CARRERA DE ESPECIALIZACIÓN EN INTELIGENCIA ARTIFICIAL

MEMORIA DEL TRABAJO FINAL

Identificación de estados fenológicos de la flor de durazneros mediante visión por computadora

Autor:
Ing. Héctor Luis Sánchez Márquez

Director:
Ing. Juan Ignacio Cavalieri

Codirector:
Esp. Lic. Nicolás Eduardo Horro

Jurados:
Nombre del jurado 1 (pertenencia)
Nombre del jurado 2 (pertenencia)
Nombre del jurado 3 (pertenencia)

Este trabajo fue realizado en la Ciudad Autónoma de Buenos Aires, entre agosto de 2023 y junio de 2024.

Resumen

Esta memoria presenta un algoritmo desarrollado para el Instituto Nacional de Tecnología Agropecuaria que es capaz de identificar los estados fenológicos de la flor de duraznero y extraer información de su vareta a partir de imágenes. El objetivo de este desarrollo es agilizar y automatizar la toma de datos de las varetas de duraznero a través de fotos. Con ello, se busca aumentar el caudal de datos existente y conocer el estado fenológico a campo.

Para su desarrollo e implementación fueron aplicados los conocimientos de visión por computadora, análisis de datos, aprendizaje profundo y buenas prácticas de despliegue adquiridas en la carrera.

Agradecimientos

Esta sección es para agradecimientos personales y es totalmente **OPCIONAL**.

Índice general

Índice de figuras

Índice de tablas

Dedicado a... [OPCIONAL]

Capítulo 1

Introducción general

En este capítulo se presenta la problemática y la motivación que llevaron a la realización del presente trabajo.

1.1. Descripción de la problemática

La fenómica hace referencia a la obtención de un gran caudal de datos de las características de las plantas, lo que se denomina el fenotipo de la planta. Esta disciplina está en auge en la actualidad debido a sus aplicaciones potenciales. Por un lado, habilita el mejoramiento a gran escala debido a que es necesario vincular una gran cantidad de datos genéticos con datos fenotípicos para identificar la función de los genes. Por otro lado, si se incluyen otros conjuntos de datos como son los climáticos, permite realizar predicciones precisas sobre el comportamiento de las variedades, necesario para implementar lo que se conoce como agricultura de precisión. Sin embargo, la fruticultura no ha dado el salto hacia la fenómica.

En la Estación Experimental Agropecuaria (EEA) de San Pedro se ha logrado secuenciar el ADN de más de 250 variedades de duraznero [1] y se dispone de una base de datos genómica de 75 gigabases (Gb) de ADN. Esta base permite identificar genes que controlan características del duraznero mediante algoritmos de inteligencia artificial (IA). Además, se dispone de datos climáticos diarios que se toman de forma automática que incluyen: las temperaturas medias, precipitaciones, horas de frío, radiación, etc. Esta información se combina con los datos genómicos y posteriormente, con modelos de IA, se predice el comportamiento de las variedades en escenarios climáticos futuros. En la figura 1.1, se observa el proceso para obtener los datos genómicos.

En la actualidad, las heladas primaverales son el mayor problema de los frutales a nivel mundial. Este fenómeno ocurre cuando las flores abiertas se someten a temperaturas cercanas a los -2.5 °C. Las heladas primaverales tienen una temperatura parecida a cualquier otra helada que se puede presentar en la temporada de invierno. Sin embargo, estas heladas suelen presentarse después del invierno, creando un gran impacto contra las flores y los frutos. Los productores de frutas, en general, se ven altamente afectados pagando un alto precio por estas inesperadas heladas tardías. En la figura 1.2, se puede observar cómo este fenómeno meteorológico afecta a los frutales.

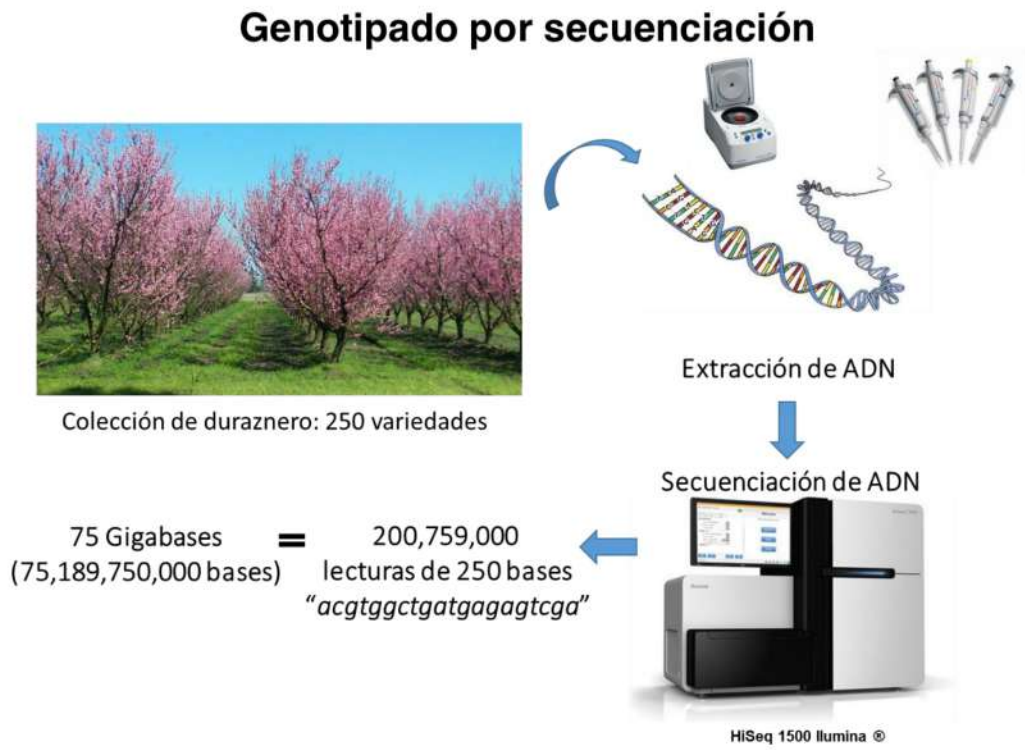

FIGURA 1.1. Proceso para obtener los datos genómicos [1].

FIGURA 1.2. Frutales afectados por las heladas primaverales [2].

1.2. Motivación

Por lo explicado anteriormente, es del interés del Instituto Nacional de Tecnología Agropecuaria (INTA) determinar el estado fenológico a campo y mejorar la tolerancia a heladas primaverales.

Para determinar el estado fenológico a campo, es necesario conocer el número de flores que se encuentran en estado vulnerable ante un pronóstico de heladas primaverales, así como también la densidad de flores.

En cuanto al mejoramiento, se ha realizado una caracterización a gran escala de la tolerancia a heladas de la colección de duraznero con el objetivo de identificar los genes responsables. Parte de ese experimento consistió en registrar el estado fenológico mediante fotos.

El presente trabajo permitirá automatizar la toma de datos de varetas de duraznero a partir de fotos para aumentar el caudal de datos y mejorar los modelos de IA.

1.3. Requerimientos

1. Requerimientos funcionales
   a) El sistema tomará como entrada imágenes de varetas de durazneros en formato JPG.
   b) El algoritmo debe detectar la presencia de las varetas de los durazneros e identificar el tipo de flor que posee.
   c) El algoritmo debe identificar el estado fenológico de cada flor de duraznero en la vareta. Este estado se clasificará como *flor abierta*, *flor cerrada*, *flor sinpetalos*, *incierto*.
   d) El algoritmo debe determinar la cantidad de flores por centímetro de vareta.
   e) El sistema debe entregar como resultado un archivo en formato CSV con los datos detectados por el algoritmo y una imagen donde se puedan visualizar las detecciones.
   f) El sistema debe funcionar en una computadora local.
2. Requerimientos de diseño e implementación
   a) El diseño debe ser modular.
   b) El algoritmo se elaborará en una notebook de Google Colab, utilizando el lenguaje de programación Python y bibliotecas de IA correspondientes.
3. Requerimiento de evaluación y prueba
   a) El modelo se evaluará con imágenes provenientes del mismo dataset de imágenes entregado por el cliente.
   b) La métrica que se utilizará para la evaluación del modelo de detección será *mean average precision* (mAP) y para el clasificador se tomarán en cuenta las métricas *accuracy*, *precision* y *recall*.
4. Requerimientos de documentación
   a) El funcionamiento del sistema debe estar correctamente explicado y documentado.
   b) El código estará correctamente comentado como parte de buenas prácticas del desarrollo de software.
   c) Inclusión de documentación en un repositorio, mediante un archivo README.md (opcional).

1.4. Objetivo y alcances

El objetivo de este trabajo es desarrollar un algoritmo que permita automatizar la toma de datos de las flores de duraznero a través de fotos de varetas.

El presente trabajo incluye:

- El preprocesamiento de las fotos para entrenar el modelo.
- La selección del modelo a entrenar.
- La elaboración del notebook de pruebas en Python.
- La implementación local del modelo.

El trabajo no incluye:

- La recolección de datos/fotos.
- La integración con otros modelos que utilice el cliente.

1.5. Estado del arte

El presente trabajo contiene distintos algoritmos que, integrados, logran tomar los datos deseados. Es por este motivo que para determinar el estado del arte es necesario desglosar cada algoritmo y evaluarlo individualmente como se hace a continuación.

1.5.1. Medición de la vareta

En la actualidad, se han desarrollado algoritmos que pueden determinar el tamaño de distintos objetos a través de imágenes usando visión por computadora. Muchos parten de encontrar un objeto de referencia al que se le conocen sus dimensiones (alto y ancho). Este objeto de referencia, normalmente se selecciona por ser fácil de detectar, por conocer sus dimensiones y por ser un objeto único. El procedimiento habitual para su detección, es pasar la imagen a escala de grises, aplicar filtros gaussianos para eliminar el ruido, utilizar detección de bordes y por último utilizar detección de contornos, tal y como se realiza en el trabajo [3]. Cabe destacar que usualmente el fondo es de un color blanco, lo que facilita la detección.

En el presente trabajo se tienen imágenes con fondos de color naranja en su mayoría, el objeto de referencia a veces se encuentra ocluido, las imágenes se encuentran en horizontal o vertical, el objeto de referencia no siempre tiene la misma posición, etc. Por estos motivos, se utilizó un método de detección más complejo con un modelo de detección de objetos que se conoce como *YOLOv8* y se considera el estado del arte a la fecha.

1.5.2. Detección de los estados fenológicos de la flor de duraznero

La detección de flores ha sido estudiada con diferentes enfoques y arquitecturas de *deep learning*, como por ejemplo el estudio [4] que exploró la viabilidad de detección de estados fenológicos de las rosas con técnicas del contraste del color y comparando con el modelo de detección *Faster R-CNN*. Sin embargo, no utilizó

ninguna arquitectura de una etapa para la detección, lo que podría ser más eficiente. Por otro lado, se tienen trabajos que sí utilizaron la arquitectura de una etapa, en específico de *YOLO* en sus versiones 4 y 5 como se presenta en [5] [6], pero su enfoque fue basado para las flores de kiwi.

La presente memoria busca en particular los estados fenológicos de la flor de duraznero utilizando y comparando dos modelos de detección, donde el primero tiene una arquitectura de dos etapas y el segundo tiene una arquitectura de una etapa. El detector de una etapa sería la arquitectura de *YOLO* en su version 8, la cual se propuso para la realización efectiva y eficiente de esta tarea, además de representar el estado del arte en la actualidad.

1.5.3. Conteo de flores

El conteo de objetos en imágenes a través de visión por computadora es otro campo que ha sido altamente estudiado y tiene muchos enfoques tanto simples como complejos. De esta forma, se expone el caso [7] donde se hace uso de redes convolucionales para predecir mapas de densidad que permiten hacer un conteo preciso de objetos en imágenes de alta densidad. Así como también se tiene el trabajo [8] que utiliza *transformers* con el mismo propósito.

Por otro lado, este trabajo propone una combinación entre el uso del modelo de detección de objetos y métodos de visión por computadora tradicionales para el conteo de las flores de duraznero.

Capítulo 2

Introducción específica

En este capítulo se presenta una introducción teórica detallada de los algoritmos utilizados para la elaboración del presente trabajo.

2.1. Red neuronal convolucional

Una red neuronal convolucional es un tipo red neuronal diseñada para identificar patrones en imágenes. Particularmente esta arquitectura de *deep learning*, suele ser utilizada en problemas de visión por computadora relacionados a la clasificación o detección de objetos en una imagen. Estas redes, pueden llegar a tener cientos de capas y llegan a estar compuestas por tres tipos principales como son las capas convolucionales, las capas de agrupación (*pooling*) y una o varias capas totalmente conectadas (*fully connected*) [9][10][11].

La capa convolucional consiste en un conjunto de filtros o *kernels* entrenables que se mueven por el ancho y alto de la imagen de entrada, donde, se calcula el producto escalar entre los píxeles de entrada y el filtro en cualquier posición. El resultado de este calculo se incorpora en una matriz de salida. De esta forma, se desplaza el filtro repitiendo la operación anterior hasta que el *kernel* recorre toda la imagen. La serie de productos escalares de la imagen de entrada con los filtros se conoce como mapa de activación. Finalmente, la red aprenderá filtros que se activan cuando detectan algún tipo de característica, borde o patrón visual [10][11].

La capa de agrupación, se encuentra normalmente entre capas sucesivas convolucionales. Su función es simplificar la salida mediante la reducción no lineal de la tasa de muestreo, lo que termina resultando en una disminución de la cantidad de parámetros que la red debe aprender. Aunque se pierde información en esta capa, tiene beneficios para la red convolucional, como por ejemplo se reduce la complejidad, mejora la eficiencia y evita el sobreajuste [10][11].

La capa totalmente conectada a diferencia de las otras capas mencionadas, es que tiene todos los nodos de la salida conectados directamente a los nodos anteriores. El objetivo de esta última capa, es realizar una clasificación, basándose en las características extraídas anteriormente. La función de activación generalmente utilizada en esta capa es la función *softmax* para obtener la probabilidad de que un objeto pertenezca a una clase u otra entre un intervalo de 0 a 1.

En la figura 2.1, se puede observar un ejemplo de como se estructuran dichas capas en la red neuronal convolucional.

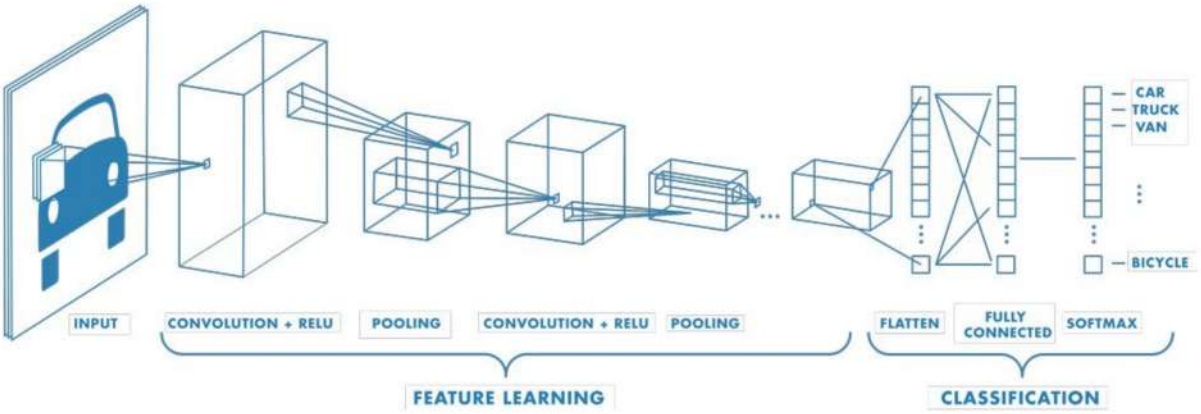

FIGURA 2.1. Ejemplo de arquitectura de red neuronal convolucional [9].

2.2. Detección de objetos

La detección de objetos es una técnica utilizada en visión por computadora para localizar e identificar uno o varios objetos en una imagen o vídeo. A diferencia de otras técnicas de *machine learning*, como es el caso de la clasificación o reconocimiento de imágenes, la detección busca localizar el lugar exacto donde se encuentra el objeto de interés y lo delimita con un rectángulo también llamado caja delimitadora o *bounding box* por su término en inglés. Por otro lado, una vez delimitado el objeto se clasifica entre las categorías disponibles [12].

La detección de objetos se puede implementar utilizando métodos de *machine learning* clásicos o de *deep learning*, dependiendo del problema a resolver[13]. El uso de *deep learning* es más eficaz cuando se requiere tratar imágenes con muchas etiquetas, es decir, muchos objetos a detectar y las imágenes tienen otras variaciones como cambio de brillo, rotaciones, cambio de escala, etc. Por otro lado, el uso de *machine learning* clásico es favorable cuando no se tienen altas capacidades de procesamiento y el número de etiquetas distintas a identificar es menor.

Algunos ejemplos de detección de objetos con *machine learning* clásico son *SIFT*, *Temple Matching*, etc. En la figura 2.2, se puede observar el uso de *Temple Matching* para detectar el logo de Coca-Cola en una imagen.

FIGURA 2.2. Ejemplo de *template matching* para identificar el logo de Coca-Cola.

Por otro lado, se tienen métodos de *deep learning* para detección de objetos más avanzados que involucran detectores de dos etapas y de una etapa como son *R-CNN*, *Faster R-CNN*, *SSD*, *YOLO*, entre otros.

2.2.1. Detectores de dos etapas

Los detectores de dos etapas, como *R-CNN* y sus variantes, primero extraen las características de la imagen y se propone la región de interés (ROI). El ROI, consiste en un *bounding box* donde se presume que se encuentra el objeto a buscar. Luego, en la segunda etapa, se analizan las características encontradas en conjunto con el ROI, para seleccionar los *bounding boxes* finales y calcular las probabilidades de que el objeto en las regiones pertenezca a una clase específica [14].

Las redes neuronales de dos etapas son muy precisas al momento de detectar un objeto, sin embargo, son redes que son consideradas lentas durante la inferencia. Esta desventaja llevó a mejorar los modelos iniciales como *R-CNN* a sus variantes *Fast R-CNN* y *Faster R-CNN*.

En la figura 2.3, se puede observar la arquitectura de una red neuronal de dos etapas *R-CNN*.

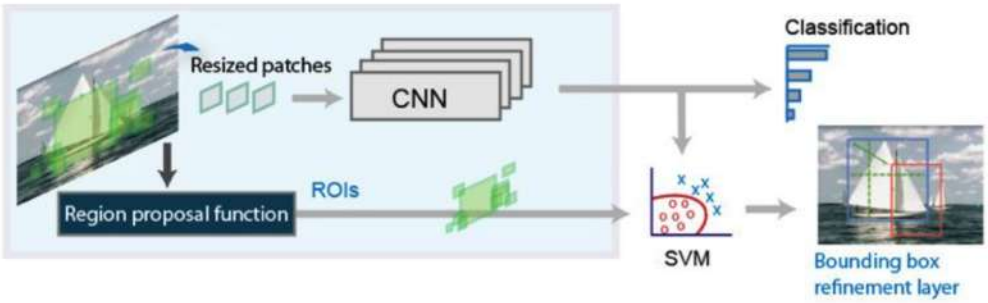

FIGURA 2.3. Ejemplo de arquitectura de red neuronal de dos etapas *R-CNN* [13].

Detección de objetos con *Fast R-CNN*

Las mejoras adicionadas en *Fast R-CNN* en comparación con *R-CNN* incluyen una nueva capa llamada *ROI Pooling*, que se encarga de extraer vectores de características de igual longitud de todas las regiones de interés propuestas. Por otro lado, *R-CNN* contiene tres etapas como son la generación de región propuesta, la extracción de características y la clasificación usando *SVM*, sin embargo, *Fast R-CNN* crea una red neuronal que tiene una única etapa, reduciendo así el número de etapas de su predecesor. Además, este modelo comparte cálculos computacionales a través de todas las ROIs propuestas en vez de hacerlo una a una de manera independiente. Por último, *Fast R-CNN* no guarda en cache las características, lo que reduce el uso de disco de memoria [15].

Detección de objetos con *Faster R-CNN*

El modelo *Faster R-CNN* fue diseñado para superar muchos de los errores encontrados en sus predecesores como *Fast R-CNN* y *R-CNN*, en general como su nombre en inglés sugiere, sus mejoras van relacionadas a la rapidez en comparación a las otras variantes.

Faster R-CNN mejora con respecto a *Fast R-CNN* incorporando la red de región propuesta o por sus siglas en ingles RPN, que es una red completamente convolucional que produce propuestas con diferentes escalas y relaciones de aspecto. La RPN aplica la terminología de redes neuronales con atención para indicar al modelo a donde mirar.

En *Faster R-CNN* se introduce el concepto de *anchor boxes*, lo que permite detectar objetos en distintas escalas y relaciones de aspecto. Por último, se comparten cálculos computacionales a través de la RPN y *Fast R-CNN*, lo que reduce el tiempo computacional [16]. En la figura 2.4, se muestra un ejemplo de la arquitectura de *Faster R-CNN*.

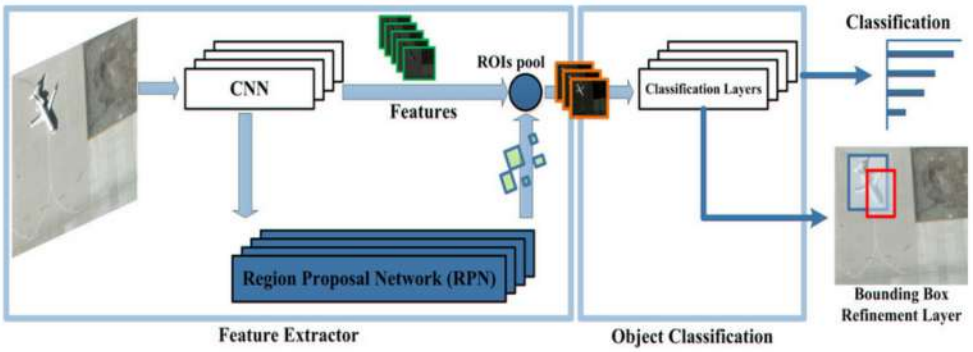

FIGURA 2.4. Ejemplo de arquitectura de red neuronal de dos etapas *Faster R-CNN* [17].

2.2.2. Detectores de una etapa

Los detectores de una etapa son modelos que omiten la etapa de RPN y ejecutan la detección directamente en un muestreo denso de ubicaciones. Estos modelos hacen uso de *grid-box* y *anchor box* para localizar la región de detección y delimitar al objeto de interés. En general, se consideran más rápidos que los detectores de dos etapas, sin embargo, son menos precisos [18].

Algunos ejemplos de detectores de una etapa son las arquitecturas *YOLO*, *SSD*, *RetinaNet*, *SqueezeDet* y *DetectNet*.

Detección de objetos con *YOLO*

You Only Look Once (*YOLO*) es considerado el estado del arte para la detección en tiempo real y fue introducido en el año 2015. En el paper [19], los autores trataron la detección de objetos como un problema de regresión simple en vez de hacerlo como un problema de clasificación, donde, se separa espacialmente los *bounding boxes* y se les asocia una probabilidad a cada detección usando una red neuronal convolucional.

El algoritmo se basa en el uso de bloques residuales, *bounding boxes* por regresión, *Intersection Over Unions* (*IoU*) y *Non-Maximum Suppression*.

El primer paso es el uso de bloques residuales que consiste en dividir la imagen original en celdas cuadriculadas de NxN de igual dimensión. Cada celda en la cuadricula es responsable de localizar y predecir la clase de objeto que cubre con su respectivo valor de confianza. Luego, se determinan los *bounding boxes* con los atributos que *YOLO* obtiene usando un modulo de regresión lineal. La respuesta

que se recibe de este modulo es una representación vectorial de las coordenadas para cada *bounding box*.

Por otro lado, un mismo objeto puede tener múltiples detecciones durante la predicción y no todas serán relevantes. Por este motivo, se usa *Intersection Over Unions* que ayuda a descartar aquellas predicciones no relevantes, dándole un valor entre 0 y 1. En este proceso, se establece un umbral para el *Intersection Over Unions*, que posteriormente, se compara con el calculo generado por *YOLO* para cada cuadrícula y si el resultado es menor al umbral establecido, se descarta dicha predicción. Por ultimo, se aplica *Non-Maximum Suppression* para preservar solo aquellas predicciones que tengan un valor de confianza alto [20].

YOLO cuenta con varias versiones desde su lanzamiento donde se ha mejorado su velocidad, precisión y el uso de recursos. Estas versiones, van desde la uno a la nueve, siendo la ocho la usada en la presente memoria.

2.3. Clasificación de imágenes

La clasificación de imágenes es una tarea fundamental en el ámbito de la visión por computadora, que consiste en asignar etiquetas o clases a las imágenes. Este proceso implica trabajar a nivel de píxeles, donde se extraen características de la imagen en formato vectorial, que luego son utilizadas por modelos de *machine learning* para realizar predicciones.

La clasificación puede llevarse a cabo mediante métodos de *machine learning* clásicos o utilizando *deep learning*. El uso de *deep learning* para esta tarea ha demostrado un rendimiento sobresaliente, incluso ante imágenes con cambios de iluminación, rotación, deformación, oclusión, entre otros desafíos.

La clasificación de imágenes mediante *deep learning* emplea redes neuronales convolucionales, un proceso que inicia con una imagen que pasa por una capa de entrada, donde se realiza un preprocesamiento antes de ser enviada a una capa convolucional. En la capa convolucional, se aplican filtros a la imagen para extraer características y generar un mapa de características que indica la presencia o ausencia de ciertos atributos. Posteriormente, la imagen se somete a una reducción de dimensionalidad en una capa de agrupación, de esta forma se preserva la información importante extraída, antes de pasar por una capa de activación no lineal, generalmente una capa ReLU. Finalmente, el resultado se alimenta a una capa totalmente conectada para obtener las probabilidades de que la imagen pertenezca a una de las clases predefinidas.

```
    updateSensors();
    updateAlarms();
    controlActuators();
    vTaskDelayUntil(&ticks, period);
  }
}
```

CÓDIGO 3.1. Pseudocódigo del lazo principal de control.

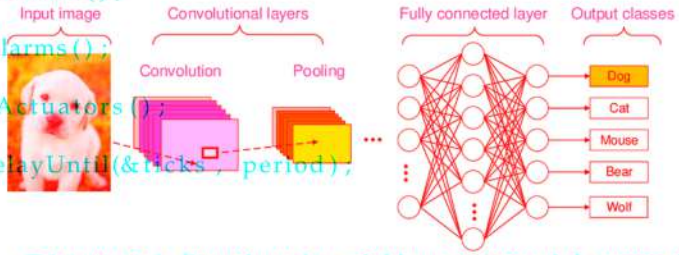

FIGURA 2.5. Clasificación de imagenes con *deep learning* [21].

Capítulo 3

Diseño e implementación

3.1. Análisis del software

La idea de esta sección es resaltar los problemas encontrados, los criterios utilizados y la justificación de las decisiones que se hayan tomado.

Se puede agregar código o pseudocódigo dentro de un entorno lstlisting con el siguiente código:

```
\begin{lstlisting}[caption= "un epígrafe descriptivo"]
las líneas de código irían aquí...
\end{lstlisting}
```

A modo de ejemplo:

```
#define MAX_SENSOR_NUMBER 3
#define MAX_ALARM_NUMBER  6
#define MAX_ACTUATOR_NUMBER 6

uint32_t sensorValue[MAX_SENSOR_NUMBER];
FunctionalState alarmControl[MAX_ALARM_NUMBER]; //ENABLE or DISABLE
state_t alarmState[MAX_ALARM_NUMBER];           //ON or OFF
state_t actuatorState[MAX_ACTUATOR_NUMBER];     //ON or OFF

void vControl() {

  initGlobalVariables();

  period = 500 ms;

  while(1) {

    ticks = xTaskGetTickCount();

    updateSensors();

    updateAlarms();

    controlActuators();

    vTaskDelayUntil(&ticks, period);
  }
}
```

CÓDIGO 3.1. Pseudocódigo del lazo principal de control.

Capítulo 4

Ensayos y resultados

4.1. Pruebas funcionales del hardware

La idea de esta sección es explicar cómo se hicieron los ensayos, qué resultados se obtuvieron y analizarlos.

5.2. Próximos pasos

Acá se indica cómo se podría continuar el trabajo más adelante.

Capítulo 5

Conclusiones

5.1. Conclusiones generales

La idea de esta sección es resaltar cuáles son los principales aportes del trabajo realizado y cómo se podría continuar. Debe ser especialmente breve y concisa. Es buena idea usar un listado para enumerar los logros obtenidos.

Algunas preguntas que pueden servir para completar este capítulo:

- ¿Cuál es el grado de cumplimiento de los requerimientos?
- ¿Cuán fielmente se puedo seguir la planificación original (cronograma incluido)?
- ¿Se manifestó algunos de los riesgos identificados en la planificación? ¿Fue efectivo el plan de mitigación? ¿Se debió aplicar alguna otra acción no contemplada previamente?
- Si se debieron hacer modificaciones a lo planificado ¿Cuáles fueron las causas y los efectos?
- ¿Qué técnicas resultaron útiles para el desarrollo del proyecto y cuáles no tanto?

5.2. Próximos pasos

Acá se indica cómo se podría continuar el trabajo más adelante.

[1] Maximiliano Martín Aballay; Natalia Cristina Aguirre; Carla Valeria Filippi; Gabriel Hugo Valentini; Gerardo Sánchez. «Fine-tuning the performance of ddRAD-seq in the peach genome». En: *Scientific Reports* (2021).

[2] Departamento de Matemática Aplicada. *Redes Neuronales Convoluciones.* https://dcain.etsin.upm.es/~carlos/bookAA/05.7_RRNN_Convoluciones_CIFAR_10_INFORMATIVO.html. Mar. de 2024. (Visitado 26-03-2024).

[3] T. Dhikhi; Allagada Naga Suhas; Gosula Ramakanth Reddy; Kanadam Chandu Vardhan. «Measuring Size of an Object using Computer Vision». En: *International Journal of Recent Technology and Exploring Engineering (IJITEE)* (2019).

[4] Jose Luis Osorio Naranjo. «VIABILIDAD IDENTIFICACIÓN DE ESTADOS FENOLOGICOS EN LA ROSA APLICANDO ALGORITMOS DE RECONOCIMIENTO DE IMÁGENES». En: (2019).

[5] J. Li; R. Li; Y. Cui G. Li; L. Fu; C. Gao; W. Fang; G. Zhao; F. Shi. «Multi-class detection of kiwifruit flower and its distribution identification in orchard based on YOLOv5l and euclidean distance». En: *Comput. Electron. Agric.* (2022).

[6] J. Li; R. Li; Y. Cui G. Li; L. Fu; C. Gao; W. Fang; G. Zhao; F. Shi. «Real-time detection of kiwifruit flower and bud simultaneously in orchard using YOLOv4 for robotic pollination». En: *Comput. Electron. Agric.* (2022).

[7] D. Onoro-Rubio; R.J. López-Sastre. «Towards perspective-free object counting with deep learning». En: *Computer Vision ECCV 2016, Springer International Publishing* (2016).

[8] Y. Tian; X. Chu; H. Wang. «Cctrans: simplifying and improving crowd counting with transformer». En: *Computer Vision and Pattern Recognition (cs.CV)* (2021).

[9] Inc. The MathWorks. *¿Qué son las redes neuronales convolucionales?* https://es.mathworks.com/discovery/convolutional-neural-network.html. Mar. de 2024. (Visitado 26-03-2024).

[10] Departamento de Matemática Aplicada. *Redes Neuronales Convoluciones.* https://dcain.etsin.upm.es/~carlos/bookAA/05.7_RRNN_Convoluciones_CIFAR_10_INFORMATIVO.html. Mar. de 2024. (Visitado 26-03-2024).

[11] IBM. *¿Qué son las redes neuronales convolucionales?* https://www.ibm.com/es-es/topics/convolutional-neural-networks. Mar. de 2024. (Visitado 26-03-2024).

[12] Zoumana Keita. *¿Qué son las redes neuronales convolucionales?* https://www.datacamp.com/es/blog/yolo-object-detection-explained. Mar. de 2024. (Visitado 27-03-2024).
[13] Inc. The MathWorks. *Object Detection*. https://la.mathworks.com/discovery/object-detection.html. Mar. de 2024. (Visitado 27-03-2024).
[14] Sergio Hernán Valenzuela Cámara. «Detección y Clasificación de Enfermedades en el Tomate Mediante Deep Learning y Computer Vision». En: *Universidad Nacional de La Plata Facultad de Informática* (2021).
[15] Ross Girshick. «Fast R-CNN». En: *Computer Vision Foundation* (2015).
[16] Shaoqing Ren; Kaiming He; Ross Girshick; Jian Sun. «Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks». En: *Advances in neural information processing systems* (2015).
[17] Shahid Karim; Ye Zhang; Shoulin Yin; Irfana Bibi; Ali Anwar Brohi. «A brief review and challenges of object detection in optical remote sensing imagery». En: *Multiagent and Grid Systems An International Journal* (2020).

Bibliografía

[1] Maximiliano Martín Aballay; Natalia Cristina Aguirre; Carla Valeria Filippi; Gabriel Hugo Valentini; Gerardo Sánchez. «Fine-tuning the performance of ddRAD-seq in the peach genome». En: *Scientific Reports* (2021).

[2] DEBORAH PUEBLA. *lujan-y-tunuyan-las-zonas-mas-afectadas-por-las-heladas-tardias*. https://www.mendozapost.com/sociedad/lujan-y-tunuyan-las-zonas-mas-afectadas-por-las-heladas-tardias/. Oct. de 2023. (Visitado 12-10-2023).

[3] T. Dhikhi; Allagada Naga Suhas; Gosula Ramakanth Reddy; Kanadam Chandu Vardhan. «Measuring Size of an Object using Computer Vision». En: *International Journal of Innovative Technology and Exploring Engineering (IJITEE)* (2019).

[4] Jose Luis Osorio Naranjo. «VIABILIDAD IDENTIFICACIÓN DE ESTADOS FENOLOGICOS EN LA ROSA APLICANDO ALGORITMOS DE RECONOCIMIENTO DE IMAGENES». En: *UNIVERSIDAD EAFIT* (2019).

[5] J. Dhupia; K. Zhao; R. Li; Y. Cui G. Li; L. Fu; C. Gao; W. Fang; G. Zhao; F. Shi. «Multi-class detection of kiwifruit flower and its distribution identification in orchard based on YOLOv5l and euclidean distance». En: *Comput. Electron. Agric.* (2022).

[6] G. Li; R. Suo; G. Zhao; C. Gao; L. Fu; F. Shi; J. Dhupia; R. Li; Y. Cui. «Real-time detection of kiwifruit flower and bud simultaneously in orchard using YOLOv4 for robotic pollination». En: *Comput. Electron. Agric.* (2022).

[7] D. Oñoro-Rubio; R.J. López-Sastre. «Towards perspective-free object counting with deep learning». En: *Computer Vision ECCV 2016, Springer International Publishing* (2016).

[8] Y. Tian; X. Chu; H. Wang. «Cctrans: simplifying and improving crowd counting with transformer». En: *Computer Vision and Pattern Recognition (cs.CV)* (2021).

[9] Inc. The MathWorks. *¿Qué son las redes neuronales convolucionales?* https://es.mathworks.com/discovery/convolutional-neural-network.html. Mar. de 2024. (Visitado 26-03-2024).

[10] Departamento de Matemática Aplicada. *Redes Neuronales Convoluciones*. https://dcain.etsin.upm.es/~carlos/bookAA/05.7_RRNN_Convoluciones_CIFAR_10_INFORMATIVO.html. Mar. de 2024. (Visitado 26-03-2024).

[11] IBM. *¿Qué son las redes neuronales convolucionales?* https://www.ibm.com/es-es/topics/convolutional-neural-networks. Mar. de 2024. (Visitado 26-03-2024).

[12] Zoumana Keita. *¿Qué son las redes neuronales convolucionales?* https://www.datacamp.com/es/blog/yolo-object-detection-explained. Mar. de 2024. (Visitado 27-03-2024).

[13] Inc. The MathWorks. *Object Detection*. https://la.mathworks.com/discovery/object-detection.html. Mar. de 2024. (Visitado 27-03-2024).

[14] Sergio Hernán Valenzuela Cámara. «Detección y Clasificación de Enfermedades en el Tomate Mediante Deep Learning y Computer Vision». En: *Universidad Nacional de La Plata Facultad de Informática* (2021).

[15] Ross Girshick. «Fast R-CNN». En: *Computer Vision Foundation* (2015).

[16] Shaoqing Ren; Kaiming He; Ross Girshick; Jian Sun. «Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks». En: *Advances in neural information processing systems* (2015).

[17] Shahid Karim; Ye Zhang; Shoulin Yin; Irfana Bibi; Ali Anwar Brohi. «A brief review and challenges of object detection in optical remote sensing imagery». En: *Multiagent and Grid Systems An International Journal* (2020).

[18] Aditya Lohia; Kalyani Dhananjay Kadam; Rahul Raghvendra Joshi; Dr. Anupkumar M. Bongale. «Bibliometric Analysis of One-stage and Two-stage Object Detection». En: *University of Nebraska - Lincoln* (2021).

[19] Joseph Redmon; Santosh Divvala; Ross Girshick; Ali Farhadi. «You Only Look Once: Unified, Real-Time Object Detection». En: *Computer Vision and Pattern Recognition* (2015).

[20] Zoumana Keita. *YOLO Object Detection Explained*. https://www.datacamp.com/blog/yolo-object-detection-explained. Sep. de 2022. (Visitado 01-04-2024).

[21] Alireza Ghaffari; Yvon Savaria. *CNN2Gate: An Implementation of Convolutional Neural Networks Inference on FPGAs with Automated Design Space Exploration*. https://www.researchgate.net/profile/Alireza-Ghaffari-14/publication/347863856. Dic. de 2020. (Visitado 02-04-2024).