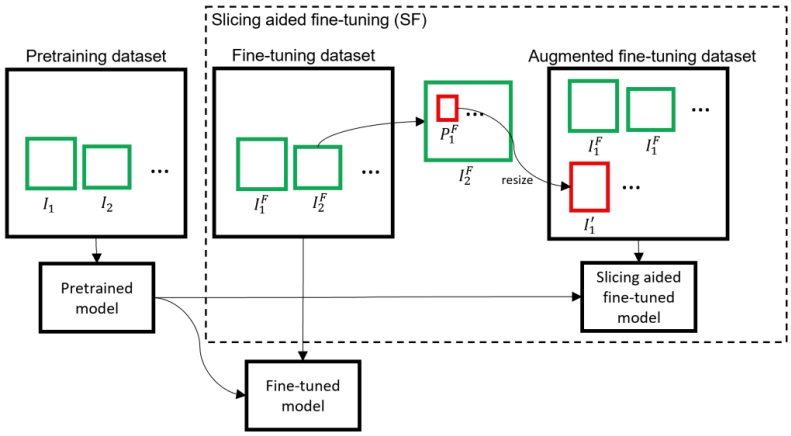

Figura 4.2: Esquema del proceso de Slicing Aided Fine Tuning propuesto en Akyon y cols. (2022). A partir de un modelo previamente entrenado (Pretrained model), se realiza una primera etapa de ajuste fino con un conjunto de datos ($I_1^F, I_2^F, \ldots$) (Fine-tuning dataset). Luego, se generan parches (P_1^F) a partir del conjunto inicial de ajuste fino (Fine-tuning dataset), los cuales se incorporan a un nuevo conjunto aumentado (Augmented fine-tuning dataset). Este nuevo conjunto es utilizado en una segunda etapa de entrenamiento, dando como resultado el modelo final (Slicing aided fine-tuned model).

sobre cada parche individualmente y sobre la imagen original. El proceso de mapear las detecciones de los parches a la imagen original conlleva la aplicación de técnicas de reconciliación para detecciones duplicadas.

Durante la inferencia con SAHI, al realizar la predicción sobre una imagen completa y sobre múltiples parches solapados de la misma, es común que un mismo objeto sea detectado más de una vez, generando predicciones duplicadas o solapadas. Para abordar este problema, se aplican estrategias de post-procesamiento que permiten fusionar o suprimir dichas detecciones.

Para determinar si múltiples predicciones corresponden a la misma instancia de objeto, se emplean métricas de superposición que cuantifican el grado de coincidencia espacial entre las cajas delimitadoras, puede utilizarse Intersection over Union (sección 5.1) o Intersection over Smaller (sección 5.2). Se considera que una detección es duplicada si supera cierto umbral basado en el cálculo de la métrica, 0.5 por ejemplo. En tareas de detección de objetos pequeños, como es el caso de las flores de manzano, la métrica IoU podría no ser del todo representativa para indicar una detección duplicada. Esto se debe a que IoU da valores bajos cuando una de las cajas tiene un área pequeña y, por tanto, su intersección también es reducida; al dividir esta intersección sobre la unión, la cual puede ser considerablemente mayor si la otra caja es más grande, se obtienen valores engañosamente bajos, incluso cuando las cajas se solapan de

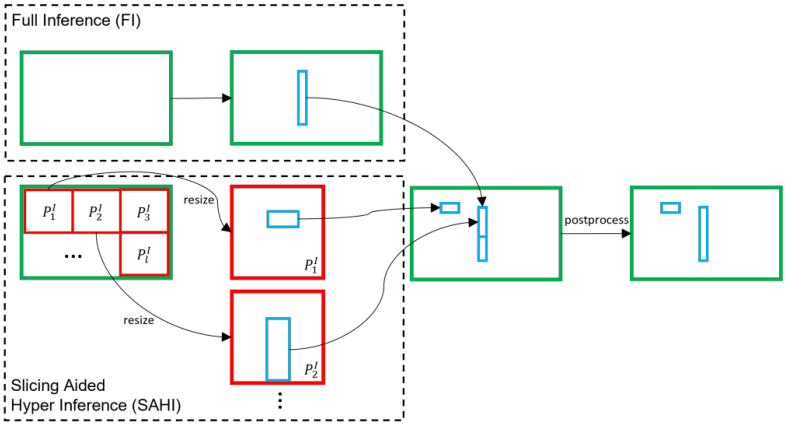

Figura 4.3: Esquema del proceso de inferencia Slicing Aided Hyper Inference (Akyon y cols., 2022). La parte superior representa la inferencia convencional completa (Full Inference, FI), donde la imagen completa es procesada directamente por el modelo. SAHI (parte inferior) divide la imagen original en múltiples parches ($P_1^I, P_2^I, \ldots$), los cuales son redimensionados al momento de ser procesados por el modelo. Las predicciones generadas por cada parche son luego combinadas y reubicadas en la imagen original, generando un conjunto unificado de detecciones. Finalmente, se aplica un paso de posprocesamiento (postprocess) para eliminar duplicaciones o inconsistencias, obteniendo las detecciones finales sobre la imagen

manera razonable. Por el contrario, la métrica IoS evalúa el solapamiento relativo tomando como referencia el área de la caja más pequeña entre las dos, lo que puede volverla más adecuada para objetos de baja escala. Al normalizar la intersección en función del tamaño del objeto más pequeño, IoS permite capturar con mayor sensibilidad cuándo una detección cubre de forma significativa el objeto objetivo, incluso si una de las cajas es considerablemente mayor a otra. Esta propiedad resulta interesante para nuestro caso de estudio por lo que se buscará evaluar diferencias en rendimiento variando entre Intersection over Union e Intersection over Smaller como métrica para la identificación de detecciones duplicadas durante el posprocesamiento de la inferencia con SAHI.

Una vez identificadas las predicciones duplicadas, se aplican estrategias de fusión para consolidarlas en una sola detección. El marco de trabajo propone también la utilización de diferentes algoritmos para fusionar detecciones duplicadas.

Non-Maximum Suppression (**NMS**), es un algoritmo que elimina predicciones redundantes seleccionando aquellas con la mayor puntuación de confianza y descartando las que se solapan por encima de un umbral definido (Neubeck y Van Gool, 2006).

Non-Maximum Merging (**NMM**), similar a NMS, pero en lugar de eliminar, fusiona las predicciones solapadas. Las cajas se combinan promediando sus posiciones y puntuaciones de confianza, útil para detecciones cercanas pero no idénticas (X. Zhang y Li, 2017).

Greedy Non-Maximum Merging (**GREEDYNMM**), una variante de NMM que fusiona las predicciones solapadas de forma greedy, priorizando aquellas con mayor puntuación de confianza antes de combinar las posiciones y valores (Kim y cols., 2019).

El marco de trabajo Slicing Aided Hyper Inference (SAHI) puede integrarse con cualquier flujo de inferencia de detección de objetos. Experimentos realizados con detectores como FCOS, VFNet y TOOD en los conjuntos de datos VisDrone (Deng y cols., 2018) y xView (Lam y cols., 2018), ambos compuestos por imágenes aéreas o satelitales de alta resolución que presentan escenas complejas con múltiples objetos pequeños (vehículos, personas, edificaciones, entre otros), han demostrado mejoras de hasta un 6.8 % en AP al aplicar esta técnica. Además, el uso de Slicing Aided Fine-Tuning aporta un incremento adicional del 14.5 % en AP para objetos pequeños, y la incorporación de una superposición del 25 % entre parches mejora aún más el desempeño en un 2.9 % AP (Akyon y cols., 2022).

4.2. Diseño de la solución

El flujo de trabajo seguido en este estudio se representa en la figura 4.4. Para facilitar el almacenamiento, organización y accesibilidad de los conjuntos de datos empleados, se recurrió a la plataforma Roboflow (Nelson y cols., 2020), la cual permite gestionar las versiones de los datasets y realizar tareas de preprocesamiento de forma centralizada.

El entrenamiento de los modelos se llevó a cabo mediante scripts desarrollados en el lenguaje de programación Python (versión 3.8), utilizando principalmente la biblioteca de Ultralytics (Ultralytics, 2024) para la implementación de YOLO y la de Roboflow para la gestión de los datos de entrada.

La generación de los nuevos conjuntos de datos resultantes de aplicar la estrategia SF sobre los datasets originales se realizó utilizando la librería SAHI (Yolcu y cols., 2021). Permitiendo la ejecución mediante línea de comandos para generar las imágenes.

Del mismo modo, la implementación de la estrategia SI y la posterior evaluación frente a los conjuntos de verificación se efectuaron mediante la ejecución de scripts y comandos que integran las herramientas mencionadas. Este procedimiento permitió medir métricas de desempeño sobre los resultados obtenidos.

El código para experimentar con el sistema se encuentra disponible en el repositorio referenciado en Tessore (2025b). Un instructivo detallado para su utilización se incluye en el anexo B.

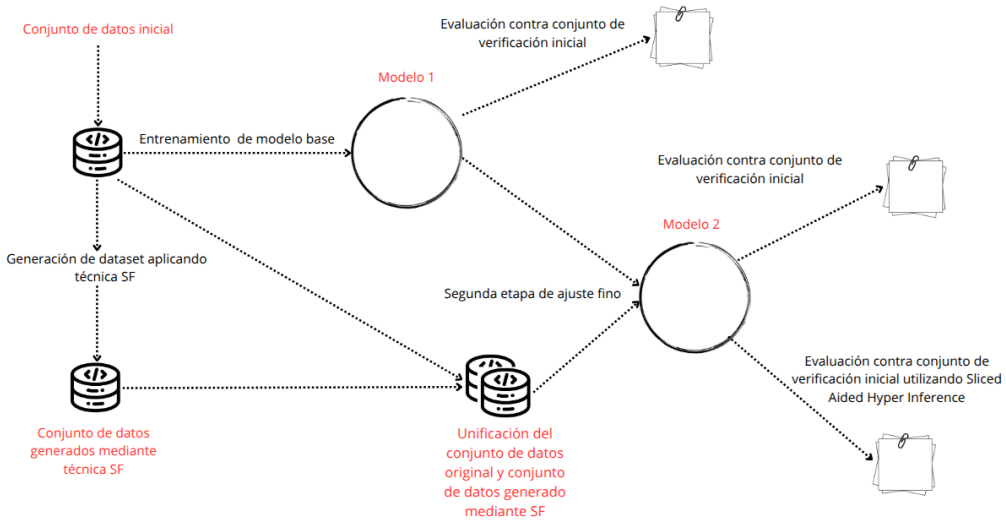

Figura 4.4: Esquema del flujo de trabajo propuesto. A partir del conjunto de datos inicial, se entrena un modelo (Modelo 1), el cual es evaluado contra el conjunto de verificación inicial. Paralelamente, se aplica la técnica SF sobre el conjunto inicial para obtener un nuevo conjunto de datos aumentado. Posteriormente, se realiza una unificación entre este conjunto generado y el conjunto inicial, sobre el cual se entrena un nuevo modelo (Modelo 2) en una segunda etapa de ajuste fino. Modelo 2 es evaluado contra el conjunto de verificación inicial utilizando el método Slicing Aided Hyper Inference y sin utilizarlo, permitiendo así comparar el impacto de la técnica SF y del posprocesamiento en el rendimiento final del sistema.

Capítulo 5

Experimentación

Este capítulo describe la metodología experimental empleada para evaluar el impacto del marco de trabajo *SAHI* en la detección de flores de manzano en montes frutales, caracterizados por la presencia de múltiples plantas con flores pequeñas, densamente agrupadas y parcialmente ocluidas. Se detallan las métricas utilizadas para cuantificar el rendimiento de los modelos, los conjuntos de datos seleccionados, así como la metodología seguida durante los entrenamientos y pruebas.

5.1. Métricas

Para medir la efectividad de los modelos desarrollados, se utilizan métricas estándar ampliamente aceptadas en el campo de la detección de objetos (Everingham y cols., 2010). Estas métricas permiten evaluar tanto la calidad de la localización de los objetos como la exactitud en su clasificación.

Una de las métricas fundamentales para evaluar la calidad de la localización en detección de objetos es el **IoU** (Intersection over Union) (Everingham y cols., 2010), que cuantifica la superposición entre la caja predicha por el modelo y la verdad de referencia (ecuación, 5.1). Un valor alto de IoU indica una mayor coincidencia espacial entre ambos. Para determinar si una predicción es considerada correcta, se calcula el IoU entre la caja predicha y la caja de referencia; si este valor supera un umbral predefinido (por ejemplo, 0.5), entonces la predicción se considera válida.

$$\text{IoU} = \frac{\text{Área de la Intersección}}{\text{Área de la Unión}} \tag{5.1}$$

Por otro lado, la métrica **IoS** (Intersection over Smaller) (L. Zhang y cols., 2016) ha sido propuesta como una alternativa al IoU, particularmente útil en el contexto de objetos pequeños. Esta medida evalúa la superposición entre la predicción y la verdad de referencia (ecuación, 5.2), pero normaliza con respecto al área menor entre ambas, lo que la hace más permisiva en casos donde los

objetos ocupan regiones reducidas. Se propone utilizar IoS como una alternativa en el posprocesamiento con SAHI, específicamente para identificar detecciones duplicadas.

$$\text{IoS} = \frac{\text{Área de la Intersección}}{\text{mín}(\text{Área de la Predicción}, \text{Área de Referencia})} \tag{5.2}$$

Precisión:

Mide qué tan exactas son las predicciones positivas del modelo (ecuación, 5.3). Es el porcentaje de predicciones positivas correctas entre todas las predicciones positivas realizadas por el modelo. Una alta precisión indica que cuando el modelo predice un objeto o clase, lo hace correctamente la mayoría de las veces.

Recall:

Mide qué tan bien el modelo puede encontrar todas las instancias positivas en los datos (ecuación, 5.4). Es el porcentaje de verdaderos positivos que el modelo logra detectar, en relación con todos los ejemplos reales positivos. Un alto recall indica que el modelo no está perdiendo muchas instancias positivas, es decir, detecta la mayoría de los objetos reales.

F1-score:

Es la media armónica entre la precisión y el recall (ecuación, 5.5). Esta métrica resume el equilibrio entre ambos, siendo particularmente útil cuando se busca un compromiso entre evitar falsos positivos y no perder verdaderos positivos. Un F1-score alto indica que el modelo mantiene una buena proporción de aciertos entre lo que predice como positivo y lo que realmente es positivo.

Precisión Promedio (AP):

Evalúa la precisión a lo largo de la curva de precisión vs. recall (o P-R curve) y calcula el área bajo esa curva (ecuación, 5.6). Esta métrica se basa en las predicciones ordenadas por la puntuación de confianza del modelo. La métrica **AP@0.5** refiere a calcular el AP utilizando un umbral de IoU de 0.5.

Media de Precisión Promedio (mAP):

Es el promedio de las AP calculadas sobre todas las clases o instancias en el dataset (ecuación, 5.7). La mAP es una métrica fundamental para evaluar el rendimiento global del modelo de detección de objetos.

$$\text{Precisión} = \frac{TP}{TP + FP} \tag{5.3}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{5.4}$$

$$\text{F1-score} = 2 \cdot \frac{\text{Precisión} \cdot \text{Recall}}{\text{Precisión} + \text{Recall}} \tag{5.5}$$

$$\text{AP} = \sum_{n} P_n (R_n - R_{n-1}) \tag{5.6}$$

$$\text{mAP} = \frac{1}{C} \sum_{c=1}^{C} AP_c \tag{5.7}$$

Donde:

- TP = Verdaderos Positivos
- FP = Falsos Positivos
- FN = Falsos Negativos
- P_n = Precisión en el n-ésimo punto de recall
- R_n = Recall en el n-ésimo punto de la curva
- C = Número de clases en el dataset
- AP_c = Precisión promedio para la clase c

Loc (Localization Error Ignored)

En el marco de este trabajo, el interés principal radica en evaluar la viabilidad de estrategias de detección orientadas al conteo preciso de flores de manzano en imágenes de montes de frutales, más que en lograr una localización exacta de cada instancia individual. La precisión con la que los modelos logran localizar las flores depende en gran medida del conjunto de datos utilizado y de su capacidad de entrenamiento. En este contexto, métricas tradicionales como mAP@50, que requieren una superposición mínima (IoU ≥ 0.5) entre la predicción y la verdad de referencia para considerar una detección como válida, pueden resultar estrictas. Esta exigencia se vuelve especialmente problemática en la detección de objetos pequeños, donde pequeñas desalineaciones entre las cajas, aun cuando la clase y la ubicación general sean correctas, pueden derivar en penalizaciones significativas durante la evaluación.

Para ofrecer otra perspectiva, se incorpora al análisis la métrica Loc (Localization error ignored), introducida en el estudio de errores en detección de objetos de Hoiem et al. (Hoiem y cols., 2012). Esta métrica considera como verdaderos positivos aquellas detecciones que identifican correctamente la clase y presentan cierto grado de solapamiento con la verdad de referencia. Loc clasifica como errores de localización aquellas predicciones cuyo IoU se encuentra en el rango [0.1, 0.5), es decir, aquellas detecciones que reconocen correctamente el objeto pero tienen un alineamiento parcial. En el presente trabajo, adoptamos el cálculo de la métrica con un umbral de IoU ≥ 0.1, lo que equivale a calcular el desempeño del modelo mediante mAP@0.1. La métrica ignora errores de localización y se enfoca en evaluar la capacidad del modelo para reconocer la presencia y clase del objeto, proporcionando una referencia útil sobre el potencial de detección del modelo en caso de mejorar la precisión espacial de las

predicciones. Esta aproximación resulta particularmente adecuada para este caso de estudio, donde el objetivo es estimar con precisión la cantidad total de flores, sin requerir una segmentación espacial precisa.

Adicionalmente, aunque con menor relevancia en el contexto de este trabajo, se consideran otras métricas complementarias también propuestas en (Hoiem y cols., 2012).

- **C75**: Corresponde a los resultados de AP obtenidos al aplicar un umbral más estricto de IoU $\geq 0{,}75$. Proporciona una visión más exigente sobre la precisión espacial de las predicciones.
- **Sim (Similar)**: Reporta los resultados de AP@50 tras ignorar falsos positivos que corresponden a supercategorías relacionadas. Esto suaviza penalizaciones cuando las confusiones ocurren entre clases similares.
- **Oth (Other)**: Refleja el AP@50 tras ignorar todas las confusiones entre categorías distintas, permitiendo evaluar la capacidad del modelo para detectar objetos sin considerar la clasificación fina.
- **BG (Background)**: Mide el AP@50 al eliminar los falsos positivos que surgen de predicciones sobre el fondo o regiones sin objeto.
- **FN (False Negatives)**: Presenta el AP@50 los resultados al excluir los falsos negativos, evaluando así qué tan bien el modelo realiza predicciones sin considerar los objetos no detectados.

Estas métricas fueron calculadas utilizando la implementación disponible en la librería SAHI (Yolcu y cols., 2021)

5.2. Datasets relevados

Se relevaron diversos conjuntos de datos públicos relacionados con flores de manzano y otras especies frutales en entornos agrícolas. La selección de los conjuntos utilizados para las pruebas se basó en su similitud con imágenes que potencialmente podría capturar el robot terrestre "Jacky" del grupo MINA de la Facultad de Ingeniería (Universidad de la República), en los campos experimentales del INIA Las Brujas (Uruguay), figura 5.1. En particular, se priorizaron conjuntos con imágenes tomadas a una distancia aproximada de 1.5 metros, que mostraran la planta completa, con alta densidad de floración y flores de tamaño pequeño.

Además, se consideró fundamental que las anotaciones fuesen de alta calidad y consistentes entre sí, de modo de garantizar una base confiable para el entrenamiento. También se dio preferencia a conjuntos de datos que incluyeran una proporción elevada de anotaciones correspondientes a objetos pequeños, según el criterio establecido por el estándar COCO (Lin y cols., 2014). Los conjuntos finalmente seleccionados se detallan a continuación.

Figura 5.1: Imagen del monte de plantación de manzano en Las Brujas

5.2.1. Dataset 1: Pear Computer Vision Project

Es un conjunto de datos anotado para detección de objetos, compuesto por 471 imágenes de montes de perales en floración, con una resolución de 1242×2208 píxeles (Roboflow, 2023). El conjunto de datos consta de 92891 anotaciones que incluyen una única clase correspondiente a flores de peral. A pesar de no tratarse específicamente de flores de manzano, la similitud morfológica entre ambas especies es considerable, como se ilustra en la figura 5.2. La principal motivación para incorporar este conjunto en los experimentos radica en la disposición de las flores: numerosas, parcialmente ocluidas y agrupadas, en un entorno complejo con interferencias visuales entre los árboles y el fondo. Además, la proporción de objetos pequeños en las anotaciones lo convierte en un recurso especialmente adecuado para los objetivos de este estudio (véase tabla 5.1).

Con el objetivo de analizar las anotaciones de este conjunto de datos en función de los criterios DORI establecidos en el capítulo 4, los cuales definen un umbral mínimo del 10 % de la altura de la imagen para detección y del 20 % para reconocimiento, se implementó un script para calcular la altura promedio de las anotaciones. El resultado obtenido indica que la altura promedio de las cajas delimitadoras es de 32.31 píxeles, lo que representa aproximadamente un 1.46 % de la altura de las imágenes (2208 píxeles). Este valor se encuentra considerablemente por debajo de los umbrales establecidos por DORI tanto para la detección como para el reconocimiento. Este resultado refuerza el interés de experimentar con este conjunto de datos, dado que representa un escena-

rio particularmente desafiante para los algoritmos de detección y especialmente adecuado para evaluar las estrategias propuestas en este trabajo.

Figura 5.2: Imagen y respectiva anotación con cuadros delimitadores ejemplo del Dataset 1 (Roboflow, 2023)

Tamaño	Cantidad anotaciones	Porcentaje del total
Pequeñas	53164	57.3 %
Medianas	39347	42.4 %
Grandes	380	0.4 %

Tabla 5.1: Cantidad de anotaciones por tamaño según estándar COCO para el Dataset 1

5.2.2. Dataset 2 : apple-flowers-flowering-rosebuds

Se trata de un conjunto de datos compuesto por 100 imágenes anotadas para detección de objetos mediante cajas delimitadoras (véase figura 5.3), con

una resolución de 1920×1080 píxeles (Kiktev y Kutyrev, 2023a), capturadas en montes de manzanos. El conjunto incluye 6310 anotaciones correspondientes a dos clases: "floración", que representa el 95.6 % del total, y "capullo", con el 4.4 %. Aunque se trata de un conjunto relativamente pequeño, cumple con los criterios establecidos previamente en cuanto a distancia de las flores, similitud del entorno y distribución de tamaños de las anotaciones (véase tabla 5.2).

Además, se analizó la altura promedio de las cajas delimitadoras en relación con los criterios DORI. El cálculo arrojó un valor de promedio de altura 27.60 píxeles para las anotaciones, lo que equivale aproximadamente al 2.56 % de la altura de las imágenes. Este valor se encuentra alrededor de cuatro veces por debajo del umbral establecido para la detección (10 %), lo que reafirma el carácter desafiante del conjunto para las tareas de detección de objetos. Este conjunto fue publicado en 2024 por Alexey Kutyrev, coautor del estudio referenciado en la sección 3.2, a través de la plataforma Roboflow, y constituye un subconjunto del dataset total utilizado en dicho estudio. Su inclusión en este trabajo no solo responde a su adecuación técnica, sino también a su valor como referencia comparativa para la evaluación de los resultados experimentales obtenidos.

Tamaño	Cantidad anotaciones	Porcentaje del total
Pequeñas	3743	59.3 %
Medianas	2526	40.0 %
Grandes	41	0.7 %

Tabla 5.2: Cantidad de anotaciones por tamaño según estándar COCO para el Dataset 2

5.3. Ensayos

La arquitectura seleccionada para los modelos de detección entrenados en los ensayos fue YOLOv8. En comparación con versiones anteriores como YOLOv3 y YOLOv5, YOLOv8 presenta mejoras sustanciales tanto en precisión como en velocidad de inferencia, según pruebas comparativas estandarizadas sobre conjuntos de datos como COCO (Ultralytics, 2025). Además, ofrece un ecosistema más completo que incluye herramientas integradas para entrenamiento y validación, lo que facilita su integración en distintos entornos de desarrollo. Si bien al momento de realizar este trabajo fue anunciada la versión YOLOv11 (Jocher y Qiu, 2024), esta se encuentra aún en etapa temprana de adopción, carece de validaciones extensivas y no presenta mejoras concluyentes en los principales indicadores de rendimiento respecto a YOLOv8. Por estas razones, se decidió emplear YOLOv8 como base para este estudio. Aunque existen otras arquitecturas con buen desempeño (por ejemplo, Faster R-CNN o EfficientDet), las variaciones relativas de rendimiento obtenidas tras la aplicación de las estrategias *Slicing Aided Fine Tuning* y *Slicing Aided Hyper Inference* pueden evaluarse de forma análoga en distintas arquitecturas, ya que no dependen de la red de detección de objetos subyacente.

Figura 5.3: Ejemplo imagen y anotaciones del Dataset 2. (Kiktev y Kutyrev, 2023a)

Con el fin de reducir los tiempos de entrenamiento y acelerar la convergencia durante el ajuste fino sobre los conjuntos de datos específicos de flores, implementamos la estrategia de transferencia de pesos. Como se describió en la sección 2.3.1, YOLOv8 facilita la aplicación de esta técnica mediante la disponibilización de modelos preentrenados sobre el conjunto de datos COCO. Se dispone de distintas variantes que difieren principalmente en la cantidad de parámetros, estas se presentan en la tabla 2.1. Teniendo en consideración que el

principal objetivo de este trabajo es evaluar estrategias para mejorar el rendimiento de YOLOv8 en la detección de flores de pequeño tamaño, se consideró que la versión YOLOv8n (nano) ofrece un buen balance entre rendimiento y eficiencia computacional. Esto se debe a que, al contar con una menor cantidad de parámetros en comparación con versiones más grandes como YOLOv8s, permite reducir significativamente los tiempos de entrenamiento sin sacrificar de manera importante la precisión del modelo. De esta manera, YOLOv8n facilita ciclos de entrenamiento y validación más rápidos, lo que resulta útil para poder comparar el desempeño de varios modelos.

La metodología adoptada para llevar a cabo las pruebas se organizó en varias etapas, siguiendo el esquema ilustrado en la figura 4.4. En primer lugar, se realizó una etapa de ajuste fino al modelo YOLOv8n, utilizando el conjunto de datos correspondiente al ensayo. Finalizado este proceso, el modelo fue evaluado utilizando el conjunto de verificación.

Como segunda etapa del procedimiento, se aplicó la técnica denominada *Slicing Aided Fine Tuning (SF)* al conjunto de datos empleado en el ajuste fino inicial, como se especifica en la sección 4.2. Este procedimiento consistió en dividir las imágenes del conjunto en parches de dimensiones $H_p \times W_p$, incorporando una superposición O entre parches adyacentes. Los valores específicos de H_p, W_p y O fueron determinados en función de las características del conjunto de datos y de las restricciones computacionales del experimento (véase anexo A).

Posteriormente, se realizó una nueva fase de ajuste fino sobre el modelo entrenado en la primera etapa, incorporando los parches generados mediante SF en el conjunto de entrenamiento. El resultado de este proceso fue un nuevo modelo optimizado (modelo SF).

El rendimiento del modelo SF se evaluó utilizando el mismo conjunto de verificación empleado previamente con el modelo inicial. En una primera instancia, se realizó una evaluación directa del modelo SF; en una segunda, se implementó la estrategia *Slicing Aided Hyper Inference (SI)* (sección 4.3) durante la etapa de inferencia y se evaluó nuevamente el desempeño. La optimización de los parámetros H_p, W_p y O para la técnica SI, ajustados según las características de cada conjunto de datos, se detalla en el anexo A.

La evaluación principal de los modelos se realizó utilizando la métrica mAP@50, que por ser un estándar permite una comparación directa con modelos del estado del arte. Su uso garantiza consistencia en la evaluación y facilita la interpretación de los resultados.

En cuanto a las estrategias de posprocesamiento para gestionar detecciones duplicadas con SAHI, presentadas en la sección 4.1.2, (NMS, NMM y GREEDYNMM), nos limitamos a usar NMS dado que la métrica mAP@50 requiere calcular la curva de precisión-recall variando el umbral de confianza del modelo en el intervalo [0.01, 0.99]. En este contexto, técnicas como NMM y sus variantes resultan ineficaces debido a la elevada cantidad de ruido y detecciones de baja calidad que el modelo genera en el extremo inferior de dicho rango, haciendo que fusionar detecciones, como proponen estos algoritmos, no sea una opción viable, mientras que suprimir detecciones, como propone NMS, ofrece resultados más coherentes.

En la tabla 5.3 pueden encontrarse los parámetros de entrenamiento utilizados para todos los modelos de los ensayos y en la tabla 5.4 los parámetros utilizados para las técnicas referentes al marco de trabajo SAHI, optimizados según lo indicado en el anexo A

Parámetro	**Valor**
Optimizador	AdamW
Tasa de aprendizaje	0.01
Número de épocas	300
Tamaño de imagen	640 píxeles
Momento	0.937
Decaimiento de los pesos	0.0005

Tabla 5.3: Parámetros para entrenamiento de modelos en los ensayos

Ensayo	**Slicing Aided Hyper Inference**	**Slicing Aided Fine Tuning**
1, 2 y 3	{Hp = 640px, Wp = 640px, O = 0.2}	{Hp = 640px, Wp = 640px, O = 0.2}

Tabla 5.4: Parámetros utilizados en los ensayos para *Slicing Aided Hyper Inference* y *Slicing Aided Fine Tuning*

Todos los entrenamientos se llevaron a cabo en la plataforma ClusterUY (Nesmachnow e Iturriaga, 2019). La evaluación de los modelos fue realizada utilizando como dispositivo una CPU AMD Ryzen 7 1700 Eight-Core Processor.

5.3.1. Ensayo 1

Para este experimento se utilizó el *Dataset 1* (Pear Computer Vision Project sección 5.2.1), el cual se dividió en tres subconjuntos: 331 imágenes para entrenamiento, 95 para validación y 45 para verificación.

Se entrenó un modelo YOLOv8 conforme a las especificaciones previamente detalladas en la sección 5.3. Los parámetros principales del entrenamiento se presentan en la tabla 5.3. El tamaño de imagen elegido (640 píxeles) permitió reducir el costo computacional y acelerar el proceso de entrenamiento. En consecuencia, las imágenes originales de tamaño 1242 $\times$ 2208 píxeles son redimensionadas a partir del valor de este parámetro, manteniendo la relación de aspecto; como resultado, la dimensión más larga (altura) se ajusta a 640 píxeles y el ancho se escala proporcionalmente, obteniéndose imágenes de aproximadamente 360 $\times$ 640 píxeles durante el entrenamiento. Este modelo se denominará en adelante modelo Base.

A partir del conjunto original de 471 imágenes, se aplicó la técnica *Slicing Aided Fine Tuning*, generando múltiples parches de tamaño 640px $\times$ 640px por cada imagen, con una superposición del 20 % ($O = 0.2$), hasta cubrir completamente cada imagen. Como resultado, se obtuvieron 7051 imágenes procesadas

a partir del conjunto original. Las imágenes generadas son asignadas al subconjunto correspondiente (entrenamiento, validación y verificación) dependiendo del conjunto al que pertenece la imagen original, asegurando que parches de imágenes del conjunto de verificación no sean utilizados para el entrenamiento.

Con este nuevo conjunto de datos, se realizó una segunda etapa de ajuste fino sobre el modelo Base. El modelo resultante de este entrenamiento será referido como modelo SF.

En la tabla 5.5 se presenta la comparación del rendimiento de los modelos entrenados, modelo Base, SF y SF con la estrategia *Slicing Aided Hyper Inference* (SI), evaluados sobre el conjunto de verificación original, utilizando las métricas mAP@50 y Loc (ver sección 5.1). Para el modelo SF con SI (denotado como SF + SI) los parámetros utilizados para la inferencia son los especificados en la tabla 5.4 (optimizados según anexo A).

Modelo	mAP@50	Loc	Tiempo de inferencia (ms)
Base	0.414	0.615	87
SF	0.506	0.688	89
SF + SI (IoU + NMS)	**0.553**	0.753	1535
SF + SI (IoS + NMS)	0.528	**0.787**	1538

Tabla 5.5: Evaluación y comparación de los resultados obtenidos por los modelos propuestos. Las expresiones entre paréntesis indican las características del posprocesamiento utilizado durante la inferencia con *SAHI*. La columna tiempo de inferencia indica el promedio de tiempo de inferencia por imagen procesada.

La implementación de la estrategia *Slicing Aided Fine-Tuning*, incluso sin aplicar *Slicing Aided Hyper Inference*, ya demuestra una mejora sustancial, con un incremento de 9 puntos porcentuales en la métrica mAP@50 respecto al modelo Base. El mejor desempeño global se alcanza al combinar SF con SI utilizando posprocesamiento IoU + NMS, logrando una mejora de 14 puntos porcentuales tanto en mAP@50 como en Loc. A modo de comparación, en las figuras 5.4 y 5.5 se presentan las curvas precisión-recall del modelo Base y modelo SF + SI (IoU + NMS), las métricas presentadas en los gráficos corresponden a las analizadas en la sección 5.1. En relación con el tiempo de inferencia promedio por imagen, los modelos que incorporan la estrategia SI presentan un aumento considerable en el costo computacional, alcanzando diferencias de hasta tres órdenes de magnitud respecto a aquellos que no la implementan.

Al desglosar los valores de las métricas según el tamaño de las anotaciones para estos dos modelos, figura 5.6 y 5.7, se observa que las mayores ganancias se concentran en los objetos clasificados como pequeños (small), con mejoras de hasta 18 puntos porcentuales en mAP@50 (C50) y Loc, lo cual evidencia la eficacia del enfoque propuesto para la detección de objetos pequeños. La distribución de anotaciones (flores) por grupo de área en el conjunto de datos es la especificada en la tabla 5.1

Con el objetivo de analizar en mayor profundidad las diferencias en la efectividad del modelo Base y el modelo SF + SI con posprocesamiento IoU + NMS,

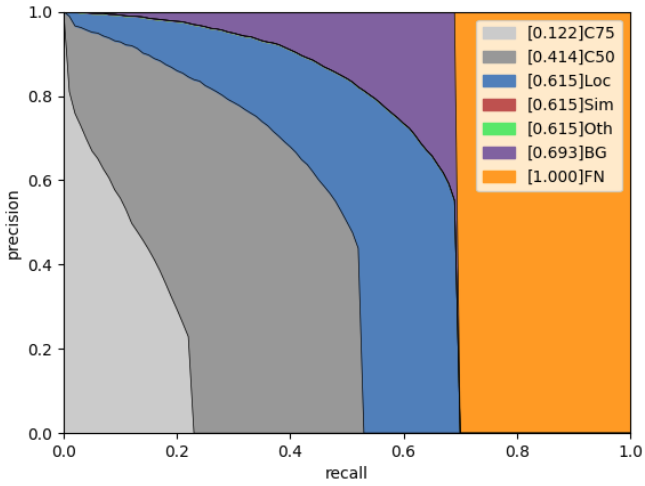

Figura 5.4: Curva precision-recall modelo Base entrenado en el ensayo 1.

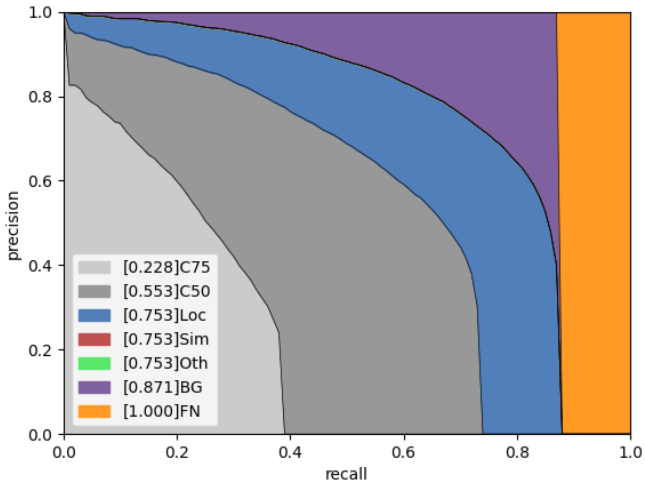

Figura 5.5: Curva precision-recall modelo SF + SI (IoU + NMS) entrenado en el ensayo 1.

se presenta en la tabla 5.6 un desglose de verdaderos positivos (TP), falsos positivos (FP), precisión, recall y F1-score. Estos valores se calcularon utilizando el valor umbral de confianza del modelo que maximiza el F1-score para cada modelo, considerando dos umbrales de IoU como referencia: 0.50 (valor empleado para el cálculo de mAP@50) y 0.10 (aplicado en el cálculo de la métrica Loc). El análisis se realiza sobre un total de 18209 anotaciones correspondientes al conjunto de verificación. Se desprende de los valores que, para ambos umbrales de IoU evaluados, el modelo SF + SI presenta una mejor proporción entre verda-

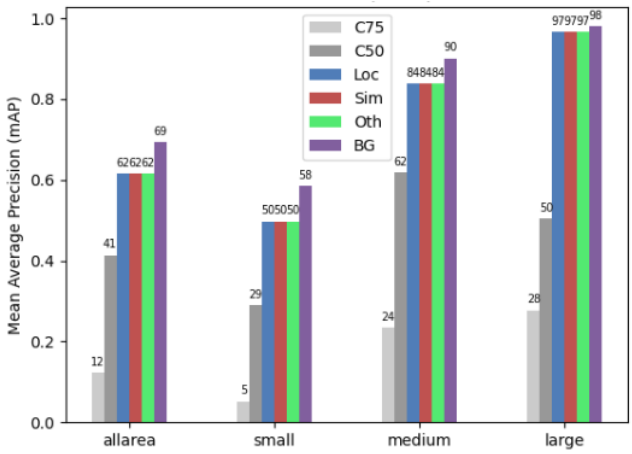

Figura 5.6: Métricas modelo Base discriminando por tamaño de anotación (Ensayo 1).

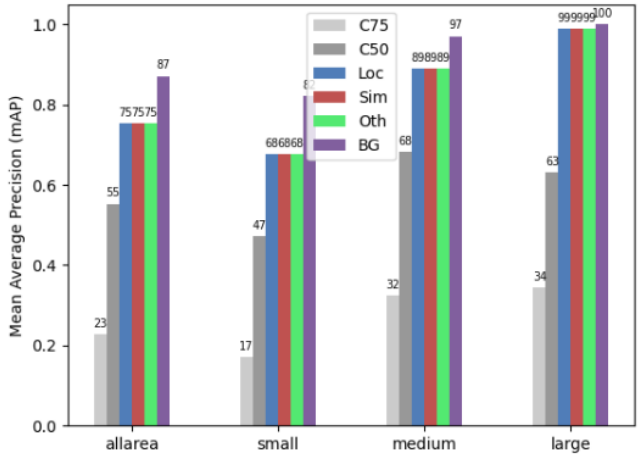

Figura 5.7: Métricas modelo SF + SI (IoU + NMS) discriminando por tamaño de anotación (Ensayo 1).

deros positivos y falsos positivos, lo que indica un balance más favorable entre sensibilidad y precisión respecto al modelo Base. Cabe resaltar, sin embargo, que ambos modelos exhiben un número elevado de falsos positivos en relación al total de anotaciones. Esta situación puede resultar problemática en aplicaciones de conteo, dado que no solo es esencial lograr una alta tasa de verdaderos positivos, sino también reducir significativamente las detecciones erróneas de instancias no presentes en la imagen, ya que estas pueden derivar en errores significativos de sobreconteo.

Modelo	**IoU**	TP	FP	Precisión	Recall	F1-score
Base	**0.50**	8517	6345	0.573	0.467	0.515
SF + SI (IoU + NMS)	**0.50**	10913	7505	0.592	0.599	0.595
Base	**0.10**	11092	4241	0.723	0.609	0.661
SF + SI (IoU + NMS)	**0.10**	13189	4742	0.735	0.724	0.729

Tabla 5.6: Análisis detallado de TP, FP, precisión, recall y F1-score para los modelos Base y SF + SI (IoU + NMS), bajo dos umbrales de IoU: 0.50 y 0.10. (Ensayo 1)

5.3.2. Ensayo 2

Para este experimento se utilizó el *Dataset 2* (apple-flowers-flowering-rosebuds sección 5.2.2), un subconjunto del conjunto original no disponible empleado en el estudio presentado en la sección 3.2, el cual contiene 100 imágenes (en contraste con las 3000 del estudio original). Aunque la cantidad de imágenes disponibles en este subconjunto es considerablemente menor, su utilización da la posibilidad de realizar una validación adicional e independiente, lo que resulta útil para corroborar la validez y consistencia de los resultados obtenidos, así como para evaluar posibles mejoras en el rendimiento del modelo.

La metodología de entrenamiento replicó la estructura del ensayo 1. Se entrenó un modelo YOLOv8 con los mismos parámetros previamente indicados, tomando como referencia el conjunto original de 100 imágenes. Este modelo se denomina aquí modelo Base.

A continuación, se aplicó la técnica *Slicing Aided Fine Tuning*, generando múltiples parches por imagen de 640 × 640 píxeles con una superposición del 20 % ($O = 0.2$), lo que resultó en un total de 900 imágenes. Con este nuevo conjunto se realizó una nueva etapa de ajuste fino adicional partiendo del modelo Base, obteniéndose el modelo SF.

Para la evaluación del modelo SF en la que se aplica *Slicing Aided Hyper Inference*, se utilizaron los parámetros $H_p = 640$ px, $W_p = 640$ px y $O = 0.2$ (véase anexo A).

En la tabla 5.7 se presenta la comparación del rendimiento de los modelos entrenados, modelo Base, SF y SF con la estrategia *Slicing Aided Hyper Inference*, evaluados sobre el conjunto de verificación original.

El modelo SF + SI con posprocesamiento basado en IoU + NMS obtuvo los mejores resultados, con mejoras de 7 y 4 puntos porcentuales en las métricas mAP@50 y Loc, respectivamente, en comparación con el modelo Base. La comparativa de curvas precisión-recall de ambos modelos puede verse en las figuras 5.8 y 5.9.

Al comparar los resultados obtenidos con los del estudio relevado en la sección 3.2, que reporta una precisión promedio para el modelo entrenado de 0.569, se evidencia que, pese a la disparidad en el volumen de datos de entrenamiento, nuestro modelo Base alcanza un mAP@50 comparable (0.547). Cabe resaltar que el modelo SF + SI (IoU + NMS) supera este valor de referencia establecido,

logrando un incremento de 6 puntos porcentuales en mAP@50.

Modelo	mAP@50	Loc	Tiempo de inferencia (ms)
Base	0.547	0.743	70
SF	0.556	0.762	71
SF + SI (IoU + NMS)	**0.621**	**0.781**	1031
SF + SI (IoS + NMS)	0.567	0.775	1022

Tabla 5.7: Evaluación y comparación de los resultados obtenidos por los modelos propuestos. Las expresiones entre paréntesis indican las características del posprocesamiento utilizado durante la inferencia con *SAHI*. La columna tiempo de inferencia indica el promedio de tiempo de inferencia por imagen procesada (Ensayo 2).

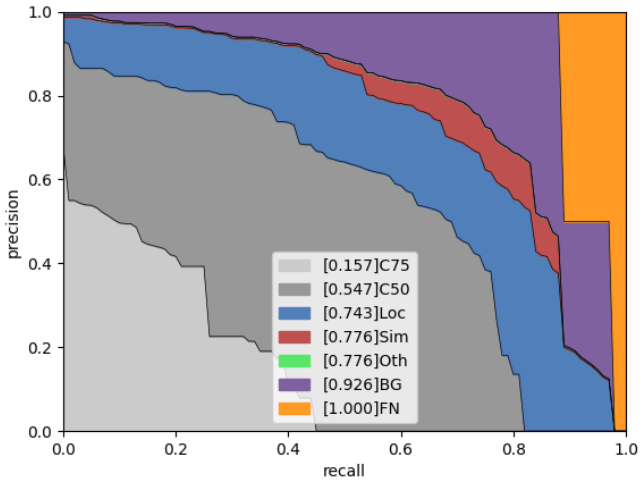

Figura 5.8: Curva precision-recall modelo Base (Ensayo 2).

El análisis por tamaño de anotación (figuras 5.10 y 5.11) confirma nuevamente que las mayores mejoras se dan en objetos pequeños, con incrementos de hasta 16 puntos en ambas métricas. Las anotaciones medianas también muestran una mejora de 5 puntos, mientras que, para objetos grandes, se observa una ligera disminución de 4 puntos en comparación con el modelo Base. La distribución de anotaciones (flores y capullos) por grupo de área en el conjunto de datos es la especificada en la tabla 5.2.

Para profundizar el análisis, en la tabla 5.8 se presentan los valores de verdaderos positivos (TP), falsos positivos (FP), precisión, recall y F1-score para los modelos Base y SF + SI (IoU + NMS), calculados a partir del umbral de confianza del modelo que maximiza la métrica F1-score, y evaluados bajo dos valores de referencia de IoU: 0.50 y 0.10. El conjunto de verificación cuenta con un total de 1101 anotaciones.

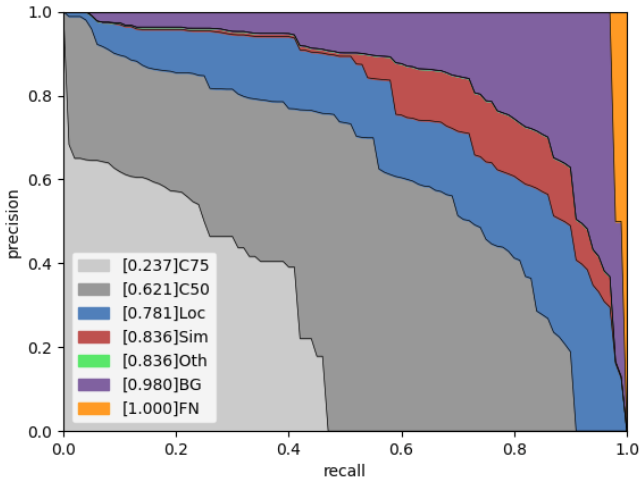

Figura 5.9: Curva precision-recall modelo SF + SI (IoU + NMS) (Ensayo 2).

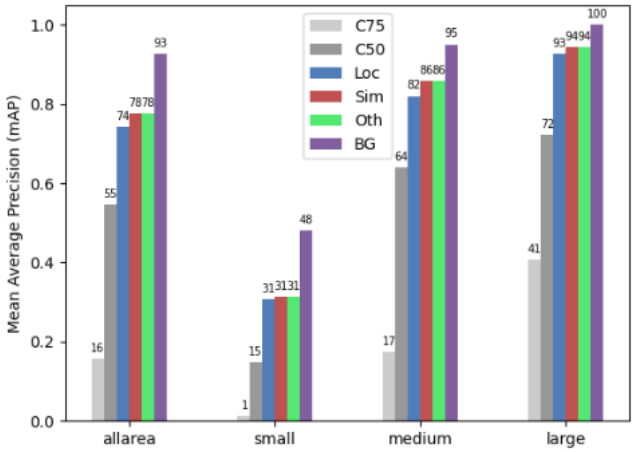

Figura 5.10: Rendimiento modelo Base discriminando por tamaño de anotación (Ensayo 2).

Los resultados evidencian una mejora general del modelo SF + SI (IoU + NMS) en comparación con el modelo Base, manifestada en incrementos en la cantidad de verdaderos positivos y, a diferencia de los resultados en el ensayo anterior, logrando reducir la cantidad de falsos positivos, resultando en mejoras en las métricas de precisión, recall y F1-score para ambos umbrales evaluados, reflejando un avance en la capacidad del modelo para equilibrar sensibilidad y precisión. Al igual que en el ensayo anterior, la proporción de falsos positivos en relación con el total de anotaciones continúa siendo un aspecto crítico a tener en cuenta al evaluar la aplicabilidad de los modelos en tareas de conteo.

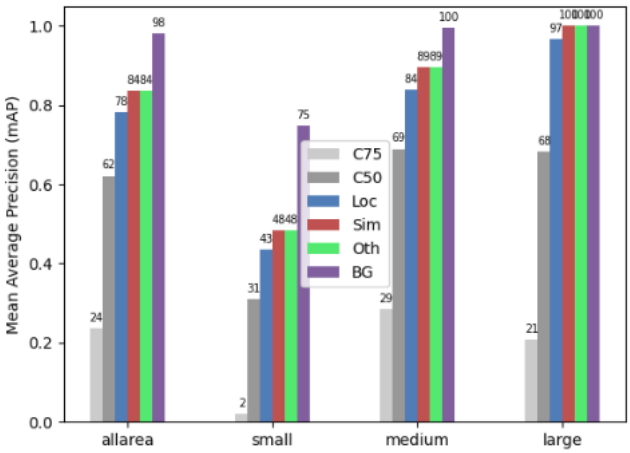

Figura 5.11: Rendimiento mejor modelo SF + SI discriminando por tamaño de anotación (Ensayo 2).

Modelo	**IoU**	TP	FP	Precisión	Recall	F1-score
Base	**0.50**	595	671	0.470	0.540	0.502
SF + SI (IoU + NMS)	**0.50**	622	453	0.578	0.564	0.571
Base	**0.10**	774	424	0.646	0.703	0.673
SF + SI (IoU + NMS)	**0.10**	857	402	0.680	0.778	0.726

Tabla 5.8: Análisis detallado de TP, FP, precisión, recall y F1-score para los modelos Base y SF + SI (IoU + NMS), bajo dos umbrales de IoU: 0.50 y 0.10. (Ensayo 2)

5.3.3. Ensayo 3

Con el objetivo de evaluar el impacto del aumento de la cantidad de datos de entrenamiento sobre la eficacia de los modelos para detectar flores, se realizó un tercer experimento en el que se unificaron los conjuntos *Dataset 1* (Pear Computer Vision Project) y *Dataset 2* (apple-flowers-flowering-rosebuds), compuestos por imágenes de flores de peral y manzano, respectivamente. Si bien las clases no son exactamente las mismas, ambas representan estructuras florales similares, por lo que se considera razonable explorar si esta combinación puede beneficiar la capacidad de generalización del modelo.

La metodología experimental adoptada replicó la estructura de los ensayos anteriores. El nuevo conjunto combinado consta de un total de 401 imágenes para entrenamiento, 115 para validación y 55 para verificación. La distribución de anotaciones por grupo de área del nuevo conjunto de datos es la especificada en la tabla 5.9.

Se entrenó un modelo YOLOv8 utilizando los mismos parámetros definidos previamente (tabla 5.3), obteniéndose el modelo Base de este ensayo.

Tamaño	Cantidad de anotaciones	Porcentaje del total
Pequeñas	56907	57.4 %
Medianas	41873	42.3 %
Grandes	421	0.4 %

Tabla 5.9: Cantidad de anotaciones por tamaño según el estándar COCO para el dataset resultante de la unificación del *Dataset 1* y *Dataset 2*.

Se reutilizaron los conjuntos de imágenes generados en los ensayos previos mediante la técnica *Slicing Aided Fine Tuning*, los cuales fueron unificados para conformar un nuevo conjunto de datos ampliado, compuesto por un total de 7951 imágenes. A partir de este conjunto, se efectuó una nueva etapa de ajuste fino sobre el modelo Base, dando lugar al modelo SF. Finalmente, se evaluó el rendimiento del modelo sobre el conjunto de verificación original, individualmente y utilizando *Slicing Aided Hyper Inference* (SI), empleando los mismos parámetros que en los ensayos anteriores. Los resultados obtenidos se presentan en la tabla 5.10.

Modelo	mAP@50	Loc
Base	0.501	0.675
SF	0.532	0.750
SF + SI (IoU + NMS)	**0.602**	**0.791**

Tabla 5.10: Evaluación y comparación de los resultados obtenidos por los modelos propuestos en el ensayo 3.

El mayor incremento en las métricas mAP@50 y Loc se obtuvo nuevamente con el modelo SF + SI con posprocesamiento basado en IoU + NMS, superando al modelo base en 10 y 12 puntos porcentuales, respectivamente. La comparación del rendimiento discriminado por tamaño de anotación puede observarse en las figuras 5.12 y 5.13. El incremento de precisión media más sustancial se mantiene para las anotaciones clasificadas como pequeñas y medianas, mientras que para la categoría grandes se registra una leve disminución.

Los resultados presentados en la tabla 5.11, que detalla métricas relevantes para los modelos Base y SF + SI (IoU + NMS) sobre el conjunto de verificación con un total de 19310 anotaciones, exhiben patrones similares a los observados en el ensayo 1, lo cual es razonable dada la prevalencia de imágenes y anotaciones del conjunto de flores de peral. El modelo SF + SI muestra para ambos umbrales de IoU un aumento tanto en la cantidad de verdaderos positivos como en la de falsos positivos, en proporciones que justifican la mejora observada en la métrica F1-score respecto al modelo Base.

Con el objetivo de analizar más en profundidad si la unificación de los conjuntos de datos mejoró la capacidad de generalización del modelo para detectar estructuras florales, se evaluó el rendimiento del mejor modelo de este ensayo diferenciando los resultados por clase. Para ello, se presenta la métrica mAP@50 de las siguientes clases: flores de peral (clase perteneciente al dataset utilizado

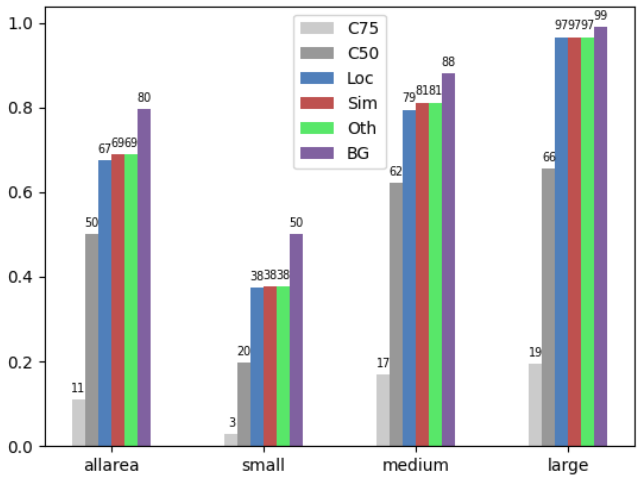

Figura 5.12: Rendimiento modelo Base discriminando por tamaño de anotación (Ensayo 3).

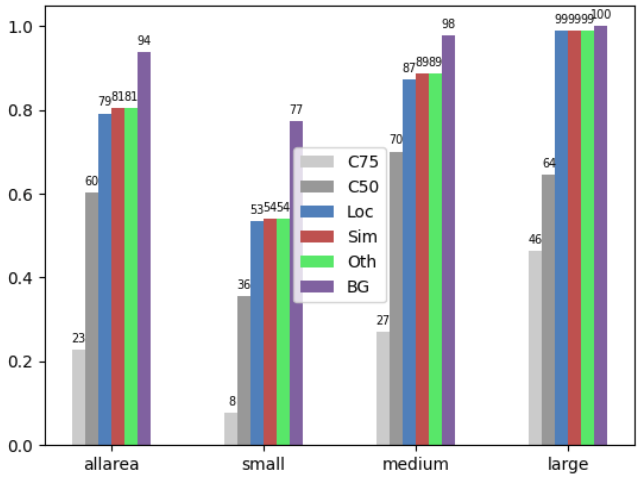

Figura 5.13: Rendimiento del modelo SF + SI (IoU + NMS) discriminando por tamaño de anotación (Ensayo 3).

en el ensayo 1), capullos de flores de manzano y flores de manzano en floración, que corresponden a las clases del dataset del ensayo 2. Los resultados se presentan en la tabla 5.12.

Al comparar estos valores de mAP@50 y Loc con los obtenidos en los ensayos 1 y 2, 0.553 mAP@50 y 0.753 Loc para flores de peral, y 0.621 mAP@50 y 0.781 Loc para flores de manzano (tablas 5.5 y 5.7), se observa que el modelo entrenado sobre el conjunto de datos unificado alcanza un rendimiento muy

Modelo	**IoU**	TP	FP	Precisión	Recall	F1-score
Base	**0.50**	8805	6018	0.594	0.456	0.515
SF + SI (IoU + NMS)	**0.50**	11497	7788	0.596	0.595	0.595
Base	**0.10**	11962	4796	0.717	0.619	0.665
SF + SI (IoU + NMS)	**0.10**	14131	5154	0.732	0.731	0.732

Tabla 5.11: Análisis detallado de TP, FP, precisión, recall y F1-score para los modelos Base y SF + SI (IoU + NMS), evaluados con el umbral de confianza del modelo que maximizan la métrica F1-score, bajo dos umbrales de IoU: 0.50 y 0.10. (Ensayo 3)

Clase	mAP@50	Loc
Flor de peral	0.559	0.753
Flor de manzano (floración y capullo)	0.624	0.810

Tabla 5.12: mAP@50 y Loc por clase del modelo SF + SI (IoU + NMS) obtenido en el ensayo 3

similar. Si bien no se evidencia una mejora significativa, tampoco se registra una degradación en el desempeño, lo cual sugiere que la combinación de datos no perjudica la capacidad del modelo para detectar estructuras florales distintas. Por el contrario, habilita la posibilidad de integrar datasets más extensos y heterogéneos, lo que podría favorecer una mejor generalización y una mejora en los valores de precisión media alcanzados.

5.3.4. Análisis comparativo de ensayos

En línea con los fundamentos que motivaron la experimentación con SAHI, las figuras 5.14 y 5.15 presentan los resultados obtenidos para los modelos entrenados en cada ensayo. La primera muestra los incrementos en la precisión media y en la métrica Loc correspondientes a las instancias clasificadas como pequeñas, grupo de anotaciones en el que se reporta el mayor incremento de rendimiento, mientras que la segunda reporta las mismas métricas considerando la totalidad de las anotaciones.

De la gráfica correspondiente a la comparación sobre el total de anotaciones (figura 5.15) se evidencia que la implementación de SAHI resultó favorable en todos los ensayos, evidenciando incrementos considerables en las métricas evaluadas.

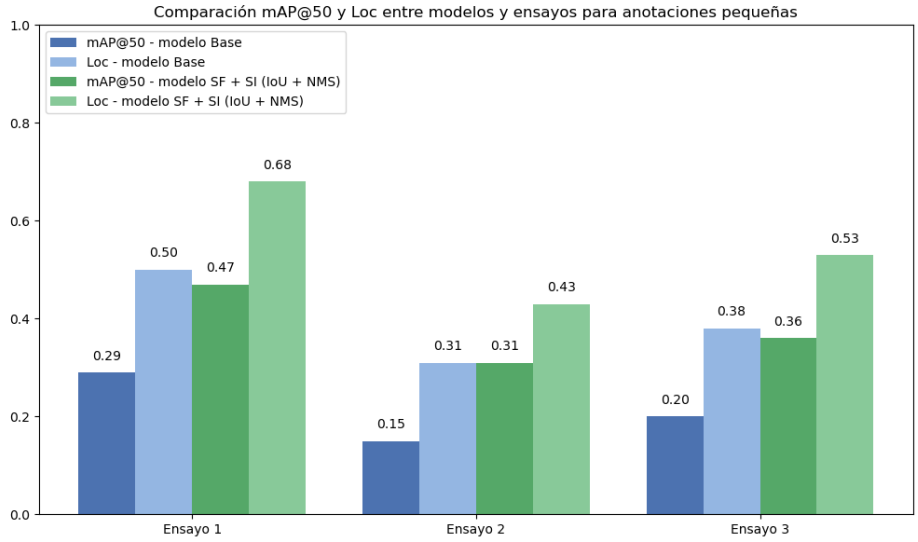

Figura 5.14: Precisión media y Loc para anotaciones clasificadas como pequeñas de los modelos entrenados en los ensayos

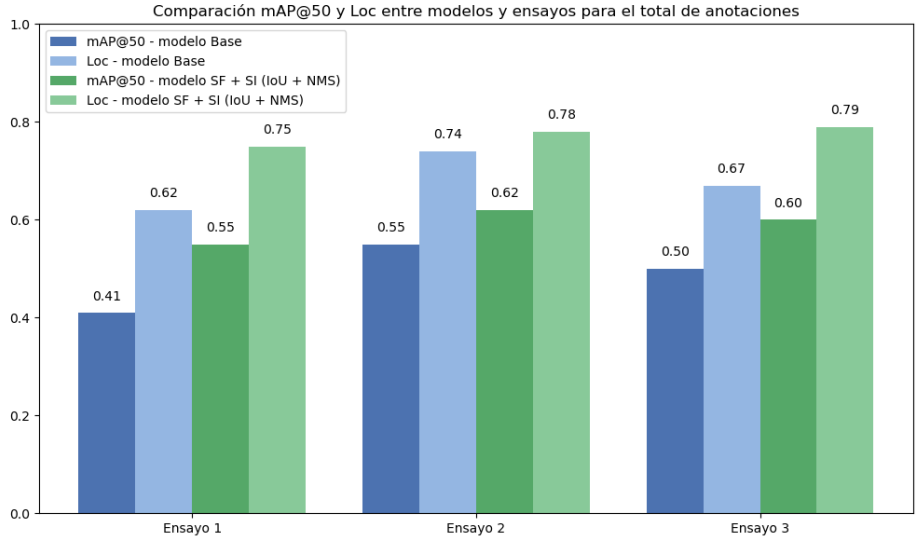

Figura 5.15: Precisión media para el total de anotaciones de los modelos entrenados en los ensayos

Capítulo 6

Conclusiones y Trabajo Futuro

Los resultados obtenidos en los ensayos ponen de manifiesto la eficacia del enfoque propuesto, basado en la combinación de *Slicing Aided Fine-Tuning* y *Slicing Aided Hyper Inference* mediante el marco de trabajo SAHI, para mejorar el desempeño de modelos de detección aplicados al conteo de flores en entornos agrícolas. En el caso del conjunto de datos *Pear Computer Vision Project* utilizado en el ensayo 1, el modelo SF + SI con posprocesamiento IoU + NMS alcanzó una precisión media de 0.55, lo que representa una mejora de 14 puntos porcentuales respecto a la del modelo base de 0.41. De manera análoga, en el ensayo 2, realizado con el conjunto *apple-flowers-flowering-rosebuds*, el modelo SF + SI (IoU + NMS) obtuvo un mAP@50 de 0.62, superando en 7 puntos porcentuales al modelo base, que alcanzó un valor de 0.55. Esto representa también un incremento de 5 puntos en mAP@50 respecto al modelo presentado en el estudio de la sección 3.1, que obtuvo un valor de 0.57 utilizando el conjunto de datos original del cual se derivó el dataset empleado en nuestro ensayo. En lo que respecta al uso de IoS como alternativa para la identificación de detecciones duplicadas durante el posprocesamiento de las inferencias con SAHI, los resultados de los ensayos indican que no representa una opción superior al uso de IoU, incluso en nuestro caso de uso, donde existe predominancia de objetos pequeños.

Los incrementos en la métrica de precisión media se encuentran estrechamente vinculados a la distribución de tamaños de las anotaciones en los conjuntos de datos, caracterizados por una alta proporción de objetos pequeños y medianos. Este sesgo en la distribución es un factor determinante en la efectividad de las estrategias presentadas, dado que las mejoras más pronunciadas se registran precisamente en estas categorías. Por el contrario, en el caso de objetos grandes, la estrategia de generación de parches tiende a fragmentar las instancias, lo que compromete la capacidad del modelo para reconocerlos de forma consistente, limitando así las ganancias en desempeño para esta clase.

El modelo entrenado con la unificación de los conjuntos de datos de flores de perales y manzano en el ensayo 3, desprende las mismas conclusiones en cuanto a la eficacia de las estrategias aplicadas para mejorar detecciones en flores de tamaño pequeño. Si bien el mejor modelo entrenado en este ensayo (SF + SI (IoU + NMS)) no presenta mejoras significativas en cuanto a precisión media por clase frente a los modelos entrenados con los conjuntos de datos separados en los ensayos previos, tampoco evidencia una degradación. Esto sugiere que la inclusión de datos heterogéneos, incluso pertenecientes a clases florales distintas, no afecta negativamente el desempeño del modelo, y podría, en cambio, favorecer su capacidad de generalización. Esto abre una vía para futuras investigaciones, donde la incorporación progresiva de nuevos conjuntos de datos, con mayor volumen y diversidad morfológica, podría contribuir al desarrollo de modelos más robustos.

Si bien las mejoras reportadas podrían parecer moderadas en comparación con los resultados obtenidos en el estudio relevado en la sección 3.3, donde se reporta un mAP@50 de 0.94 para la detección de flores de manzano, resulta relevante destacar las diferencias sustanciales entre los conjuntos de datos utilizados. En particular, el dataset propuesto en dicho estudio (ver figura 3.5) presenta condiciones notablemente más controladas, las flores se encuentran a corta distancia de la cámara, con fondos más uniformes y un nivel de oclusión prácticamente nulo. Por el contrario, las imágenes empleadas en nuestros ensayos (ver figura 3.1) exhiben escenas más complejas, con mayor variabilidad en iluminación, mayor distancia a los objetos de interés y un entorno natural más heterogéneo. Esta disparidad en las condiciones experimentales podría contribuir, al menos en parte, a las diferencias en precisión media observadas entre los modelos, aunque también deben considerarse otras variables, como la diferencia en la arquitectura de red utilizada en este estudio. Cabe destacar, además, que los conjuntos de datos seleccionados para los ensayos del presente estudio resultan más representativos del entorno real, al asemejarse a las imágenes que podría capturar un robot terrestre autónomo operando en un monte de cultivo de manzanas.

La estrategia SAHI propuesta en este trabajo se presenta como una herramienta prometedora para potenciar la etapa de detección de flores en el procesamiento de videos obtenidos durante recorridos en montes de frutales de manzanos. En particular, el procesamiento en diferido de video puede resultar más adecuado, dado que, si bien SAHI contribuye significativamente a mejorar la precisión media de los modelos de detección, su implementación incrementa los tiempos de inferencia, ya que requiere que la red realice múltiples inferencias por imagen o cuadro de video. Esto puede representar una limitación para aplicaciones en tiempo real, especialmente cuando se requiere procesar una gran cantidad de parches por fotograma. No obstante, la factibilidad de emplear SAHI en escenarios de procesamiento en vivo dependerá en gran medida del equipamiento utilizado, considerando que dispositivos con capacidades superiores, tales como unidades de procesamiento gráfico de alto rendimiento, podrían mitigar este incremento en los tiempos de inferencia y permitir su aplicación en tiempo real. Se considera pertinente en este sentido una futura exploración

para evaluar los tiempos de inferencia con SAHI en dispositivos con mayores capacidades computacionales.

En un contexto de aplicación real de conteo de flores mediante inferencias de un modelo de detección de objetos, maximizar el rendimiento del sistema implica contar con un modelo base robusto, entrenado sobre un conjunto de datos amplio y con anotaciones de alta calidad. Esto resulta crucial para mitigar uno de los principales problemas observados en los modelos entrenados en este trabajo, que es la dificultad para localizar con precisión las coordenadas de las flores. Esta limitación se ve reflejada en los resultados obtenidos para la métrica de precisión media con errores de localización ignorados (Loc). A ello se suma la alta proporción de falsos positivos reportados en las evaluaciones de los modelos (tablas 5.6, 5.8 y 5.11). En este sentido, además del robustecimiento del conjunto de datos, como trabajo futuro sería valioso explorar estrategias adicionales durante la etapa de entrenamiento que contribuyan a reducir la tasa de falsos positivos.

Se propone profundizar en diversas líneas de investigación orientadas a incrementar la efectividad en la aplicación de SAHI. Un primer abordaje consistiría en evaluar el impacto de modificar la arquitectura de la red neuronal subyacente. En este sentido, resulta pertinente analizar el desempeño de alternativas al modelo utilizado, tales como Faster R-CNN, arquitecturas con mecanismos de atención, como el descrito en el estudio relevado en la sección 3.3, u otras arquitecturas con buen rendimiento, con el objetivo de identificar configuraciones que optimicen las métricas de detección en el dominio estudiado. Además, de forma independiente a la aplicación de SAHI, queda como trabajo pendiente la exploración y evaluación de redes convolucionales orientadas a la predicción de mapas de densidad de flores para conteo.

Por otra parte, la variación de los parámetros de entrenamiento de la red, en particular la resolución de las imágenes, representa un aspecto de especial interés. Si bien un aumento en el tamaño de entrada conlleva mayores costes computacionales durante el entrenamiento, este ajuste podría traducirse en una mayor capacidad del modelo para identificar correctamente las flores, al permitir una representación más detallada de los objetos de estudio.

Asimismo, un punto relevante de la solución implementada que puede mejorarse es la optimización del tamaño del parche empleado al aplicar la técnica SF para la generación del conjunto de datos utilizado en la segunda etapa de ajuste fino. En el presente trabajo, dicho tamaño se mantuvo fijo en 640 px $\times$ 640 px; sin embargo, explorar el efecto de tamaños menores, por ejemplo, 256 px $\times$ 256 px o 128 px $\times$ 128 px, podría aportar mejoras significativas, al posibilitar un mayor nivel de detalle en la segmentación de regiones de interés y, potencialmente, un mejor desempeño en la detección de instancias pequeñas.

Finalmente se plantea explorar la integración de este enfoque en la etapa de detección de un flujo de procesamiento de video para conteo de flores de manzano, que incluya tanto detección como seguimiento de objetos. Esta idea toma como referencia el sistema presentado por Guchin y Sheppard (2024), donde se implementa una arquitectura similar para el seguimiento y geolocalización de manzanas en recorridos por campos de cultivo. La incorporación de SAHI en

la etapa de detección contribuiría a incrementar la calidad de las predicciones, lo que fortalecería la etapa de seguimiento y permitiría obtener estimaciones de conteo de flores más precisas y confiables.

Referencias

Akyon, F. C., Altinuc, S. O., y Temizel, A. (2022). Slicing aided hyper inference and fine-tuning for small object detection. En *2022 ieee international conference on image processing (icip)* (pp. 966–970). IEEE. doi: 10.1109/ICIP46576.2022.9897990

Bay, H., Tuytelaars, T., y Van Gool, L. (2006). Surf: Speeded up robust features. En *European conference on computer vision* (pp. 404–417).

Bochkovskiy, A., Wang, C.-Y., y Liao, H.-Y. M. (2020). Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*. doi: 10.48550/arXiv.2004.10934

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32. doi: 10.1023/A:1010933404324

Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *PAMI-8*(6), 679–698.

Cortes, C., y Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297. doi: 10.1007/BF00994018

Dalal, N., y Triggs, B. (2005). Histograms of oriented gradients for human detection. En *2005 ieee computer society conference on computer vision and pattern recognition (cvpr'05)* (Vol. 1, pp. 886–893).

Datascientest. (2021). *Redes neuronales convolucionales: definición y funcionamiento.* Descargado de https://datascientest.com/es/convolutional-neural-network-es (Consultado el 29 de abril de 2025)

Deng, W., Shi, J., y cols. (2018). *Visdrone dataset.* https://github.com/VisDrone/VisDrone-Dataset. (Accessed: 2025-05-01)

Duda, R. O., y Hart, P. E. (1972). Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, *15*(1), 11–15.

Elhanashi, A., Dini, P., Saponara, S., y Zheng, Q. (2024). Telestroke: Real-time stroke detection with federated learning and yolov8 on edge devices. *Journal of Real-Time Image Processing*, *21*(4), Article 121. Descargado de https://doi.org/10.1007/s11554-024-01500-1 doi: 10.1007/s11554-024-01500-1

Estrada, J. S., Vasconez, J. P., Fu, L., y Auat Cheein, F. (2024). Deep learning based flower detection and counting in highly populated images: A peach grove case study. *Journal of Agriculture and Food Research*. (En prensa)

European Committee for Electro-technical Standardization. (2012, agosto). *Alarm systems - cctv surveillance systems for use in security applications* (Inf. Téc.). CENELEC. (European Standard EN 62676)

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., y Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, *88*(2), 303–338. doi: 10.1007/s11263-009-0275-4

Girshick, R., Donahue, J., Darrell, T., y Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. En *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 580–587).

Goodfellow, I., Bengio, Y., y Courville, A. (2016). *Deep learning*. MIT Press. Descargado de https://www.deeplearningbook.org/

Guchin, A., y Sheppard, T. (2024, abril). *Pipeline de detección y seguimiento de manzanas para geolocalización de anomalías.* Informe de Proyecto de Grado, Facultad de Ingeniería, Universidad de la República. Montevideo. (Supervisores: Gonzalo Tejera, Mercedes Marzoa)

Haykin, S. (2008). *Neural networks and learning machines* (3.ª ed.). Upper Saddle River, NJ: Pearson Education.

Hoiem, D., Chodpathumwan, Y., y Dai, Q. (2012). Diagnosing error in object detectors. En *Computer vision–eccv 2012* (pp. 340–353). Springer. doi: 10.1007/978-3-642-33783-3_25

Ioffe, S., y Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. En *Proceedings of the 32nd international conference on machine learning (icml)* (pp. 448–456).

Jocher, G., y Qiu, J. (2024). *Ultralytics yolov11.* Descargado de https://docs.ultralytics.com/es/models/yolo11/ (Accedido: 2025-06-11)

Khaki, S., y Wang, L. (2019). Crop yield prediction using deep neural networks. *Frontiers in Plant Science*.

Kiktev, N., y Kutyrev, A. (2023a). *apple-flowers-flowering-rosebuds.* Descargado de https://universe.roboflow.com/alexey-kutyrev/flower-isqoj

Kiktev, N., y Kutyrev, A. (2023b). Flower detection and counting using cnn for thinning decisions in apple trees. *Agriculture*, *13*(2), 370. Descargado de https://doi.org/10.3390/agriculture13020370 doi: 10.3390/agriculture13020370

Kim, H., Park, S., y Lee, J. (2019). Greedy non-maximum merging for improved object detection performance. *International Journal of Computer Vision*, *124*(7), 1032–1045. doi: 10.1007/ijcv.2019.024

Kingma, D. P., y Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*. Descargado de https://arxiv.org/abs/1412.6980

Lam, D., Yarlagadda, S., y cols. (2018). *xview: Objects in context in overhead imagery.* https://challenge.xviewdataset.org/. (Accessed: 2025-05-01)

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., y Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, *1*(4), 541–551. Descargado de https://doi.org/10.1162/neco.1989.1.4.541 doi: 10.1162/neco.1989.1.4.541

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... Zitnick, C. L. (2014). Microsoft COCO: Common objects in context. En *Computer vision – eccv 2014* (pp. 740–755). Springer. doi: 10.1007/978-3-319-10602-1_48

Liu, S., Sun, M., Liu, X., y Mei, T. (2018). Path aggregation network for instance segmentation. En *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 8759–8768).

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., y Berg, A. C. (2016). Ssd: Single shot multibox detector. En *European conference on computer vision (eccv)* (pp. 21–37). Springer.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, *60*(2), 91–110.

Nair, V., y Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. En *Proceedings of the 27th international conference on machine learning (icml)* (pp. 807–814). Omnipress.

Nelson, J., Dwyer, B., Solawetz, J., y cols. (2020). *Roboflow: Organize, label, and export computer vision datasets.* https://roboflow.com. (Accessed: 2025-06-21)

Nesmachnow, S., y Iturriaga, S. (2019). Cluster-uy: Collaborative scientific high performance computing in uruguay. En M. Torres y J. Klapp (Eds.), *Supercomputing. isum 2019* (Vol. 1151, pp. 23–35). Springer, Cham. doi: 10.1007/978-3-030-34914-1_2

Neubeck, A., y Van Gool, L. (2006). Efficient non-maximum suppression. *Proceedings of the 18th International Conference on Pattern Recognition (ICPR)*, *3*, 850–855.

Rawat, W., y Wang, Z. (2017). Deep convolutional neural networks for image classification: A comprehensive review. *Neural computation*, *29*(9), 2352–2449.

Redmon, J., Divvala, S., Girshick, R., y Farhadi, A. (2016). You only look once: Unified, real-time object detection. En *Proceedings of the ieee conference on computer vision and pattern recognition (cvpr)* (pp. 779–788). Descargado de https://doi.org/10.1109/CVPR.2016.91 doi: 10.1109/CVPR.2016.91

Roboflow. (2023, oct). *Pear computer vision project.* Autor. Descargado de https://universe.roboflow.com/flower-testing/pear-tkh32/dataset/1

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, *65*(6), 386–408.

Rumelhart, D. E., Hinton, G. E., y Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., y Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*(1), 1929–1958.

Szeliski, R. (2022). *Computer vision: Algorithms and applications* (2nd ed.). Springer Nature.

Tessore, F. (2025a). *Documento de estado del arte: Detección y conteo de flores de manzano.* https://github.com/facundot/proy-grado/blob/master/estado%20del%20arte.pdf. (Último acceso: 23 de junio de 2025)

Tessore, F. (2025b). *proy-grado: Código fuente del trabajo final de grado sobre detección y conteo de flores de manzano.* https://github.com/facundot/proy-grado. (Accedido el 7 de junio de 2025)

Tureckova, A., Turecek, T., y Oplatkova, Z. K. (2022). ICIP 2022 Challenge: PEDCMI, TOOD Enhanced by Slicing-Aided Fine-Tuning and Inference. En *Proceedings of the 2022 ieee international conference on image processing (icip).* IEEE. doi: 10.1109/ICIP46576.2022.9897826

Ultralytics. (2024). *Ultralytics yolov8 documentation.* https://docs.ultralytics.com/models/yolov8/#overview. (Accessed: 2025-05-01)

Ultralytics. (2025). *Ultralytics yolo benchmarks.* https://github.com/ultralytics/ultralytics#benchmarks. (Sección "Benchmarks" del repositorio, con detalles de comparación de velocidad (FPS) y precisión para distintos formatos y modelos. Consultado en junio de 2025.)

Wang, Z., Li, J., Zhang, Z., Li, H., y Ma, Y. (2020). Cspnet: A new backbone that can enhance learning capability of cnn. En *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 3900–3909).

Xu, X., Lin, C., Zhang, Z., Huang, Z., Lin, H., y Lin, S. (2024). *YO-AFD: An improved YOLOv8-based deep learning approach for rapid and accurate apple flower detection.* Figshare. Descargado de https://figshare.com/articles/dataset/_b_YO-AFD_An_improved_YOLOv8-based_deep_learning_approach_for_rapid_and_accurate_apple_flower_detection_b_/28262693/2 doi: 10.6084/m9.figshare.28262693.v2

Yolcu, C., Guler, S., Aksoy, S., Erdem, E., y Erdem, A. (2021). *Sahi: Slicing aided hyper inference.* https://github.com/obss/sahi. (Último acceso: junio de 2025)

Zeiler, M. D., y Fergus, R. (2014). Visualizing and understanding convolutional networks. En *European conference on computer vision (eccv)* (pp. 818–833). Springer. Descargado de https://doi.org/10.1007/978-3-319-10590-1_53 doi: 10.1007/978-3-319-10590-1_53

Zhang, H., Hao, C., Song, W., Jiang, B., y Li, B. (2023). Adaptive slicing-aided hyper inference for small object detection in high-resolution remote sensing images. *Remote Sensing*, *15*(5), 1249. doi: 10.3390/rs15051249

Zhang, L., Lin, L., Liang, X., y He, K. (2016). Is faster r-cnn doing well for pedestrian detection? En *European conference on computer vision (eccv)* (pp. 443–457). Springer. doi: 10.1007/978-3-319-46478-7_18

Zhang, X., y Li, J. (2017). A novel approach to object detection using non-maximum merging. *Journal of Computer Vision*, *58*(2), 132–144. doi: 10.1007/jcv.2017.015

Zhao, Y., Zhang, Z., Yang, Z., y Yan, Z. (2023). YO-AFD: an improved YOLOv8-based deep learning approach for rapid and accurate apple flower detection. *Frontiers in Plant Science*, *14*, 1541266. Descargado de https://doi.org/10.3389/fpls.2023.1541266 doi: 10.3389/fpls.2023.1541266

Zhao, Z., Zheng, P., Xu, S.-t., y Wu, X. (2020). Object detection with deep learning: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. doi: 10.1109/TPAMI.2020.2981333

Anexo A

Anexo 1: Optimización de parámetros SAHI

En este anexo se presentan los resultados del proceso de optimización de los parámetros de segmentación por parches, específicamente la altura (*Hp*) y el ancho (*Wp*) de los parches utilizados durante la inferencia mediante el enfoque SI (Slicing Aided Hyper Inference).

Es importante destacar que, para la generación de parches mediante la técnica de *Slicing Aided Fine Tuning*, se fijó el tamaño de los parches en 640 píxeles. Esta decisión se tomó con el objetivo de mantener la experimentación dentro de límites computacionales razonables. A medida que se reduce el tamaño de los parches, se incrementa considerablemente la cantidad de imágenes generadas, así como el tiempo de entrenamiento. Si bien este parámetro también podría ser objeto de optimización, se decidió mantenerlo constante para los fines del presente estudio.

El objetivo de esta experimentación fue determinar las configuraciones de *Hp* y *Wp* que maximizan el desempeño del modelo, considerando métricas (*mAP@50*) y *Loc (Localization Error Ingored)* (sección 5.1).

Dado que el espacio de búsqueda para estos parámetros es extenso, se asumió que valores cercanos en píxeles tienden a producir resultados similares. Por lo tanto, se optó por una exploración con valores discretos y representativos, seleccionados estratégicamente para capturar los cambios significativos en el rendimiento del método. Los valores considerados para los experimentos fueron: *Hp* y *Wp* = {210, 420, 640} píxeles. Se detallan los resultados obtenidos para cada configuración evaluada, donde 'N° de parches' corresponde a la cantidad de parches sobre los que se realiza inferencia por imagen de entrada y 'Tiempo (ms)' corresponde al tiempo total de inferencia por imagen.

A.1. Ensayo 1

En la tabla A.1 se presentan los resultados de la optimización de parámetros *Hp*, *Wp* para la aplicación de la técnica *Slicing Aided Hyper Inference* en el ensayo 5.3.1 sobre el dataset *Pear Computer Vision Project* (sección 5.2.1).

Modelo	Tamaño de parche	AP@50	Loc	N° de parches	Tiempo (ms)
SF + SI (IoU + NMS)	210px	0.530	0.730	104	8060
SF + SI (IoU + NMS)	420px	0.532	0.730	28	2535
SF + SI (IoU + NMS)	640px	**0.553**	**0.753**	15	1535

Tabla A.1: Resultados obtenidos para variando tamaño de parche

Encontramos que el mejor rendimiento del modelo se da para el parámetro de tamaño de parche 640px (mismo valor que el utilizado para la técnica *SF*).

A.2. Ensayo 2

En la tabla A.2 se presentan los resultados de la optimización de parámetros *Hp* y *Wp* para la aplicación de técnica *Slicing Aided Hyper Inference* en el ensayo 5.3.2 sobre el dataset *apple-flowers-flowering-rosebuds* 5.2.2

Modelo	Tamaño de parche	AP@50	Loc	N° de parches	Tiempo (ms)
SF + SI (IoU + NMS)	210px	0.557	0.730	84	6230
SF + SI (IoU + NMS)	420px	0.559	0.734	18	1769
SF + SI (IoU + NMS)	640px	**0.621**	**0.781**	8	1031

Tabla A.2: Resultados obtenidos para variando tamaño de parche

Nuevamente encontramos que el mejor rendimiento del modelo se da para el parámetro de tamaño de parche 640px.

Anexo B

Anexo 2: Instructivo de uso del repositorio

El código desarrollado para este trabajo se encuentra disponible en el repositorio indicado en Tessore (2025b). Este instructivo describe los pasos necesarios para utilizar el *pipeline* de experimentación.

Inicialmente, es necesario disponer de un dataset de detección de objetos con el que se desee realizar los experimentos. Dicho dataset puede ser preprocesado o particionado según los requerimientos específicos del caso de uso. Para el entrenamiento del modelo base, se debe ejecutar el `script` de entrenamiento ubicado en la carpeta `train/` del repositorio.

Instalación de SAHI

Desde la carpeta `sahi/` del repositorio, el paquete puede instalarse localmente mediante el siguiente comando:

```
cd sahi
pip install .
```

Generación de dataset con técnica SF

Para aplicar la técnica SF es necesario que el dataset de partida esté en formato COCO. Existen diversas bibliotecas que permiten convertir anotaciones entre formatos (por ejemplo, `roboflow`, `fiftyone`, o `labelme2coco`). Una alternativa práctica es utilizar la plataforma Roboflow, que facilita estas conversiones de forma automática.

Una vez preparado el dataset, la generación de los parches puede realizarse ejecutando:

```
sahi coco slice --image_dir $IMAGES_PATH \
```

```
--dataset_json_path $ANNOTATIONS_COCO_JSON_PATH \
--slice_size $SLICE_SIZE --overlap_ratio $OVERLAP_RATIO
```

Donde:

- `IMAGES_PATH`: ruta a la carpeta que contiene las imágenes originales.
- `ANNOTATIONS_COCO_JSON_PATH`: ruta al archivo de anotaciones en formato COCO.
- `SLICE_SIZE`: tamaño (en píxeles) del lado del parche cuadrado (por ejemplo, 640).
- `OVERLAP_RATIO`: fracción de solapamiento entre parches adyacentes (por ejemplo, 0.2 para un 20 %).

El nuevo dataset generado podrá emplearse en una segunda etapa de entrenamiento o ajuste fino utilizando el mismo script de la carpeta `train/`.

Inferencia y evaluación

Inferencia sin SI

```
sahi predict --source $DATASET_VALID \
--dataset_json_path $DATASET_VALID_COCO_ANNOTATIONS \
--model_path $MODEL_PATH --model_type ultralytics \
--no_sliced_prediction \
--model_confidence_threshold $MODEL_CONFIDENCE_THRESHOLD
```

Donde:

- `DATASET_VALID`: ruta a la carpeta con imágenes de validación.
- `DATASET_VALID_COCO_ANNOTATIONS`: ruta al archivo de anotaciones COCO del dataset de validación.
- `MODEL_PATH`: ruta al archivo del modelo entrenado (por ejemplo, `best.pt`).
- `MODEL_CONFIDENCE_THRESHOLD`: umbral mínimo de confianza para conservar una predicción (se recomienda utilizar 0.01 si se pretende evaluar el modelo utilizando mAP@50, dado que necesitamos la curva precision-recall de [0.01.0.99]).

Inferencia con SI

```
sahi predict --source $DATASET_VALID \
--dataset_json_path $DATASET_VALID_COCO_ANNOTATIONS \
--model_path $MODEL_PATH --model_type ultralytics \
--slice_height $SLICE_HEIGHT --slice_width $SLICE_WIDTH \
```

```
--overlap_height_ratio $OVERLAP_HEIGHT_RATIO \
--overlap_width_ratio $OVERLAP_WIDTH_RATIO \
--model_confidence_threshold $MODEL_CONFIDENCE_THRESHOLD
```

Donde los parámetros son equivalentes a los previamente descritos, y además:

- `SLICE_HEIGHT`, `SLICE_WIDTH`: dimensiones (en píxeles) de cada parche para inferencia (por ejemplo, 640).
- `OVERLAP_HEIGHT_RATIO`, `OVERLAP_WIDTH_RATIO`: fracción de solapamiento entre parches en las dimensiones correspondientes (por ejemplo, 0.2 para un 20 %).

Evaluación de resultados

```
sahi coco analyse --dataset_json_path $DATASET_VALID_COCO_ANNOTATIONS \
--result_json_path $PREDICT_RESULT_PATH
```

Donde:

- `PREDICT_RESULT_PATH`: ruta al archivo JSON con los resultados de la predicción generados por SAHI.

Para mayor información sobre SAHI, puede consultarse el repositorio oficial referenciado en Yolcu y cols. (2021).