UNIVERSIDAD DE LA REPÚBLICA URUGUAY

FACULTAD DE INGENIERÍA

Sistema de aprendizaje continuo para la detección de caminos de hormigas

Informe de Proyecto de Grado presentado por

Gabriel Borges e Imanol González

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 López
Mercedes Marzoa Tanco

Montevideo, 7 de noviembre de 2025

Agradecimientos

En primer lugar, queremos agradecer a nuestros tutores, Mercedes y Gonzalo, por darnos la libertad y guiarnos a encontrar un punto de vista sobre el cual intentar aportar al control de hormigas.

En segundo lugar queremos agradecer especialmente a nuestras familias, que fueron nuestro apoyo en todos estos años de la carrera, y en particular durante el transcurso de este proyecto, todos se convirtieron en fotógrafos de hormigas para ayudarnos.

Por último, agradecemos tanto a los profesores y compañeros que fueron parte del proceso de formación que nos condujo hasta aquí.

Resumen

Las hormigas cortadoras de hojas representan una de las principales plagas agrícolas en Uruguay y la región, provocando pérdidas económicas significativas. Los métodos de control más efectivos actualmente dependen de la aplicación de cebos tóxicos, pero suelen implementarse de manera poco optimizada, con altos costos y riesgos ambientales. Un obstáculo central para desarrollar sistemas automáticos que permitan optimizar esta aplicación es la escasez de conjuntos de datos de entrenamiento adecuados para modelos de detección visual. Frente a esta limitación, el presente trabajo propone un sistema de aprendizaje activo orientado a la detección de caminos de hormigas, que permite construir de manera iterativa el conjunto de datos necesario mientras se mejora progresivamente el desempeño del modelo. Al mismo tiempo, se busca que el enfoque desarrollado pueda generalizarse a otras tareas de visión por computadora en las que exista una disponibilidad limitada de datos, ofreciendo una forma sistemática y eficiente de mejorar modelos de detección en escenarios prácticos.

Para ello se integró la plataforma CVAT (Computer Vision Annotation Tool), destinada al etiquetado asistido de imágenes, con un sistema que permite seleccionar automáticamente los ejemplos más informativos para mejorar el modelo. Se implementaron y compararon distintas estrategias de selección, incluyendo métodos basados en confianza, en similitud entre representaciones de imágenes, en disimilitud y en la incorporación de fondos (*backgrounds*) controlados.

Para la experimentación, se trabajó con dos conjuntos de datos, uno más extenso basado en la detección de ciclistas que permitió hacer un análisis más completo de los resultados obtenidos y otro más reducido pero aplicado a la detección de los caminos de hormigas para validar su utilización en esta realidad. Los experimentos mostraron que la utilización del sistema permite obtener mejoras progresivas en el desempeño del modelo con menor cantidad de datos etiquetados.

En conclusión, el sistema desarrollado demostró ser una solución viable y adaptable tanto para el problema del control de hormigas como para otras tareas de reconocimiento visual en las que la disponibilidad de datos etiquetados sea limitada. Se establece así una base para futuras investigaciones orientadas a ampliar el marco de aprendizaje activo, diversificar los criterios de selección de datos y explorar su aplicación en distintos dominios de aprendizaje automático enfocado a tareas visuales.

Palabras clave: Hormigas cortadoras de hojas, aprendizaje automático, aprendizaje activo, YOLO, CVAT

Índice general

1. Introducción **1**
- 1.1. Motivación . . . 1
- 1.2. Objetivos . . . 2
- 1.3. Estructura del informe . . . 2

2. Revisión de antecedentes **5**
- 2.1. Comportamiento de las hormigas . . . 5
  - 2.1.1. Organización . . . 5
  - 2.1.2. Fuente de alimentos . . . 7
  - 2.1.3. Efectos naturales en el forrajeo . . . 7
- 2.2. Métodos de control . . . 8
  - 2.2.1. Manipulación ambiental del campo cultivado . . . 8
  - 2.2.2. Insecticidas . . . 10
- 2.3. Detección . . . 11
  - 2.3.1. Detección con vehículos aéreos . . . 12
  - 2.3.2. Detección autónoma terrestre . . . 13
- 2.4. Aprendizaje Activo . . . 18
  - 2.4.1. Técnicas de selección de candidatos . . . 19
  - 2.4.2. Posibles problemas . . . 21
- 2.5. CLIP . . . 22
  - 2.5.1. Embeddings en CLIP . . . 22
- 2.6. YOLO . . . 23
  - 2.6.1. Métricas de rendimiento en modelos YOLO . . . 23
  - 2.6.2. Variantes de los modelos YOLO . . . 25
- 2.7. Oso hormiguero artificial . . . 26
  - 2.7.1. Active Road Pipeline (ARP): *Pipeline* de detección de caminos activos . . . 27
  - 2.7.2. Desarrollo de un oso hormiguero artificial . . . 28

3. Problema y diseño de la solución **29**
- 3.1. Descripción de la solución . . . 30
  - 3.1.1. Conceptos claves . . . 30
  - 3.1.2. Solución planteada . . . 30
- 3.2. Técnicas de selección de candidatos . . . 36

3.2.1. Método basado en confianza . . . 36
3.2.2. Método basado en confianza y similitud . . . 36
3.2.3. Método basado en confianza y disimilitud . . . 37
3.2.4. Método basado en confianza y *backgrounds* con similitud . 38
3.3. Arquitectura de la solución . . . 39
3.3.1. Capa de datos . . . 39
3.3.2. Capa Lógica . . . 41
3.3.3. Diagrama de componentes . . . 41
3.3.4. Despliegue de la solución . . . 43

4. Experimentación **45**
4.1. Datos experimentales . . . 45
4.2. Entorno de pruebas . . . 48
4.3. Modelos iniciales . . . 48
4.4. Pruebas de aprendizaje activo, método basado en confianza . . . 52
4.5. Pruebas de evaluación de la similitud entre imágenes mediante CLIP . . . 57
4.6. Agregando *backgrounds* . . . 58
4.7. Similitud al conjunto de datos disponible . . . 60
4.8. Disimilitud al conjunto de entrenamiento . . . 62
4.9. Evaluación comparativa de iteraciones con distintas cantidades de imágenes . . . 62
4.10. Análisis de los resultados . . . 65
4.11. Experimentación sobre caminos de hormigas . . . 67
4.11.1. Análisis cualitativo de la evolución de los modelos . . . 72
4.11.2. Comparación con los resultados obtenidos en los antecedentes de búsqueda de caminos de hormigas . . . 75

5. Conclusiones y Trabajo Futuro **79**
5.1. Trabajos a futuro . . . 80
5.1.1. Líneas de investigación en el control de hormigas . . . 80
5.1.2. Líneas de investigación para la mejora del sistema . . . 81

Referencias **85**

A. Anexo 1 **89**
A.1. Evaluación de la medida de similitud propuesta . . . 90
A.2. Evaluación seleccionando imágenes aleatorias . . . 95
A.3. Evaluación seleccionando imágenes basados unicamente en la confianza . . . 95
A.4. Evaluación seleccionando una cantidad mínima de *backgrounds* . . . 96
A.5. Evaluación seleccionando imágenes similares al conjunto . . . 98
A.6. Evaluación seleccionando imágenes disimilares al conjunto de entrenamiento . . . 100
A.7. Evaluación comparativa de iteraciones con distintas cantidades de imágenes . . . 102

A.8. Experimentación sobre caminos de hormigas 102

Capítulo 1

Introducción

1.1. Motivación

Las hormigas cortadoras de hojas son los insectos nativos herbívoros dominantes en diversos ecosistemas naturales de Sudamérica y, por lo tanto, cuando se implantan en los cultivos, pueden transformarse en plagas primarias que ocasionan daños de importancia a la agricultura y la forestación. Los insecticidas son el método de control más eficaz con el que contamos para lidiar con esta problemática, en particular los cebos tóxicos. Estos últimos funcionan por ingestión. Las hormigas los confunden con el material que recolectan, lo llevan a su nido y contagian a las demás.

No existe una metodología clara de como aplicar este cebo. Una de las estrategias sugeridas por Zerbino (2002), investigador del INIA, es la de esparcirlo en un patrón predeterminado (cada cierta cantidad de metros). Heurísticas como esta pueden ser una buena guía inicial de dónde aplicar este insecticida. Sin embargo, si se tuviera un mejor conocimiento del terreno, quizás conociendo aproximadamente las zonas en las que circulan las hormigas, se podría hacer una aplicación más adecuada en cuanto a la cantidad de cebo utilizada, y que posiblemente además mejore la eficacia de este. Se plantea la hipótesis de que las zonas en las que circulan las hormigas corresponden a áreas donde se encuentran caminos de hormigas, y colocar el cebo cerca de estos aumenta las posibilidades de ser detectado y trasladado al nido. La identificación de estos trayectos representa una tarea compleja y demandante. Por ello, se explorarán distintos métodos de reconocimiento visual que permitan abordar esta problemática de forma eficiente.

Sin embargo, estos procedimientos enfrentan un problema general: la necesidad de contar con un conjunto de datos lo suficientemente amplio como para entrenar un modelo de aprendizaje automático que pueda reconocer eficazmente los caminos de hormigas.

Así es que en el transcurso de este trabajo se propone un *framework* general de aprendizaje activo que permita construir el conjunto de datos requerido para

esta y otras tareas similares de manera iterativa, mejorando el modelo predictivo a medida que se utiliza en la práctica.

Este *framework* se aplicará experimentalmente al problema de reconocimiento de caminos de hormigas, mostrando más adelante los resultados obtenidos. Pero además se espera que el *framework* pueda generalizarse a distintas tareas de reconocimiento visual, haciendo que este sea de utilidad al aplicarlo a otros problemas para los que se cuente con una cantidad limitada de datos entrenamiento, o simplemente para la utilización de aquellos usuarios que deseen mejorar su modelo predictivo, de una manera eficiente y sistemática, con datos obtenidos en la práctica.

1.2. Objetivos

El objetivo principal del proyecto es desarrollar un sistema de aprendizaje activo, el cual permita iterativamente mejorar un modelo de aprendizaje automático que sea utilizado para tareas de reconocimiento visual. Pudiendo contar con distintas versiones del modelo y a su vez poder obtener estadísticas de ellas.

El resto de los objetivos son referentes a la tarea particular de reconocimiento de caminos de hormigas. Por lo tanto, se plantean además los siguientes objetivos:

- Estudiar las especies de hormigas cortadoras de hojas que se encuentran en Uruguay. Además, estudiar el comportamiento de estas, su organización, fuentes de alimento y patrones de recolección.
- Investigar los métodos de control más eficaces utilizados en Uruguay para la problemática de las hormigas.
- Relevar el estado del arte en cuanto a tareas de reconocimiento visual, en especial aplicadas a problemas similares al de detección de caminos de hormigas.
- Relevar el estado del arte de los métodos de aprendizaje activo, principalmente referido a la selección de candidatos.
- Experimentar con el sistema de aprendizaje activo diseñado, aplicándolo al problema de la detección de caminos de hormigas.

1.3. Estructura del informe

El documento se divide en los capítulos descritos a continuación. El segundo capítulo cuenta con la investigación previa de todos los conocimientos necesarios para el proyecto. El estudio de las hormigas cortadoras de hojas en el Uruguay, especies, comportamiento y métodos de control. Así como antecedentes de proyectos que utilicen visión por computadora, y en particular para detectar

hormigas. Por último, cuenta con la investigación del Aprendizaje Activo. En el capítulo tres se presenta el problema a resolver. A su vez, se muestra la solución planteada a dicho problema, las técnicas utilizadas y la arquitectura del sistema. En el capítulo cuatro se realizan los experimentos, se presentan los datos utilizados y los resultados de utilizar el sistema. Por último, en el capítulo cinco se presentan las conclusiones y líneas futuras de investigación que aportarían al desarrollo del proyecto.

Capítulo 2

Revisión de antecedentes

2.1. Comportamiento de las hormigas

En el artículo "La problemática del manejo de hormigas cortadoras"de la revista INIA (Bollazzi, Sabattini, Pilón, Vega, y Martínez, 2023), se presentan las cuatro especies de hormigas cortadoras de hojas que habitan en Uruguay. Estas son *Acromyrmex heyeri*, *Acromyrmex lundii*, *Acromyrmex balzani*, *Amoimyrmex striatus*, imágenes de estas se presentan en la Figura 2.1. Algunas de ellas, construyen nidos en forma de montículo, como la *Acromyrmex heyeri* y *Acromyrmex balzani*, los cuales son fáciles detectar a simple vista. Por otro lado, *Acromyrmex lundii* y *Amoimyrmex striatus* construyen sus nidos de forma subterránea, lo que los hace particularmente difíciles de detectar. Se pueden ver ejemplos de los hormigueros de cada especie en la Figura 2.2.

Las cuatro especies predominantes de Uruguay pertenecen al género *Acromyrmex*, el cual, junto al género *Atta* conforman la tribu *Attina*, las conocidas como cortadoras de hojas. En su trabajo, Fisher y colaboradores (1994) presentan la razón de su nombre, tanto las *Atta* como las *Acromyrmex* cortan material vegetal fresco como sustrato para el hongo *Leucoagaricus gongylophorus* el cual será el único alimento de toda la colonia. Esta es la razón por la cual son una plaga para las actividades agropecuarias: atacan principalmente plantas poco maduras que están en su proceso de crecimiento, una etapa vulnerable en la cual los daños causados por las hormigas podría llevar a la muerte total de las plantas, causando pérdidas económicas por la baja en la producción.

2.1.1. Organización

Según el trabajo realizado por Della Lucia, Gandra, y Guedes (2014), en una colonia existen cuatro tipos de hormigas, la reina, machos y hembras alados y obreras. El objetivo de estas está bien definido, la reina es única, es la encargada de la reproducción de la colonia y en caso de morir, la colonia termina. Por otro lado, las obreras se encargan del cuidado de las larvas, defensa de la colonia, y recolección del alimento del hongo. Por último, una vez la colonia se desarrolló,

(a) Acromyrmex heyeri

(b) Acromyrmex lundii

(c) Acromyrmex balzani

(d) Acromyrmex striatus

Figura 2.1: Especies de hormigas en Uruguay. Imágenes obtenidas de Antweb (2025).

(a) Hormiguero Acromyrmex heyeri

(b) Hormiguero Acromyrmex lundii

(c) Hormiguero Acromyrmex balzani

(d) Hormiguero Acromyrmex striatus

Figura 2.2: Tipos de hormigueros en Uruguay. Imágenes (a) y (b) adaptadas de (Bollazzi, 2014). Imágenes (c) y (d) adaptadas de (Bollazzi y cols., 2023).

habrá periodos donde nacen machos y hembras alados, los cuales, mediante vuelos nupciales, se reproducirán, para luego abandonar el nido, siendo las hembras nuevas reinas de sus propios nidos.

2.1.2. Fuente de alimentos

El micelio vegetativo de *Leucoagaricus* (Figura 2.3) es el único alimento de las larvas de hormiga. Las hormigas adultas llevan fragmentos del hongo a las cámaras de cría para alimentar a las larvas. La supervivencia de la colonia depende de que puedan mantener el bienestar de este hongo. Es por esto que las hormigas tienen mecanismos de higiene para cuidar que este no sea contaminado. La mirmicacina, $C_{10}H_{20}O_3$, es un antibiótico secretado por las hormigas, que inhibe el crecimiento de muchos hongos, incluidos los organismos típicos del suelo como *Penicillium* y epífitas en plantas como *Cladosporium* y *Alternaria* (Fisher y cols., 1994). Esta información será relevante más adelante en la sección 2.2, ya que, unos de los métodos de control estudiados implica la utilización de esta fuente de alimento.

2.1.3. Efectos naturales en el forrajeo

En las épocas de forraje de las hormigas, distintos factores naturales pueden afectar el comportamiento de las mismas. Un claro ejemplo de esto es el viento. En su trabajo, Marina Alma, Farji-Brener, y Elizalde (2016) plantean que la

Figura 2.3: Cultivo de hongos dentro de un nido de hormigas. Imagen adoptada de (Fisher y cols., 1994).

distribución del tamaño de las hormigas está sesgada hacia hormigas más grandes en condiciones ventosas, en comparación con condiciones sin viento. Esto para ambas hormigas, salientes del nido y las que traen la carga. La diferencia en el tamaño se puede observar en la Figura 2.4

2.2. Métodos de control

Con el fin de reducir el nivel de perdidas en el sector agropecuario a causa de la acción de las hormigas cortadoras de hojas, se han definido métodos de control para reducir o eliminar las colonias.

2.2.1. Manipulación ambiental del campo cultivado

La diversificación de los sistemas de cultivos puede utilizarse como método de control. Recurriendo nuevamente al trabajo realizado por Della Lucia y cols. (2014) se explica que la conservación de vegetación de sotobosque en plantaciones de Eucalipto y/o el mantenimiento de franjas de vegetación nativa en el campo, permite que las hormigas tengan una variedad grande de plantas para forrajear, haciendo que estas se inclinen por aquellas que ya conocen, cómo las nativas, así evadiendo las que se desea cultivar. Además, la diversificación permite que haya un incremento de enemigos naturales que probablemente limiten la expansión de las colonias.

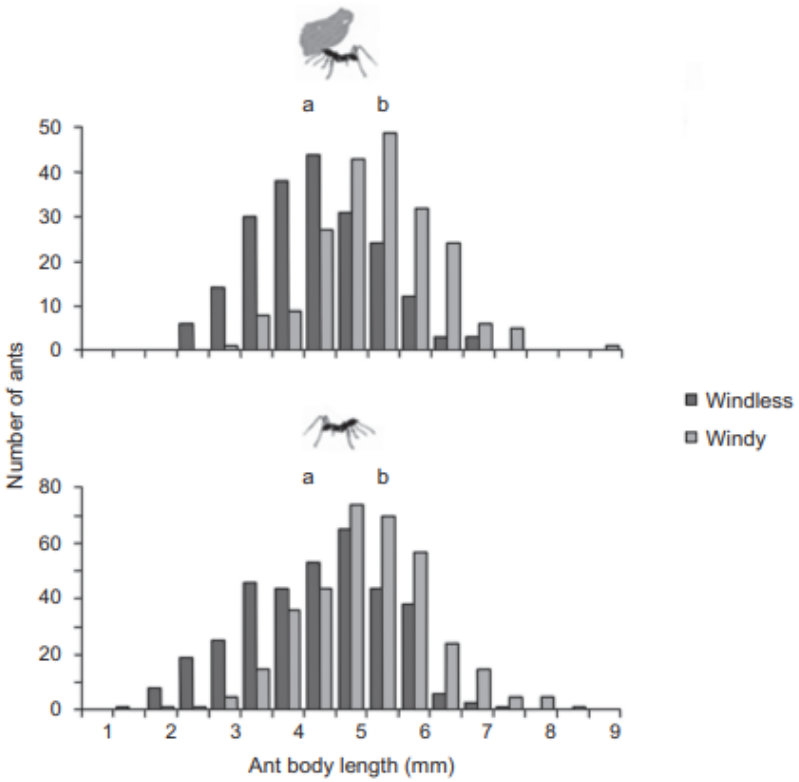


Figura 2.4: Diferencias observadas en el tamaño de las hormigas (en milímetros) dependiendo del viento. Imagen obtenida de (Marina Alma y cols., 2016).

2.2.2. Insecticidas

Los insecticidas se presentan como el método más eficaz para el control de las hormigas cortadoras de hojas si se tiene identificada la ubicación del nido. Existen cuatro métodos principales de aplicación en las colonias: aplicación en polvo, fumigación, nebulización térmica y cebos tóxicos.

La aplicación de formulaciones en polvo seco se logra bombeando el insecticida al interior del nido a través de su apertura principal y orificios activos. Della Lucia y cols. (2014) presentan que, utilizando organofosfatos, carbamatos y particularmente piretroides como polvo seco, se ha logrado proporcionar un control eficaz en condiciones ambientales secas y contra nidos pequeños y poco profundos. No así para los nidos profundos, especialmente los observados en la especie Atta, ya que el polvo no llega a las cámaras más protegidas donde habita la reina.

El uso de fumigantes se restringía principalmente al uso de bromuro de metilo. Inyectándolo en el nido en estado líquido, por algún orificio abierto, y evaporándose dentro por la reducción en la presión. En estado gaseoso podía llegar incluso a las cámaras más profundas, suprimiendo la colonia. En el protocolo de Montreal (1987), se establece la prohibición de este producto debido a sus efectos adversos sobre la capa de ozono. En particular, citando al ministerio de ambiente de Uruguay (2024): "Actualmente su uso permitido se reduce a las aplicaciones de cuarentena y preembarque de cargas de exportación en el caso de que el destino así lo exija."

Cebos tóxicos

Los cebos tóxicos funcionan por ingestión. El objetivo es que las hormigas obreras los confundan por el material vegetal que forrajean y lo transporten al nido, donde serán procesados. Las hormigas despedazan los trozos de cebo, en pequeñas partículas que se incorporan al hongo simbionte. En todo este proceso las hormigas se van contagiando una a una el insecticida mediante intercambios de alimento. Resultando en que toda la colonia quede infectada.

Es de suma importancia que el insecticida sea de acción lenta. De lo contrario, las primeras obreras que se intoxiquen serán eliminadas rápidamente y no podrán esparcir el insecticida por el resto de la colonia.

Según el artículo del INIA publicado por Bollazzi y cols. (2023), los cebos raramente alcanzan a controlar el total de las colonias y en general logran eliminar aproximadamente entre el 70 y el 90 % de la población de colonias de un sitio determinado. El cebo se aplica en los días que hay actividades de corte, pero puede ocurrir que la colonia no lo acepte, o peor aún, que se vea la aceptación y el transporte, pero que lo rechacen luego dentro de la colonia, descartándolo fuera del nido varias horas después. Como se mencionó anteriormente, la acción del cebo es tardía, y podría hacer efectos hasta 30 días después de su uso, esto implica que no parará el forrajeo inmediatamente, algo a tener en cuenta sobre todos cuando las plantaciones son nuevas y están vulnerables.

La estrategia recomendada por el INIA explica que: "si se desea manejar

ÉPOCAS DE CONTROL: recomendación cebo granulado											
MALA	BUENA			MUY MALA			EXCELENTE				MALA
Ene	Feb	Mar	Abr	May	Jun	Jul	Ago	Sep	Oct	Nov	Dic

Tabla 2.1: Esquema que muestra los meses en los cuales es más efectivo utilizar los métodos de control para las hormigas cortadoras de hojas. Tabla obtenida de (Bollazzi y cols., 2023).

las hormigas cortadoras con resultados de mediano-largo plazo, las tácticas de control deben ser ejecutadas en los meses cuando las probabilidades de que las colonias estén visibles sean las mayores". Estos períodos se observan en la Tabla 2.1. A su vez, debe evitarse su uso en días de lluvia, ya que esta los inhabilita. En cuanto a su aplicación, no depende de encontrar el hormiguero. Zerbino (2002) presenta la estrategia heurística de realizar una pasada de fertilizadora de péndulo con un ancho de trabajo de 6 metros, y repetir este proceso, dejando 41 metros sin aplicar, reduciendo la cantidad usada sin grandes cambios en la efectividad.

2.3. Detección

Como se expuso en la sección anterior, el método de control mediante cebos tóxicos es muy bueno, pero este se basa en realizar un uso desmedido de cebos, ya que se busca evitar encontrar los nidos. Por esto sería importante detectar previamente rastros de hormigas con el fin de poder utilizar el cebo de forma más precisa, reduciendo costos y posibles riesgos medioambientales. Es por esto que se investiga la detección de hormigas, automatizada usando visión por computadora. Este punto es aún más relevante cuando se quiere implementar otros medios de control, donde es necesario primero encontrar el hormiguero que requiere mucha mano de obra.

La detección de hormigas es un área de trabajo aún bastante inmadura, su implementación suele ser en entornos controlados, donde el foco está en el seguimiento de las mismas para estudiar su comportamiento, mas no para encontrarlas. Por ejemplo, Sabattini y colegas (2023) presentan un dispositivo que permite el seguimiento específico de cada hormiga, pudiendo contar cuántas atraviesan cierto tramo de un camino y si lo hacen cargadas o no. Esta corriente de investigación es útil para entender el comportamiento de las hormigas, pero poco aporta a la detección de las mismas en entornos no controlados.

A continuación se presentan diferentes ramas de investigación, basadas en detección con vehículos aéreos y vehículos terrestres, que buscan el poder detectar la presencia de hormigas en la naturaleza de forma automática. Esto con el fin de controlar su población. [1]

[1] Sumados a estos, existen también los métodos de detección satelital. Estos no serán discutidos en este informe debido a las limitaciones en su aplicación.

Figura 2.5: Nidos de *formica exsecta* en la tundra sueca. Obtenido de (Monsimet y cols., 2024).

2.3.1. Detección con vehículos aéreos

La detección mediante vehículos aéreos no tripulados parece ser la más relevante en cuanto a cantidad de artículos académicos se refiere. Entre los más interesantes se encuentra el trabajo realizado por Monsimet y cols. (2024). Donde utilizando UAVs (vehículos aéreos no tripulados) buscan la detección de nidos de la hormiga *formica exsecta* en la tundra sueca (Figura 2.5). El foco de su investigación está en probar diferentes espectros de luz y ver su desempeño en la detección de nidos. Para esto usaron cámaras RGB, cámaras multiespectrales y térmicas. A partir de esta estas tres fuentes llegan a un conjunto de diferentes espectros.

- RGB - imagen natural.
- Red - toman el espectro de luz del rojo.
- NIR - espectro cercano al infra rojo.
- Thermal - información obtenida de cámaras térmicas.
- NDVI - es el cálculo de: NIR-R / (NIR+R).

Sus pruebas fueron realizadas con un conjunto de datos de 2130 imágenes, con un total de 1013 montículos de hormigueros, teniendo 703 imágenes que al menos contenían uno. Los resultados se pueden ver en la Tabla 2.2, donde se observa que las imágenes visibles para el ojo humano (RGB) son las mejores en cuanto al desempeño, comentan que incluso el resultado es superior al obtenido por observadores humanos. A su vez, los observadores tardaban alrededor de 90 minutos por imagen para detectar los nidos, cuando el modelo lo hacía en menos de tres minutos. Es importante notar que esta imagen es la unión de todas las imágenes tomadas por el UAV en un área de 1,58 hectáreas, esta es la razón de los tiempos tan elevados.

Dentro de sus conclusiones, se encuentra que es muy factible la utilización de UAVs para la detección de montículos, pero dentro de sus limitaciones está que los hormigueros que estén debajo de árboles no eran visibles, algo que ocurre muy poco en su ecosistema, pero en otros, podría ser un gran problema, más

UAV	Precisión	Recall
RGB	99 ± 1	68 ± 11
Thermal	80 ± 18	76 ± 3
NDVI	62 ± 17	68 ± 2
Red	74 ± 9	58 ± 10
NIR	52 ± 43	15 ± 32

Tabla 2.2: Resultados de la detección para los distintos espectros visuales utilizados por los UAV. Tabla adaptada de: (Monsimet y cols., 2024).

adelante en la sección 2.3.2 se hablara sobre la detección autónoma terrestre que no tiene este problema.

Además, otro inconveniente es que únicamente buscan montículos, son incapaces de detectar hormigueros que no generen grandes estructuras hacia el exterior, esto mismo sucede en el artículo presentado por Almeida, Bertacine, Hernandes, y Carmona (2023), que se reducen a la búsqueda de *Atta Capiguara* en Brasil, que en su hormiguero genera un montículo con forma de volcán. En este caso, se decidió utilizar un dron FIMI X8 20 SE, el cual tiene un costo reducido y únicamente cuenta con una cámara RGB. Se tomaron fotos entre cinco a veinte metros, siempre al medio día y en días despejados para mantener la cromaticidad, de las cuales se utilizaron treinta y cinco. En su caso, utilizaron el algoritmo *Single Linear Iterative Clustering*, el cual divide las imágenes en una cuadrícula no fija, buscando agrupar píxeles similares en lo que llaman *SuperPíxeles*. Los cuales, posteriormente, etiquetan según a la clase que pertenezcan la mayoría de los píxeles del SuperPíxel. Como la mayoría de sus SuperPíxeles pertenecían a las clases "Tierra" o "Hierba" su conjunto de datos resultó muy desbalanceado. Es por esto que se optó por utilizar un algoritmo *Random Forest Classifier*, que funciona bien para este tipo de conjuntos de datos desbalanceados. Como resultados, obtuvieron un $71\,\% \pm 2\,\%$ de precisión global.

2.3.2. Detección autónoma terrestre

Yendo al enfoque que se busca en el proyecto, solo se encontró evidencia de un único proyecto que busque la detección de hormigueros mediante robots autónomos terrestres. Su y cols. (2023) buscan enfrentarse a la problemática existente en China con la invasión de hormigas RIFA (Red imported fire ant), las cuales afectan negativamente los ecosistemas autóctonos. Durante el transcurso del proyecto buscan la detección y posterior control únicamente de Nidos de RIFA, pudiendo diferenciarlos de los de las especies autóctonas. En esta sección se desglosará los puntos más relevantes de este proyecto, los cuales pueden ser tomados como puntapié inicial para subsecuentes proyectos dentro de esta temática.

Figura 2.6: CyberDog de Xiaomi. Imagen obtenida de la pagina oficial de Xiaomi (www.mi.com).

Tecnologías

Para la implementación, como base utilizaron el *CyberDog* de Xiaomi (Figura 2.6), un cuadrúpedo de alrededor de nueve kilos que puede alcanzar una velocidad de 3,2 m/s. Para el procesamiento cuenta con un Jetson Xavier NX AI de NVIDIA con 384 núcleos CUDA, 48 núcleos Tensor, una CPU 6 Carmel ARM y dos motores de aceleración de aprendizaje profundo. En cuanto a sensores, cuenta con una cámara RGB, cuatro sensores TOF, un sensor LIDAR, una cámara de profundidad, un sensor de ultrasonido y cuatro micrófonos. El código del mismo es *open source*, pudiendo modificar y aplicar las mejoras que se crean necesarias.

Para definir el comportamiento buscado se utilizó ROS (un conjunto de bibliotecas y herramientas de software que ayudan a construir aplicaciones orientadas a la robótica), en donde se modificó el módulo de *tracking* del CyberDog. Usando *Open CV* se obtenían las imágenes de las cámaras, las cuales era procesada por una red YOLOX para la detección de hormigueros. La información de navegación y de objetivos se guarda fuera del robot para su posterior análisis, mientras que su distribución en tiempo real entre los distintos módulos del sistema se realiza mediante Cyclone DDS, un *middleware* de comunicación orientado a datos.

Flujo de trabajo

Se define un flujo de trabajo que permite la detección y el control de hormigueros, así como el mantenimiento y la mejora continua del sistema de detección. Dicho flujo se muestra en la Figura 2.7.

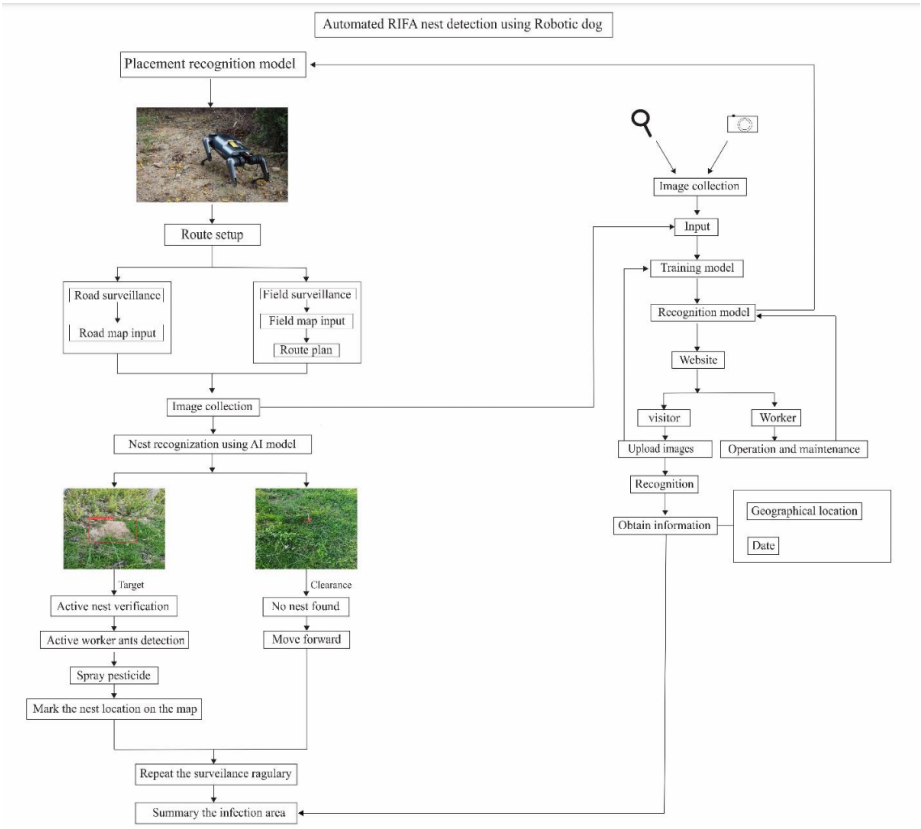


Figura 2.7: Flujo de trabajo para la detección de hormigueros de RIFA. Imagen obtenida de (Su y cols., 2023).

Una vez planificada la ruta de patrullaje por los operadores, el *CyberDog* la sigue mientras captura imágenes con su cámara integrada, estas imágenes son respaldadas utilizando como medio de transmisión Cyclone DDS, las cuales posteriormente serán revisadas por un técnico y etiquetadas para confirmar si son un nido o no. Esto se realiza para cada una, sin importar si lo es o no. En el artículo no se profundiza sobre cómo es este proceso, pero más adelante, en la sección 2.4.1 se presentan métodos para filtrar imágenes de un conjunto, según la información que puedan aportar a nuevas versiones del modelo. A su vez, además de las imágenes que son obtenidas por el robot, en su flujo de trabajo, proponen la implementación de una página web, en la que las personas pueden subir sus imágenes alimentando el conjunto a utilizar por el modelo.

Luego de utilizar la imagen proporcionada por el robot como entrada para futuras versiones del modelo, se evalúa, y en el caso de quedar dentro del intervalo de confianza necesario, se procede a realizar una confirmación, se puede observar una representación del mismo en la Figura 2.8. El comportamiento se basa en situarse a 1,2 metros del hormiguero y obtener el nivel de confianza que retorna el modelo, posteriormente, acercase hasta los 0,6 metros y volver a tomar el nivel de confianza para esta nueva visualización. En el caso de que el nivel de confianza aumente, se considera que efectivamente es un nido de RIFA. Posteriormente, dada la naturaleza luchadora de las hormigas RIFA, el robot se acerca, y perturba el nido. En el caso de que las hormigas salgan se confirma que es un nido activo. Si es así, se esparce el pesticida y se marca la ubicación.

Entrenamiento y Resultados

El entrenamiento del modelo se realizó utilizando 1.519 imágenes con una proporción de aproximadamente 75-25 entre el conjunto de entrenamiento y validación. Aunque no especifican utilizar un *dataset* de evaluación separado, más adelante evalúan el sistema comparando su eficacia contra humanos. Dado que en su proyecto buscaban únicamente detectar a las hormigas de la especie RIFA, dentro del conjunto de datos se incluyeron 100 imágenes de otras especies, con el fin de anotarlas como clases negativas. Para el entrenamiento, se utilizó la implementación *YOLOX* de una red *YOLO*, usando 400 épocas. Como resultado, obtuvieron que el tiempo de procesamiento sea de 20,16 milisegundos en la especificación mencionada en la sección 2.3.2 sobre la capacidad de cómputo del CyberDog. Para la validación del modelo, junto con la eficiencia del robot, capacitaron a tres personas durante tres horas y los hicieron ubicar nidos en un área de 300 metros cuadrados, misma tarea que le fue encomendada al robot. De este experimento se obtuvo que tanto los humanos como el robot tardaron menos de tres minutos, siendo los humanos un poco más veloces. En contraparte, el robot detecto tres veces más nidos y fallo menos en falsas detecciones (nidos inactivos o no nidos) y faltas (nidos de otras especies). Reportan tener una tasa de éxito del 90 % en la detección de nidos, siendo las fallas en nidos muy pequeños que aún están en desarrollo. Por último, presentan los resultados de su modelo en diferentes casos de nidos. Como se puede ver en la Figura 2.9, los nidos despejados son más fáciles de visualizar, encontrándose todos en

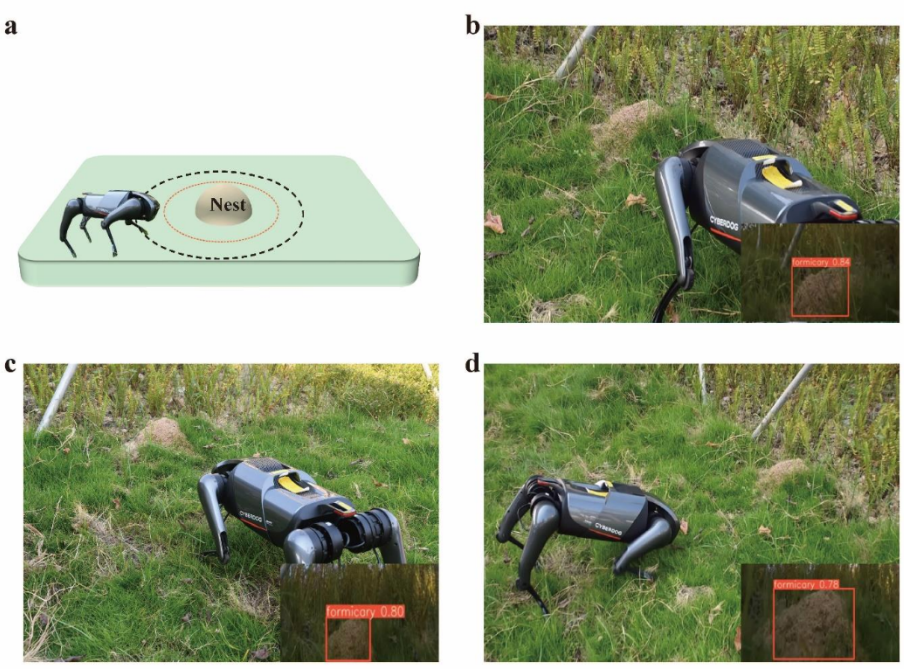


Figura 2.8: Representación del comportamiento del Cyberdog acercándose a obtener nuevas predicciones para confirmar el nido. Imagen obtenida de (Su y cols., 2023).

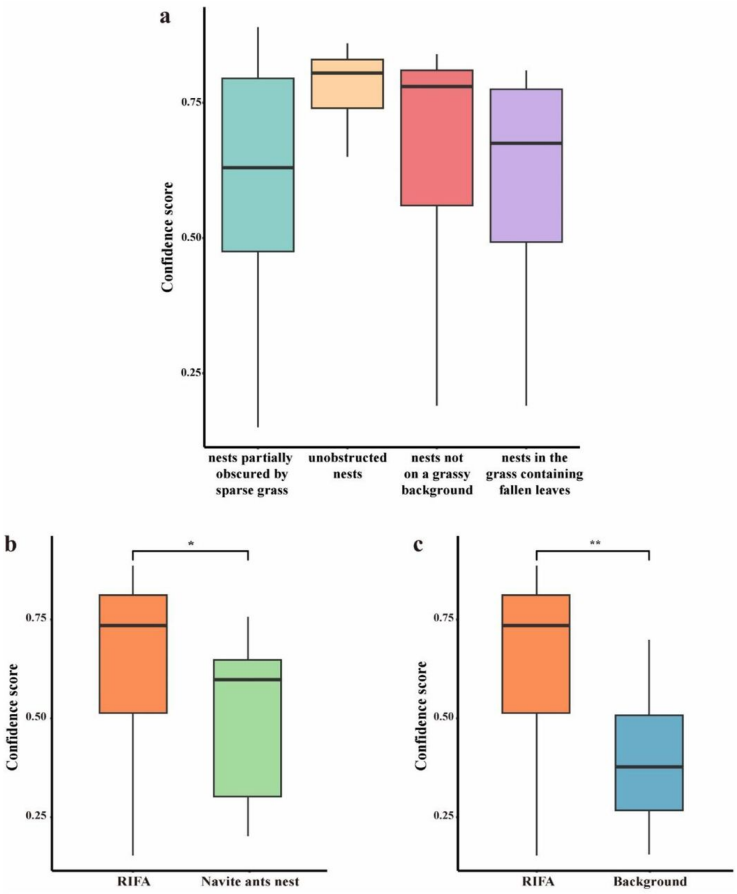


Figura 2.9: Intervalos de confianza en la detección de los diferentes tipos de nidos. Imagen obtenida de (Su y cols., 2023).

el intervalo de alrededor de 75 % de confianza, por otro lado, a medida que las imágenes de los nidos están más contaminadas por factores externos como la maleza, es necesario reducir este intervalo de confianza hasta el 50 % para detectar todos los casos.

2.4. Aprendizaje Activo

Durante la búsqueda de conjuntos de datos para construir un modelo propio, se identificó una problemática importante: la escasez de datos etiquetados (principalmente de buena calidad) que contengan rastros de hormigas. Dentro de la información existente se hace foco en hormigueros y hormigas, sin tener en cuenta apariciones de caminos u otro indicador de presencia de hormigas. Es por esto, que se decidió buscar formas de realizar un entrenamiento más

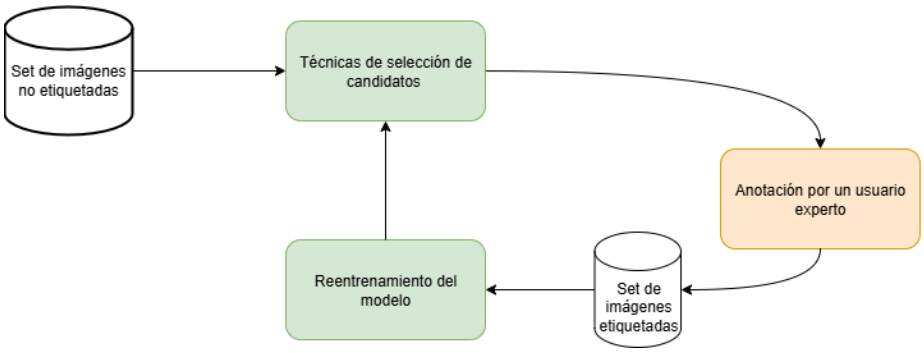


Figura 2.10: Pipeline general de Aprendizaje Activo.

eficiente, con una menor cantidad de datos, que a su vez, permita una mejora continua del modelo utilizando la información que sea recolectada mediante el uso del sistema. En la publicación de Li y cols. (2024) se presenta al aprendizaje activo como una solución viable para abordar el desafío de etiquetar grandes cantidades de datos en aplicaciones que requieren un uso intensivo de datos, como la visión artificial y la traducción automática. El objetivo principal del aprendizaje activo es identificar automáticamente un subconjunto de muestras de datos no etiquetados para su anotación, tratando de maximizar la mejora del modelo al incluir estos nuevos datos. Se presenta un *Pipeline* general de Aprendizaje Activo en la Figura 2.10.

2.4.1. Técnicas de selección de candidatos

El paso más importante del aprendizaje activo es la selección de los datos candidatos a ser etiquetados. Se tiene que obtener de alguna forma aquellos datos que, al ser etiquetados y utilizados en el entrenado del modelo, maximicen la mejora de este. En su trabajo, Wu, Li, y Yao (2022) discuten distintas estrategias utilizadas:

Selección aleatoria

El muestreo aleatorio consiste en hacer una selección aleatoria de cierta cantidad de imágenes del conjunto de datos sin etiquetar. Es usado normalmente como punto de comparación de las otras estrategias de selección.

Basados en incertidumbre

La selección basada en la incertidumbre es el método de selección más común. Trabaja sobre la hipótesis que la muestra con el menor valor de confianza respecto a la clase predicha será la que mayor información podrá aportarle al modelo. Existen distintas heurísticas sobre como seleccionar estas muestras. Entre ellas se encuentran, tomar las de menor confianza, es decir, para los cuales el modelo

tiene una incertidumbre mayor. Otra posible estrategia, en el caso de tener más de una clase, es calcular la diferencia entre las probabilidades de pertenecer a cada una y tomar las de menor diferencia, a esta técnica se le llama muestreo de margen.

Basados en diversidad

Para escenarios con múltiples candidatos surge la estrategia de selección basada en la diversidad de la distribución de las características de la muestra. Donde es atractivo seleccionar los datos más diferentes entre sí.

Selección basada en un comité

Otro posible método de selección es el basado en un comité. Los cuatro pasos típicos de una selección de este tipo son los siguientes:

1. Se utilizan múltiples modelos $\{M_1,\ldots,M_N\}$ para construir un comité C para la votación, es decir, C=$\{M_1,\ldots,M_N\}$.

2. Los modelos en el comité C se entrenan en el conjunto de datos etiquetados L y obtienen diferentes parámetros.

3. Todos los modelos en el comité realizan predicciones por separado sobre las muestras sin etiquetar de U. Se votan las muestras con la información más relevante.

4. Las muestras que obtienen más desacuerdos se seleccionan como candidatas para ser etiquetadas.

Otros métodos

Distintos investigadores han propuesto métodos de selección de datos candidatos, usualmente combinando distintas métricas. Un ejemplo de esto es el que presentan Li y Guo (2013), donde además de tomar en cuenta la incertidumbre del modelo en la clasificación de la instancia, toma en cuenta la representatividad de esta con respecto a las demás que no han sido etiquetadas, comentan que "la motivación es que las instancias representativas de la distribución de entrada pueden ser muy informativas para mejorar el rendimiento de generalización del clasificador de destino". Esta representatividad se mide utilizando una medida de similitud basada en información mutua dentro de un marco de procesos gaussianos, donde las imágenes son representadas mediante descriptores manuales como GIST y PHOW. La fórmula principal para medir el valor de cada muestra no etiquetada es la siguiente:

$$h_\beta(x_i) = f(x_i)^\beta \, d(x_i)^{1-\beta}$$

dónde x_i representa una muestra no etiquetada, $f(x_i)$ mide la incertidumbre de la muestra y $d(x_i)$ mide la representatividad de esta con respecto al resto

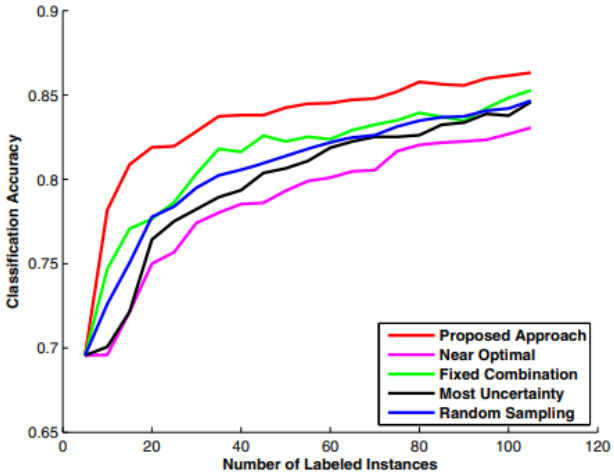


Figura 2.11: Resultados obtenidos comparando distintos métodos de selección para el aprendizaje activo. Imagen basada en: Li y Guo (2013).

del conjunto no etiquetado. Finalmente, $0 \leq \beta \leq 1$ es un parámetro de control de compensación entre los dos términos. Este parámetro β no se fija de antemano, sino que se selecciona de manera adaptativa en cada iteración. Para ello, se definen distintos valores candidatos de β, y para cada uno se elige la instancia que maximiza h_β. Con este conjunto reducido de posibles instancias, se simula el impacto de añadirlas al entrenamiento midiendo el error esperado de clasificación sobre los datos no etiquetados. La instancia que minimiza dicho error es finalmente seleccionada para ser etiquetada, lo que permite ajustar dinámicamente la importancia relativa entre incertidumbre y representatividad a lo largo del proceso de aprendizaje activo.

En la Figura 2.11 podemos ver una comparación entre el método propuesto y el resto de métodos más comunes para un conjunto de datos de imágenes que consta de escenas tanto naturales (costa, bosque, montaña, etc.) como artificiales (cocina, edificio alto, calle, etc.).

2.4.2. Posibles problemas

Para finalizar, Wu y cols. (2022) presentan escenarios donde el aprendizaje activo es perjudicial. En el caso de que la distribución de los datos sea muy dispersa o presente una alta variabilidad y se tenga una baja redundancia, la selección aleatoria podría tener un mejor desempeño para poca cantidad de datos, sucediendo que en un principio el aprendizaje activo lleve a obtener muestras menos representativas, y con ello peores resultados. Esto, una vez se obtienen más datos, deja de ocurrir, y el aprendizaje activo tiene un mejor desempeño para una menor cantidad de datos comparado con la utilización de una selección

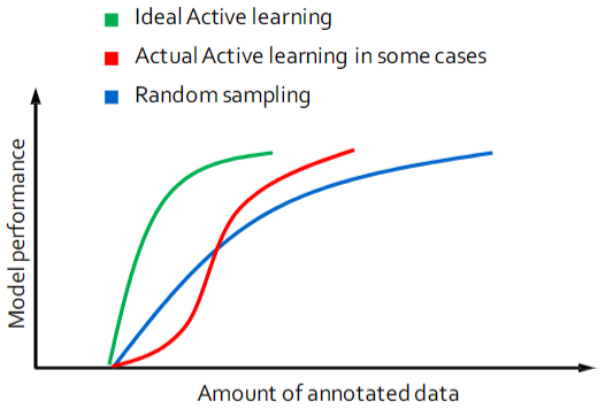


Figura 2.12: Imagen mostrando la problemática del arranque en frío, dónde la selección aleatoria (en azul) supera a la selección utilizando aprendizaje activo (en rojo) en las primeras etapas. Imagen basada en: Wu y cols. (2022).

aleatoria. Se puede ver una representación de este problema en la Figura 2.12, el cual se denomina arranque en frío.

Como solución al arranque en frío se presenta utilizar modelos preentrenados y luego utilizar aprendizaje activo; o la utilización de la selección aleatoria en las primeras iteraciones.

2.5. CLIP

CLIP es un modelo propuesto por *OpenAI* y presentado en Radford y cols. (2021), que es entrenado utilizando un enfoque autosupervisado en el cual se utilizan representaciones visuales a partir de texto natural, siendo el entrenamiento de la siguiente manera: dada una imagen, tiene el objetivo de predecir cuál, de un conjunto de 32.768 fragmentos de texto muestreados aleatoriamente, se emparejó con ella en el conjunto de datos. En el sitio de OpenAI destacan que: "Para resolver esta tarea, intuimos que los modelos CLIP necesitarán aprender a reconocer una amplia variedad de conceptos visuales en imágenes y asociarlos con sus nombres. Como resultado, los modelos CLIP pueden aplicarse a tareas de clasificación visual prácticamente arbitrarias".

2.5.1. Embeddings en CLIP

El aporte central de CLIP es la capacidad de proyectar tanto imágenes como texto en un mismo espacio compartido, mediante el uso de *embeddings*. Un *embedding* es una representación vectorial en alta dimensión que captura relaciones semánticas: imágenes con conceptos visuales similares quedan cercanas entre sí en dicho espacio, y lo mismo ocurre con descripciones textuales.

En el caso de las imágenes, CLIP genera *embeddings* a partir de sus características visuales. Estos *embeddings* permiten comparar instancias mediante métricas de similitud, siendo la más habitual la similitud del coseno, definida como

$$\text{sim}(A, B) = \frac{A \cdot B}{\|A\| \, \|B\|}$$

donde A y B son los *embeddings* de dos imágenes, $A \cdot B$ representa su producto punto y $\|A\|$, $\|B\|$ sus normas euclidianas. El resultado toma valores entre -1 y 1, siendo 1 cuando los vectores son idénticos (máxima similitud) y 0 cuando son ortogonales (sin relación).

2.6. YOLO

La detección de objetos en imágenes ha tenido un desarrollo notable en la última década, y uno de los avances más importantes en este campo ha sido la aparición del modelo YOLO (por sus siglas en inglés, *You Only Look Once*). Este modelo fue presentado originalmente por Redmon, Divvala, Girshick, y Farhadi (2016), y propone una arquitectura de una sola etapa que permite detectar y clasificar objetos en una única pasada de red neuronal convolucional, lo que le otorga una gran ventaja en velocidad frente a otros métodos tradicionales como Faster R-CNN.

Desde su publicación original, YOLO ha tenido múltiples versiones, cada una incorporando mejoras tanto en precisión como en rendimiento. En la revisión realizada por Ali y Zhang (2024) se destacan avances clave introducidos en las versiones más recientes, como el uso de módulos de atención, técnicas de entrenamiento más robustas y arquitecturas optimizadas para detección de objetos pequeños. También se resalta su capacidad para operar en tiempo real y adaptarse a entornos con limitaciones computacionales, lo que lo hace especialmente útil en contextos como la agricultura de precisión. En este campo, YOLO ha sido aplicado para tareas como detección de frutos, identificación de enfermedades en cultivos, monitoreo de malezas y detección de plagas. En particular, ha mostrado buenos resultados cuando se combina con imágenes obtenidas desde UAVs, permitiendo el análisis automatizado de grandes extensiones de terreno. En Ahmad y cols. (2023) mencionan además que YOLO puede trabajar con diferentes tipos de imágenes, incluyendo espectros RGB, térmico y multiespectral, lo que amplía sus aplicaciones a ecosistemas diversos.

2.6.1. Métricas de rendimiento en modelos YOLO

En Ultralytics (2024) se nos presenta que la evaluación del rendimiento de los modelos YOLO se basa en métricas estándar de detección de objetos. Entre las más relevantes se encuentran **precisión**, ***recall***, **mAP@0.5**, **mAP@0.5:0.95** y ***fitness***. Estas métricas permiten analizar distintos aspectos del comporta-

miento del modelo, desde la capacidad de detectar objetos hasta la precisión en su localización.

Precisión

La *precisión* mide la proporción de verdaderos positivos (TP) sobre el total de predicciones positivas realizadas por el modelo, es decir, cuántas de las detecciones fueron correctas:

$$\text{Precisión} = \frac{TP}{TP + FP} \tag{2.1}$$

Una precisión alta indica que el modelo produce pocas detecciones incorrectas (falsos positivos, FP).

Recall

El *recall* mide la proporción de verdaderos positivos sobre el total de objetos reales presentes en la imagen. Refleja la capacidad del modelo para detectar todos los objetos relevantes:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2.2}$$

Un valor alto de recall implica que el modelo detecta la mayoría de los objetos, aunque podría incluir algunas detecciones incorrectas.

mAP@0.5 (Mean Average Precision)

La métrica *mean Average Precision* a un umbral de IoU de 0.5 (mAP@0.5) evalúa la precisión promedio considerando como correcta una predicción si la superposición con la etiqueta anotada (*ground truth*) es mayor o igual al 50 %.

El *IoU* (*Intersection over Union*) se define como:

$$\text{IoU} = \frac{\text{Área de Superposición}}{\text{Área de Unión}} \tag{2.3}$$

El *Average Precision* (AP) de una clase k a un umbral de IoU de 0.5 se define como el área bajo la curva precisión–recall:

$$AP_{0,5}^{(k)} = \int_0^1 P_k(R)\, dR \tag{2.4}$$

donde $P_k(R)$ es la precisión interpolada en función del recall para la clase k. Finalmente, el *mAP@0.5* se obtiene promediando sobre todas las clases:

$$\text{mAP@0.5} = \frac{1}{N_{\text{clases}}} \sum_{k=1}^{N_{\text{clases}}} AP_{0,5}^{(k)} \tag{2.5}$$

Esta métrica proporciona una estimación global del desempeño del modelo, combinando tanto la precisión como la cobertura de las detecciones en todas las clases.

mAP@0.5:0.95

Esta métrica promedia los valores de mAP sobre varios umbrales de IoU, desde 0.5 hasta 0.95 con incrementos de 0.05. Es la métrica estándar utilizada en *datasets* como COCO y evalúa tanto la detección como la calidad de localización:

$$\text{mAP@[0.5:0.95]} = \frac{1}{10}\sum_{i=0}^{9}\text{mAP@IoU}_{0,5+0,05i} \tag{2.6}$$

Fitness

La métrica de *fitness* de YOLO puede variar, siendo una ponderación del resto de métricas obtenidas. Por defecto es una combinación de los resultados de mAP@0.5:0.95 y mAP@0.5 :

$$\text{fitness} = 0{,}1 \cdot \text{mAP}_{0,5} + 0{,}9 \cdot \text{mAP}_{0,5:0,95} \tag{2.7}$$

2.6.2. Variantes de los modelos YOLO

Además de existir versiones de YOLO, que son liberadas con el transcurso del tiempo y tienen mejoras de rendimiento con respecto a sus predecesores, también existen variaciones del modelo en cada versión que permiten su utilización en ciertos escenarios específicos. A continuación se detallan las variaciones que son importante tener en cuenta durante el transcurso de este proyecto.

Variantes por tamaño

Existen cinco variantes de tamaño en los modelos YOLO: n, s, m, l, x. Cada una de estas versiones busca equilibrar de manera distinta la relación entre velocidad de inferencia, consumo de memoria y precisión. El modelo *nano* se caracteriza por ser extremadamente liviano, con alrededor de 4 millones de parámetros, lo que le permite realizar inferencias en tiempo real incluso en dispositivos con recursos limitados, como CPUs o sistemas embebidos. Por su parte, el modelo *small* (s) ofrece un mejor equilibrio entre velocidad y precisión, mientras que las variantes *medium* (m), *large* (l) y *xlarge* (x) incrementan gradualmente la capacidad del modelo, alcanzando mayores valores de precisión (mAP) a costa de un mayor consumo de memoria y tiempo de entrenamiento. Estas variantes permiten adaptar la arquitectura YOLO a distintos escenarios de aplicación. Por ejemplo, los modelos n y s son adecuados para sistemas en tiempo real o dispositivos móviles, mientras que las versiones l y x son más apropiadas para entrenamientos en entornos con GPUs de alto rendimiento o cuando se busca la máxima precisión posible.

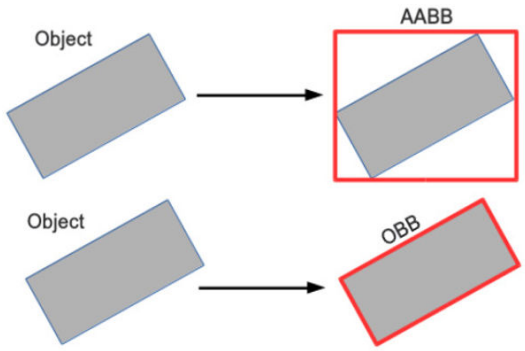


Figura 2.13: Funcionamiento de las Oriented bounding boxes. Imagen recuperada de Bhandari (2025). Arriba se ve un objeto rotado que al ser delimitado con una caja tradicional necesita ser mucho más grande, y porciones que no son el objeto quedan dentro de la caja. Abajo se ve como una Oriented bounding boxes resuelve este problema.

Variantes orientadas

Además de las diferencias en tamaño, las versiones más recientes de YOLO, particularmente a partir de YOLOv8 incorporan variantes adaptadas a tareas específicas, como la detección de Oriented bounding boxes. A diferencia de las cajas delimitadoras tradicionales, que son rectángulos alineados con los ejes de la imagen, las Oriented bounding boxes permiten representar objetos rotados mediante un ángulo de orientación. Esta característica es especialmente útil en contextos donde los objetos no se encuentran alineados horizontalmente. En la Figura 2.13 se puede ver un ejemplo de como funcionan las Oriented bounding boxes y el problema que resuelven.

2.7. Oso hormiguero artificial

Dentro del marco de los proyectos realizados por la Facultad de Ingeniería de la Universidad de la República sobre la problemática del control de hormigas, del cual el proyecto presentado en este informe es parte, surgen antecedentes que fueron desarrollados en paralelo, pero finalizaron anteriormente, por lo que alguno de los resultados obtenidos de estos proyectos fueron insumos para este. En particular, se presentarán dos de estos, uno destinado a la creación de un robot que pueda desplazarse por entornos desafiantes y otro más relevante a este proyecto que busca la detección de caminos de hormigas activos.

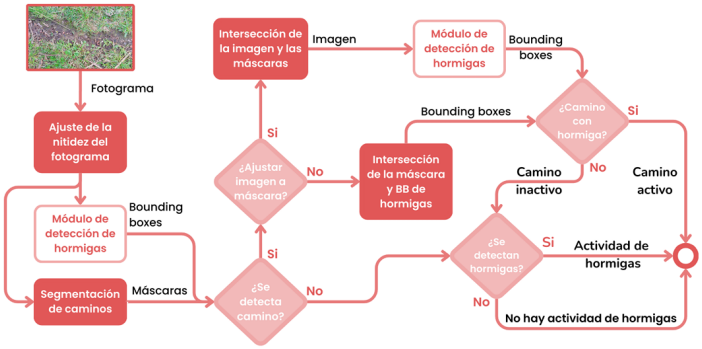


Figura 2.14: Pipeline ARP. Imagen recuperada de: Nadile y cols. (2025).

2.7.1. Active Road Pipeline (ARP): *Pipeline* de detección de caminos activos

En su trabajo, Nadile, Marr, y Perera (2025) presentan un *Pipeline* para la detección de caminos de hormigas activos. En la Figura 2.14 se puede ver una representación del *pipeline* presentado. Básicamente la detección del camino de hormigas se basa en dos etapas principales, una inicial donde se busca segmentar los caminos de hormigas para posteriormente, basado en esa segmentación, buscar la presencia de estas para considerarlo un camino activo. Durante el *pipeline* se presentan pasos para la mejora de la detección, como son ajustes en la nitidez de las imágenes o la utilización de SAHI, un *framework* desarrollado para la detección de objetos pequeños que se basa en dividir la imagen en secciones lo que se traduce en que el objeto sea más grande en dichas secciones. Además, durante el proyecto se realizaron experimentaciones sobre la utilización de diferentes arquitecturas para la detección y segmentación, como lo son YOLO, Mask-R-CNN y Faster-RCNN. Por otro lado, para la realización de pruebas se construyó un conjunto de datos de caminos de hormigas localizados en diferentes lugares de Montevideo. Para el entrenamiento de los modelos, se realizó un etiquetado iterativo, donde utilizando la funcionalidad *Nuclio* de CVAT, a medida que se realizaba el etiquetado de caminos de hormigas se entrenaba un modelo de segmentación que era utilizado para ayudar a la anotación de las imágenes restantes, realizando una auto anotación que facilitaba el trabajo, aunque era necesaria la supervisión humana. De los resultados obtenidos, lo más destacado es la creación de un conjunto de datos, hasta ahora inexistente, de caminos de hormigas cortadoras de hojas. Por otro lado, se probó un mejor desempeño de YOLO sobre arquitecturas como Mask-RNN y Faster-CRNN para la segmentación y detección de caminos y hormigas. Pese a que se obtienen buenos resultados, los autores plantean la importancia de incrementar y diversificar los conjuntos de datos para mejorar el rendimiento del sistema.

2.7.2. Desarrollo de un oso hormiguero artificial

En su proyecto Berois, De Oliveira, y Gastelú (2024) describen el desarrollo de un robot hexápodo, un subsistema de percepción basado en redes de detección y un módulo heurístico de toma de decisiones para seguir caminos de hormigas en contextos agrícolas. El objetivo declarado fue diseñar una solución práctica para detectar y seguir rutas de hormigas con el fin de explorar alternativas al uso indiscriminado de pesticidas.

El pipeline de trabajo propuesto por los autores puede resumirse en tres etapas: captura de imagen desde la cámara montada en el robot, detección de hormigas mediante un modelo de visión (YOLOv8) y decisión de movimiento mediante una heurística que interpreta las detecciones como indicios de un camino activo. Para las pruebas se utilizaron experimentos en simulador (Unreal Engine 5) donde se modelaron errores físicos y de percepción observados en campo, lo que permitió evaluar el comportamiento integrado bajo condiciones adversas.

Para el entrenamiento del modelo se construyó un conjunto de datos mixto de imágenes reales y un *dataset* sintético generado desde el simulador en Unreal. El *dataset* combinado reportado consta de aproximadamente 21.846 imágenes (17.534 para entrenamiento y 4.312 para validación), además de 5.107 imágenes sintéticas. El uso de datos sintéticos tuvo un doble propósito: aumentar la variedad de fondos e iluminaciones y permitir pruebas controladas del sistema integrado. En cuanto a la implementación del seguimiento de los caminos, los autores integraron el detector con una política sencilla y determinista: orientar el robot hacia la concentración de detecciones de hormigas.

En cuanto a la detección, el modelo logra identificar en promedio alrededor del 40 % de las hormigas presentes en un camino real estándar. En videos con 10–12 hormigas, el detector recupera típicamente entre 2 y 5 detecciones por fotograma. El desempeño resulta sensible al contraste y al tamaño aparente de las hormigas en la imagen: funciona muy bien en superficies de alto contraste (por ejemplo, baldosas claras), pero disminuye notablemente en pasto o tierra. La incorporación de un *dataset* sintético mejoró la capacidad del modelo en el entorno virtual y permitió explorar condiciones límite de iluminación y contraste.

Respecto a la movilidad y estabilidad, el robot pasó por varias iteraciones, entre ellas el cambio de geometría de patas, aumento del área de apoyo e impresión 3D de una nueva base, que incrementaron la estabilidad en la mayoría de superficies. Sin embargo, persisten limitaciones en pendientes ascendentes y en pasto largo.

Finalmente, en relación con la integración y simulación, las pruebas realizadas en Unreal Engine mostraron que una heurística simple resulta suficiente en muchos escenarios representativos para seguir caminos.

Capítulo 3

Problema y diseño de la solución

Dada la extensión del problema del control de hormigas, en esta sección se presentará el problema en el que se hace foco a resolver durante este proyecto, además de la solución propuesta.

Del capítulo anterior se desprende que la mejor forma de control de hormigas a día de hoy es la utilización de cebos tóxicos (ver Sección 2.2.2), esta es la más efectiva y más utilizada por los agricultores. El problema de esta técnica es que es utilizada de forma desmedida, se riegan los campos con los cebos, sin optimizar su uso, muchos de ellos no terminan en los hormigueros. Este uso desmedido genera problemas ambientales y mayores costos para los productores rurales. Es por esto que en esta solución se busca la presencia de hormigas, el objetivo es que cada cebo sea colocado con sentido, en un lugar estratégico.

La presencia de las hormigas tiene un claro indicio, los caminos que estas dejan en sus rutas de forrajeo. Dado que los caminos son el elemento más abundante, se decide que este será el objetivo de la detección para el proyecto. En caso de detectar un camino de hormigas, se toma como un buen lugar para la colocación de un cebo.

Una vez definido que se quiere buscar caminos de hormigas, surge una problemática, como se mostró en el capítulo anterior, la mayoría de trabajos anteriores se basan en la detección del propio hormiguero, lo que reduce considerablemente la cantidad de datos disponibles para el entrenamiento de un modelo de detección. La recolección y procesamiento de datos, es de las mayores cargas de trabajo humano al implementar un modelo de detección. Por ejemplo, la implementación de *Ultralytics* de YOLO, en su documentación oficial (Ultralytics, 2025), recomienda tener más de 10.000 instancias de cada clase para un buen modelo de detección. Esto implica una gran cantidad de horas de obtención de imágenes y su posterior anotación. Considerando esto, el proyecto se enfocará en atacar esta problemática, creando un sistema que permita una obtención de datos eficiente, permitiendo modelos que mejoren iterativamente, utilizando

Aprendizaje Activo.

3.1. Descripción de la solución

3.1.1. Conceptos claves

A continuación se presentan los conceptos que resultarán importantes para comprender el resto del documento.

Cuando se hace referencia a agente, se refiere a un vehículo autónomo capaz de tomar imágenes de su alrededor. A su vez, es necesario que este agente tenga la capacidad de comunicar las imágenes al sistema, ya sea de forma sincrónica o de almacenarla y enviarlas posteriormente. Cabe aclarar que siempre que se hable del sistema se refiere a la solución propuesta. Para tener una visión más clara, el agente sería lo que es el *Cyberdog* de *Xiaomi* en el trabajo explicado en la sección 2.3.2. Por otro lado, cuando se hace referencia a un usuario es una persona. Existen dos tipos de usuarios, un usuario anotador, el cual tiene la capacidad de etiquetar imágenes para su posterior uso en nuevas versiones de los modelos. Estos tendrán acceso a la herramienta CVAT y únicamente tienen que ser capaz de saber como utilizar esta herramienta y de reconocer los caminos de hormigas. El otro tipo de usuario es el usuario administrador, este podrá visualizar las métricas de rendimiento de los diferentes modelos y seleccionar la versión en producción en cada momento, así como poder modificar hiperparámetros del entrenamiento. Es importante que este usuario tenga conocimientos sobre modelos de aprendizaje automático para tomar las decisiones más acertadas. Cabe destacar que los dos tipos de usuarios no colisionan entre sí, y una misma persona puede cumplir tanto el rol de anotador como el de administrador. Por otro lado, cuando se habla de iteración, se hace referencia a un ciclo del sistema que es iniciado por el usuario administrador, obtiene las mejores imágenes según los parámetros seleccionados y da como resultado una nueva versión del modelo con sus métricas asociadas.

3.1.2. Solución planteada

Se propone crear un sistema que implemente distintos métodos de aprendizaje activo, en dónde un usuario administrador pueda cargar un modelo de aprendizaje automático, e iterativamente, mediante la carga de imágenes tomadas por un agente que utilice el modelo y el uso de los métodos de selección de candidatos implementados, pueda reentrenar el modelo y tenga la posibilidad de lanzar versiones mejoradas de este. Además, se integrará el sistema con la plataforma CVAT para ser utilizada en el etiquetado de las imágenes.

El flujo general del sistema propuesto es el que se puede ver en la Figura 3.1. Existen tres componentes principales en este flujo. Estos son el agente externo ya mencionado, el sistema de aprendizaje activo desarrollado y la plataforma CVAT. Las interacciones entre los componentes y el usuario se pueden ver en la Figura 3.2.

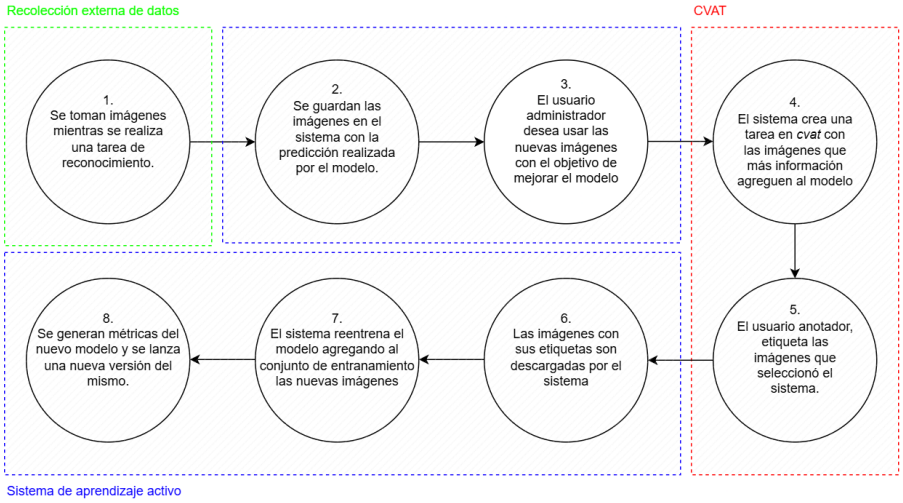


Figura 3.1: Flujo de la solución.

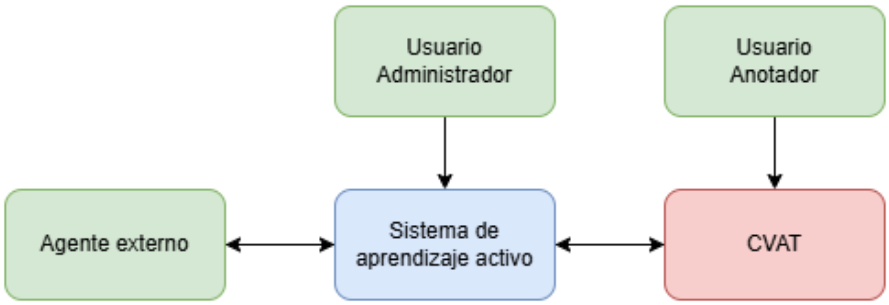


Figura 3.2: Interacciones del sistema.

El proceso tiene una etapa inicial y luego puede tener cualquier cantidad de etapas iterativas como sea convenido por el usuario. Estas son la carga inicial del modelo con el que se va a trabajar y las iteraciones de selección, etiquetado y re-entrenado de este modelo. A continuación se describen en detalle estas etapas.

En la etapa inicial se carga en el sistema el modelo con el cual se va a trabajar. Además, se almacena el conjunto de entrenamiento utilizado en la creación de dicho modelo. Luego, el sistema crea un conjunto de datos no etiquetados en donde almacenará las imágenes candidatas a utilizar en el reforzamiento del modelo. Este conjunto puede construirse de manera asíncrona al flujo general. El origen de las imágenes puede ser variado. Si bien, como se comentó antes, las imágenes más apropiadas serían aquellas que el agente encuentre en la práctica, esto no quiere decir que no se puedan utilizar imágenes de otras fuentes.

Las etapas iterativas serán iniciadas por el usuario administrador. Cada una de estas funcionará de la siguiente manera: En primer lugar, el sistema tomará basado en un método de selección de candidatos, una cantidad parametrizable de imágenes desde el conjunto de imágenes no etiquetadas. Una vez obtenidas, se creará en la plataforma CVAT una tarea de etiquetado. Esta tarea contendrá las imágenes seleccionadas por el sistema, las cuales deberán ser etiquetadas por el usuario anotador. Una vez finalizada esta tarea, el sistema obtiene las imágenes etiquetadas, en su mayoría serán estas las que serán utilizadas por el sistema para el nuevo entrenamiento del modelo. Una fracción de estas imágenes será utilizada para aumentar el conjunto de validación, esta fracción es parametrizable. El reentrenamiento del modelo se realizará utilizando los pesos de la última época del modelo anterior como pesos iniciales y constará de las imágenes seleccionadas en la iteración actual, así como también de todas las imágenes de las iteraciones anteriores. La necesidad de utilizar las imágenes pasadas radica en que durante pruebas preliminares se detectó que al realizar los reentrenamientos el modelo olvidaba lo aprendido en las iteraciones anteriores y solo obtenía buenos resultados en imágenes similares a las de la iteración actual. Este fenómeno es llamado *Catastrophic forgetting* y Rolnick, Ahuja, Schwarz, Lillicrap, y Wayne (2019) proponen la reutilización de datos pasados para solucionarlo. Otra opción considerada fue la de congelar las capas iniciales de la red neuronal, pero esta se descartó porque podía afectar la capacidad de generalización del modelo.

Luego de reentrenado el modelo, el sistema lo evaluará, generando métricas que el usuario administrador podrá utilizar para comprobar la progresión de este. No solo se evalúa al modelo nuevo, sino también al modelo de la iteración anterior. Ambos contra el mismo conjunto de validación aumentado, conformado por el conjunto de validación previo más la porción de nuevos datos etiquetados elegidos para validación. La necesidad de revaluar el modelo de la iteración previa radica en que si se utilizaran las métricas ya obtenidas en la iteración anterior (contra el conjunto de validación previo) las comparaciones entre ambos serían utilizando métricas obtenidas contra conjuntos de datos distintos. Dependiendo de la composición original de los datos con los que se entrenó y validó el modelo inicial y los de las etapas subsecuentes, pueden surgir problemas como que el modelo esté sobreajustado (sobre todo en etapas iniciales) contra el conjunto de validación original. En las diferentes iteraciones el objetivo será que el

conjunto de datos se asemeje más a la realidad práctica con la que se encuentra el agente. En la Figura 3.3 se tiene una ayuda visual de como crece el conjunto de validación con cada iteración con el fin de diversificarlo con nuevos ejemplos encontrados en la realidad. Esto ayuda a entender el porqué una evaluación como la mostrada en la Figura 3.4 es injusta con las iteraciones posteriores del modelo si no se vuelve a evaluar el modelo anterior con el nuevo conjunto de validación. De esto, también se desprende la razón de aumentar el conjunto de validación, si el conjunto inicial está sesgado a cierto tipo de imágenes que le llamaremos tipo de imagen A, que comparten características muy similares podría perjudicar a nuevas versiones del modelo que sean más generales si se mantiene este conjunto de validación. Por ejemplo, en el caso de estudio podrían ser caminos de hormigas sobre césped muy verde donde es fácil reconocer el camino. Tanto el entrenamiento como la validación serán realizadas con imágenes A, lo que llevará a que el modelo dé buenas métricas de rendimiento. En cambio, si en una nueva iteración es entrenado con datos más complejos y plurales, como podría ser suelos más áridos donde se dificulta la visualización del camino, probablemente si en esa iteración no se agregaron imágenes de tipo A, la evaluación sobre el conjunto de validación inicial dará peores o al menos iguales resultados que el modelo inicial. Pero tendrá un mejor desempeño en la realidad, ya que detectará mejor este tipo de caminos en entornos áridos.
Finalmente, en caso de que el modelo sea aprobado por el usuario administrador, se utilizará la versión nueva del modelo en el sistema externo para futuras tareas de clasificación, y la siguiente iteración del proceso se realizará sobre esta nueva versión.
Se proponen dos formas de uso del sistema, una *offline* y otra *online*. En la forma *online* el agente no necesita tener una versión del modelo localmente, este se comunica con el sistema enviando las imágenes de su entorno y el sistema le comunica la predicción obtenida. En este caso, el agente no debe realizar ninguna tarea extra debido a que el sistema almacenará automáticamente las imágenes que se le envían para predecir. Es importante notar el contexto en el que trabajaría el agente, muchas veces la ubicación del agente será un descampado con baja o nula conectividad, por lo cual la comunicación con el sistema sería imposible o cuanto menos ineficiente. Es por esto que se propone una alternativa *offline*, en donde el agente obtendrá la última versión del modelo previamente y con este realizará las predicciones de las imágenes de su entorno de forma autónoma. En este caso, es importante que el agente almacene las imágenes recolectadas y posteriormente, una vez que disponga mejor conexión, las comunique al sistema, para que este las pueda utilizar en subsecuentes iteraciones para la mejora continua del modelo.

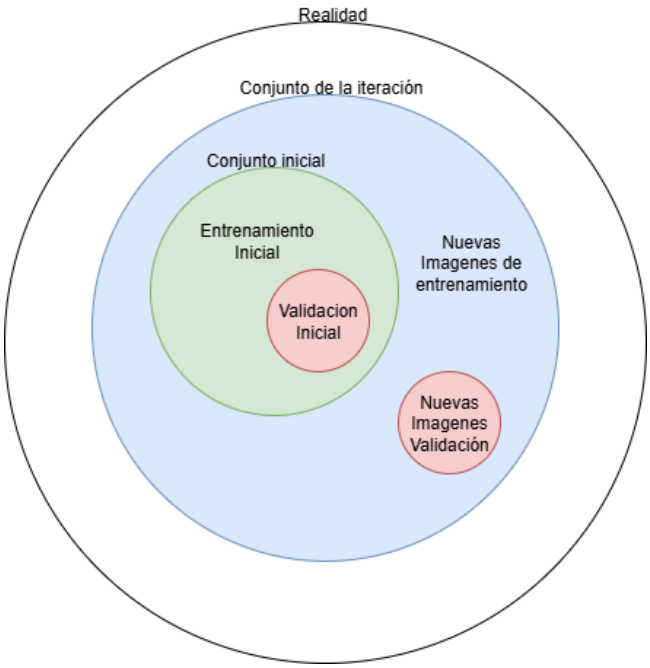


Figura 3.3: Crecimiento del conjunto de validación en cada iteración. Con una nueva iteración, ademas de agregar imágenes para entrenamiento (en azul), se utiliza una porción (en rojo) para alimentar el conjunto de validación, en busca de que cada vez sea una representación más fiel de la realidad.

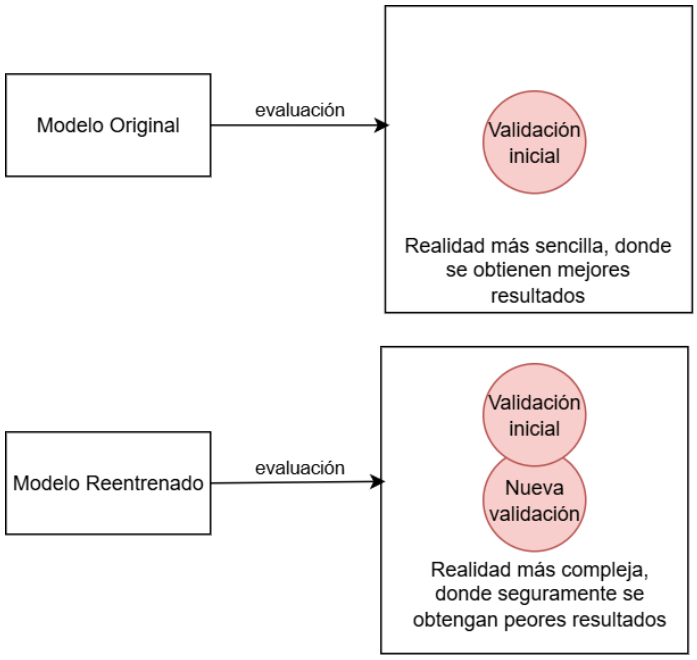


Figura 3.4: Ejemplo de comparación injusta entre modelos. Al aumentar el conjunto de validación es necesario volver a evaluar los modelos anteriores sobre este nuevo conjunto de validación para que la comparación de los modelos sea justa.

3.2. Técnicas de selección de candidatos

El objetivo de los métodos de selección implementados dentro del sistema es el de optimizar la selección de un conjunto manejable[1] de imágenes que serán luego presentadas a un usuario anotador (el usuario administrador puede seleccionar cuál algoritmo utilizar en cada iteración). A continuación se explicarán los métodos de selección implementados y sobre los cuales se realizarán los distintos experimentos.

3.2.1. Método basado en confianza

En primer lugar, el método más básico con el que se cuenta es el basado en confianza. El funcionamiento de este es el siguiente: se toman todas las imágenes sin etiquetar, predichas por el modelo, y se ordenan priorizando, en primer lugar, aquellas que tienen predicciones y, dentro de ese subconjunto, aquellas cuya predicción presenta menor nivel de confianza. Se muestra en el Algoritmo 1 el pseudocódigo del método de selección por confianza.

Algorithm 1 Selección basada en confianza

Require: Conjunto de imágenes no etiquetadas U, número de imágenes a seleccionar k

Ordenar U priorizando:
    1. Imágenes con predicciones
    2. $f(x) \leftarrow 1 - \text{confianza}(x)$ *(incertidumbre de x)*
Seleccionar las k primeras imágenes
return Conjunto de imágenes seleccionadas

3.2.2. Método basado en confianza y similitud

El segundo método que se propone está inspirado en el presentado en la Sección Otros métodos dónde se utiliza como método de selección un algoritmo basado en métricas de confianza y representatividad de las imágenes candidatas con respecto a las demás no etiquetadas. En este proyecto se propone un método que adapta esta idea al escenario de clasificación, pero reemplazando la medida de representatividad original por una basada en el modelo CLIP, descrito en la Sección 2.5. De esta manera, CLIP se utiliza como mecanismo para obtener *embeddings* de las imágenes, lo que permite medir la similitud entre estas imágenes mediante la distancia entre estos *embeddings*. En este caso se utilizará la similitud del coseno entre el *embedding* de la imagen y el promedio de todos los restantes del conjunto de datos con el que se quiera comparar. La principal diferencia con el método original es que, en este caso, no se ajusta dinámicamente el parámetro de ponderación entre confianza y representatividad, sino

[1] Manejable en el sentido de que un anotador pueda realizar la tarea de etiquetado en un tiempo razonable.

que se emplea un valor fijo de β entre 0 y 1. Se muestra en el Algoritmo 2 el pseudocódigo del método de selección por confianza y similitud.

Algorithm 2 Selección basada en confianza y similitud

Require: Conjunto de imágenes no etiquetadas U, número de imágenes a seleccionar k, parámetro β

for cada imagen $x \in U$ **do**
  if x no tiene predicciones **then**
    $f(x) \leftarrow 0$ *(sin predicciones, incertidumbre nula)*
  else
    $f(x) \leftarrow 1 - \text{confianza}(x)$ *(incertidumbre de x)*
  end if
  $e(x) \leftarrow \text{CLIP}(x)$ *(obtener embedding de la imagen con CLIP)*
  $\mu(U \setminus \{x\}) \leftarrow$ promedio de los embeddings del resto del conjunto
  $s(x) \leftarrow \text{dist\_cos}(e(x), \mu(U \setminus \{x\}))$ *(similitud respecto al conjunto U)*
end for
$f_{norm}(x) \leftarrow$ Normalización *min–max* de $f(x)$ sobre todos los valores de f calculados en U
$s_{norm}(x) \leftarrow$ Normalización *min–max* de $s(x)$ sobre todos los valores s calculados en U
for cada imagen $x \in U$ **do**
  Calcular score $scr_\beta(x) = \beta \cdot f_{norm}(x) + (1 - \beta) \cdot s_{norm}(x)$
end for
Ordenar imágenes por $scr_\beta(x)$ en orden descendente
Seleccionar las k primeras imágenes
return Conjunto de imágenes seleccionadas

3.2.3. Método basado en confianza y disimilitud

El siguiente método propuesto es análogo al anterior, con la diferencia de que esta vez se toma la **disimilitud** al conjunto de imágenes etiquetadas, al contrario del anterior donde se tomaba la **similitud** a las **no** etiquetadas. Se muestra en el Algoritmo 3 el pseudocódigo del método de selección por confianza y disimilitud.

Algorithm 3 Selección basada en confianza y disimilitud

Require: Conjunto de imágenes no etiquetadas U, conjunto de imágenes etiquetadas L, número de imágenes a seleccionar k, parámetro β

for cada imagen $x \in U$ **do**
  if x no tiene predicciones **then**
    $f(x) \leftarrow 0$ *(sin predicciones, incertidumbre nula)*
  else
    $f(x) \leftarrow 1 - \text{confianza}(x)$ *(incertidumbre de x)*
  end if
  $e(x) \leftarrow \text{CLIP}(x)$ *(obtener embedding de la imagen con CLIP)*
  $\mu(L \setminus \{x\}) \leftarrow$ promedio de los embeddings del conjunto de imágenes etiquetadas
  $d(x) \leftarrow 1 - \text{dist\_cos}(e(x), \mu(L \setminus \{x\}))$ *(disimilitud respecto al conjunto L)*
end for
$f_{norm}(x) \leftarrow$ Normalización *min–max* de $f(x)$ sobre todos los valores de f calculados en U
$d_{norm}(x) \leftarrow$ Normalización *min–max* de $d(x)$ sobre todos los valores d calculados en U
for cada imagen $x \in U$ **do**
  Calcular score $scr_\beta(x) = \beta \cdot f_{norm}(x) + (1 - \beta) \cdot d_{norm}(x)$
end for
Ordenar imágenes por $scr_\beta(x)$ en orden descendente
Seleccionar las k primeras imágenes
return Conjunto de imágenes seleccionadas

3.2.4. Método basado en confianza y *backgrounds* con similitud

El último método propuesto pretende sumar a los métodos de selección anteriores un cierto porcentaje de imágenes (*porcentaje_backgrounds*) que sean *backgrounds*. Es decir, imágenes que no tienen presente al objeto que se intenta detectar (en la realidad presentada, sería una imagen del suelo sin caminos de hormigas). En Ultralytics (2024) se recomienda que alrededor del 10 % de las imágenes de entrenamiento sean *backgrounds* ya que dan ejemplos al modelo de lo que no debe etiquetar como un camino de hormigas, esto ayuda a mejorar su precisión. Como están construidos los métodos de selección anteriores estas imágenes sin predicciones tienen pocas probabilidades de ser consideradas, a no ser que fueran un falso positivo o muy similares/disimilares al resto. Por esto es importante dar la posibilidad de forzar su inclusión en las iteraciones. El método funciona de la siguiente manera: El (100 -*porcentaje_backgrounds*) % de las imágenes se selecciona de la misma manera que en el método basado en confianza. El *porcentaje_backgrounds* restante se elige siguiendo el siguiente criterio: primero se consideran todas las imágenes no etiquetadas predichas por el modelo, de ellas, se toman únicamente aquellas sin predicciones y, dentro de ese subconjunto, se priorizan según su similitud con el conjunto completo de

imágenes no etiquetadas. De no ser suficientes, se agregan las restantes de la selección basada en confianza inicial hasta completar el total. Se muestra en el Algoritmo 4 el pseudocódigo del método de selección por confianza y similitud con *backgrounds*.

Algorithm 4 Selección basada en confianza y backgrounds con similitud

Require: Conjunto de imágenes no etiquetadas U, número de imágenes a seleccionar k, porcentaje de backgrounds p, parámetro β

$num\_backgrounds \leftarrow \lfloor p \cdot k \rfloor$
$num\_non\_backgrounds \leftarrow k - num\_backgrounds$
Seleccionar $num\_non\_backgrounds$ imágenes utilizando el método basado en confianza $\rightarrow S$
$R \leftarrow$ imágenes restantes de la selección inicial que no están en S
$B \leftarrow$ imágenes en U que no tienen predicciones y no están en S
for cada imagen $x \in B$ **do**
    $e(x) \leftarrow \text{CLIP}(x)$ *(obtener embedding de la imagen con CLIP)*
    $\mu(U \setminus \{x\}) \leftarrow$ promedio de los embeddings del resto del conjunto
    $s(x) \leftarrow \text{dist\_cos}(e(x), \mu(U \setminus \{x\}))$ *(similitud respecto al conjunto U)*
end for
$s_{norm}(x) \leftarrow$ Normalización *min–max* de $s(x)$ sobre todos los valores s calculados en U
Ordenar B de forma descendente en cuanto a $s_{norm}(x)$
Agregar las primeras $num\_backgrounds$ imágenes de B a S
if tamaño de $S < k$ **then**
    Completar S con las primeras imágenes de R hasta llegar a k
end if
return Conjunto S de k imágenes seleccionadas

3.3. Arquitectura de la solución

Para la solución se plantea una arquitectura en capas, conteniendo una capa de datos y una capa lógica. En esta sección se presentarán estas capas, junto a sus especificaciones y las decisiones importantes que fueron tomadas en su diseño y desarrollo.

3.3.1. Capa de datos

En cuanto a la capa de datos, hay un punto muy importante a tener en cuenta: se maneja una gran cantidad de información multimedia que puede llegar a deteriorar fuertemente el rendimiento del sistema si no es manejada eficientemente. Tomando en cuenta esto, se decide realizar la siguiente división: por un lado, en una base de datos se almacenan la información relacionada con los archivos, por ejemplo, información de la predicción realizada por el modelo, o si fue utilizada en alguna iteración del modelo. Y, por otro lado, el

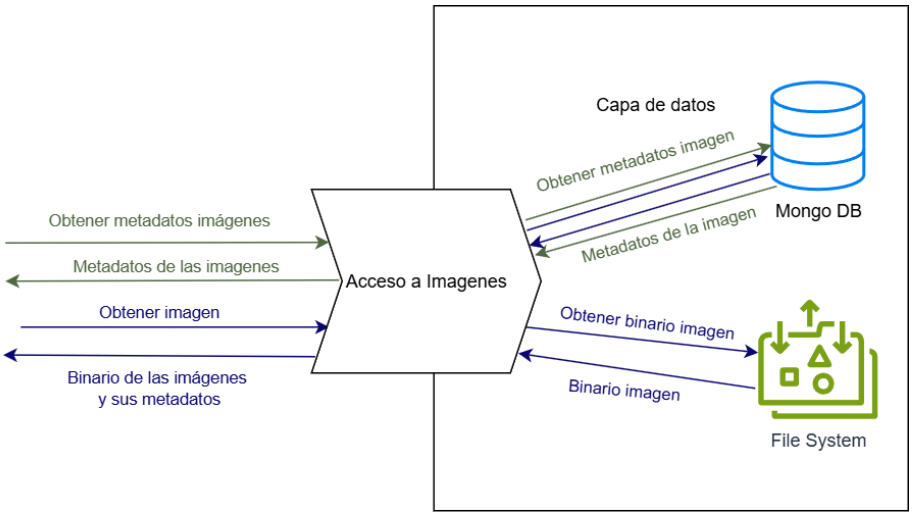


Figura 3.5: Representanción del proceso de obtención de imágenes con las interacciones entre la base datos documental y el *file system*, orquestradas por el componente de Acceso a Imágenes.

archivo es almacenado a nivel del *file system*. La responsabilidad de la capa de datos es abstraer el hecho de que el archivo y su información están separados, transparentando esto a las capas superiores. En caso de necesitar un archivo junto con sus metadatos, únicamente tiene que solicitarlo sin necesidad de estar controlando en donde se almacena cada uno. La segunda decisión que se tomó con el fin de mejorar este aspecto es realizar una carga perezosa de los archivos multimedia: si la capa superior necesita de la información del archivo binario perteneciente a la imagen, debe solicitarlo a demanda. De esta forma se limita el uso de memoria RAM utilizada. Es responsabilidad de la capa lógica en caso de trabajar con múltiples imágenes realizar un procesamiento de a lotes, que vaya cargando las imágenes del lote a demanda y al finalizar libere la memoria utilizada. En la Figura 3.5 se tiene un ejemplo gráfico de este comportamiento.

Para la base de datos se decidió utilizar una base documental, se tomó esta decisión basada en que los metadatos que se pueden generar para los archivos pueden variar mucho según lo que se busque medir. Por ejemplo, sería perfectamente válido que las imágenes tomadas por el agente tengan información extra como: luminosidad, temperatura, fecha, etc. Pero si se trabaja con más de un agente, quizás no todos tengan capacidad para determinar estas características, esto genera datos variables. Estos esquemas variables son de los principales motivos por los cual utilizar una base de datos no relacional. El punto anterior, sumado a la ausencia de relaciones estrictas, da lugar a la utilización de una base no relacional documental.

Al comparar las implementaciones disponibles de bases documentales se consideró el escenario de mayor estrés de la base de datos, esto ocurre al momento

de obtener los metadatos de las imágenes según las características deseadas por el usuario. En este punto, la carga de trabajo se basa en una lectura intensiva. En el reporte técnico presentado por United Software Associates (2015), se realizan pruebas considerando 95 % operaciones de lecturas y 5 % de escrituras, lo que es una representación bastante cercana a lo esperado en estos momentos de estrés. En configuraciones con durabilidad máxima, donde se garantiza que no existe pérdida de datos, MongoDB demuestra ventajas significativas: alcanza dos veces mayor rendimiento que Cassandra y seis veces mayor que Couchbase. Adicionalmente, MongoDB mantiene latencias considerablemente menores, con 1ms para lecturas y 10ms para escrituras. Comparado con los 15ms de lectura y 66ms de escritura de Cassandra, y los extremos 238ms de escritura de Couchbase. Es por esto que fue seleccionada como la base de datos para el proyecto.

3.3.2. Capa Lógica

La lógica del sistema está implementada en *Python*, esto es debido al gran soporte que tiene este lenguaje para las aplicaciones de aprendizaje automático, siendo el estado del arte en esta temática. En particular, para el entrenamiento del modelo y sus predicciones, se decidió utilizar la implementación de YOLO de *Ultralytics*. Esta decisión se basa en la facilidad que tiene dicha implementación para realizar entrenamientos sencillos en las primeras etapas del proyecto. Pero a su vez, permite un alto nivel de personalización en los futuros entrenamientos, punto muy necesario para la experimentación y análisis de resultados.

Para la implementación de la API se utilizó el *framework Flask*, esta decisión se basa en que con la ausencia de una lógica de negocio pesada, se buscaba un *framework* muy liviano y sencillo de utilizar. El entrenamiento de los modelos utiliza una gran cantidad de recursos, la RAM suele ser un cuello de botella importante y en el caso de que el usuario requiera que el sistema realice el entrenamiento utilizando la CPU, esto se ve empeorado, ya que la información almacenada por la VRAM en, caso de usar una GPU, pasa a ser responsabilidad de la RAM. Es por esto, que se decide utilizar un *framework* liviano, que no acapare demasiados recursos que podría estar usando el módulo de entrenamiento. En el siguiente repositorio de *GitHub* (Siimp, 2021) se puede observar que al realizar pruebas de rendimientos, más precisamente las de consumo de RAM, *Flask* queda muy bien posicionado, haciendo a este un buen candidato para su utilización.

3.3.3. Diagrama de componentes

En la Figura 3.6 se puede ver un diagrama de componentes del sistema, donde se presenta la estructura de la capa lógica de la solución e interacciones con la capa de datos y servicios externos.

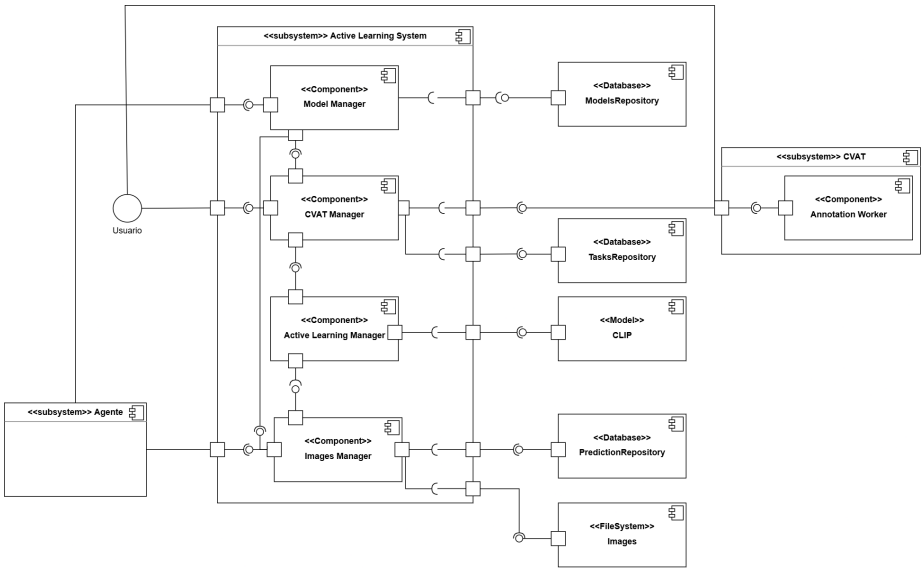


Figura 3.6: Diagrama de componentes del sistema.

Los componentes principales son:

- **Usuario**: representa al usuario encargado de iniciar el sistema y lanzar las iteraciones.
- **Agente**: es el agente externo que recopila las imágenes y puede hacer tanto llamados al sistema para almacenarlas para futuras mejoras del modelo como también para pedir predicciones sobre esas imágenes en tiempo real.
- **Active Learning System**: representa el subsistema central que contiene los módulos:
  - **Model Manager**: encargado de gestionar los modelos entrenados y su almacenamiento en el *ModelsRepository*.
  - **CVAT Manager**: gestiona la comunicación con la herramienta de anotación CVAT y sus tareas (almacenadas en el *TasksRepository*.)
  - **Active Learning Manager**: componente clave que implementa las distintas técnicas de selección de candidatos.
  - **Images Manager**: administra las imágenes, su recuperación desde el *file system* y la información asociada en cuanto a sus predicciones y otros metadatos en el *PredictionRepository*.
- **Bases de datos y almacenamiento**: el sistema interactúa con tres repositorios:
  - **ModelsRepository**: almacena información de los modelos entrenados.

    - **TasksRepository**: contiene las tasks de CVAT y su estado.
    - **PredictionRepository**: guarda las predicciones realizadas sobre las imágenes y otros metadatos de ellas.
    - **File System**: se utiliza para almacenar los archivos binarios de las imágenes y para separar los conjuntos de datos de entrenamiento para cada modelo, así como también los de validación y test (generales para todos los modelos.)
- **Modelo CLIP**: utilizado para generar los *embeddings de las imágenes*.
- **CVAT**: subsistema externo utilizado para la anotación manual de imágenes.

Este diseño permite una clara separación de responsabilidades y facilita la extensibilidad del sistema, por ejemplo, nuevas técnicas de selección de candidatos pueden incorporarse en el *Active Learning Manager*, sin afectar el resto de la arquitectura.

3.3.4. Despliegue de la solución

Con el fin de simplificar la instalación del sistema, se tomó la decisión de trabajar sobre contenedores *Docker*. Como se mencionó anteriormente se utiliza la aplicación CVAT, y a pesar de que esta tiene una alternativa *Cloud*, también cuenta con la posibilidad de realizar un despliegue propio. Se decidió ir por esta segunda opción para mantener la información en sistemas propios. A su vez, aunque no es necesario, en este caso comparte *host* con la aplicación desarrollada, esta decisión se tomó para una comunicación más eficiente de los archivos multimedia. Por otro lado, la aplicación desarrollada junto con la base *MongoDB* están desplegados en otro contenedor. La imagen de la aplicación utiliza *Ubuntu* 22.04 como sistema base, más en particular, se utilizó la imagen *nvidia/cuda:12.1.1-runtime-ubuntu22.04* la cual contiene los *drivers* necesarios para utilizar CUDA en caso de contar con una GPU *Nvidia*. Como se mencionó anteriormente, los archivos multimedia se guardan en el *file system* para un mejor rendimiento, es por esto que se crea un Docker Volume en donde se almacena esta información. El resto de detalles sobre el despliegue, se pueden observar en la Figura 3.7.

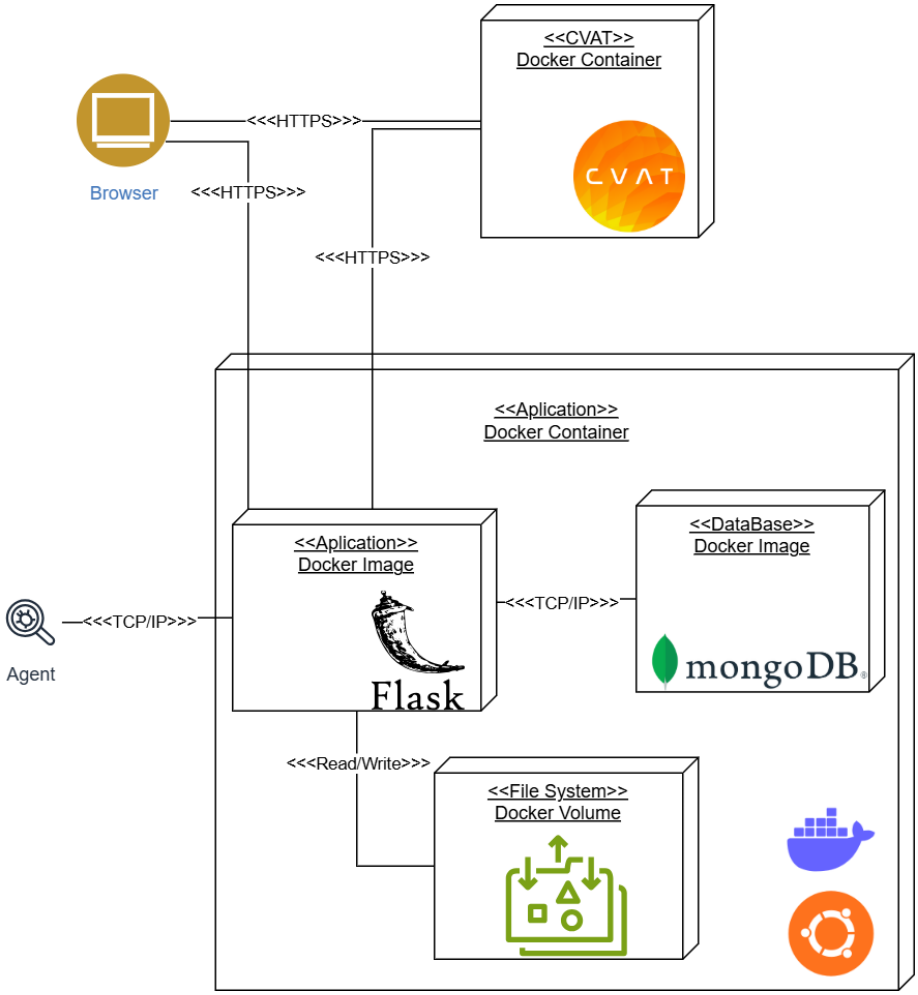


Figura 3.7: Diagrama de despliegue del sistema.

Capítulo 4

Experimentación

4.1. Datos experimentales

Al momento de evaluar la eficiencia del sistema implementado se presenta un problema claro, el conjunto de datos de caminos de hormigas con el que se cuenta es de un volumen muy bajo. Pese a que el sistema plantea una solución para esta problemática, es necesario para la evaluación contar con una gran cantidad de imágenes. Es por esto, que con el fin de realizar pruebas de mayor calidad se utilizará una realidad distinta. En la búsqueda de otros contextos similares sobre los cuales trabajar, se encontró uno muy interesante y en el que se pueden extrapolar ciertos puntos de interés: la detección de ciclistas. En este campo de estudio se puede acceder fácilmente a una gran cantidad de datos etiquetados, esto impulsado por la creciente industria de conducción autónoma, lo que resuelve el problema de escasez comentado. Uno de los puntos de similitud entre los contextos es, que al igual que en la detección de caminos de hormigas, se busca la presencia o ausencia del objeto y no la detección e identificación de múltiples ocurrencias del objeto. Básicamente, interesa que el ciclista este presente o no en la imagen, como también era el objetivo con los caminos de hormigas. Otra similitud radica en que se puede considerar que son un objeto pequeño en la amplia perspectiva de las cámaras presentes en el tráfico o en aquellas presentes dentro de vehículos, paralelo a como se verían los caminos de hormigas en imágenes tomadas por un agente. Estos dos puntos de comparación conciernen a las imágenes en sí, pero hay un punto de paralelismo más importante que se da entre los conjuntos de datos que utilizaremos en la evaluación. En particular, serán dos que modelan muy bien la problemática que se presenta en este trabajo en cuanto a los métodos de obtención de las imágenes de entrenamiento. El primer conjunto de datos, al que partir de ahora llamaremos *general-ciclistas-dataset* contiene imágenes donde el ciclista es el foco, la imagen se tomó teniendo como objetivo capturar al ciclista, estas son imágenes fáciles de conseguir e identificar, en Figura 4.1 se puede ver una imagen representativa del conjunto de datos. Por otro lado, el segundo conjunto de datos

Figura 4.1: Imagen de ejemplo de *general-ciclistas-dataset*. El ciclista es el protagonista de la imagen, es fácil identificarlo. Imagen obtenida de: (Cycler, 2023).

Figura 4.2: Imagen de ejemplo de *dashcam-ciclistas-dataset*. La imagen no fue sacada a los ciclistas, sino que son parte del panorama, lo cual requiere observar detenidamente para identificarlos. Imagen obtenida de: (SemiEmptyGlass, 2022).

al que a partir de ahora llamaremos *dashcam-ciclistas-dataset* es una grabación tomada desde un auto por una *dashcam* en un recorrido por una ciudad y sobre los fotogramas de ese video se etiquetan las apariciones de ciclistas. En la Figura 4.2 se puede ver una imagen representativa del *dashcam-ciclistas-dataset*.

Estos dos conjuntos de datos se asemejan a la realidad con la que nos encontraríamos en relación a los caminos de hormigas. Se tiene un conjunto de datos de imágenes de caminos de hormigas que fueron tomadas con el objetivo de fotografiar al camino en sí. Estas imágenes son utilizadas para crear el modelo predictivo inicial, en el caso de los ciclistas serían las imágenes del *general-ciclistas-dataset*. En la Figura 4.3 se puede ver un ejemplo. Pero la realidad es que, en el uso práctico, las imágenes que toma un agente móvil van a ser muy diferentes a esas imágenes ideales y se asemejan más a lo mostrado en la Figura 4.4, su paralelo para los ciclistas serían las del *dashcam-ciclistas-dataset*.

Por lo tanto, para representar este paralelismo durante la experimentación, al comienzo se entrenará un modelo inicial a partir del *general-dataset-ciclistas* y las imágenes que entrarán al sistema presentado en este trabajo serán las del *dashcam-ciclistas-dataset*, de estas últimas se obtendrán los mejores candidatos y se reentrenará el modelo.

Un punto en contra, que tiene la selección de los conjuntos de datos, es que

Figura 4.3: Imagen utilizada para entrenar modelo inicial de detección de hormigas. La calidad de la imagen es excelente y el camino esta muy definido.

Figura 4.4: Representación de imagen obtenida por el Agente. La imagen tiene una baja calidad y el camino es difícil de identificar.

Nombre	Total Imágenes	Entrenamiento	Validación	Testing
general-ciclistas-dataset	1416	70 %	20 %	10 %
dashcam-ciclistas-dataset	13674	70 %	20 %	10 %

Tabla 4.1: Distribución de imágenes por conjunto de datos.

el *dashcam-ciclistas-dataset*, el que simulará la realidad a la que se enfrenta el agente, contiene alrededor de un 12 % de imágenes en las que no aparecen ciclistas. Algo que en la realidad, sobre todo pensando en los caminos de hormigas, será un valor mucho mayor, dado que la mayoría del tiempo el agente va a estar en terrenos donde no hay caminos de hormigas y encontrarlo será la excepción.

4.2. Entorno de pruebas

Las pruebas mostradas en esta sección fueron realizadas en una computadora con una tarjeta gráfica con 8 GB de memoria de video, 16 GB de memoria RAM y un procesador Intel Core i7-13620H de 10 núcleos (6 de rendimiento y 4 de eficiencia), con una frecuencia base de 2.40 GHz y una frecuencia turbo de hasta 4.90 GHz.

4.3. Modelos iniciales

Para comenzar esta sección, es importante aclarar precisamente de qué tratará la evaluación, esto ayudará a entender el porqué de las pruebas que se plantean a continuación. Se busca evaluar la eficiencia del aprendizaje activo al momento de reducir la carga de trabajo de anotación con respecto a la posible pérdida de eficiencia del modelo al tener una menor cantidad de información. A su vez, también se quiere apreciar la capacidad del sistema para adaptar el modelo a la realidad a la que se encuentra el agente.

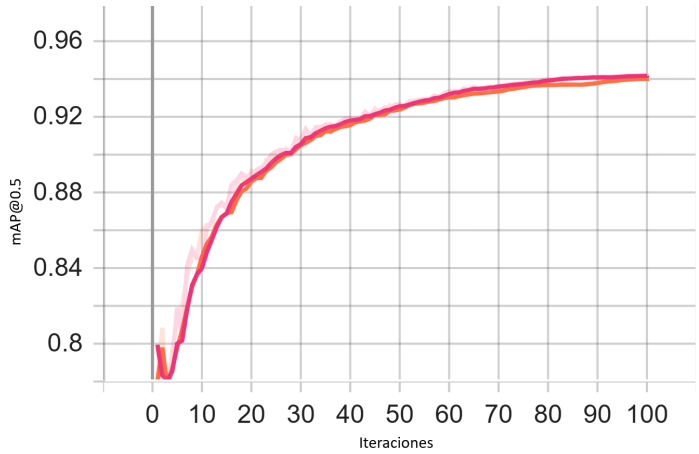
Para realizar la experimentación se llevará a cabo la siguiente partición de los conjuntos de datos que se observa en la Tabla 4.1. El conjunto de validación, será el utilizado para que el modelo pueda evaluar su entrenamiento, mientras que el de *testing* se utilizará para la comparación entre los diferentes modelos. Como se comentó anteriormente, uno de los objetivos es evaluar qué tan bien funciona el aprendizaje activo en reducir la carga de trabajo. Es por esto que en un principio se entrena un modelo de referencia utilizando la combinación de ambos conjuntos de datos—el *general-ciclistas-dataset* y el *dashcam-ciclistas-dataset*. Con esto, se quiere representar el hecho de que no se aplicó ninguna técnica de aprendizaje activo, sino que, de forma voraz, se etiquetaron todas las imágenes que se obtuvieron. En principio, este es el mejor modelo que se puede entrenar con los datos existentes, y se usará de referencia para ver el costo de rendimiento que tiene la utilización del aprendizaje activo. De ahora en más, se hará referencia a este modelo como “Modelo de referencia”. Por

otro lado, también se entrena un modelo inicial únicamente a partir del *general-ciclistas-dataset*. Este será el modelo inicial del sistema presentado, el cual se irá mejorando iterativamente a partir del aprendizaje activo. Como se comentó anteriormente, *general-ciclistas-dataset* representa un conjunto más básico, que no necesariamente se asemeja a la realidad final a la cual se enfrentará el modelo. Es por esto que será de utilidad para evaluar las capacidades del modelo para adaptarse a la realidad a partir de datos iniciales de entrenamiento no ideales. A partir de este momento, se hará referencia a este modelo, como "Modelo Inicial".

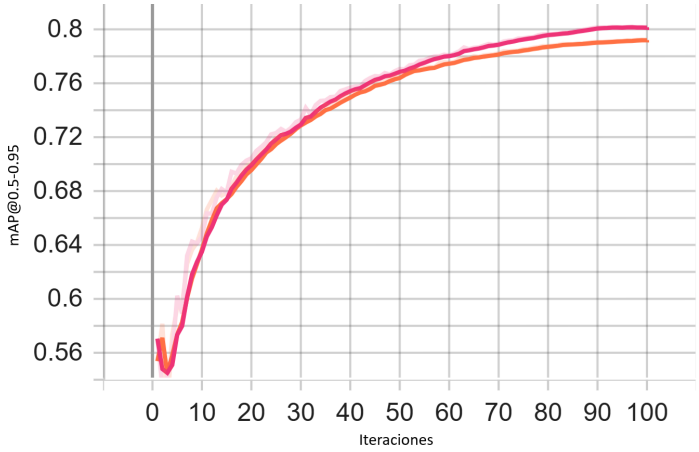
Para entrenar el modelo de referencia se utilizó como base el modelo YOLOv11n, la versión 11 es la última presentada y por ende la que tiene las mejores especificaciones. La razón de utilizar la variante *nano* se respalda en lo comentado en la sección 2.6.2, es la más liviana y con los tiempos de procesamiento más bajos, algo fundamental considerando que se espera sea utilizado en tiempo real y el agente tendrá una capacidad de procesamiento limitada. En cuanto a configuración se utilizaron 100 épocas, un tamaño de imagen de 640 X 640 píxeles y un tamaño de lote de 16. También se experimentó utilizando un tamaño de lote de 8, aunque los resultados son muy similares, hay una cierta ventaja en usar el tamaño de 16 como se puede ver en la Figura 4.5 sobre todo si consideramos la ganancia en *mAP@.5:.95*—qué tan bien detecta y localiza objetos con precisión. Otra posible variable a modificar en las pruebas podría ser el tamaño de imagen utilizado. El problema con esto es que aumenta exponencialmente la utilización de memoria de video al momento de entrenar el modelo, lo que hace inviable la ejecución en el entorno de pruebas actual. Para los demás parámetros de entrenamiento se mantuvo la configuración predeterminada de *YOLO*, con el objetivo de simplificar las pruebas y evitar que la modificación de dichos valores afectara los resultados obtenidos al variar únicamente los parámetros del sistema y evaluar su eficiencia al obtener las mejores imágenes disponibles. Es importante aclarar que todos los modelos mencionados en este documento tendrán esta misma configuración.

Por otro lado, al momento de entrenar el modelo inicial, se puede ver lo comentado en la sección 3.1 sobre la importancia de aumentar el conjunto de validación y que los modelos anteriores vuelvan a ser evaluados sobre los nuevos conjuntos de validación. Si observamos los resultados obtenidos al entrenar el modelo inicial mostrados en la Figura 4.6 vemos que tiene muy buenos resultados, tanto los valores de precisión como de *recall* son mayores al 90 %. Si únicamente esto guiara la elección del modelo, sería una mejor opción que el modelo de referencia. El problema está en que la validación del modelo se está realizando con una realidad muy sencilla, que únicamente da buenos resultados porque es similar a la de entrenamiento. Pero si se pone a prueba al modelo contra el conjunto de test que contiene tanto los datos de *general-ciclistas-dataset* como de *dashcam-ciclistas-dataset* se obtienen los resultados presentados en la Tabla 4.2, donde se ve que los valores de precisión y *recall* decaen fuertemente, y da una idea de lo que ocurrirá si el modelo fuese puesto en práctica en la realidad.

Una vez definido el modelo inicial y el modelo de referencia, se puede comenzar a evaluar el sistema propuesto. Para esto, en las secciones subsecuentes se



(a) Comparación de *mAP@50*



(b) Comparación de *mAP@0.50:0.95*

Modelo entrenado con 16 de tamaño de lote.

Modelo entrenado con 8 de tamaño de lote.

Figura 4.5: Curvas de mAP@50 y mAP@[.5:.95] durante el entrenamiento del modelo de referencia.

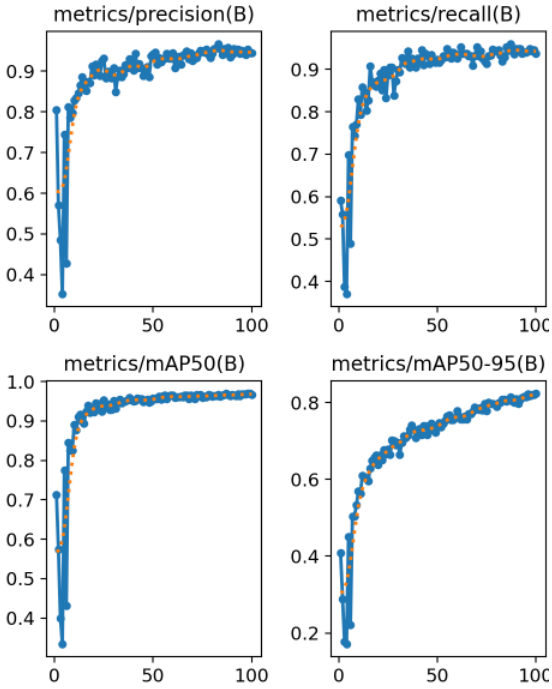


Figura 4.6: Distintas métricas del entrenamiento del modelo inicial en base a la cantidad de épocas.

Métrica	**Valor**
Precisión	0.653
Recall	0.463
mAP@0.50	0.512
mAP@0.50:0.95	0.238

Tabla 4.2: Resultados de la validación del modelo inicial sobre el conjunto de test combinado del general-ciclistas-dataset y dashcam-ciclistas-dataset.

Método	Configuración	Imágenes por iteración	Iteraciones	Imágenes en el sistema	Sección
Aleatorio	NA	1000	1	5000	4.4
Confianza	NA	1000	1	5000	4.4
Confianza	NA	500	2	5000	4.4
Aleatorio	NA	1000	8	10000	4.4
Confianza	NA	1000	8	10000	4.4
Confianza + *backgrounds*	10 % backgrounds	1000	8	10000	4.6
Similitud	$\beta = 0{,}75$	1000	8	10000	4.7
Similitud	$\beta = 0{,}5$	1000	8	10000	4.7
Disimilitud	$\beta = 0{,}75$	1000	8	10000	4.8
Disimilitud	$\beta = 0{,}5$	1000	8	10000	4.8
Confianza	NA	250	8	10000	4.9

Tabla 4.3: Resumen de configuraciones utilizadas en las pruebas de selección de imágenes.

mostrarán los resultados de realizar una selección de candidatos basada en los métodos de selección presentados en la sección 3.2. A su vez, se realizarán experimentos para validar la utilización de CLIP como base para medir la similitud de las imágenes. Por otro lado, se buscará variar la cantidad de imágenes disponibles y cantidad de imágenes utilizadas por iteración con el fin de ver como se comporta el sistema ante estos cambios. En la Tabla 4.3 se puede ver un resumen de las pruebas que serán realizadas sobre el conjunto de datos presentados iniciando el sistema con el modelo Inicial.

4.4. Pruebas de aprendizaje activo, método basado en confianza

Para comenzar las pruebas del sistema se propone observar el comportamiento utilizando el método basado en confianza como método de selección de candidatos. Es decir, cuanto menor es el valor de confianza asignado por el modelo sobre una imagen, más se considera como posible imagen de entrenamiento para la siguiente iteración. Para realizar estas pruebas se inicializó el sistema con el modelo inicial. Luego se tomaron aleatoriamente 5000 imágenes del *dashcam-ciclistas-dataset* y fueron enviadas al sistema para que sean evaluadas por el modelo inicial (estas serán las imágenes que estarán disponibles para ser tomadas por los métodos de selección de candidatos). En la Tabla 4.4 se pueden ver las imágenes disponibles para la selección y cómo está conformado el *dataset* de entrenamiento inicial del modelo.

Experimento	**Precision**	***Recall***	**mAP@50**	**mAP@50-95**	**Fitness**
Modelo de referencia	0.937	0.884	0.948	0.810	0.824
1000 aleatorias	0.850	0.762	0.831	0.634	0.653
1000 de un paso	0.856	0.742	0.824	0.626	0.646
1000 en dos pasos	0.846	0.750	0.820	0.627	0.646

Tabla 4.5: Resultados de las variantes en el conjunto de test combinado de *general-ciclistas-dataset* y *dashcam-ciclistas-dataset* utilizando 5000 imágenes en el sistema.

Nombre	**Modelo inicial**	**Sistema**
general-ciclistas-dataset	1416	0
dashcam-ciclistas-dataset	0	5000

Tabla 4.4: Composición de los *dataset* para las pruebas iniciales utilizando el método de selección basado en confianza. "Modelo inicial" indica la composición del *dataset* de entrenamiento del modelo inicial, mientras que "Sistema" son las imágenes disponibles que pueden ser seleccionadas por los métodos de selección.

A partir de este punto se realizaron iteraciones del sistema con tres métodos distintos de selección de imágenes: la primera fue elegir 1000 imágenes aleatorias para tener un modelo como referencia. Por otro lado, se realizó una iteración con las 1000 imágenes con menor confianza asignada por el modelo inicial, a este modelo se hará referencia como "1000 de un paso". Y por último se tomaron las 500 imágenes con menor confianza, con estas se realizó una iteración intermedia que predijo nuevamente las 4500 imágenes restantes que fueron enviadas al sistema, y de estas se tomaron las 500 con menor confianza nuevamente para realizar la siguiente iteración. A este modelo se le llamará "1000 en dos pasos".

En la Tabla 4.5 se pueden ver los resultados obtenidos por los modelos usando las tres variantes mencionadas trabajando con la combinación del conjunto de *test* del *general-ciclistas-dataset* y *dashcam-ciclistas-dataset*. Lo primero que se puede notar, es que el aprendizaje activo no está funcionando como se esperaría, se obtienen mejores resultados utilizando una selección aleatoria de los datos. Una posible hipótesis de por qué está sucediendo esto es que esté sufriendo la problemática mencionada en la sección 2.4.2: arranque en frío, ya que pese a que se está usando un modelo preentrenado, este difiere bastante de los nuevos datos y quizás no sea suficiente para sobrellevar el arranque en frío inicial. A su vez, en esta primera prueba no se considera agregar imágenes sin ciclistas, algo que en la selección aleatoria sí ocurre. Fuera de este problema, que será ahondado más adelante, hay otros puntos interesantes que se pueden observar de estos resultados. Aunque el modelo desarrollado en un paso y el de dos pasos tienen un comportamiento general similar, las diferencias son interesantes de analizar. El aumento en el recall indica que el modelo en dos pasos es mejor al identificar mayor variedad de ciclistas, lo que tiene sentido, ya que mientras que el modelo de un paso trabaja sobre las 1000 mejores imágenes que puede

identificar el modelo inicial, el modelo de dos, tiene una nueva porción de datos que únicamente es identificado por el modelo intermedio (entrenado con las mejores 500) y no por el modelo inicial.

Es importante aclarar que en estas pruebas tanto el conjunto de entrenamiento como el de validación está construido principalmente en base a datos del *general-ciclistas-dataset*, pero se puede ver que aún así, el modelo gana fácilmente generalidad y funciona relativamente bien en el conjunto de test combinado donde predominan datos pertenecientes al *dashcam-ciclistas-dataset*.

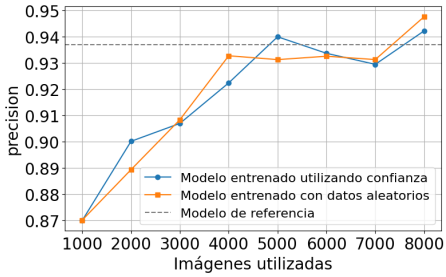
A continuación, se harán pruebas agregando más imágenes y realizando más iteraciones hasta que los valores se acerquen a los valores del modelo de referencia. Para realizar estas pruebas, se aumentó el conjunto de imágenes disponibles de 5.000 a 10.000 y se realizaron iteraciones agregando de a 1000 imágenes (70 % para entrenamiento, 20 % para validación, 10 % para test). En la Tabla 4.6 se pueden ver las imágenes disponibles para la selección y cómo está conformado el *dataset* de entrenamiento inicial del modelo.

Nombre	**Modelo inicial**	**Sistema**
general-ciclistas-dataset	1416	0
dashcam-ciclistas-dataset	0	10000

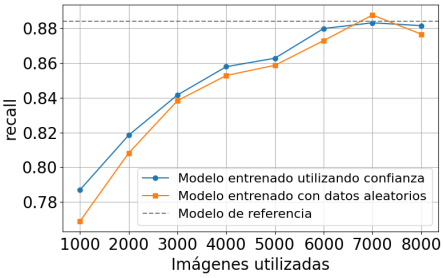
Tabla 4.6: Composición de los *dataset* para las pruebas iniciales utilizando el método de selección basado en confianza. "Modelo inicial" indica la composición del *dataset* de entrenamiento del modelo inicial, mientras que "Sistema" son las imágenes disponibles que pueden ser seleccionadas por los métodos de selección.

En la Figura 4.7 se pueden ver gráficas que muestra la comparación de los resultados obtenidos para Precisión, *Recall*, *mAP@0.50*, *mAP@0.50:0.95* y *fitness* respectivamente.

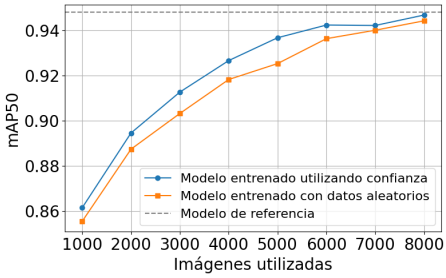
Al observar los resultados hay un primer punto que destaca, lo mostrado en la Tabla 4.5 se deja de cumplir, ahora para 1000 imágenes se obtiene que funciona mejor la selección de candidatos utilizando la confianza. Una posible explicación a este suceso es el aumento de la base de imágenes, al tener 10.000 posibles imágenes el modelo inicial logra identificar que alrededor de 4.000 de estas tienen un ciclista, estas van a ser las consideradas para la selección de candidatos, al aumentar este número aumenta la pluralidad de los datos, lo que permite al modelo generalizar de mejor forma. En la Tabla 4.7 se puede observar más claramente este fenómeno, pese a que es cierto que el aumento se da tanto para la selección aleatoria como para la selección utilizando la confianza, en este último el aumento es mucho más pronunciado (con excepción de la métrica de precisión). Esto se puede interpretar como que en la selección por confianza se optimiza la selección de candidatos, ya que se toman los 1000 mejores (según esta métrica) mientras que al seleccionar aleatoriamente pudo ocurrir que se seleccionaran los mismos 1000 que cuando había 5000 y el aumento en las estadísticas sería nulo. Es importante notar que la utilización de una selección de candidatos aleatoria solo es viable en entornos de prueba, donde el conjunto de datos tiene una proporción muy alta de los objetos a identificar. En



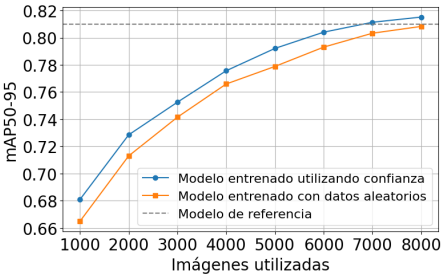
(a) Comparación de precisión.
Valor inicial: 0.653



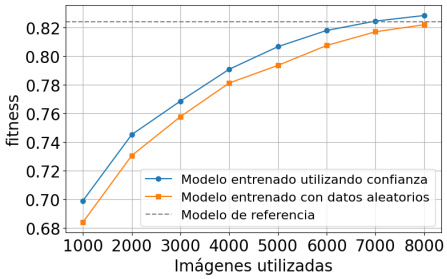
(b) Comparación de recall.
Valor inicial: 0.463



(c) Comparación de *mAP@0.50*.
Valor inicial: 0.512



(d) Comparación de *mAP@0.50:0.95*.
Valor inicial: 0.238



(e) Comparación de *fitness*.
Valor inicial: 0.265

Figura 4.7: Comparación de métricas de rendimientos para los modelos, en etapas de selección de 1000 imágenes cada una, utilizando el método de selección basado en confianza (línea azul) y la selección aleatoria (línea naranja).

Experimento	Precision	Recall	mAP@50	mAP@50-95	Fitness
Selección Aleatoria	+2.35 %	+0.89 %	+2.93 %	+4.87 %	+4.73 %
Utilizando Confianza	+1.63 %	+6.07 %	+4.53 %	+8.77 %	+8.21 %

Tabla 4.7: Aumento de las métricas de rendimiento para la iteración inicial al aumentar la base de 5000 a 10000 imágenes.

Número de iteración	Cantidad de *backgrounds* con una predicción
1	140
2	100
3	61
4	32
5	18
6	11
7	8
8	3

Tabla 4.8: Cantidad de *backgrounds* que contenían una predicción, es decir que podían ser elegidos para la próxima iteración, a lo largo de las iteraciones del modelo entrenado utilizando el método basado en confianza.

un ambiente real, en la mayoría de los casos, si se realiza una selecciona aleatoria de las imágenes entrantes al sistema, se obtendrían prácticamente en su totalidad imágenes sin el objeto, lo cual no tendría sentido para un entrenamiento.

Al continuar con la comparación entre ambas técnicas de selección de candidatos, se observa que, en la mayoría de los casos, la técnica basada en la confianza ofrece mejores resultados, con la excepción de la métrica de precisión. Sin embargo, existe una explicación razonable para este fenómeno. Con cada iteración el modelo va mejorando su precisión, esto hace que se detecten menos *backgrounds* como falsos positivos, esto se puede observar en la Tabla 4.8. Como únicamente se utiliza la confianza, por ejemplo, para seleccionar los candidatos para la última iteración, únicamente existían ocho *backgrounds* con predicciones, por lo que dentro del conjunto de todas las imágenes que era realmente *backgrounds*, solo estas podían ser seleccionadas como posibles candidatas, el resto, como no eran predichas como un posible objeto, no tienen una confianza asociada. Esto tiene como consecuencia que en cada iteración la cantidad de *backgrounds* vaya disminuyendo su proporción dentro del conjunto de datos, aunque para la selección aleatoria esta disminución también ocurre ya que la proporción de *backgrounds* es menor en el *dashcam-ciclistas-dataset* que en el *general-ciclistas-dataset*, sucede de una forma menos abrupta que para la selección basada en confianza, como se puede ver en la Figura 4.8. Este bajo porcentaje de *backgrounds* podría estar mermando la precisión de los modelos con selección basada en confianza, es por esto que más adelante en este capítulo se experimentara formas de obtener *backgrounds* del conjunto de imágenes disponibles para realizar un entrenamiento de mayor calidad.

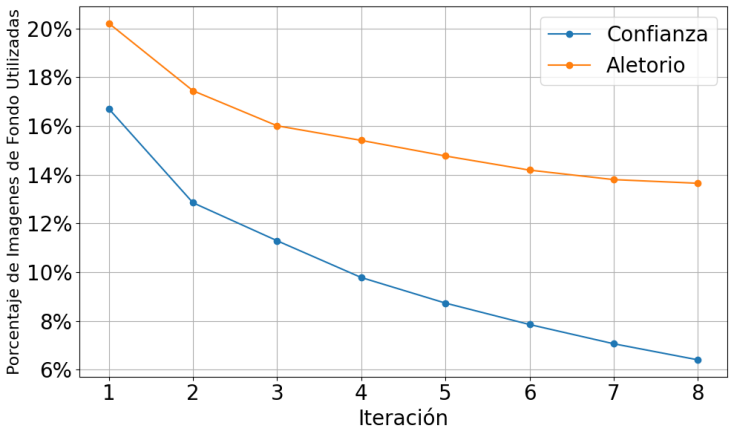


Figura 4.8: Proporción de *backgrounds* para cada iteración del sistema, para los modelos entrenados con el método de selección basado en confianza (línea azul) y la selección aleatoria (línea naranja).

4.5. Pruebas de evaluación de la similitud entre imágenes mediante CLIP

Con el objetivo de entender cómo funcionará la medida de similitud utilizando el modelo CLIP, se realizaron una serie de pruebas utilizando los *embeddings* generados por una versión de este modelo publicada por *OpenAI* y disponible a través de la plataforma *Hugging Face*. Las pruebas fueron realizadas en los mismos conjuntos de datos que se utilizaron para las anteriores pruebas. En particular se tomó un subconjunto de 200 imágenes del *general-ciclistas-dataset*, de este subconjunto se obtuvieron los *embeddings* generados por el modelo de todas las imágenes, para luego tomar una aleatoria y compararla con todas las demás. En la Figura 4.9, se puede observar la imagen original con la que se comparó a todas las demás, y las tres imágenes que obtuvieron un mayor valor de similitud, este valor fue obtenido utilizando la métrica de similitud del coseno como se explicó en la sección 3.2.2. Se puede apreciar cómo la similitud en este sentido está fuertemente ligada a una evaluación semántica de las imágenes y no por ejemplo a una evaluación que considere simplemente los píxeles abstraídos de ellas. En las cuatro el factor determinante es que hay mujeres conduciendo en bicicleta, nada fuera de esto lleva a pensar que las imágenes son similares. El hecho de que la similitud esté fuertemente ligada a la semántica de las imágenes tiene sentido, ya que el modelo fue entrenado con el objetivo de reconocer qué imagen está asociada a qué descripción de texto, y en efecto podríamos suponer que las cuatro imágenes tendrían una descripción muy similar en lenguaje natural. Sin embargo, en evaluaciones similares en otros contextos más reducidos, como es el caso de los caminos de hormigas, donde la variación entre las imágenes es menor, se observó que cobra mayor relevancia una similitud de carácter más

(a) imagen base

(b) similitud: 0.90

(c) similitud: 0.88

(d) similitud: 0.88

Figura 4.9: Ejemplo de eficiencia de CLIP al determinar la similitud entre imágenes al utilizar la similitud del coseno entre los embeddings resultantes. Se muestran las tres imagenes más similares a la imagen base.

visual, cercana a una comparación pixel a pixel. Por ejemplo, cuando la imagen seleccionada era perteneciente a un video, y dentro del subconjunto de prueba había otras imágenes del mismo video, eran estas las que mayor similitud tenían, los resultados de estos experimentos se pueden encontrar en la sección A.1 del anexo.

4.6. Agregando *backgrounds*

El haber comprobado en la sección anterior la eficiencia de usar modelos para comparar la similitud entre imágenes abre la puerta para retomar la discusión que se había dejado abierta en el final de la sección 4.4. La utilización de la similitud permite obtener *backgrounds* de calidad, los cuales son similares al conjunto de datos, es decir, son similares a los que el agente se encuentra. Es por esto que se sumó a los métodos de selección de candidatos implementados la

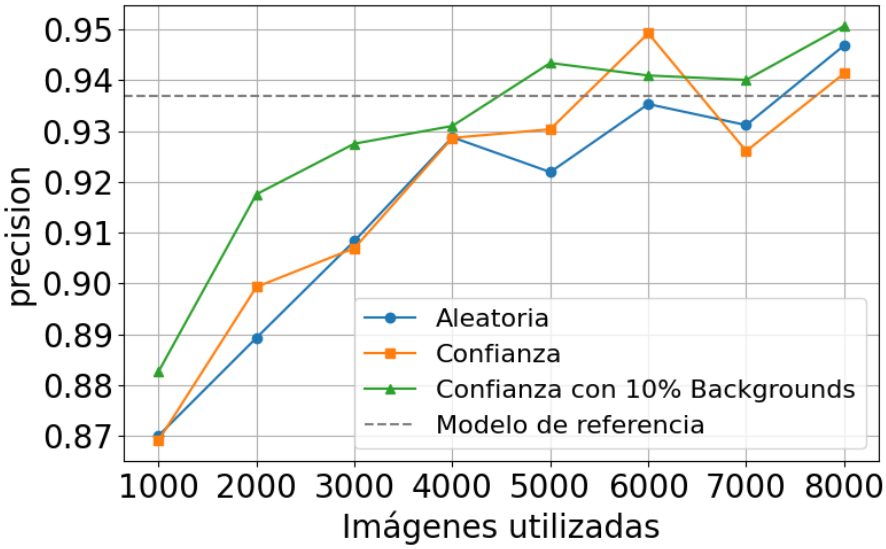


Figura 4.10: Comparación de la medida de precisión entre el método basado en confianza (anaranjado), el aleatorio (azul) y el basado en confianza sumando 10 % de *backgroundss* (verde). Valor inicial: 0.653

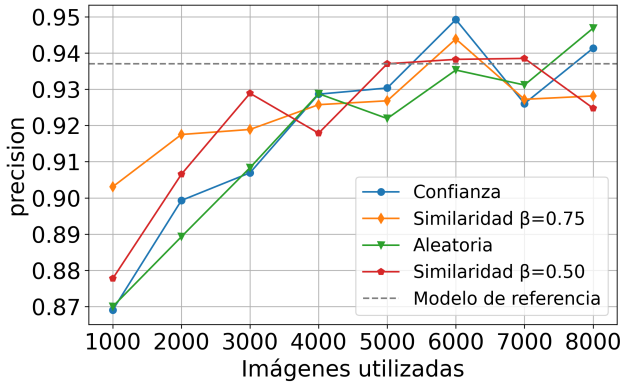
posibilidad de incluir *backgrounds* cuando se considere necesario y que no sean una consecuencia de que el modelo lo identifique de forma errónea como un objeto. Estos *backgrounds* son obtenidos basados en la similitud con el conjunto total de datos, las imágenes más similares al conjunto total de imágenes y que el modelo no detecte un objeto son las seleccionadas como *backgrounds*. Es importante notar que esto puede fallar, al igual que existen los falsos positivos al momento de incluir imágenes con objeto al utilizar la confianza, en este caso se puede incluir *backgrounds* que en realidad son un falso negativo. De todas formas, estos fallos no son un gran inconveniente, de hecho, pueden llegar a ser positivos, ya que se le está dando un ejemplo donde el modelo está funcionando de forma incorrecta. El porcentaje puede ser ajustado como un parámetro de los métodos de aprendizaje activo.

Se realizaron los experimentos de la misma manera que en la sección 4.4. Los resultados de los mismos los podemos ver en la imagen 4.10. Se puede observar en ellos lo que se había teorizado: la caída en la precisión estaba relacionada con falta de *backgrounds*. Al pedirle al sistema que el 10 % de las imágenes en cada iteración fueran backgrounds, el modelo supera en precisión a los otros métodos de selección.

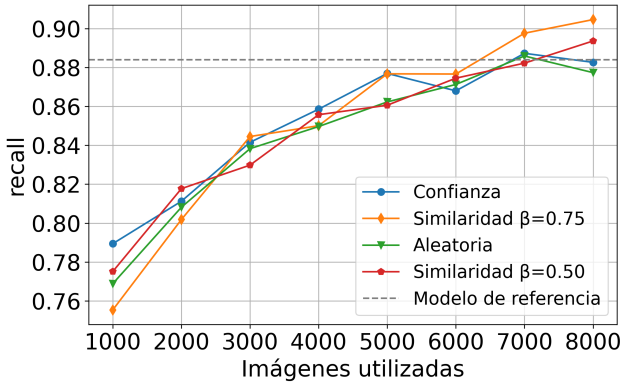
4.7. Similitud al conjunto de datos disponible

Además de la posibilidad de agregar *backgrounds*, el contemplar la similitud de las imágenes abre otras puertas, entre ellas la de obtener las imágenes más representativas del conjunto, como se menciona en la sección 3.2.2. Tiene sentido que la utilización de las imágenes más similares a la realidad que se encuentra el agente repercuta en un mejor desempeño en esa realidad, es por esto que se realiza un experimento obteniendo imágenes utilizando la función de *puntuación* mencionada en la sección 3.2.2, en particular utilizando $\beta = \{0{,}5, 0{,}75\}$. En la Figura 4.11 se puede observar los resultados obtenidos para ambas configuraciones de β así como tener de referencia la selección basada en confianza y aleatoria. A primera vista, el beneficio más notable en la utilización de la similitud es el aumento de la precisión del modelo en las primeras iteraciones. En particular, la utilización de $\beta = 0{,}75$ conlleva un aumento de alrededor de 3.4 puntos de la precisión en la primera iteración. A cambio de este beneficio, se tiene una perdida de 3.4 puntos en el *recall*. Con el fin de intentar explicar este fenómeno, algo interesante sería observar la cantidad de *backgrounds* que se están agregando al modelo: para la primera iteración esta cantidad es de 16 % mismo porcentaje que la cantidad de *backgrounds* agregados en la primera iteración que el experimento de la sección anterior. Pero algo importante a notar, es que en la sección anterior se obtenían *backgrounds* que eran los más similares al conjunto de datos y que no estaban siendo predichos por el modelo, es decir, el modelo ya los identificaba como *backgrounds*, ya que esta es la mejor forma de asegurarse que efectivamente sea un *backgrounds* (aunque obviamente puede fallar). En cambio, en el experimento presentado en esta sección no se busca obtener *backgrounds*, sino que se busca obtener imágenes basado en una función ponderada entre que sean las más similares al conjunto de datos con el que se está trabajando y que el modelo no sea bueno para determinar si existe un objeto o no. Pese a que no se busca activamente esta consecuencia, se está teniendo una ganancia orgánica de *backgrounds*. Se puede afirmar que estos *backgrounds* aportan una mejor calidad de información al modelo, ya que al momento este no estaba prediciendo esos *backgrounds* correctamente. Este fenómeno es el que dispara la precisión, ya que se le está dando una fuerte corrección al sesgo del modelo de predecir esos *backgrounds* como objetos.

Hacia las iteraciones finales, se obtiene un resultado más esperable, el *recall* es mejor con la utilización de similitud. Esta era la hipótesis más esperada antes de realizar la experimentación: al obtener las imágenes más similares al conjunto de datos, se obtienen mayor *recall*, ya que la mayoría de las imágenes contienen ciclistas y estas son las más representativas del conjunto de datos. Tiene sentido que esto se empieza a notar hacia al final, ya que el modelo comienza a ser mejor en su tarea y cada vez hay menos *backgrounds* en los que el modelo tenga incertidumbre si es un objeto.



(a) Precision. Valor inicial: 0.653



(b) Recall. Valor inicial: 0.463

Figura 4.11: Métricas de precisión y recall por iteración para los métodos de selección basados en similitud (con valores de $\beta = \{0{,}5, 0{,}75\}$), selección basada en confianza y selección aleatoria.

4.8. Disimilitud al conjunto de entrenamiento

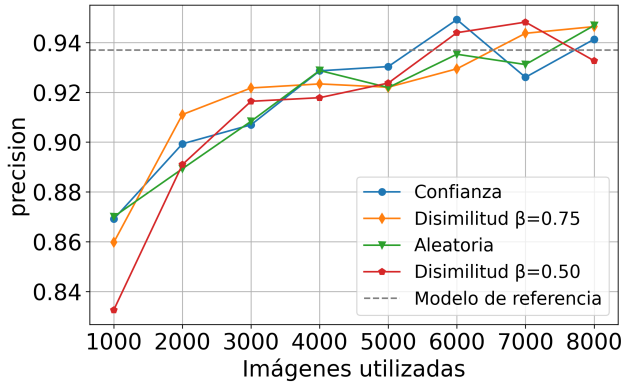
Además de la obtención de las imágenes más representativas del conjunto de datos, otra posible hipótesis para mejorar el modelo es obtener las imágenes más diferentes al conjunto de entrenamiento del modelo. Esto se basa en que estas imágenes pertenecen a una realidad en la que el modelo no fue entrenado y seguramente vaya a tener un mal rendimiento al momento de ejercer su función de predicción sobre estas imágenes. Básicamente, la hipótesis trata de que el modelo logre generalizar mejor a todas las imágenes que tiene de entrenamiento.

Al momento de observar los resultados (Figura 4.12), nuevamente las métricas más destacadas son la precisión y el recall. En las primeras iteraciones la utilización de imágenes disimilares al conjunto de entrenamiento existente se puede observar que la precisión merma fuertemente dependiendo del β seleccionado, al darle más peso a la diferencia con el *dataset* de entrenamiento, se va reduciendo la precisión. Es importante tomar cuidadosamente esta información, en la primera iteración, el conjunto de entrenamiento es muy disimilar al conjunto de posibles imágenes candidatas. En este punto el conjunto de entrenamiento proviene en su totalidad de *general-ciclistas-dataset*, mientras que el de posibles candidatos se compone por *dashcam-ciclistas-dataset*, es decir la mayoría de las imágenes son muy disimilares. Esta parece ser la razón por la cual los valores de la precisión mejoran tan abruptamente en la segunda iteración, una vez completada la primera iteración, el modelo ya es bastante más parecido al conjunto de datos y la disimilitud empieza a tener más peso, ya que hasta el momento todas eran muy disimilares. Al enfocarnos únicamente en las iteraciones finales, se ve algo interesante, utilizando similitud con $\beta = 0{,}75$ se obtuvo un mejor resultado en la precisión que el modelo entrenado unicamente con confianza y se mantuvo valores muy similares de recall. Esta modificación al algoritmo de obtención de candidatos es la primera que logra dicho resultado, ya que anteriormente la utilización de *backgrounds* aumentaba la precisión, pero ante un coste significativo de recall y lo opuesto ocurría para la utilización de la similitud al conjunto de datos.

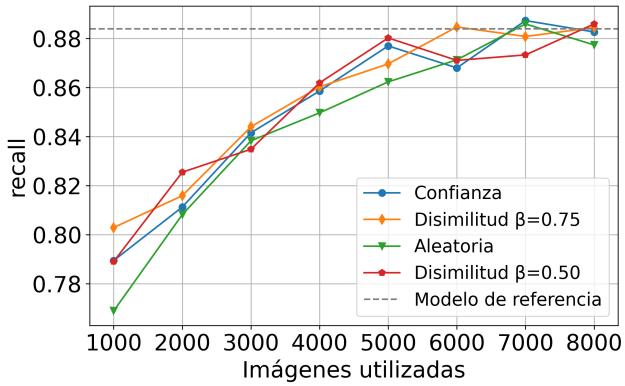
4.9. Evaluación comparativa de iteraciones con distintas cantidades de imágenes

Se decidió comprobar el impacto de reducir la cantidad de imágenes por iteración. En las etapas de experimentación anteriores se tomaban iteraciones de 1000 imágenes, para este análisis se disminuyó esa cantidad en un 75 %, realizando iteraciones de solo 250 imágenes. Se realizan ocho iteraciones, habiendo sumado en la última un total de 2000 imágenes al modelo, y se compara con dos iteraciones tomando de a 1000 imágenes que sumaría la misma cantidad.

En la Figura 4.13 se ve la gráfica con la diferencia final más grande para todas las metodologías de selección probadas bajo la métrica de *fitness*, el resto de los resultados puede encontrarse en la sección A.7 del anexo. Si bien en todos los casos los resultados son mejores realizando más iteraciones, esto no es en gran



(a) Precision. Valor inicial: 0.653



(b) Recall. Valor inicial: 0.463

Figura 4.12: Métricas de precisión y recall por iteración para los métodos de selección basados en disimilitud (con valores de $\beta = \{0{,}5, 0{,}75\}$), selección basada en confianza y selección aleatoria.

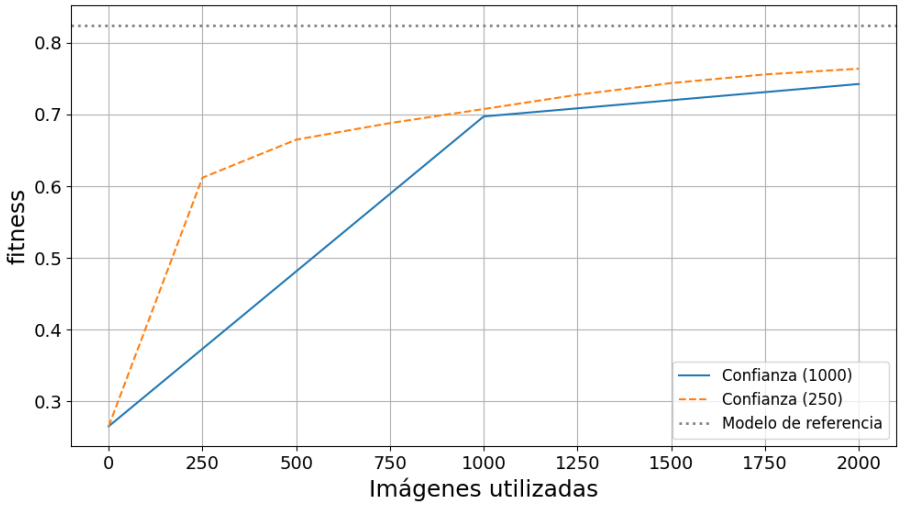


Figura 4.13: Comparación de *fitness* entre tomar iteraciones de 250 imágenes (línea naranja) e iteraciones de 1000 imágenes (línea azul) para el método basado en confianza.

medida y además no parecería explicarse en relación a que los métodos estén seleccionando mejor las imágenes en etapas intermedias, dado que la mejora también se observa en la selección aleatoria. Es probable que esta ventaja se deba al mayor número de épocas de entrenamiento: cada iteración implica 100 épocas, por lo que ocho iteraciones de 250 imágenes resultan en 800 épocas en total, frente a solo 200 épocas al usar dos iteraciones de 1000 imágenes.

Aun así, utilizar menos imágenes por iteración parece ser una estrategia efectiva en ciertos casos. El modelo mejora significativamente en las primeras etapas incluso con pocas imágenes, lo que indica que si se busca una mejora rápida con bajo esfuerzo de etiquetado, reducir el tamaño de cada iteración es una opción recomendable. Esto no compromete los resultados finales y puede acelerar el rendimiento del modelo en etapas tempranas.

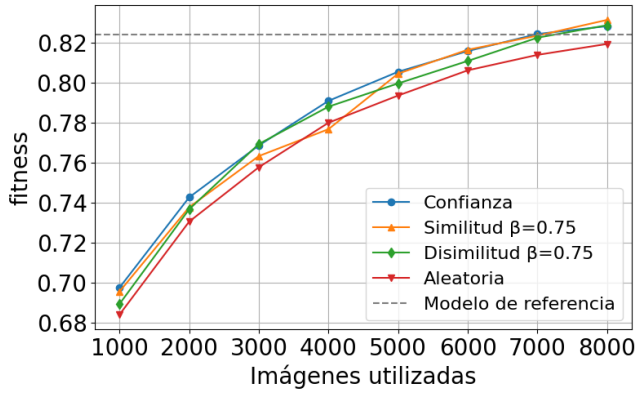
Otro punto notable al utilizar menor cantidad de imágenes es la baja de eficiencia para los algoritmos de selección presentados. En la Figura 4.14 se puede observar como en las iteraciones de 250 imágenes la utilización de únicamente confianza es superior al resto. Esto se puede interpretar como que cuando se obtienen pocas imágenes las de menor confianza ya son las mejores imágenes para el modelo, en cambio, cuando se obtiene un número alto de imágenes las peores posicionadas son imágenes en las que el modelo tiene bastante confianza de su predicción y por ende, no aporta tanta información relevante para la iteración. Acá es cuando entra en juego la utilización del resto de algoritmos de selección, las imágenes que realmente tienen poca confianza seguirán siendo consideradas por la forma en la que se calcula la puntuación de la imagen, en

cambio, las que tienen una confianza más elevada pueden ser sustituidas por imágenes que aportan otro tipo de información, ya que el peso ponderado de la confianza no es lo suficientemente alto y es substituida por imágenes que son más similares (o disimilares) y aporta información más interesante al modelo. También es una realidad que incluso para las iteraciones de a 1000 imágenes, en las primeras interacciones la confianza funciona mejor que el resto. Lo que se considera en este caso es que el modelo es bastante malo en su tarea y aún no se ha estabilizado, por esto, lo mejor es utilizar solo confianza, como por ejemplo se vio en la sección 4.8 donde se obtenían resultados variables, ya que el oráculo que se utiliza para obtener los candidatos es el propio modelo y en un comienzo este tiene mayor cantidad de falsos positivos y verdaderos negativos por lo que hace difícil la utilización de los algoritmos de selección más complejos.

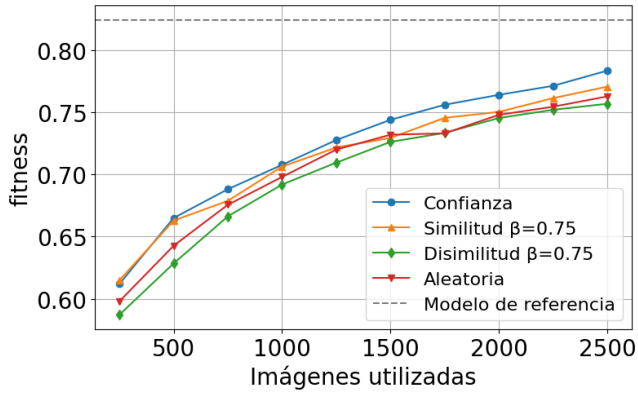
4.10. Análisis de los resultados

Antes de continuar con los experimentos finales, es importante destacar los resultados obtenidos hasta el momento. Esta información será utilizada para desarrollar un último experimento sobre un conjunto de datos de caminos de hormigas, aplicando los consejos que serán mostrados en esta sección. Lo primero que se desprende de lo mostrado en las secciones anteriores es que no hay un algoritmo de selección que sea claramente superior al resto. Esto no quiere decir que su utilización sea en vano, los algoritmos dan buenos resultados al modificar el modelo para aplicaciones específicas. Por ejemplo, la utilización de la similitud al conjunto de entrenamiento aumenta el *recall* perjudicando la precisión, esto podría ser muy interesante para ciertas aplicaciones como puede ser la detección de enfermedades, donde es prioritario la detección en la mayoría de los casos sobre no fallar nunca. Pero en otras aplicaciones, donde el costo de detectar incorrectamente un objeto esté muy penalizado, quizás la mejor opción es utilizar la agregación de *backgrounds*, lo que priorizará la precisión. Es importante notar, que no es necesario utilizar un único algoritmo de selección para el modelo. Es trabajo del administrador en cada iteración analizar el funcionamiento del modelo y a partir de esto tomar una decisión para mejorar el modelo. En el caso concreto de la detección de caminos de hormigas, el administrador debería evaluar si el agente está identificando pocos caminos. Esto puede deberse a nuevas condiciones (como un cambio de estación) o a que el modelo aún no está consolidado. En tales situaciones, una decisión acertada sería realizar una iteración utilizando el método basado en similitud. Por otro lado, si el costo operativo que está generando los falsos positivos detectados por el agente es muy alto, sería buena idea utilizar un porcentaje de *backgrounds* en la siguiente iteración.

Otro punto destacado de los resultados obtenido es la mejora en las medidas de *mAP@0.50:95*, en todos los algoritmos se obtiene una mejora con respecto a la selección aleatoria, y se maximiza al utilizar la similitud con un $\beta = 0{,}75$, aunque el aumento no es considerablemente mayor, existe, y puede ser beneficioso en aplicaciones donde se necesita identificar el objeto con precisión, como



(a) Comparación de *fitness* para iteraciones de 1000 imágenes. Valor inicial: 0.265



(b) Comparación de *fitness* para iteraciones de 250 imágenes. Valor inicial: 0.265

Figura 4.14: Comparación entre tomar iteraciones de 250 imágenes e iteraciones de 1000 imágenes para cada método de selección.

Algoritmo	Característica
Similitud	Aumenta el recall a cambio de una pérdida de precisión.
Disimilitud	Aumenta muy levemente todas las métricas.
backgrounds	Aumenta la precisión a cambio de una pérdida de *recall*.

Tabla 4.9: Efectos de los algoritmos en las métricas de evaluación.

es la conducción autónoma. Para la realidad de la detección de hormigas, este resultado es interesante para realizar técnicas como las mencionadas en la sección 2.3.2 donde el robot se acercaba a confirmar que sea un hormiguero, en este caso es importante tener una detección precisa de donde se encuentra el objeto para poder realizar un acercamiento correcto.

De la experimentación también se desprende que es importante ser cautos con la utilización de los algoritmos en las primeras iteraciones, o al menos, cuando se sabe que el modelo aún no es bueno en la realidad con la que se está trabajando. En esta situación ocurre que muchas de las imágenes seleccionadas son falsos positivos o verdaderos negativos, esto hace que la utilización de parámetros como la agregación de *backgrounds* puedan tener resultados inesperados, ya que por defecto en las imágenes que no son consideradas *backgrounds* puede haber un alto nivel de falsos positivos, si a esto además se le suman más *backgrounds* a propósito haría que la iteración tenga una cantidad demasiado elevada de *backgrounds*. Es por esto que sería una buena estrategia que en las iteraciones iniciales únicamente se utilice la selección basada en confianza como algoritmo de selección. Más adelante, con el modelo más consolidado, si se podrán utilizar otros de los algoritmos de selección planteados para obtener resultados más específicos.

Para concluir esta sección en la Tabla 4.9 se puede ver un resumen del efecto que tienen los algoritmos presentados al ser utilizados comparados con la selección únicamente basados en confianza. En todos los casos se tuvo en cuenta $\beta = 0{,}75$ ya que fue la que dio los mejores resultados.

4.11. Experimentación sobre caminos de hormigas

Con el objetivo de dar cierre a la experimentación, se busca volver al inicio del proyecto y trabajar sobre la realidad del control de hormigas. Pese a que se obtuvieron resultados interesantes al utilizar los ciclistas, es importante validar el uso del sistema en esta área de interés. Aunque se buscó una realidad que tiene ciertas similitudes que fueron presentadas en la Sección 4.1, también existen diferencias muy notorias como lo es que los caminos son más difíciles de identificar y delimitar para los seres humanos, ya que pueden ser confundidos con imperfecciones del terreno, lo que hace muy compleja su anotación. Es por esto, que es necesario validar el uso del sistema en esta realidad y no simplemente confiar en los resultados obtenidos al trabajar con los conjuntos de datos

Figura 4.15: Imagen de ejemplo del conjunto *caminos-definidos*.

Figura 4.16: Imagen de ejemplo del conjunto *caminos-desde-videos*.

de ciclistas. Teniendo en cuenta la cantidad de datos disponibles, estas pruebas serán mucho más acotadas.

Para realizar esta experimentación, se utilizarán tres fuentes de imágenes. La primera, y la de mayor calidad, es el conjunto de datos armado durante el proyecto de Nadile y cols. (2025), el cual consta de 1.174 imágenes de caminos de hormigas. Este conjunto cuenta con imágenes de alta calidad, donde es fácil identificar el camino. Se llamará a este conjunto de datos *caminos-definidos*, en la Figura 4.15 se puede ver un ejemplo tomado de este conjunto de datos. En segundo lugar, se cuenta con videos de caminos tomados en diferentes locaciones y obtenidos durante el transcurso de este proyecto, además de un video extenso de una zona agropecuaria. Una vez procesados estos videos se obtuvieron 459 imágenes, a este conjunto de datos se le llamará *caminos-desde-videos*, una imagen de ejemplo se puede ver en la Figura 4.16. Por último, para realizar una mejor simulación de la realidad se utilizó el conjunto de datos publicado por Wigness, Eum, Rogers, Han, y Kwon (2019) el cual tiene 7.436 imágenes tomadas por un robot en diferentes locaciones, con esta información extra, se tiene un acercamiento a como funcionaria el sistema en la realidad, donde encontrarse caminos de hormiga sería la excepción en las imágenes tomadas por el agente, en la Figura 4.17 se encuentra un ejemplo de este conjunto, al que se le llamará *dataset-backgrounds*. De cada uno de estos conjuntos de datos se separó el 10 %

Figura 4.17: Imagen de ejemplo del conjunto *dataset-backgrounds*.

Conjunto	**Total**	**Modelo inicial**	**Sistema**	**Test**
Caminos-definidos	1174	500	560	114
Caminos-desde-videos	459	0	413	46
dataset-backgrounds	7436	0	6692	744

Tabla 4.10: Distribución de imágenes en los diferentes conjuntos de datos. Donde "Sistema" representa las imágenes disponibles para ser seleccionadas por las técnicas de selección de candidatos.

para formar el conjunto de test del sistema. Por otro lado, de *caminos-definidos* se tomaron 500 imágenes aleatorias para realizar el entrenamiento inicial del modelo. El resto de imágenes fueron puestas a disposición como candidatas a ser tomadas por el sistema, simulando que el agente se los encontró en su recorrido. En la Tabla 4.10 se puede ver un resumen de las asignaciones de cada conjunto de datos.

Debido a la peculiaridad de los caminos de hormigas que pueden tener orientaciones diversas, a diferencia de los ciclistas que siempre están perpendiculares a la línea del horizonte, se decidió utilizar un modelo que soporte *Oriented bounding boxes* para realizar anotaciones y predicciones más precisas. En particular, se utilizó el modelo *yolo11n-obb*. Nuevamente, en la Tabla 4.11 se puede ver que al enfrentar al modelo inicial contra el conjunto de datos de test se observa un rendimiento muy pobre. El rendimiento es bastante peor al obtenido para la detección de ciclistas en la Sección 4.3. Estos pueden ser indicios de que la detección de los caminos de hormigas es más difícil que la de ciclistas, aunque es importante tener en cuenta que el conjunto de datos inicial en el caso de los ciclistas era aproximadamente el doble que en este caso, y esto tiene un gran peso en los resultados obtenidos.

Durante estas pruebas, debido a la poca cantidad de imágenes disponibles, se agregarán 250 imágenes por iteración. Siguiendo la recomendación presentada en la sección anterior, para estas pruebas se decide hacer las iteraciones iniciales con el método basado solamente en confianza, recién a partir de la segunda iteración,

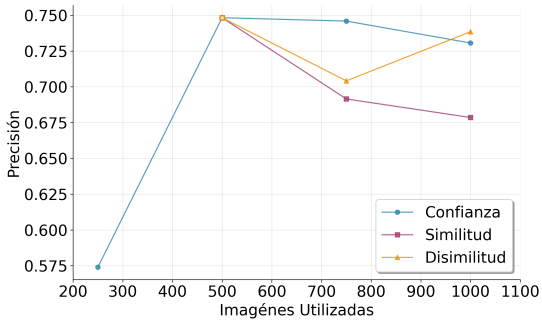
Métrica	Valor
Precisión	0.196
Recall	0.349
mAP@0.50	0.121
mAP@0.50:0.95	0.0412

Tabla 4.11: Resultados de la validación del modelo inicial sobre el conjunto de test.

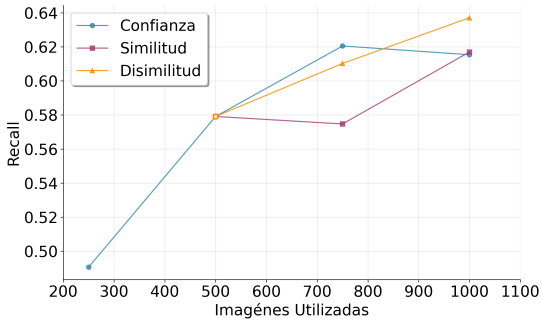
se varían los métodos y se realiza una bifurcación para cada uno. Es importante aclarar, que en los casos de similitud y disimilitud se usó $\beta = 0{,}75$ ya que fue la opción que mostró tener más eficiencia sobre el conjunto de los ciclistas. En la Figura 4.18 se puede observar la comparación de precisión, *recall*, *mAP@0.50* y *mAP@0.50:0.95* para los diferentes métodos de selección. Es claro que los resultados son peores a los obtenidos con los ciclistas, esto es algo esperable, primero por la diferencia en la cantidad de imágenes utilizadas, pero también por la dificultad de la detección de los caminos de hormigas. Diferenciar un camino de hormigas es una tarea compleja incluso para un humano. Reconocer que en cierta imagen existe efectivamente un camino, o los límites de este, en cuanto a su grosor y extensión, en muchos casos se trata de una tarea no trivial, que requiere criterios subjetivos y decisiones particulares caso por caso. Si se toma como ejemplo la Figura 4.19 la presencia del camino es clara, si se observa con detenimiento se pueden ver hormigas. Pero al momento de delimitar el camino hay varias opciones que se podrían tomar, la primera y más conservadora es únicamente etiquetar como camino la porción continua donde se encuentran las hormigas (marcado en celeste). También, otro anotador podría considerar que la sección marcada en rojo, es parte de ese mismo camino, ya que aunque no se ve la continuidad con el anterior, sus características son muy similares y parecen ser parte del mismo camino. Esta lógica se puede repetir con la porción marcada en violeta. Lo cierto es que, a no ser el caso que haya una hormiga o el camino esté muy bien delimitado, es difícil ser concluyente en sí es un camino o no, ya que perfectamente podrían ser simplemente características del terreno o la forma en la que se expande la vegetación.

Esta ambigüedad al momento de anotar puede generar información contradictoria al entrenar el modelo, lo que repercute en peores resultados. Pese a que en el transcurso de este proyecto se optó por descartar todas aquellas imágenes que generaban mayores dudas y se realizó un contraste de opiniones entre los participantes, probablemente se hayan anotado incorrectamente ciertos caminos afectando así al rendimiento del modelo. De todas formas, es clara la mejoría del modelo al utilizar el sistema, sobre todo en las primeras iteraciones, y da buenos indicios de que el sistema puede ayudar en el problema de la detección de caminos de hormigas.

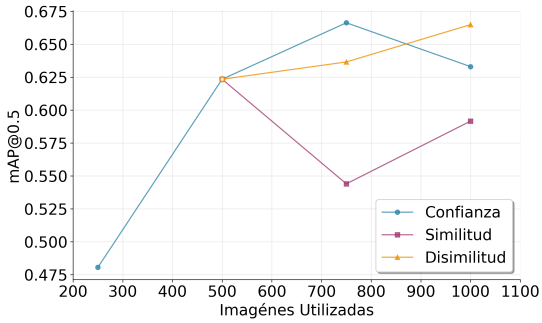
Al analizar más en específico cada una de las métricas, se puede ver como para el *mAP@0.50:0.95* se mantiene lo mostrado en la sección 4.9, donde el método de selección basado en confianza era el mejor para optimizar esta métri-



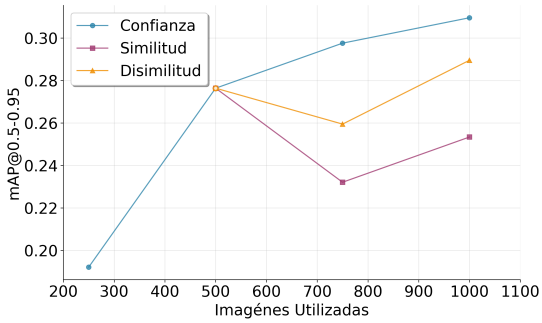
(a) Comparación de precisión. Valor inicial: 0.196



(b) Comparación de *recall*. Valor inicial: 0.349



(c) Comparación de *mAP@0.50*. Valor inicial: 0.121



(d) Comparación de *mAP@0.50:0.95*. Valor inicial: 0.041

Figura 4.18: Comparación de métricas al utilizar las diferentes técnicas de selección de candidatos. Las primeras dos iteraciones son utilizando la selección basada en confianza y luego para las siguientes dos se bifurca en seguir usando confianza (azul), usar disimilitud (amarillo) y usar similitud (rojo).

(a) Ubicación de las hormigas (b) Opciones de delimitación del camino

Figura 4.19: Ejemplo de una imagen que representa un desafío para la anotación. En la imagen a se muestran con flechas rojas la presencia de hormigas. En la imagen b se muestran opciones de delimitación del camino en azul, rojo y violeta.

ca, al utilizar iteraciones de poca cantidad de imágenes. Es interesante además notar que esta métrica es en la que se observa mayor diferencia con respecto a la realidad de los ciclistas, claramente esto está correlacionado con lo mencionado anteriormente, el hecho de que las anotaciones generen dudas y sea difícil delimitar los caminos va a repercutir fuertemente en que el modelo no logre predecir correctamente donde se encuentra y cuáles son sus dimensiones. De todas formas, considerando la realidad del control de hormigas, no es tan deseable detectar tan precisamente la ubicación del camino, como si lo es detectar si existe o no. Con respecto a esto, el método de selección basado en disimilitud tuvo un buen impacto en el *recall*, sin perder eficiencia en la precisión. No así lo sucedido con la selección basada en similitud, que es la que tiene peores resultados en todas las métricas. Esto puede estar revelando un problema intrínseco que tiene este método, a diferencia de las pruebas realizadas para los ciclistas, en este caso, la mayoría de las imágenes que constituyen el sistema son ruido, *backgrounds* que no aportan información al problema de detección. El problema está en que este método de selección se basa en la similitud a todas las imágenes sin etiquetar existentes en el sistema y en realidades como la detección de camino de hormigas la mayoría de estas serán *backgrounds*, esto hace que este método sea circunstancial, dependiendo fuertemente de la distribución total de las imágenes cargadas.

4.11.1. Análisis cualitativo de la evolución de los modelos

Con el fin de entender como va cambiando cada modelo en las diferentes iteraciones es relevante tener un análisis cualitativo, aunque menos formal, de como va evolucionando el modelo al utilizar el sistema. Para esto, se seleccionaron imágenes del conjunto de datos de test que se consideraron representativas de distintos tipos de caminos de hormigas, dependiendo de la vegetación y el tipo de suelo. Por su extensión, no todas las imágenes seleccionadas serán mostradas en esta sección, el resto serán presentadas en el anexo. Al observar la

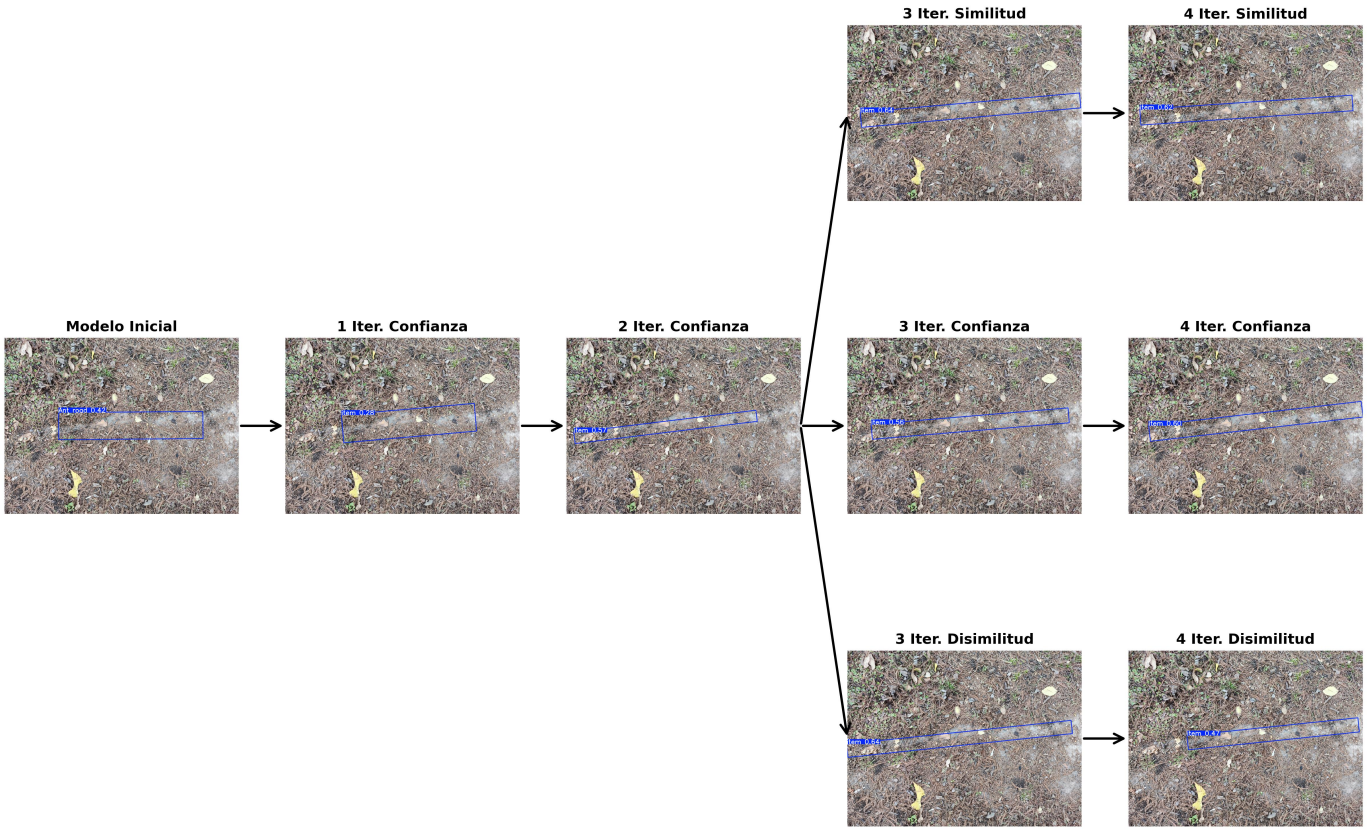


Figura 4.20: Evolución visual de cómo el modelo va mejorando en la detección precisa.

evolución, lo primero que se puede ver es que cuando el camino de hormiga es claro, el modelo lo detecta correctamente en todas las iteraciones y esta información nunca se pierde. En la Figura 4.20 se puede ver un ejemplo, un poco más complejo, en donde el modelo inicial logra detectar correctamente la presencia del camino, pero luego con las iteraciones del sistema se va afinando la precisión a la hora de definir la caja delimitadora.

Por otro lado, en la Figura 4.21 se tiene el ejemplo de un terreno árido, donde el camino es muy difícil visualizar, por esto se marca en celeste para que sea más fácil identificarlo. Es cuestionable si esto es un camino o no, pero en los ejemplos similares etiquetados que se proporcionaron al sistema se lo consideró un camino. Este tipo de imágenes no está incluido en el conjunto de entrenamiento del modelo inicial, y es muy interesante notar que el sistema logra que el modelo reconozca este tipo de imágenes, aunque inicialmente no existía nada similar en el conjunto de entrenamiento. Esto muestra que el sistema tiene cierta capacidad para lograr que el modelo extrapole hacia nuevos contextos.

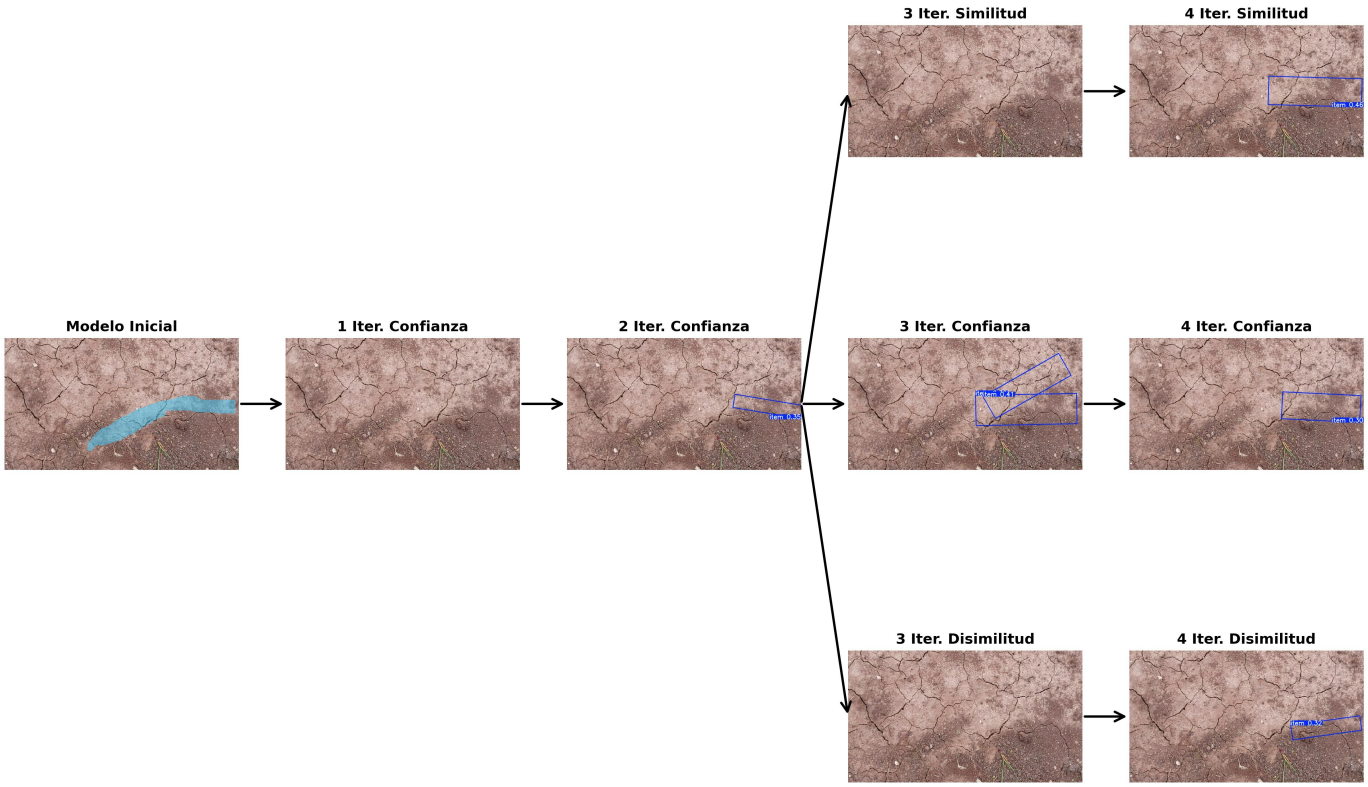


Figura 4.21: Evolución visual del modelo en suelos muy diferentes. Se marca en la primer imagen en celeste el camino que se quiere detectar.

Aunque en este caso en ningún momento logra detectar con facilidad el camino, esto podría mejorar al tener más ejemplos de este tipo de terreno árido, que aunque existe en las imágenes candidatas para el sistema, son un porcentaje mucho menor. Este problema se solucionaría al trabajar en entornos reales, donde si el agente se enfrenta constantemente a este tipo de terrenos, el sistema tendrá más información para utilizar.

Por último, se quiere cerrar mostrando un ejemplo en el que el modelo perdió poder de predicción. En la Figura 4.22, sobre las iteraciones finales, en todos los casos tiene un mal desempeño. Para similitud y disimilitud realiza una caja muy chica. Si se ve el avance que tiene en las iteraciones, parece que el modelo está sobreajustando a que la entrada del hormiguero que se ve en la imagen es el objetivo, algo interesante, pero que hace que no se detecte el camino como tal. Además, en la última iteración de confianza ni siquiera detecta la presencia del camino. Es importante mostrar este resultado, obviamente el sistema no es una solución impecable y con cada iteración aunque se gana información y el objetivo es generalizar cada vez más el modelo, también se está perdiendo

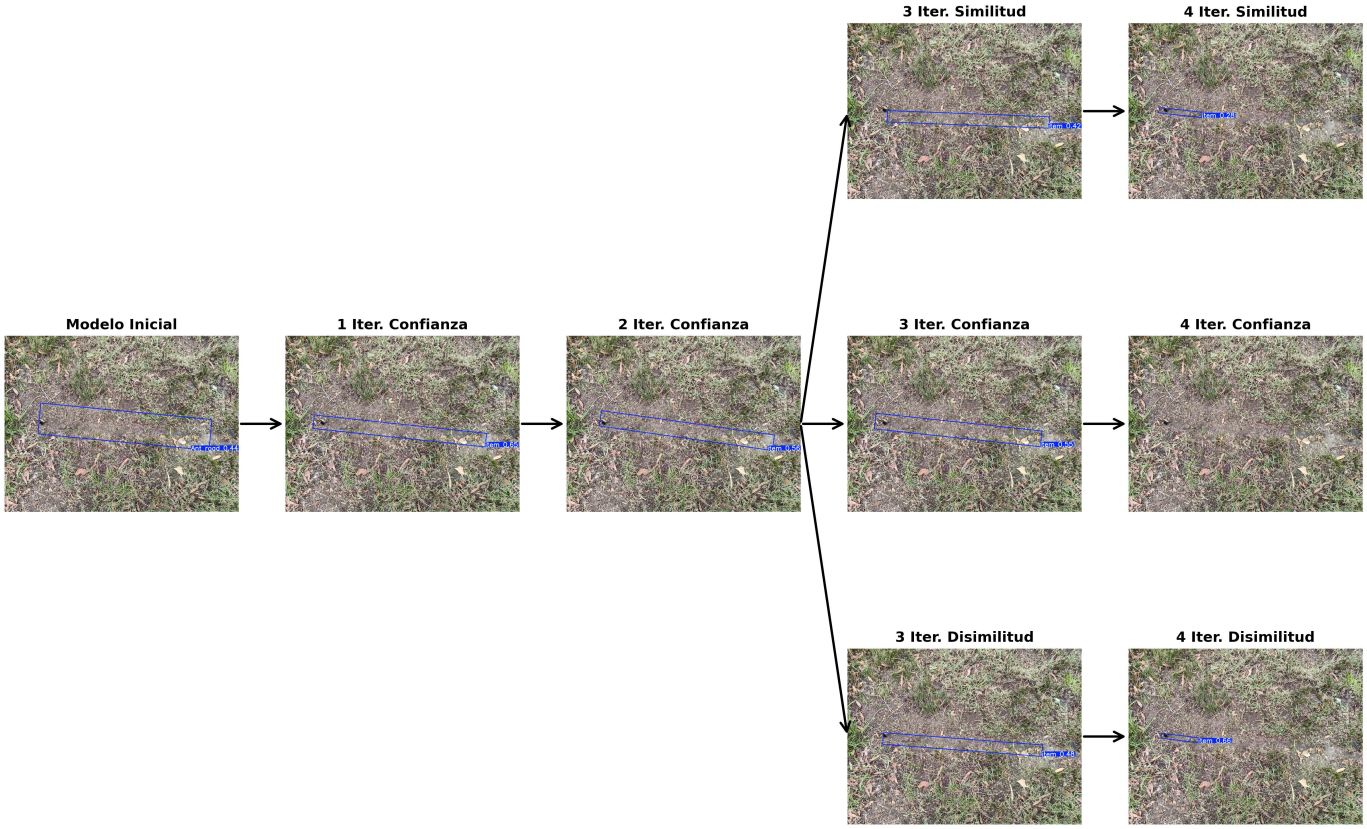


Figura 4.22: Evolución visual del modelo en suelos muy diferentes.

información para ciertos casos. En entornos reales, es importante saber cuando cortar el entrenamiento. Probablemente sea buena idea que, una vez se detecta que una versión funciona muy bien para cierto tipo de terrenos o condiciones, utilizarla en esos casos y dejar de iterar con el sistema.

4.11.2. Comparación con los resultados obtenidos en los antecedentes de búsqueda de caminos de hormigas

Para finalizar esta sección es importante comparar los resultados mostrados en este informe con los obtenidos en el trabajo de Nadile y cols. (2025). En particular se comparará la detección de caminos, sin importar si están activos o no, tomando los resultados presentados en dicho informe, con una evaluación realizada por el modelo resultante de la última iteración utilizando disimilitud presentada en este informe. Los resultados se medirán utilizando dos *datasets*. El primero esta conformado por las imágenes destinadas a test del conjunto *Caminos-definidos* al que se llamará *dataset-test-1*. El segundo es un *dataset*

Métrica	ARP	Ultima iteración disimilitud
Precisión	0.93	0.73
Recall	0.65	0.74
F1	0.76	0.75

Tabla 4.12: Resultados de la validación del modelo de deteción de ARP y el modelo resultante de la ultima iteración usando disimilitud sobre el *dataset-test-1*

utilizado para la evaluación de ARP en su totalidad, tanto la segementación de caminos como la detección de caminos activos, y contiene imágenes de caminos tomadas a más corta distancia y donde son visibles las hormigas, al que se llamara *dataset-test-2*. Para realizar la comparación se utiliza la Precisión, *Recall* y la medida F1 calculada como:

$$\text{F1} = 2 * (Precisión * Recall)/(Precisión + Recall) \tag{4.1}$$

y que busca representar el equilibrio entre Precisión y *Recall*, penalizando los extremos.

Los resultados obtenidos al evaluar los modelos sobre *dataset-test-1* se pueden observar en la tabla 4.12. Es interesante notar que aunque los resultados de F1 son muy similares la diferencia entre la precisión y recall está muy marcada. El modelo presentado en este informe es más equilibrado, y en general detectará mayor cantidad de caminos de hormigas pero con el costo de que tendrá más cantidad de errores. Mientras que el modelo presentado en ARP casi no tendrá caminos mal detectados pero con el costo de que muchos caminos pasarán desapercibidos. Estos resultados continúan sobre *dataset-test-2* y pueden observarse en la Tabla 4.13. Ambos modelo tienen una degradación en su rendimiento pero es algo esperable debido a que este *dataset* contiene imágenes considerablemente diferentes a las trabajadas anteriormente. Es difícil determinar que un modelo sea mejor que el otro, pero en un contexto como el de ARP donde el modelo será utilizado para porcionar la imagen y dentro de esa porción detectar si se encuentra con un camino activo o no, a partir de la detección hormigas, el modelo presentado en este informe podría tener mejores resultados. El tener un bajo recall en el modelo de detección de caminos es una problemática que es presentada por Nadile y cols. (2025) que degrada el sistema ARP, ya que al no ser detectado el camino ni siquiera se puede comenzar con la búsqueda de hormigas. Por otro lado, los falsos positivos ocasionados por la baja precisión son subsanados por la posterior búsqueda de hormigas, ya que se descartaran como caminos inactivos por no tener hormigas en él. En cambio, el modelo de detección de ARP puede tener un mejor desempeño al utilizarse aislado, ya que aunque el recall es bajo tiene menor cantidad de fallas. Mientras que el modelo presentado en este informe podría repercutir en un uso excesivo de cebos por detectar falsamente caminos de hormigas.

Métrica	**ARP**	**Ultima iteración disimilitud**
Precisión	0.9	0.72
Recall	0.53	0.59
F1	0.67	0.65

Tabla 4.13: Resultados de la validación del modelo de deteción de ARP y el modelo resultante de la ultima iteración usando disimilitud sobre el *dataset-test-2*

Capítulo 5

Conclusiones y Trabajo Futuro

La problemática del control de las hormigas cortadoras de hojas tiene varios puntos a resolver. Durante este proyecto se buscó atacar el problema de la falta de datos disponible y el trabajo operativo necesario para su obtención. Punto crucial para poder detectar su presencia y realizar un control preciso de su población. El sistema propuesto en este proyecto pretende agilizar la obtención de los datos basándose en una perspectiva ágil de entrenamiento de modelos. Se propone no esperar a alcanzar un modelo perfecto antes de comenzar a utilizarlo, sino adoptar una postura orientada en ponerlo en funcionamiento lo antes posible y perfeccionarlo de manera iterativa a partir de la información recolectada durante su uso. Se pueden ver dos grandes oportunidades de uso del sistema, la más básica, que no se basa en la información de la realidad, sería cargar todas las imágenes del conjunto de datos, de esta forma se puede utilizar el sistema como un marco de trabajo que permite automatizar la utilización de Aprendizaje Activo, donde el usuario únicamente tendrá que anotar las imágenes que aportan mayor información ahorrando tiempo de etiquetado. Por otro lado, de forma más general y teniendo en cuenta el fin con el cual fue desarrollado, está su utilización para reducir el trabajo operativo de obtener las imágenes para entrenar el modelo. Una vez entrenado el modelo con una cantidad baja de datos, se comienza con su utilización en la actividad de detección y el sistema será el encargado de seleccionar las mejores imágenes para mejorarlo. Evitando así el trabajo del personal dedicado a encontrar y tomar imágenes de los caminos de hormigas. Durante la experimentación, en cuanto a los métodos de selección de candidatos propuestos, se observó una tendencia general de mejora en todas las métricas consideradas. Si bien en algunas iteraciones específicas se registraron descensos puntuales en el desempeño, estos resultaron transitorios y fueron compensados en etapas posteriores. En conjunto, la evolución de los resultados refleja una pendiente positiva, lo que evidencia un progreso sostenido del modelo a lo largo de las iteraciones. Tanto en el caso de pruebas de ciclistas como en los

caminos de hormigas se buscó dificultar la tarea del sistema, usando imágenes con entornos muy diferentes entre las utilizadas para el entrenamiento inicial y las que fueron ingresadas al sistema. Con esto se pudo ver la capacidad del sistema de adaptar el modelo a una nueva realidad, así aumentando su capacidad de generalización.

5.1. Trabajos a futuro

Los puntos de mejoras se pueden dividir en dos, por un lado, se tienen los siguientes pasos a realizar en la detección y control de hormigas, y por otro, las mejoras que pueden ser realizadas en el sistema, que ayudan tanto a la realidad de las hormigas como a cualquier otro problema de detección que quiera ser aplicado.

5.1.1. Líneas de investigación en el control de hormigas

Al observar los proyectos en el marco del control de hormigas presentados por la Facultad de Ingeniería de la Universidad de la República, en particular el presentado por Nadile y cols. (2025), y el sistema presentado en este documento, claramente se puede observar la correlación y posible sinergia que hay entre ambos. Su forma de detección obtiene muy buenos resultados, pero los autores comentan la falta de datos existente, aquí es donde entraría en juego el sistema desarrollado, ayudando a obtener estos datos de forma más eficiente. Algo interesante, es que dicho proyecto tienen dos modelos, uno para segmentar los caminos y otro para detectar las hormigas, es perfectamente viable que las imágenes obtenidas por un agente sean almacenadas en una única base de datos, pero que dicha base sea utilizada por dos instancias del sistema aquí propuesto, ya que una misma imagen puede tener relevancia para los dos modelos y para cada instancia se considerará si es relevante o no para la siguiente iteración de cada modelo.

Por otro lado, se tienen enfoques en la detección que son diferentes pero complementarios. Nadile y cols. (2025) buscan una detección precisa y corroborar que los caminos de hormigas estén activos, por otro lado, en las pruebas que se realizaron en este documento se buscaba una detección más general y rápida, en entornos más desfavorables, dónde solo interesaba detectar la existencia o no de caminos. Un posible camino a explorar es la combinación de estas estrategias, en un principio, el agente puede realizar un escaneo a alta velocidad utilizando el modelo aquí propuesto de detección de caminos, y en caso de realizar efectivamente una detección, baja su velocidad y se acerca al camino detectado para obtener imágenes de mayor calidad, en este punto comienza a utilizar el modelo presentado por Nadile y cols. (2025) el cual es más preciso y permite identificar si efectivamente el camino está activo, pero no sería viable su utilización en todo momento debido a que tiene tiempos de procesamiento altos.

En cuanto al movimiento del agente, hay dos puntos a relevar, el primero sería el dirigirse hacia un camino, una vez detectado el camino se podría uti-

lizar una heurística similar a la propuesta por Berois y cols. (2024) de dividir la imagen en secciones y así tratar de acercarse al camino detectado pero usando el camino y no las hormigas. Luego, para seguir un camino, es interesante utilizar las versiones de modelos basados en Oriented bounding boxes ya que dentro de la información devuelta en cada predicción está la inclinación de la caja delimitadora del camino, lo cual puede ser beneficioso para determinar la orientación necesaria para que el robot pueda seguir el camino si se encuentra algún beneficio en esta práctica para el control de hormigas.

5.1.2. Líneas de investigación para la mejora del sistema

Validación de los métodos de selección de candidatos

Aunque durante este proyecto se realizaron pruebas sobre la utilización de diferentes métodos de selección de candidatos, los resultados no fueron concluyentes, inclusos hay variaciones al cambiar parámetros como la cantidad de imágenes o el conjunto de datos utilizado, sería interesante realizar pruebas sobre conjuntos de datos más variados y analizar si tienen una variación sistemática de los resultados al utilizar los métodos propuestos.

Purga de información

Durante este proyecto se trabajó exclusivamente sobre ambientes de pruebas, pero es una realidad que su utilización en entornos reales conllevaría una carga de información mucho mayor. Por esto es importante investigar formas de limpieza de los datos almacenados, para no estar consumiendo recursos en imágenes que no aportan ningún tipo de información. Por otro lado, también sería interesante investigar formas de podar paulatinamente imágenes de iteraciones anteriores minimizando la perdida de información con el fin de que las imágenes utilizadas para el entrenamiento no crezca de forma desproporcionada afectando considerablemente a los tiempos de entrenamiento de cada modelo.

Desacoplamiento de YOLO

YOLO es una gran herramienta y su utilización en la detección de objetos está muy extendida, por ahorrar tiempo de desarrollo el sistema está acoplado a esta implementación, pero sería mucho más versátil desacoplar de cualquier herramienta, con esto, si se implementan funciones para entrenar, validar y organizar los conjuntos de datos como lo requiera la herramienta, se podría usar cualquier implementación del mercado, o una arquitectura de modelo propia.

Glosario

backgrounds Imágenes que no tiene ninguna instancia del objeto a detectar por un modelo. v, viii, 38, 39, 52, 56–60, 62, 65, 67, 72, 96, 97, 102

dashcam Cámara instalada en un vehículo que graba continuamente el recorrido para registrar eventos del entorno.. 46

drivers Programas que permiten al sistema operativo comunicarse y controlar dispositivos de hardware.. 43

embedding Representación vectorial de datos complejos (como palabras o imágenes) que mantiene las relaciones y similitudes presentes en el dominio original. 22, 23, 36, 57

file system Estructura que organiza y gestiona cómo se almacenan y acceden los archivos en un dispositivo de almacenamiento.. 40, 42, 43

framework Marco de trabajo que proporciona una estructura y un conjunto de herramientas para desarrollar o resolver un tipo específico de tarea o problema.. 1, 2, 27, 41

confianza Es el valor asignado a la confianza con la que el modelo asegura que su predicción es correcta. Es un valor entre 0 y 1. viii, 19, 36–39, 52–54, 56, 59, 60, 62, 64, 65, 67, 69, 70, 74

CUDA Plataforma de cómputo paralelo y modelo de programación desarrollado por NVIDIA que permite utilizar las Unidades de Procesamiento Gráfico (GPU) para realizar tareas de propósito general.Esta arquitectura aprovecha la gran cantidad de núcleos de procesamiento de las GPU para acelerar operaciones matemáticas intensivas, especialmente útiles en áreas como el aprendizaje profundo, la visión por computadora y la simulación científica.. 14, 43

CVAT Herramienta de código abierto desarrollada por Intel para la creación y gestión de anotaciones en imágenes y videos, utilizada principalmente en tareas de visión por computadora. CVAT permite generar etiquetas, cajas delimitadoras, polígonos, máscaras y otros tipos de anotaciones necesarias para el entrenamiento de modelos de aprendizaje profundo. Ofrece una

interfaz web colaborativa, soporte para múltiples formatos de exportación (como COCO, YOLO o Pascal VOC) y funcionalidades avanzadas de automatización mediante modelos integrados o scripts personalizados.. v, 27, 30, 32, 42, 43

Docker Volume Mecanismo de Docker para almacenar datos persistentes fuera del ciclo de vida de los contenedores.. 43

Oriented bounding boxes Cajas delimitadoras utilizadas para anotar una imagen, tienen la peculiaridad de que pueden estar rotadas. 26, 69, 81

época Unidad de medida en el entrenamiento de modelos de aprendizaje profundo que corresponde a una iteración completa sobre el conjunto de datos de entrenamiento. Durante una época, todos los ejemplos del *dataset* se utilizan exactamente una vez para ajustar los parámetros del modelo mediante el algoritmo de optimización elegido. 16, 32, 49, 64

Referencias

Ahmad, M., Usman, M., Abdin, Z. U., Rizwan, M., Abbas, Z., y Anwar, H. (2023). Ag-yolo: An efficient lightweight yolo-based model for real-time crop monitoring and spraying. *Sensors*, *23*(7), 3892. Descargado de https://www.mdpi.com/1424-8220/23/7/3892 doi: 10.3390/s23073892

Ali, M. L., y Zhang, Z. (2024). The yolo framework: A comprehensive review of evolution, applications, and benchmarks in object detection. *Computers*, *13*(12), 336. Descargado de https://www.mdpi.com/2073-431X/13/12/336 doi: 10.3390/computers13120336

Almeida, D., Bertacine, T., Hernandes, y Carmona, A. (2023). Detection of atta capiguara ant nests from aerial images using random forest classifiers. En *2023 latin american robotics symposium (lars), 2023 brazilian symposium on robotics (sbr), and 2023 workshop on robotics in education (wre)* (p. 59-64). doi: 10.1109/LARS/SBR/WRE59448.2023.10333031

Antweb. (2025). *Antweb.* Descargado 2025-04-18, de https://www.antweb.org

Berois, M., De Oliveira, L., y Gastelú, E. (2024). Desarrollo de un oso hormiguero artificial. Descargado de https://gitlab.fing.edu.uy/tatiana.perera.iturralde/oso-hormiguero

Bhandari, N. (2025). *The era of oriented bounding boxes.* Descargado 2025-04-18, de https://www.namasbhandari.in/post/the-era-of-oriented-bounding-boxes

Bollazzi, M. (2014). *Hormigas cortadoras de hojas acromyrmex spp.* Descargado 2025-04-18, de https://ainfo.inia.uy/digital/bitstream/item/3370/1/inia-cartilla36-Acromyrmex.pdf (Cartilla N.º 36)

Bollazzi, M., Sabattini, J., Pilón, A. A., Vega, B., y Martínez, G. (2023). La problemÁtica del manejo de hormigas cortadoras. *Revista INIA (edición N° 74)*.

Cycler, U. (2023). *Cyclists.* https://universe.roboflow.com/cycler-vi9vn/cyclists-lt9pl/dataset/3. (Accedido: 2025-05-17)

Della Lucia, T., Gandra, L., y Guedes, R. (2014). Managing leaf-cutting ants: peculiarities, trends and challenges. *Pest Management Science*, *70*(1), 14-23. Descargado de https://scijournals.onlinelibrary.wiley.com/doi/abs/10.1002/ps.3660 doi: https://doi-org.proxy.timbo.org.uy/10.1002/ps.3660

Fisher, P., Stradling, D., y Pegler, D. (1994). Leaf cutting ants, their fungus gardens and the formation of basidiomata of leucoagaricus gongylophorus. *Mycologist*.

Li, X., y Guo, Y. (2013, June). Adaptive active learning for image classification. En *Proceedings of the ieee conference on computer vision and pattern recognition (cvpr)* (pp. 859–866). IEEE. doi: 10.1109/CVPR.2013.116

Li, X., Wang, X., Chen, X., Lu, Y., Fu, H., y Wu, Y. C. (2024). Unlabeled data selection for active learning in image classification. *Scientific Reports*, *14*(1), 1–13. Descargado de https://doi.org/10.1038/s41598-023-50598-z doi: 10.1038/s41598-023-50598-z

Marina Alma, A., Farji-Brener, A. G., y Elizalde, L. (2016). Collective response of leaf-cutting ants to the effects of wind on foraging activity. *The American Naturalist*, *188*(5), 576-581. Descargado de https://www.jstor.org/stable/10.2307/26519270

Ministerio de Ambiente de Uruguay. (2024). *Sustancias que agotan la capa de ozono (sao).* Descargado de https://www.gub.uy/ministerio-ambiente/politicas-y-gestion/sustancias-agotan-capa-ozono-saos (Accedido: 30 de junio de 2024)

Monsimet, J., Sjogersten, S., Sanders, N. J., Jonsson, M., Olofsson, J., y Siewer, M. (2024). Uav data and deep learning: efficient tools to map ant mounds and their ecological impact. *zoological Society of London*.

Nadile, G., Marr, M., y Perera, T. (2025). Active roads pipeline (arp): pipeline de detección de caminos activos. Descargado de https://gitlab.fing.edu.uy/tatiana.perera.iturralde/oso-hormiguero

Protocolo de montreal. (1987). (Disponible en https://ozone.unep.org/treaties/montreal-protocol)

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., ... Sutskever, I. (2021). Learning transferable visual models from natural language supervision. En *Proceedings of the 38th international conference on machine learning (icml)*. Descargado de https://arxiv.org/abs/2103.00020

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

Rolnick, D., Ahuja, A., Schwarz, J., Lillicrap, T., y Wayne, G. (2019). Experience replay for continual learning. En H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, y R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 32). Curran Associates, Inc. Descargado de https://proceedings.neurips.cc/paper_files/paper/2019/file/fa7cdfad1a5aaf8370ebeda47a1ff1c3-Paper.pdf

Sabattini, J. A., Sturniolo, F., Bollazzi, M., y Bugnon, L. A. (2023). Anttracker: A low-cost and efficient computer vision approach to research leaf-cutter ants behavior. *Smart Agricultural Technology*.

SemiEmptyGlass, U. (2022). *Cyclist dataset for object detection.* https://www

.kaggle.com/datasets/semiemptyglass/cyclist-dataset. (Accedido: 2025-05-17)

Siimp. (2021). *Benchmarks.* Descargado de https://github.com/siimp/benchmarks/tree/main (Último acceso: 30 de marzo de 2025)

Su, X., Shi, G., Zhong, J., Li, Y., Dai, W., Fox, G. X. G. P., ... Yan, Z. (2023). Use of artificial intelligence for automated detection and surveillance of red imported fire ants nests. , *50*(6), 0–10.

Ultralytics. (2024). *Yolo performance metrics — ultralytics.* https://docs.ultralytics.com/es/guides/yolo-performance-metrics/. (Último acceso: junio 2025)

Ultralytics. (2025). *Tips for best yolov5 training results.* Descargado de https://docs.ultralytics.com/yolov5/tutorials/tips_for_best_training_results/ (Último acceso: 30 de marzo de 2025)

United Software Associates. (2015, April). *High performance benchmarking: Mongodb and nosql systems.* White Paper.

Wigness, M., Eum, S., Rogers, J. G., Han, D., y Kwon, H. (2019). A rugd dataset for autonomous navigation and visual perception in unstructured outdoor environments. En *International conference on intelligent robots and systems (iros).*

Wu, M., Li, C., y Yao, Z. (2022). Deep active learning for computer vision tasks: Methodologies, applications, and challenges. *Applied Sciences*, *12*(16), 8103. doi: 10.3390/app12168103

Zerbino, M. (2002, 01). Cebos tóxicos para control de las hormigas cortadoras. *Revista del Plan AGROPECUARIO*, *103*, 46-49.

Anexo A

Anexo 1

A.1. Evaluación de la medida de similitud propuesta

(a) imagen base

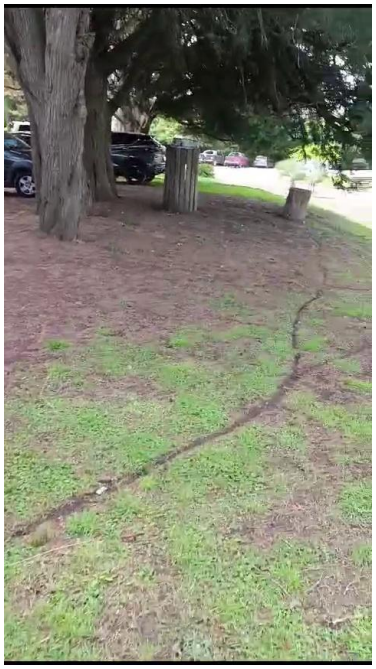
(b) similitud: 0.65

(c) similitud: 0.63

(d) similitud: 0.62

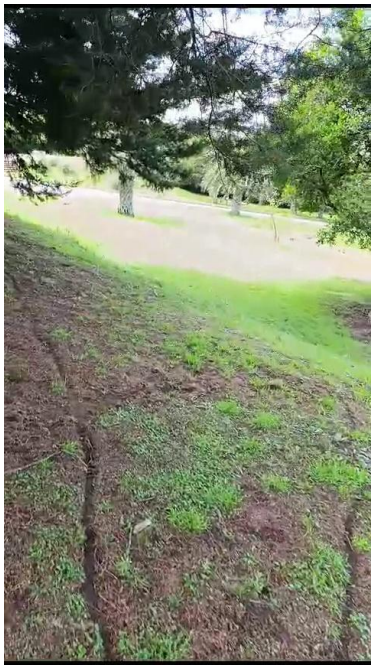
Figura A.1: Imágenes con menor similitud a la original. Obtenidas de un subconjunto aleatorio de 200 imágenes del general-ciclistas-dataset.

(a) imagen base



(b) similitud: 0.95

(c) similitud: 0.95



(d) similitud: 0.94

Figura A.2: Imágenes con mayor similitud a la original. Obtenidas de un subconjunto aleatorio de 200 imágenes del general-caminos-dataset.

(a) imagen base

(b) similitud: 0.64

(c) similitud: 0.64

(d) similitud: 0.65

Figura A.3: Imágenes con menor similitud a la original. Obtenidas de un subconjunto aleatorio de 200 imágenes del general-caminos-dataset.

(a) imagen base

(b) similitud: 0.97

(c) similitud: 0.97

(d) similitud: 0.95

Figura A.4: Imágenes con mayor similitud a la original. Obtenidas de un subconjunto aleatorio de 200 imágenes del general-caminos-dataset.

(a) imagen base

(b) similitud: 0.64

(c) similitud: 0.63

(d) similitud: 0.56

Figura A.5: Imágenes con menor similitud a la original. Obtenidas de un subconjunto aleatorio de 200 imágenes del general-caminos-dataset.

A.2. Evaluación seleccionando imágenes aleatorias

Iteración	Precisión	Recall	mAP@50	mAP@50:95	*Fitness*
1	0.8700	0.7688	0.8553	0.6648	0.6839
2	0.8893	0.8082	0.8873	0.7130	0.7304
3	0.9083	0.8383	0.9031	0.7415	0.7577
4	0.9288	0.8497	0.9161	0.7648	0.7799
5	0.9219	0.8623	0.9246	0.7790	0.7935
6	0.9353	0.8713	0.9346	0.7919	0.8062
7	0.9312	0.8859	0.9373	0.8002	0.8139
8	0.9468	0.8774	0.9417	0.8058	0.8194

Tabla A.1: Evaluación de la utilización de imágenes aleatorias, con 1000 imágenes por iteración.

A.3. Evaluación seleccionando imágenes basados unicamente en la confianza

Iteración	Precisión	Recall	mAP@50	mAP@50:95	*Fitness*
1	0.8691	0.7894	0.8619	0.6791	0.6974
2	0.8993	0.8112	0.8939	0.7257	0.7426
3	0.9069	0.8416	0.9126	0.7525	0.7685
4	0.9286	0.8586	0.9273	0.7757	0.7909
5	0.9303	0.8770	0.9364	0.7909	0.8054
6	0.9492	0.8680	0.9398	0.8020	0.8158
7	0.9260	0.8873	0.9418	0.8113	0.8243
8	0.9413	0.8827	0.9448	0.8153	0.8283

Tabla A.2: Evaluación de la utilización de imágenes basados en confianza, con 1000 imágenes por iteración.

Iteración	Precisión	Recall	mAP@50	mAP@50:95	*Fitness*
1	0.8087	0.7284	0.7990	0.5912	0.6120
2	0.8408	0.7699	0.8372	0.6458	0.6650
3	0.8673	0.7837	0.8540	0.6697	0.6881
4	0.8865	0.7959	0.8696	0.6897	0.7077
5	0.8932	0.8196	0.8860	0.7100	0.7276
6	0.9062	0.8196	0.8956	0.7270	0.7438
7	0.9180	0.8304	0.9049	0.7394	0.7559
8	0.8962	0.8515	0.9088	0.7478	0.7639
9	0.9134	0.8468	0.9141	0.7552	0.7711
10	0.9341	0.8366	0.9242	0.7677	0.7833

Tabla A.3: Evaluación de la utilización de imágenes basados en confianza, con 250 imágenes por iteración.

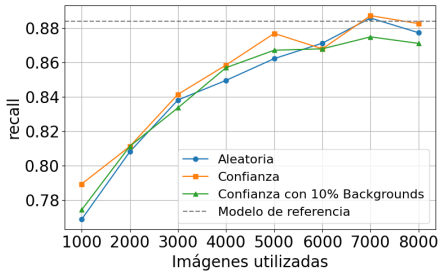
A.4. Evaluación seleccionando una cantidad mínima de *backgrounds*

Iteración	Precisión	Recall	mAP@50	mAP@50:95	*Fitness*
1	0.8826	0.7743	0.8582	0.6791	0.6970
2	0.9175	0.8113	0.8977	0.7295	0.7463
3	0.9275	0.8337	0.9112	0.7514	0.7674
4	0.9310	0.8571	0.9280	0.7735	0.7889
5	0.9434	0.8672	0.9374	0.7911	0.8058
6	0.9409	0.8680	0.9374	0.7916	0.8061
7	0.9400	0.8750	0.9434	0.8072	0.8208
8	0.9507	0.8712	0.9481	0.8175	0.8306

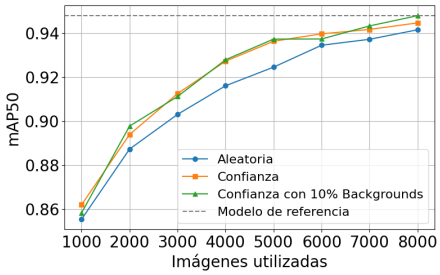
Tabla A.4: Evaluación de la utilización de 10 % de *backgrounds* por iteración, con 1000 imágenes por iteración.

Iteración	Precisión	Recall	mAP@50	mAP@50:95	*Fitness*
1	0.8127	0.7336	0.8049	0.5970	0.6178
2	0.8607	0.7678	0.8387	0.6418	0.6615
3	0.8554	0.8045	0.8608	0.6739	0.6926
4	0.8913	0.7914	0.8666	0.6898	0.7075
5	0.9010	0.8081	0.8863	0.7108	0.7283
6	0.8730	0.8371	0.8905	0.7207	0.7377
7	0.9154	0.8301	0.9037	0.7350	0.7518
8	0.9066	0.8377	0.9079	0.7424	0.7590
9	0.9146	0.8281	0.9124	0.7477	0.7641
10	0.9010	0.8568	0.9168	0.7565	0.7726

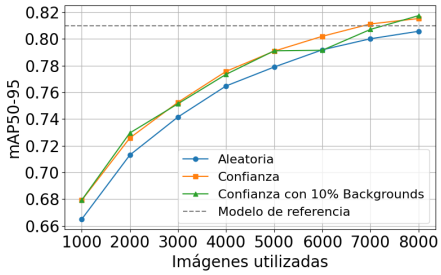
Tabla A.5: Evaluación de la utilización de 10 % de *backgrounds* por iteración, con 250 imágenes por iteración.



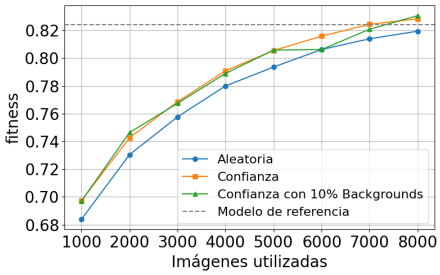
(a) Comparación de *recall*



(b) Comparación de *mAP@0.50*



(c) Comparación de *mAP@0.50:0.95*



(d) Comparación de *fitness*

Figura A.6: Comparación de métricas de rendimientos para los modelos usando confianza, selección aleatoria y confianza más 10 % de *backgrounds*.

A.5. Evaluación seleccionando imágenes similares al conjunto

Iteración	Precisión	Recall	mAP@50	mAP@50:95	*fitness*
1	0.8778	0.7751	0.8532	0.6690	0.6875
2	0.9065	0.8177	0.8882	0.7153	0.7326
3	0.9289	0.8298	0.9067	0.7451	0.7613
4	0.9179	0.8558	0.9233	0.7690	0.7844
5	0.9370	0.8606	0.9284	0.7783	0.7933
6	0.9382	0.8745	0.9403	0.7962	0.8106
7	0.9385	0.8823	0.9416	0.8087	0.8220
8	0.9248	0.8937	0.9411	0.8099	0.8230

Tabla A.6: Evaluación de la utilización de similitud por iteración para $\beta = 0{,}5$ con 1000 imágenes por iteración.

Iteración	Precisión	Recall	mAP@50	mAP@50:95	*Fitness*
1	0.9031	0.7553	0.8601	0.6771	0.6954
2	0.9175	0.8018	0.8892	0.7208	0.7376
3	0.9189	0.8445	0.9051	0.7475	0.7632
4	0.9257	0.8501	0.9209	0.7607	0.7767
5	0.9268	0.8768	0.9367	0.7898	0.8045
6	0.9439	0.8767	0.9422	0.8025	0.8165
7	0.9272	0.8977	0.9449	0.8100	0.8235
8	0.9281	0.9047	0.9480	0.8184	0.8314

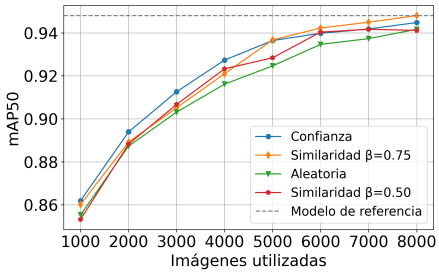
Tabla A.7: Evaluación de la utilización de similitud por iteración para $\beta = 0{,}75$ con 1000 imágenes por iteración.

Iteración	Precisión	Recall	mAP@50	mAP@50:95	*Fitness*
1	0.8502	0.7143	0.8033	0.5932	0.6142
2	0.8577	0.7405	0.8243	0.6264	0.6462
3	0.8655	0.7800	0.8414	0.6535	0.6723
4	0.8698	0.7992	0.8588	0.6787	0.6967
5	0.9007	0.7890	0.8674	0.6937	0.7110
6	0.8884	0.8170	0.8826	0.7130	0.7300
7	0.8960	0.8175	0.8844	0.7174	0.7341
8	0.9064	0.8206	0.8947	0.7321	0.7484
9	0.9029	0.8338	0.9012	0.7413	0.7573
10	0.8940	0.8538	0.9098	0.7536	0.7692

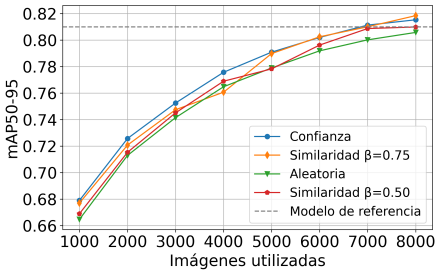
Tabla A.8: Evaluación de la utilización de similitud por iteración para $\beta = 0{,}5$ con 250 imágenes por iteración.

Iteración	Precisión	Recall	mAP@50	mAP@50:95	*Fitness*
1	0.8150	0.7568	0.8050	0.5938	0.6149
2	0.8480	0.7890	0.8437	0.6428	0.6629
3	0.8544	0.7947	0.8485	0.6599	0.6788
4	0.8700	0.7959	0.8665	0.6885	0.7063
5	0.8831	0.8081	0.8750	0.7046	0.7216
6	0.8804	0.8155	0.8795	0.7126	0.7293
7	0.8819	0.8301	0.8924	0.7291	0.7454
8	0.9154	0.8150	0.8954	0.7341	0.7502
9	0.8939	0.8428	0.9067	0.7451	0.7613
10	0.9190	0.8366	0.9103	0.7549	0.7705

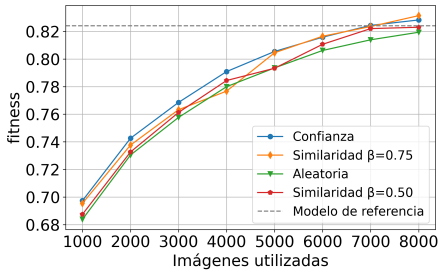
Tabla A.9: Evaluación de la utilización de similitud por iteración para $\beta = 0{,}75$ con 250 imágenes por iteración.



(a) Comparación de *mAP@50*



(b) Comparación de *mAP@0.50:0.95*



(c) Comparación de *Fitness*

A.6. Evaluación seleccionando imágenes disimilares al conjunto de entrenamiento

Iteración	Precisión	Recall	mAP@50	mAP@50:95	*Fitness*
1	0.8326	0.7890	0.8442	0.6442	0.6642
2	0.8910	0.8255	0.8894	0.7115	0.7293
3	0.9164	0.8348	0.9061	0.7363	0.7532
4	0.9178	0.8619	0.9245	0.7704	0.7858
5	0.9237	0.8802	0.9355	0.7877	0.8025
6	0.9439	0.8711	0.9405	0.7973	0.8116
7	0.9482	0.8733	0.9434	0.8077	0.8212
8	0.9327	0.8859	0.9451	0.8116	0.8249

Tabla A.10: Evaluación de la utilización de disimilitud por iteración para $\beta = 0{,}5$ con 1000 imágenes por iteración.

Iteración	Precisión	Recall	mAP@50	mAP@50:95	*Fitness*
1	0.8597	0.8029	0.8641	0.6697	0.6892
2	0.9111	0.8159	0.8955	0.7189	0.7366
3	0.9217	0.8440	0.9156	0.7530	0.7693
4	0.9234	0.8603	0.9270	0.7725	0.7880
5	0.9220	0.8697	0.9348	0.7846	0.7996
6	0.9294	0.8847	0.9404	0.7965	0.8109
7	0.9437	0.8808	0.9454	0.8088	0.8225
8	0.9464	0.8845	0.9475	0.8155	0.8287

Tabla A.11: Evaluación de la utilización de disimilitud por iteración para $\beta =$ 0,75 con 1000 imágenes por iteración.

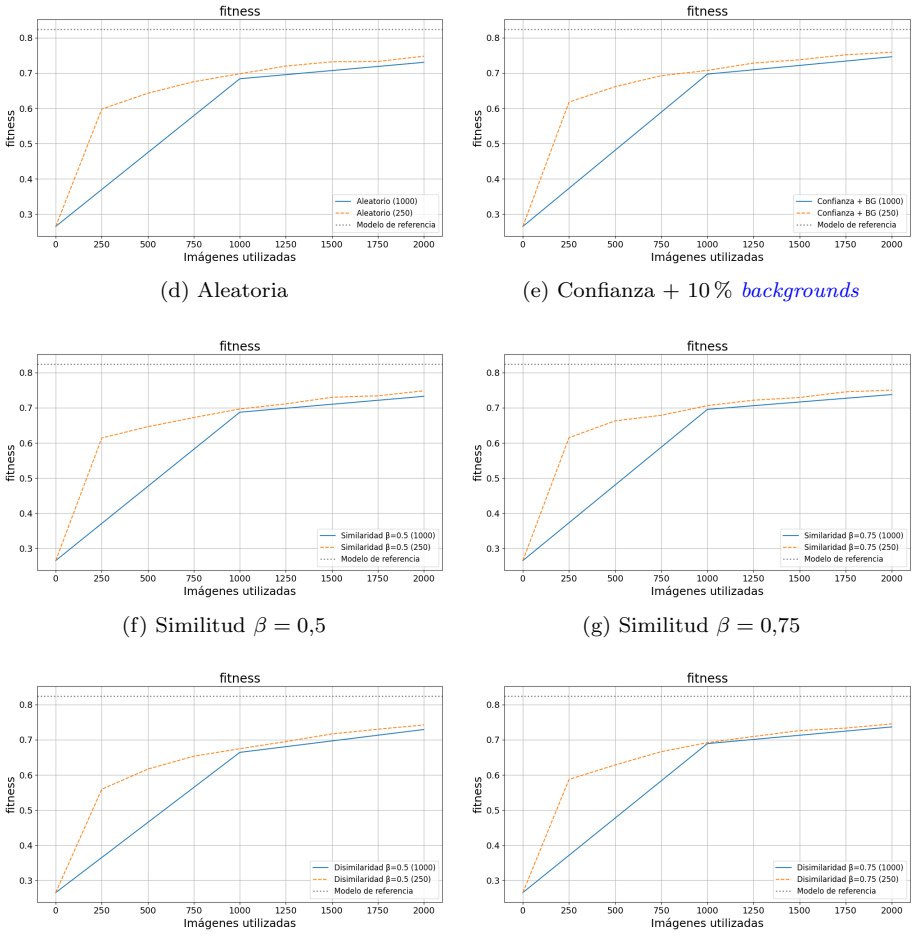
Iteración	Precisión	Recall	mAP@50	mAP@50:95	*Fitness*
1	0.8167	0.6855	0.7657	0.5365	0.5594
2	0.8259	0.7421	0.8109	0.5950	0.6165
3	0.8441	0.7650	0.8359	0.6330	0.6533
4	0.8509	0.7856	0.8475	0.6555	0.6747
5	0.8689	0.7911	0.8595	0.6766	0.6949
6	0.8965	0.7937	0.8769	0.6989	0.7167
7	0.8932	0.8141	0.8852	0.7129	0.7301
8	0.9022	0.8155	0.8938	0.7254	0.7423
9	0.9142	0.8143	0.9034	0.7390	0.7555
10	0.9000	0.8398	0.9088	0.7468	0.7630

Tabla A.12: Evaluación de la utilización de disimilitud por iteración para $\beta = 0{,}5$ con 250 imágenes por iteración.

Iteración	Precisión	Recall	mAP@50	mAP@50:95	*Fitness*
1	0.8134	0.7063	0.7884	0.5649	0.5872
2	0.8405	0.7449	0.8235	0.6068	0.6284
3	0.8504	0.7809	0.8472	0.6462	0.6663
4	0.8521	0.7992	0.8591	0.6733	0.6918
5	0.8934	0.7878	0.8711	0.6915	0.7094
6	0.8876	0.8167	0.8829	0.7087	0.7261
7	0.9133	0.7894	0.8817	0.7168	0.7333
8	0.9032	0.8191	0.8960	0.7286	0.7453
9	0.8976	0.8316	0.8976	0.7357	0.7519
10	0.9072	0.8395	0.9052	0.7403	0.7568

Tabla A.13: Evaluación de la utilización de disimilitud por iteración para $\beta =$ 0,75 con 250 imágenes por iteración.

A.7. Evaluación comparativa de iteraciones con distintas cantidades de imágenes



(d) Aleatoria

(e) Confianza + 10 % *backgrounds*

(f) Similitud $\beta = 0{,}5$

(g) Similitud $\beta = 0{,}75$

(h) Disimilitud $\beta = 0{,}5$

(i) Disimilitud $\beta = 0{,}75$

Figura A.7: Comparación de métricas de rendimientos para los modelos usando confianza, selección aleatoria y confianza más 10 % de *backgrounds*.

A.8. Experimentación sobre caminos de hormigas

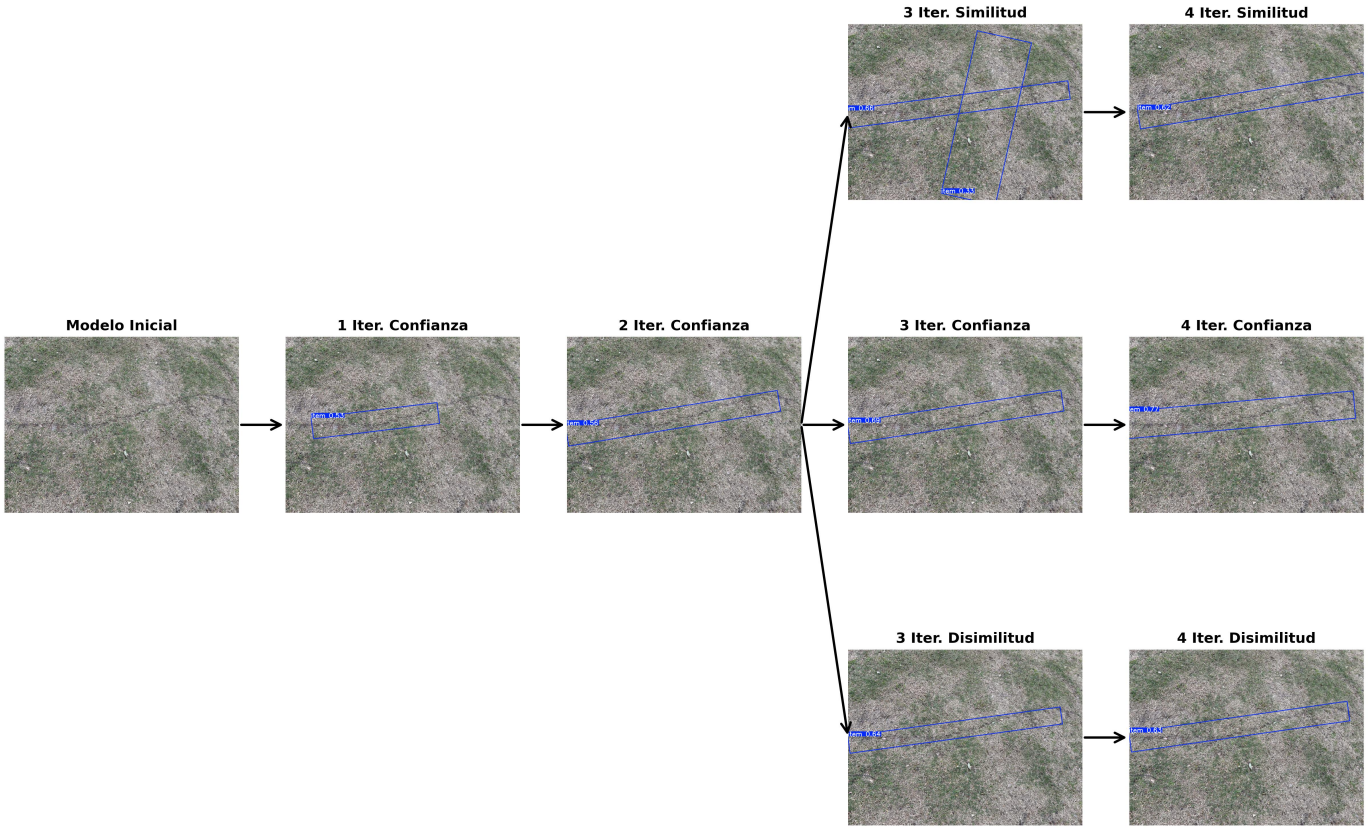


Figura A.8: Evolución visual del modelo sobre imagen 1.

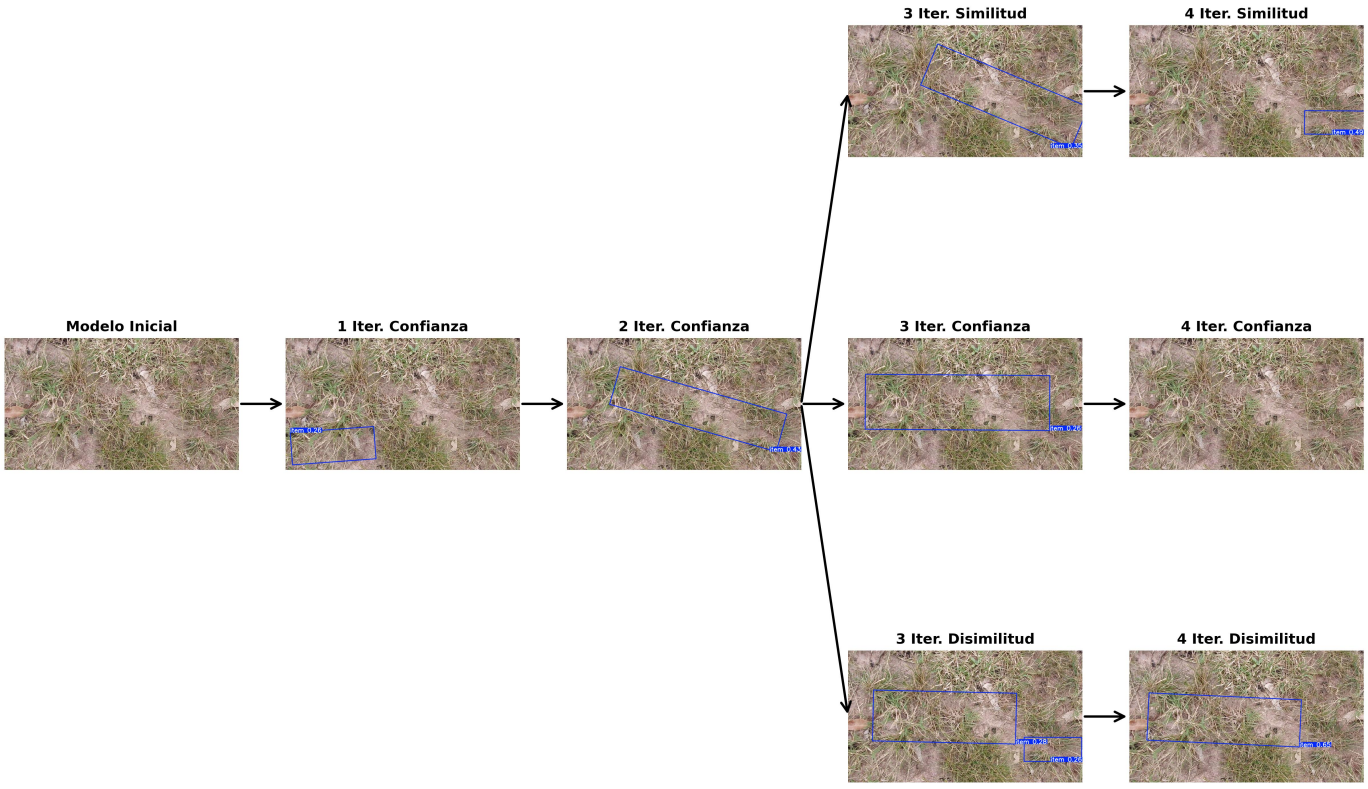


Figura A.9: Evolución visual del modelo sobre imagen 2.

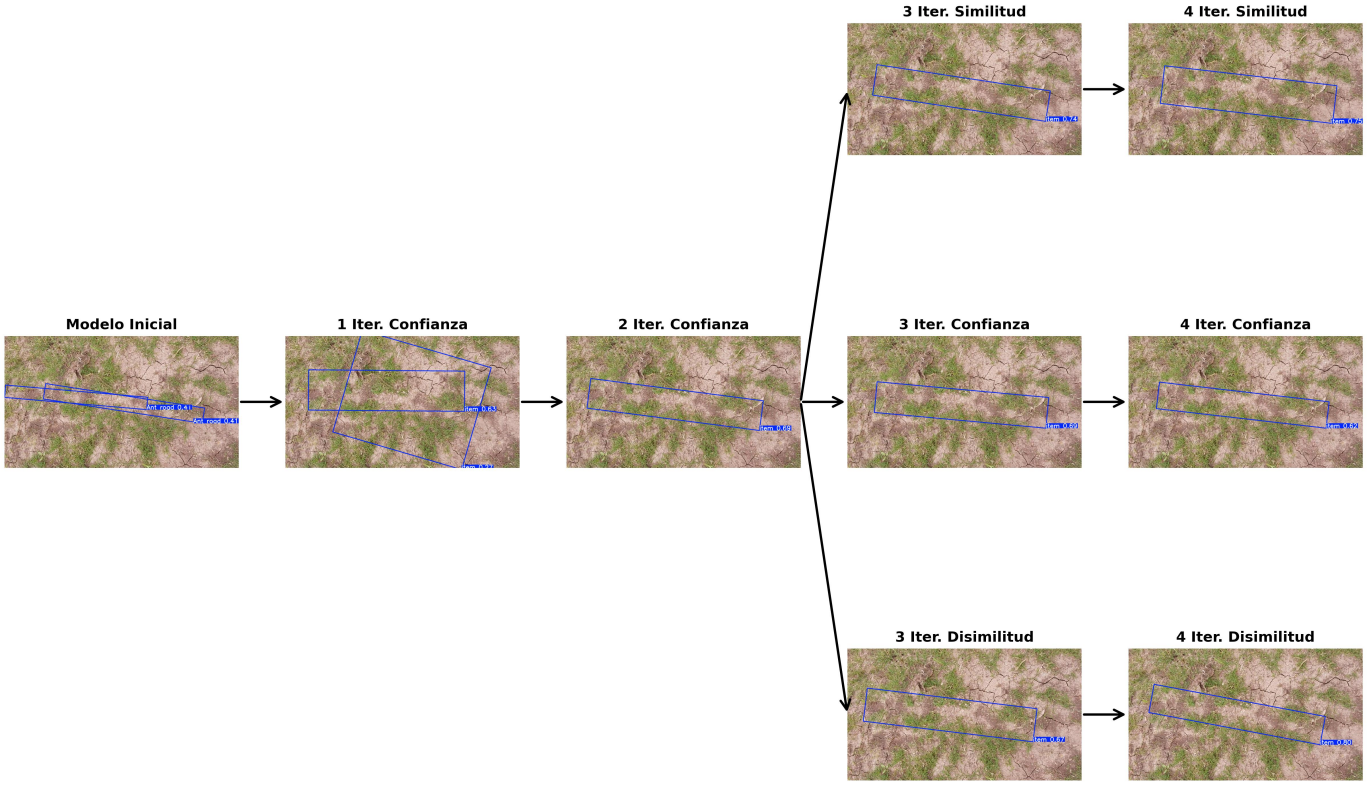


Figura A.10: Evolución visual del modelo sobre imagen 3.

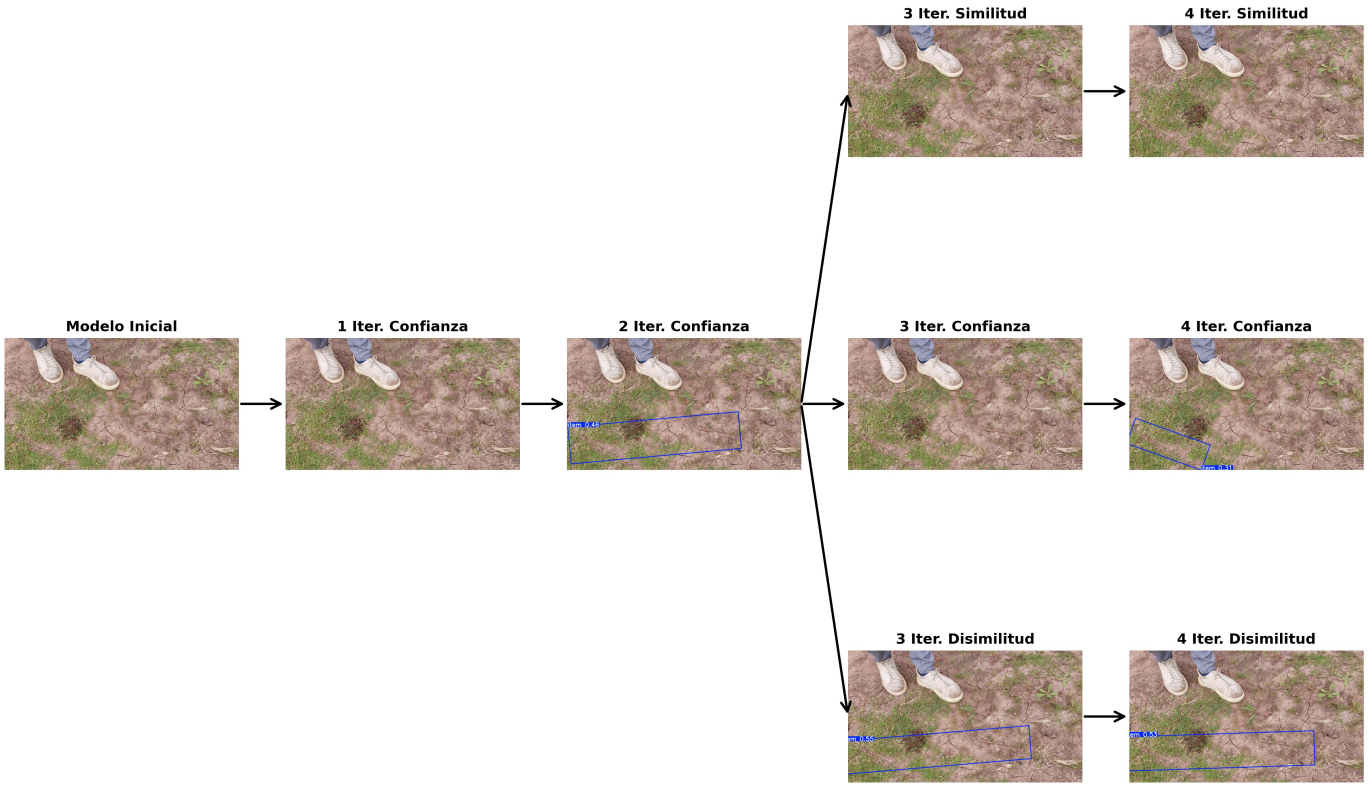


Figura A.11: Evolución visual del modelo sobre imagen 4.

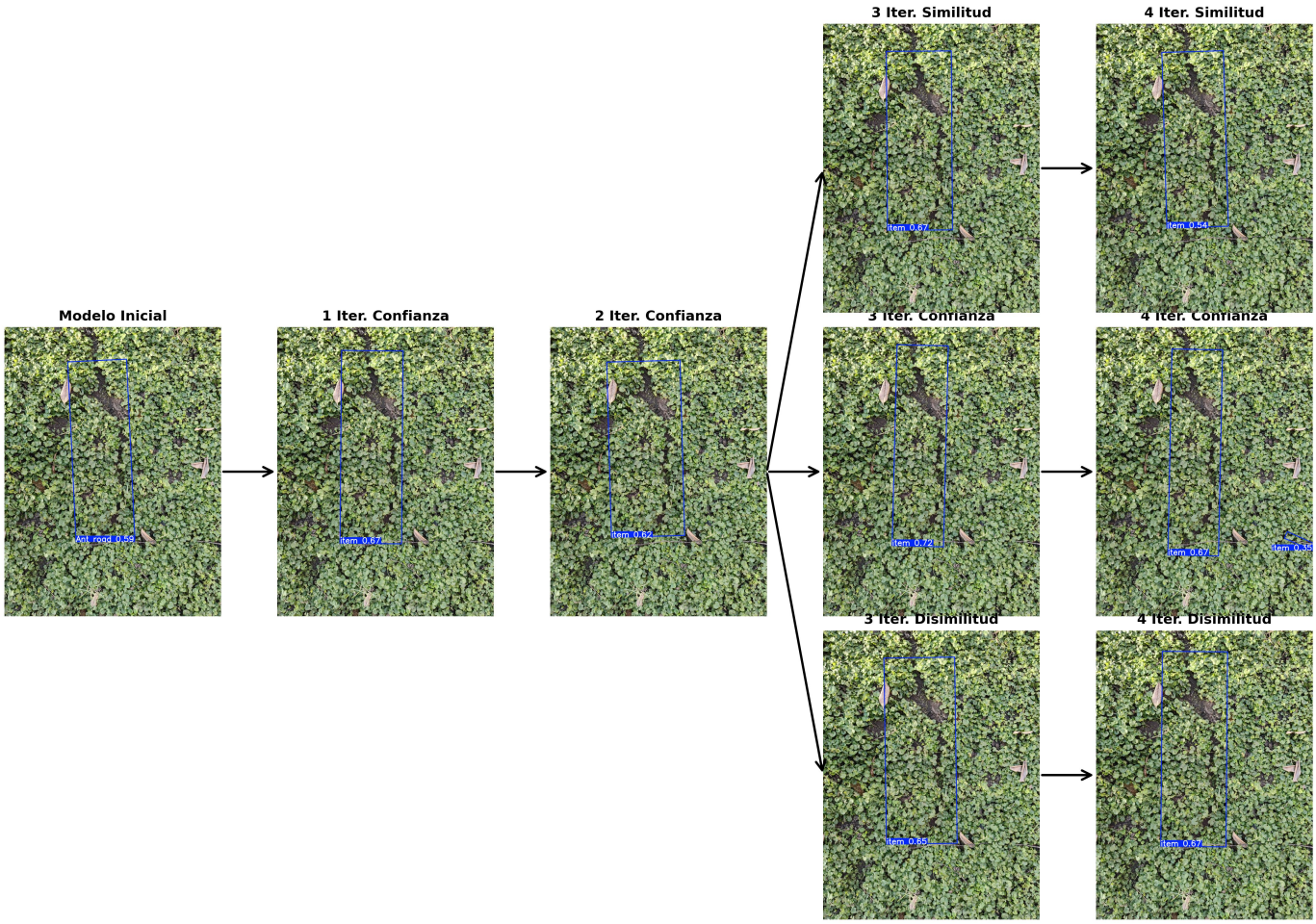


Figura A.12: Evolución visual del modelo sobre imagen 5.

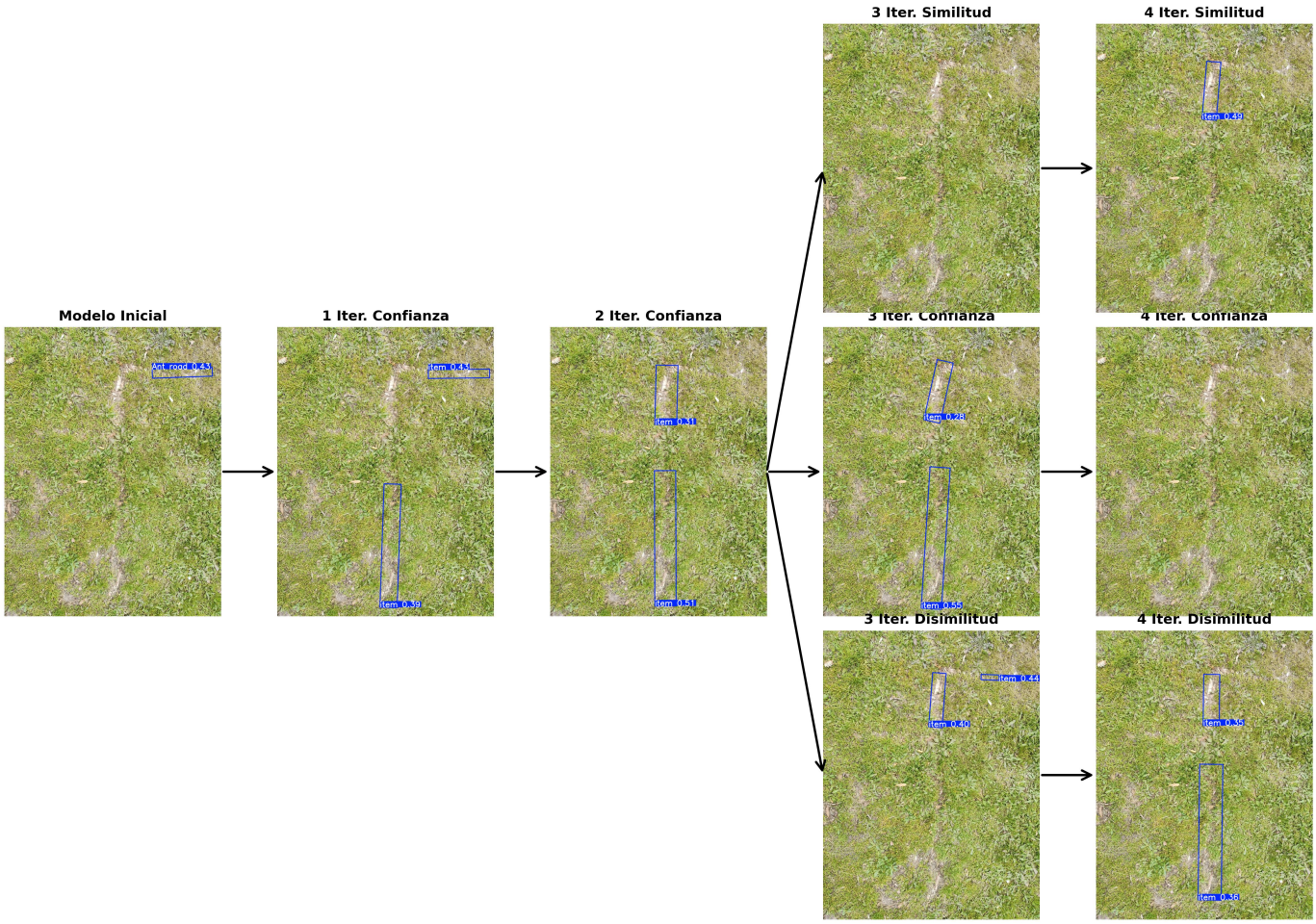


Figura A.13: Evolución visual del modelo sobre imagen 6.

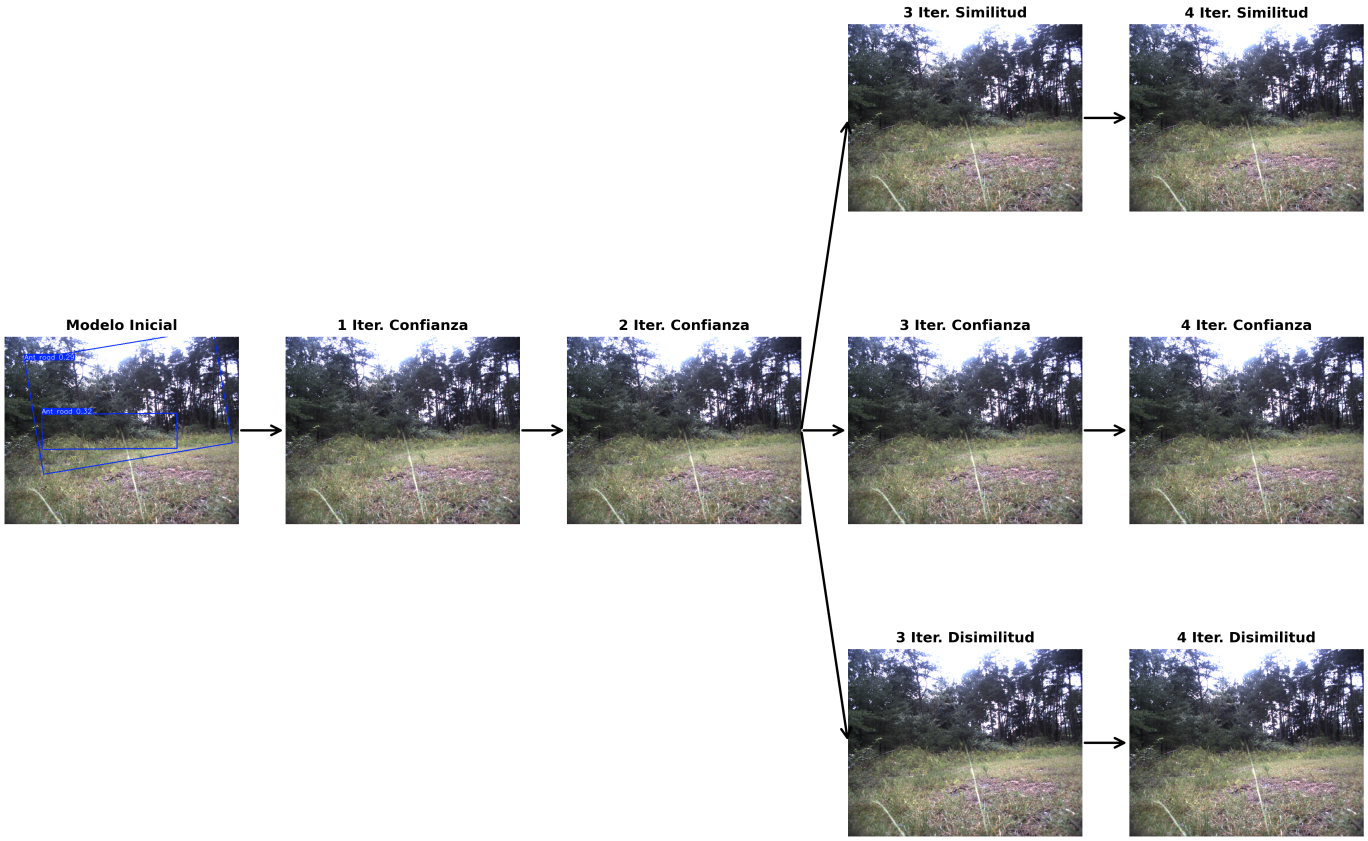


Figura A.14: Evolución visual del modelo sobre imagen 7.

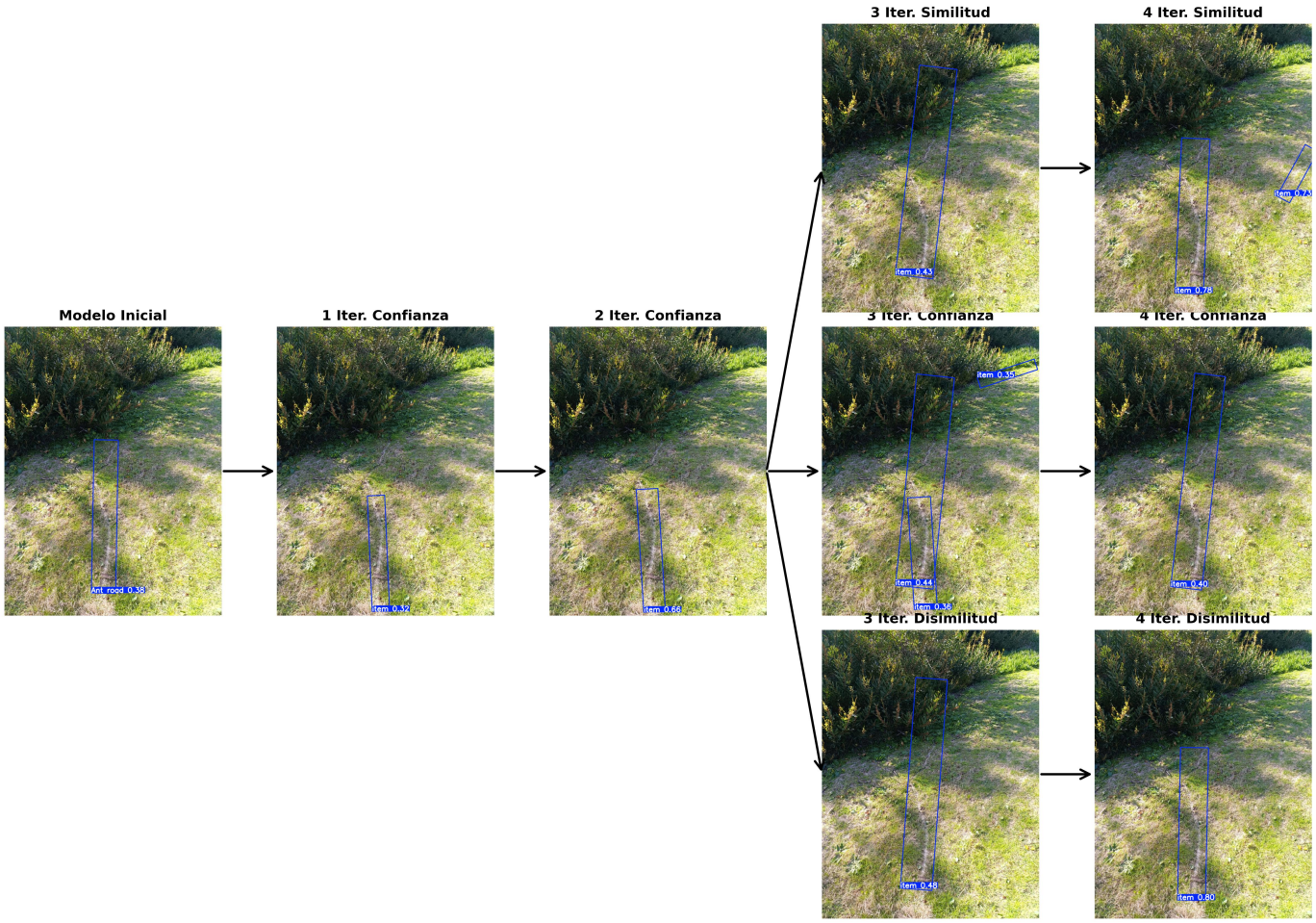


Figura A.15: Evolución visual del modelo sobre imagen 8.

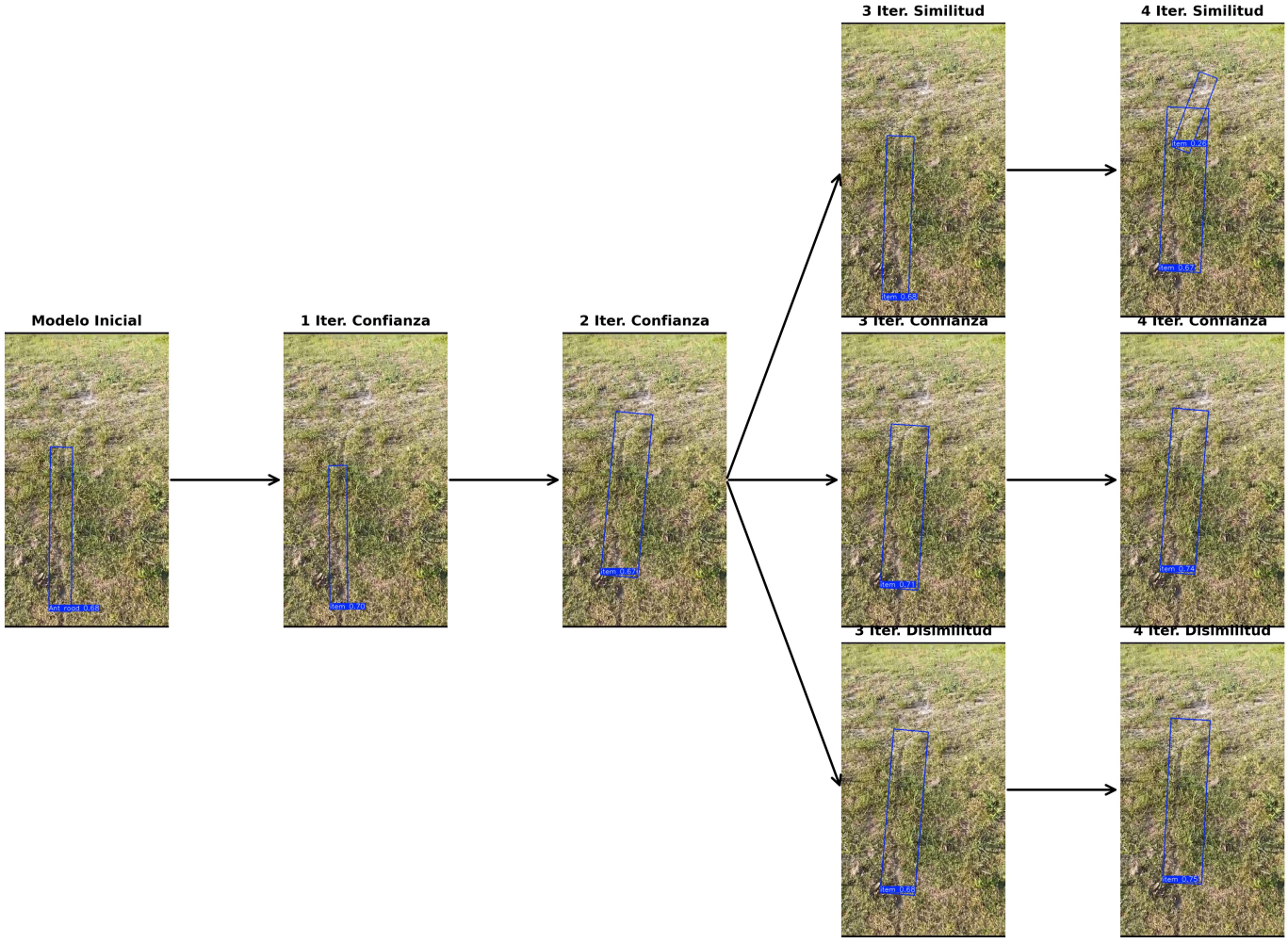


Figura A.16: Evolución visual del modelo sobre imagen 9.

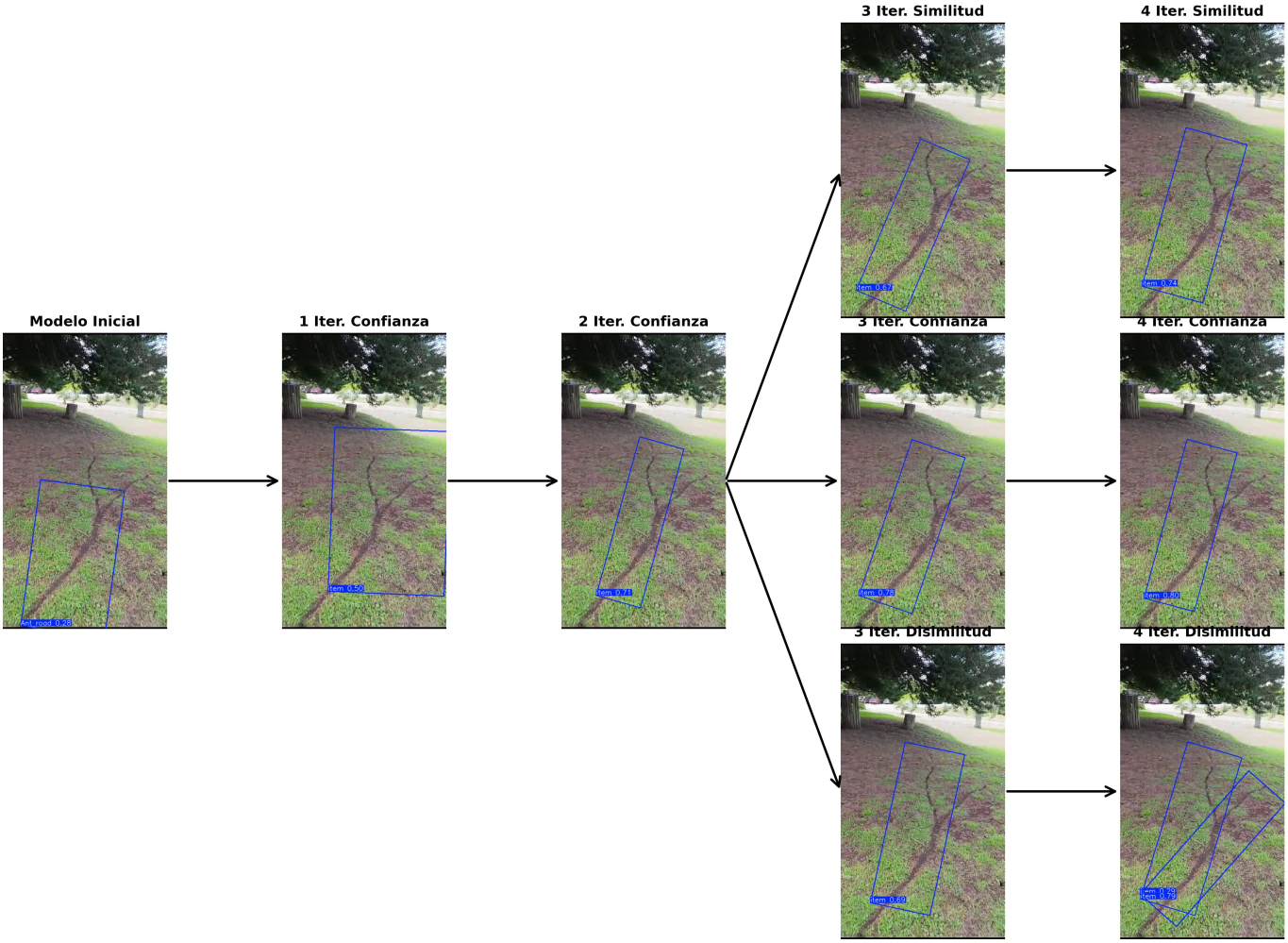


Figura A.17: Evolución visual del modelo sobre imagen 10.

Estrategia	**Iteración**	**Precision**	**Recall**	***mAP@50***	***mAP@0.50:0.95***	**Fitness**
Confianza	1	0.5740	0.4906	0.4806	0.1921	0.2209
Confianza	2	0.7482	0.5791	0.6235	0.2764	0.3111
Confianza	3	0.7459	0.6205	0.6663	0.2976	0.3345
Confianza	4	0.7306	0.6154	0.6330	0.3096	0.3419
Similitud	3	0.6915	0.5747	0.5440	0.2321	0.2633
Similitud	4	0.6785	0.6170	0.5917	0.2534	0.2872
Disimilitud	3	0.7041	0.6103	0.6366	0.2595	0.2972
Disimilitud	4	0.7385	0.6371	0.6649	0.2895	0.3271

Tabla A.14: Resultados por estrategia e iteración.