



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



**TFG del Grado en Ingeniería
Informática**

**Detección de insectos
mediante Inteligencia
Artificial**



Presentado por Arturo Carretero Mateo
en Universidad de Burgos — 9 de junio de 2024
Tutor: Carlos Cambra Baseca

Índice general

Índice general	i
Índice de figuras	iii
Índice de tablas	iv
Apéndice A Plan de Proyecto Software	1
A.1. Introducción	1
A.2. Planificación temporal	2
A.3. Estudio de viabilidad	7
Apéndice B Especificación de Requisitos	11
B.1. Introducción	11
B.2. Objetivos generales	12
B.3. Catálogo de requisitos	13
B.4. Especificación de requisitos	14
Apéndice C Especificación de diseño	23
C.1. Introducción	23
C.2. Diseño de datos	23
C.3. Diseño procedimental	27
C.4. Diseño Arquitectónico	32
Apéndice D Documentación técnica de programación	35
D.1. Introducción	35
D.2. Estructura de directorios	36
D.3. Manual del programador	39

D.4. Compilación, instalación y ejecución del proyecto	43
D.5. Pruebas del sistema	44
Apéndice E Documentación de usuario	47
E.1. Introducción	47
E.2. Requisitos de usuarios	48
E.3. Instalación	49
E.4. Manual del Usuario	51
Apéndice F Anexo de sostenibilización curricular	57
Bibliografía	61

Índice de figuras

A.1. Gráfico de horas empleadas	4
A.2. Porcentajes de horas empleadas	4
A.3. Gráfico de horas empleadas en las distintas fases	5
A.4. Distribución de las fases del proyecto	6
B.1. Diagrama de casos de uso	14
C.1. Ejemplo de resultados producidos durante el entrenamiento . . .	25
C.2. Ejemplo de identificación de insectos tras entrenar el modelo . .	26
C.3. Comunicación Cliente-Servidor Entrenamiento	29
C.4. Comunicación Detección	31
D.1. Diagrama de las estructuras de directorios	36
D.2. Vista por defecto de Anaconda	41
D.3. Debugger en Spyder	44
D.4. Debugger en Spyder	44
E.1. Despliegue Flask en dirección 127.0.0.1 y puerto 5000	51
E.2. Pantalla Inicial de la aplicación	52
E.3. Información del proceso de entrenamiento	54
E.4. Ejemplo de visualización de conteo y sus respectivos Bounding Boxes	56

Índice de tablas

A.1. Costes del proyecto.	8
A.2. Resumen de la licencia MIT	9
B.1. CU-1 Adecuar Datos.	15
B.2. CU-2 Carga de Imágenes.	16
B.3. CU-3 Establecer Proporciones de Datos de Entrenamiento.	17
B.4. CU-4 Entrenamiento del Modelo.	18
B.5. CU-5 Procesamiento de Imágenes.	19
B.6. CU-6 Predicción de Imagen por el Modelo.	20
B.7. CU-7 Mostrar Resultados y Ajustes.	21

Apéndice A

Plan de Proyecto Software

A.1. Introducción

La planificación meticulosa de proyectos de software se erige como un pilar fundamental para el éxito y la entrega oportuna de soluciones innovadoras. Este documento presenta el plan de proyecto para realizar un estudio sobre la detección y control de plagas.

El propósito de este plan es establecer un marco detallado que guíe las fases de concepción, diseño, implementación y despliegue del software. Se busca no solo asegurar la alineación con las necesidades del software y sus objetivos, sino también maximizar la eficiencia de los procesos y la calidad del producto final. La planificación estratégica abordará componentes críticos como la definición de alcance, la gestión de riesgos, la asignación de recursos y los cronogramas de desarrollo.

A.2. Planificación temporal

Antes de comenzar con el desarrollo del proyecto, fueron planteadas las distintas posibles metodologías que se utilizarían para la planificación del desarrollo de este. Tras realizarse un estudio de cuáles son las metodologías más capaces para este, se destacaron las siguientes:

- **Programación extrema.** Centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software. [2]
- **Kanban.** Gestiona un óptimo flujo de trabajo dentro del proceso. [7]
- **Scrum.** Proceso empírico, iterativo e incremental. [2]
- **Desarrollo Cascada.** Secuencia de fases, que al final de cada etapa reúne toda la documentación. [6]

Llegando a la conclusión de que la más favorable a utilizar es Kanban debido a su buena representación mediante el uso de tableros y a la mejora continua en cuanto al ajuste del trabajo de forma regular.

Para la estructuración del trabajo utilizaremos:

- **Epics.** Esta es una historia de usuario de gran tamaño o alta granularidad y que tiene, por lo tanto, mayor grado de incertidumbre.
- **Historias de usuario.** Describen, en una o dos frases, una funcionalidad de software desde el punto de vista del usuario, con el lenguaje que este emplearía. [5]

En total, fueron implementados 8 epics, los cuales fueron:

1. **Inicio del proyecto.** Este se enfocará en las tareas previas de organización y puesta en marcha del proyecto, como establecer los objetivos, la selección de metodologías y herramientas, y la asignación de recursos.
2. **Configuración del entorno de desarrollo.** Este epic abarca las actividades relacionadas con la configuración del entorno de desarrollo, así como la instalación del IDE, librerías y otras herramientas necesarias para el desarrollo del proyecto.
3. **Desarrollo del modelo.** Este epic se enfoca en la creación y desarrollo del modelo de detección de insectos, incluyendo la implementación de algoritmos y técnicas necesarias para la correcta detección.
4. **Entrenamiento del modelo.** En este epic, se lleva a cabo el proceso de entrenamiento del modelo de detección de insectos, así como el establecimiento de los conjuntos de datos dedicados al entrenamiento, validación y pruebas.
5. **Observación de los resultados.** Implica la evaluación y estudio de los resultados obtenidos por el modelo entrenado, analizando su precisión y certeza en la detección de los insectos.
6. **Afinamiento y ajuste.** En este epic, se realizan ajustes y mejoras en el modelo y en el proceso de detección de insectos, basados en los resultados obtenidos.
7. **Elaboración de la App Web.** Este epic se enfoca en la creación y desarrollo de la interfaz web para el proyecto, que incluirá la visualización de resultados, la interacción con el usuario y otras funcionalidades relacionadas.
8. **Documentación.** Finalmente, este epic implica la elaboración de la documentación del proyecto.

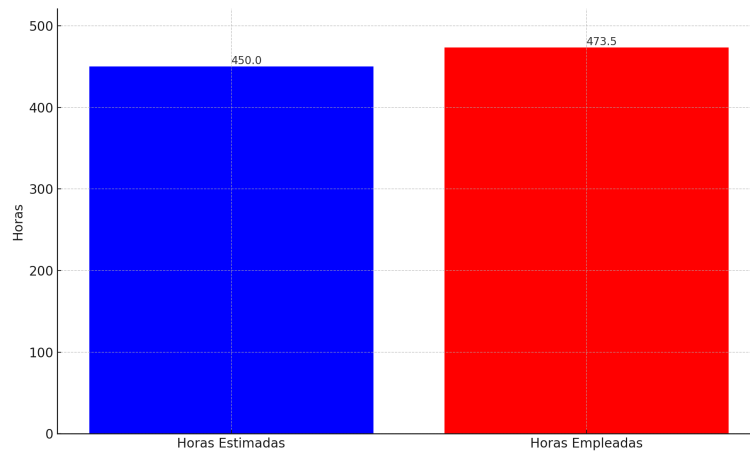


Figura A.1: Gráfico de horas empleadas

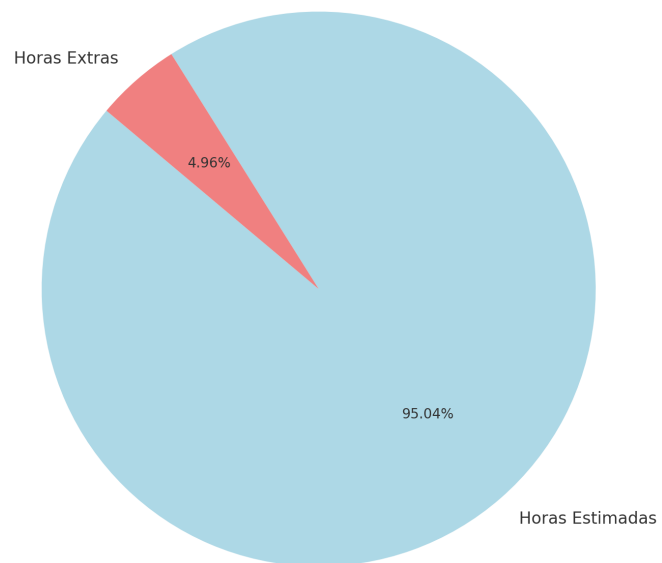


Figura A.2: Porcentajes de horas empleadas

En cuanto a las horas empleadas en el proyecto, estos son los porcentajes correspondientes a cada uno de los distintos campos del proyecto:

- 10 % Planificación previa. Engloba el inicio del proyecto y la configuración del entorno de desarrollo.
- 30 % Elaboración del desarrollo del modelo.
- 10 % Refinamiento del modelo. Engloba la observación de los resultados y el afinamiento.
- 15 % Elaboración web.
- 35 % Documentación.

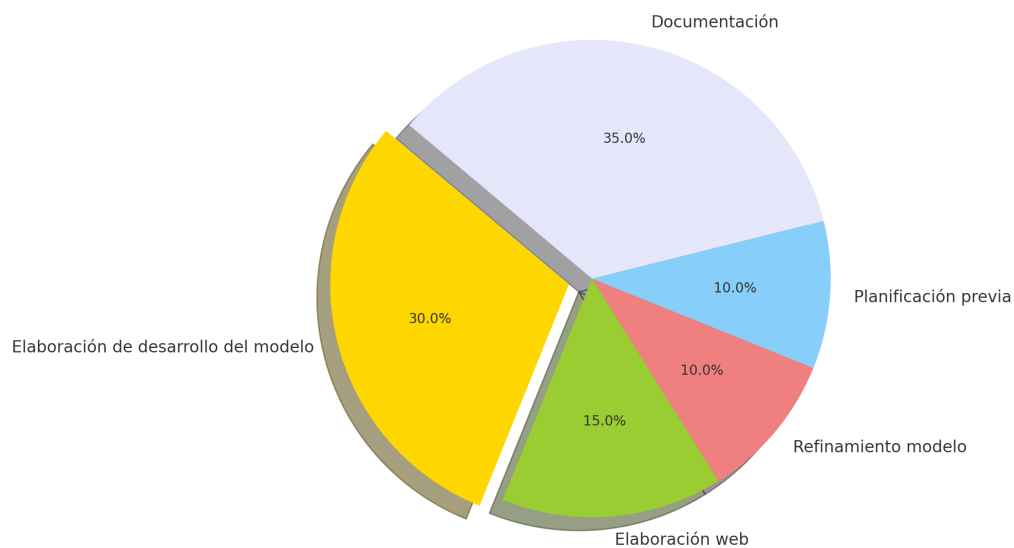


Figura A.3: Gráfico de horas empleadas en las distintas fases

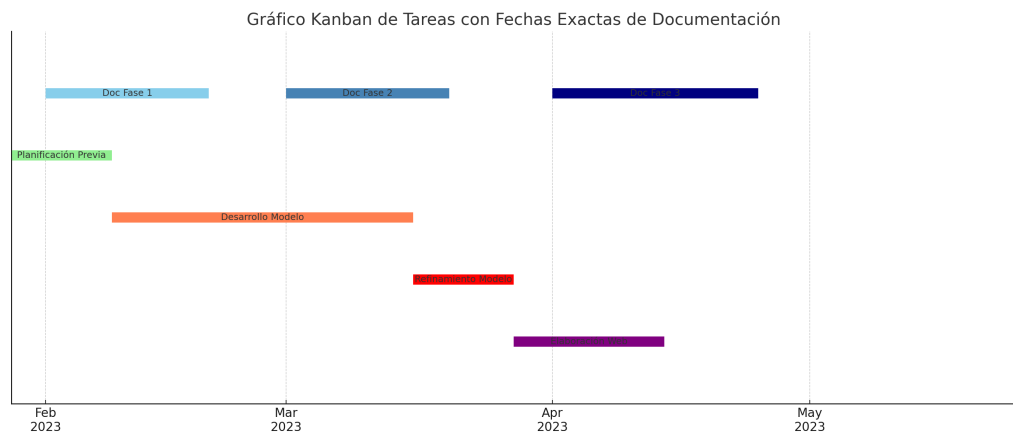


Figura A.4: Distribución de las fases del proyecto

La documentación se creó en etapas, mientras que el desarrollo del modelo y el afinamiento se realizaron sin pausas.

A.3. Estudio de viabilidad

Viabilidad económica

La viabilidad económica es un factor determinante para la realización de este proyecto. Basándonos en ella, podremos determinar si el proyecto resultará exitoso o no, en función de si el coste total es superior o inferior al presupuesto estimado. Los costes del proyecto se clasificarán en cuatro categorías: directos, indirectos, fijos y variables.

Costes fijos Son costes que permanecen constantes a lo largo del desarrollo del proyecto, tales como:

- Alquiler del espacio de oficina.
- Salarios del personal.

Costes variables Estos costes fluctúan con el progreso del proyecto y pueden incluir:

- Estrategias de marketing y publicidad.
- Facturas de servicios como la electricidad, el agua y el gas.

Costes directos Son los costes que están vinculados directamente con las actividades del proyecto:

- Compra de trampas para insectos.
- Adquisición de cámaras y otros equipos de monitoreo.

Costes indirectos Estos costes no están relacionados directamente con las actividades diarias del proyecto, pero son necesarios para su soporte:

- Mantenimiento y reparación de equipos.

Tabla A.1: Costes del proyecto.

CONCEPTO	CANTIDAD
Alquiler oficina	1200€
Salarios	5500,36€
Marketing	120€
Electricidad	312,54€
Agua	87,10€
Gas	95,02€
Trampas	80€
Cámaras	3200€
Mantenimiento de equipos	100€
GPU (GeForce RTX 3060)	600€
TOTAL	11295,02€

Beneficios

El sistema de detección de insectos mediante IA y segmentación de imágenes ofrecerá una gama de beneficios en distintos niveles de aplicación y uso.

- **Básico:** Este plan es gratuito y satisface las necesidades básicas de los usuarios interesados en la identificación de insectos para aplicaciones no comerciales. Permite un número limitado de consultas al sistema de IA, ideal para pequeñas investigaciones o estudios de campo iniciales.
- **Profesional:** Orientado a usuarios que requieren un mayor volumen de análisis, como investigadores académicos o empresas agrícolas. Incluye un número mayor de consultas así como soporte técnico avanzado. El precio será ajustado en función de los requerimientos específicos.
- **Empresarial:** Para grandes corporaciones o proyectos de investigación intensiva que requieran un uso exhaustivo del servicio.

Viabilidad legal

Licencia del proyecto Desde el inicio de nuestro proyecto, nuestra intención ha sido ofrecer el software bajo una licencia de Software Libre. Hemos considerado varias licencias y hemos elegido la licencia MIT por su claridad y amplitud, lo que promueve la colaboración y uso extenso del software de código abierto.

La licencia MIT es ampliamente reconocida por su flexibilidad y mínimas restricciones sobre la redistribución, lo que está en línea con nuestra visión de permitir un uso libre y amplio del proyecto por parte de la comunidad. Después de evaluar diversas opciones, concluimos que la licencia MIT era la más adecuada para nuestro proyecto, dada su simplicidad y la libertad que ofrece tanto a desarrolladores como a usuarios.

Optamos por la licencia MIT debido a su:

- **Permisividad:** Permite un uso extenso y variado del software, incluyendo aplicaciones comerciales.
- **Simplicidad:** Su texto breve y conciso evita complicaciones legales y es fácilmente entendible.
- **Fomento de la colaboración:** Es una licencia que anima a la mejora y contribución comunitaria.

Presentamos a continuación un resumen de los términos de la licencia MIT:

Permisos	Limitaciones	Condiciones
Uso comercial Modificación Distribución Uso privado	Sin responsabilidad Sin garantía	Aviso de licencia y derechos de autor

Tabla A.2: Resumen de la licencia MIT

La licencia completa se puede encontrar en el sitio web de Open Source Initiative en la siguiente URL: <https://opensource.org/licenses/MIT>.

Elegimos la licencia MIT porque:

- Facilita la difusión y utilización del software al no imponer restricciones significativas en la redistribución del código.
- Asegura que tanto colaboradores como usuarios puedan comprender los términos de la licencia de manera sencilla, fomentando así la colaboración.
- Refleja nuestro compromiso con la libertad de software y apoya un ecosistema de software abierto y colaborativo.

Apéndice B

Especificación de Requisitos

B.1. Introducción

Este apartado es esencial para establecer las expectativas claras y detalladas del proyecto, asegurando que todas las partes interesadas tengan un entendimiento común de los objetivos, funcionalidades y restricciones del sistema. Se centra en identificar y documentar:

- **Requisitos funcionales:** Especifican las propiedades y capacidades que debe tener el proyecto.
- **Requisitos no funcionales:** Especifican restricciones de calidad (por ejemplo, usabilidad, mantenibilidad, etc.) sobre esas propiedades y capacidades.

La comprensión profunda de estos requisitos es fundamental para guiar el diseño, desarrollo y pruebas posteriores, facilitando así la creación de una solución efectiva y eficiente para el desarrollo del proyecto.

B.2. Objetivos generales

Los objetivos generales del proyecto a desarrollar son los siguientes:

- Elaborar una aplicación web que permita al usuario interactuar con nuestro modelo de detección de insectos de forma intuitiva.
- Desarrollar un modelo de detección de insectos que permita el reconocimiento de estos mediante el uso de *redes neuronales* y *Mask-Method MRCNN*.
- Dar a entender al usuario el proceso del algoritmo y la *segmentación de imágenes*.
- Aprender sobre el entrenamiento de modelos a través del lenguaje de programación *Python* y de sus librerías especializadas *Tensorflow* y *openCV*.

B.3. Catálogo de requisitos

Requisitos funcionales

- **RF-1: Carga de Imágenes:** El usuario debe ser capaz de cargar imágenes en el sistema para su análisis.
 - **RF-1.1: Carga para Entrenamiento:** Permite subir imágenes al conjunto de entrenamiento del modelo de detección.
 - **RF-1.2: Carga para Predicción:** Facilita la subida de nuevas imágenes para realizar predicciones inmediatas.
- **RF-2: Procesamiento de Imágenes:** El sistema procesará las imágenes utilizando el algoritmo de IA.
 - **RF-2.1: Preprocesamiento:** Aplicar técnicas de preprocesamiento para preparar las imágenes para el modelo.
 - **RF-2.2: Segmentación de Imágenes:** Ejecutar el algoritmo Mask-Method MRCNN para segmentar los insectos en las imágenes.
- **RF-3: Interfaz de Usuario Intuitiva:** Proporcionar una GUI que permita a los usuarios interactuar fácilmente con el sistema.
- **RF-4: Visualización de Resultados:** Mostrar los resultados de la detección de manera que los usuarios puedan comprender fácilmente.
- **RF-5: Educación de Usuario:** Incluir una sección educativa que explique los conceptos de segmentación de imágenes y redes neuronales.

Requisitos no funcionales

- **RNF-1: Usabilidad:** La aplicación web debe ser intuitiva y de fácil navegación para usuarios de todos los niveles técnicos, así mismo, no debe superar los 10 minutos en aprender a usar la aplicación.
- **RNF-2: Rendimiento:** El modelo de detección debe realizar predicciones en un tiempo de respuesta rápido.
- **RNF-3: Escalabilidad:** El sistema debe ser capaz de escalar para manejar un aumento en la carga de trabajo sin pérdida de rendimiento.
- **RNF-4: Seguridad:** Implementar protocolos de seguridad para la protección de datos y privacidad del usuario.

B.4. Especificación de requisitos

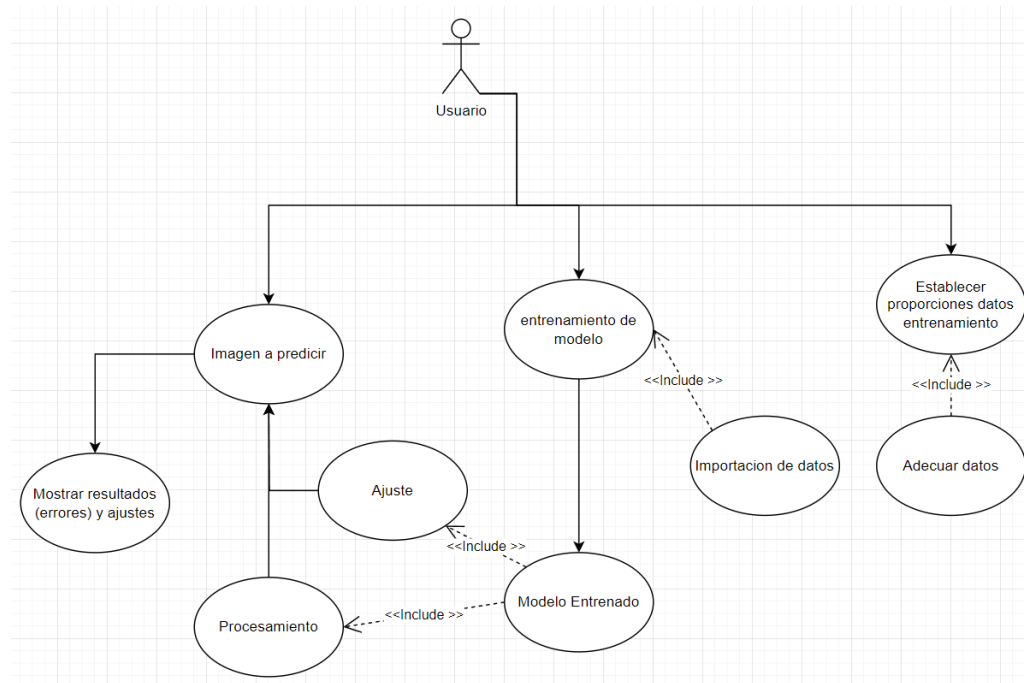


Figura B.1: Diagrama de casos de uso

CU-1	Adecuar Datos
Versión	1.0
Autor	Arturo Carretero Mateo
Requisitos asociados	RF-2.1
Descripción	Preparar el formato de las imágenes y sus equivalentes etiquetados posteriormente guardándolos en el path correspondiente .
Precondición	Imágenes cargadas en el sistema listas para ser procesadas.
Acciones	<ol style="list-style-type: none"> 1. Renombra las imágenes con el nombre 000n donde n es el número de la imagen (esto para poder referenciarlas posteriormente) 2. Asigna las imágenes a su correspondiente etiquetado 3. Establece las imágenes en el path correspondiente 4. Desordena las imágenes y su etiquetado(con el fin de ser diferente orden en los entrenamientos).
Postcondición	Las imágenes se encuentran en el correcto path para su posterior cargar.
Excepciones	Excepción al no corresponder etiqueta y imagen. Excepción al renombrar una imagen
Importancia	Alta

Tabla B.1: CU-1 Adecuar Datos.

CU-2	Carga de Imágenes
Versión	1.0
Autor	Arturo Carretero Mateo
Requisitos asociados	RF-1, RF-1.1, RF-1.2
Descripción	Permitir al usuario cargar imágenes al sistema para su análisis y procesamiento.
Precondición	Los datos han de tener una extensión y nombres correctos.
Acciones	<ol style="list-style-type: none"> 1. Comprobación de formato correcto de imágenes. 2. Se ordenan con su correspondiente etiquetado. 3. Se guardan en el path concreto. 4. Las imágenes se cargan en el sistema para el proceso seleccionado.
Postcondición	Los .jpg han de corresponder con sus correspondientes etiquetados .xml
Excepciones	Mal etiquetado Nula correspondencia Mal formato
Importancia	Alta

Tabla B.2: CU-2 Carga de Imágenes.

CU-3	Establecer Proporciones de Datos de Entrenamiento
Versión	1.0
Autor	Arturo Carretero Mateo
Requisitos asociados	RF-1.1, RF-2.1
Descripción	Definir y aplicar las proporciones adecuadas entre los conjuntos de datos de entrenamiento, validación y prueba para optimizar el rendimiento del modelo de IA.
Precondición	Los datos han sido adecuados y están listos para ser divididos en los diferentes conjuntos.
Acciones	<ol style="list-style-type: none"> 1. Determinar las proporciones óptimas para el conjunto de datos basándose en las mejores prácticas y en la cantidad de datos disponibles. 2. Dividir el conjunto de datos total en entrenamiento, validación y prueba según las proporciones establecidas. 3. Asegurarse de que la distribución de las clases sea homogénea en cada subconjunto para evitar sesgos en el entrenamiento. 4. Guardar los conjuntos de datos en ubicaciones separadas, claramente etiquetadas para su uso durante el entrenamiento y la evaluación del modelo.
Postcondición	Los conjuntos de datos de entrenamiento, validación y prueba están preparados y distribuidos adecuadamente, listos para ser utilizados en el proceso de entrenamiento y evaluación del modelo.
Excepciones	Cantidad insuficiente de datos que impide una división equitativa, distribución desigual de las clases en los subconjuntos.
Importancia	Alta

Tabla B.3: CU-3 Establecer Proporciones de Datos de Entrenamiento.

CU-4	Entrenamiento del Modelo
Versión	1.0
Autor	Arturo Carretero Mateo
Requisitos asociados	RF-1.1, RF-2.2
Descripción	Realizar el proceso de entrenamiento del modelo de IA para la detección y segmentación de insectos.
Precondición	Conjunto de datos de entrenamiento cargado y pre-procesado.
Acciones	<ol style="list-style-type: none"> 1. Preparar los datos de entrenamiento y validar su formato y calidad. 2. Configurar los parámetros del modelo y el entorno de entrenamiento. 3. Iniciar el proceso de entrenamiento del modelo con los datos disponibles. 4. Evaluar el rendimiento del modelo utilizando un conjunto de datos de validación. 5. Ajustar los parámetros del modelo si es necesario y repetir el entrenamiento.
Postcondición	El modelo ha sido entrenado y está listo para ser evaluado o utilizado en producción generando así un modelo nuevo.
Excepciones	El modelo guardado en la correspondiente dirección Problemas durante el entrenamiento como overfitting, underfitting, o errores en los datos.
Importancia	Alta

Tabla B.4: CU-4 Entrenamiento del Modelo.

CU-5	Procesamiento de Imágenes
Versión	1.0
Autor	Arturo Carretero Mateo
Requisitos asociados	RF-2, RF-2.1, RF-2.2
Descripción	Procesar las imágenes cargadas utilizando algoritmos de IA para detectar y segmentar insectos.
Precondición	Imágenes cargadas y listas para procesar.
Acciones	<ol style="list-style-type: none"> 1. El sistema aplica técnicas de preprocesamiento a las imágenes. 2. El sistema ejecuta el algoritmo Mask-Method MRCNN para la segmentación. 3. El sistema guarda los resultados del procesamiento.
Postcondición	Imágenes procesadas con insectos segmentados disponibles para revisión.
Excepciones	Fallas en el procesamiento debido a errores en los datos o en los algoritmos.
Importancia	Alta

Tabla B.5: CU-5 Procesamiento de Imágenes.

CU-6	Predicción de Imagen por el Modelo
Versión	1.0
Autor	Arturo Carretero Mateo
Requisitos asociados	RF-1.2, RF-4
Descripción	Permitir al usuario introducir una imagen en el sistema para que el modelo de IA realice una predicción, identificando y segmentando los insectos presentes.
Precondición	El modelo de IA ha sido entrenado adecuadamente y está disponible para realizar predicciones.
Acciones	<ol style="list-style-type: none"> 1. El usuario sube una imagen al sistema a través de la interfaz de usuario. 2. El sistema verifica el formato y la calidad de la imagen. 3. La imagen se procesa utilizando el modelo de IA para detectar y segmentar los insectos. 4. Los resultados de la predicción se muestran al usuario, incluyendo la localización y clasificación de los insectos detectados. 5. Se ofrece la opción de guardar los resultados o realizar ajustes si el sistema lo permite.
Postcondición	El usuario recibe los resultados de la predicción, incluyendo la identificación y segmentación de los insectos en la imagen.
Excepciones	Imagen de baja calidad que impide la detección, formato de imagen incompatible, errores en el modelo de IA.
Importancia	Alta

Tabla B.6: CU-6 Predicción de Imagen por el Modelo.

CU-7	Mostrar Resultados y Ajustes
Versión	1.0
Autor	Arturo Carretero Mateo
Requisitos asociados	RF-2, RF-4
Descripción	Visualizar los resultados de la detección y segmentación de insectos, incluyendo la identificación de errores y la posibilidad de realizar ajustes.
Precondición	La imagen ha sido procesada y analizada por el modelo.
Acciones	<ol style="list-style-type: none"> 1. Presentar los resultados de la detección de insectos en la interfaz de usuario. 2. Destacar errores o áreas de incertidumbre en los resultados. 3. Ofrecer opciones para ajustar la detección o para reentrenar el modelo con nuevos datos. 4. Guardar los ajustes realizados por el usuario para futuras predicciones.
Postcondición	Los resultados se muestran correctamente y los ajustes quedan registrados en el sistema.
Excepciones	Fallas en la visualización de resultados
Importancia	Alta

Tabla B.7: CU-7 Mostrar Resultados y Ajustes.

Apéndice C

Especificación de diseño

C.1. Introducción

En esta sección se desarrollarán las diferentes elecciones sobre el diseño realizado para el sistema, siendo estas: diseño de datos, diseño procedimental y diseño arquitectónico.

Nos centraremos en fijar los objetivos determinados que se deben seguir para el desarrollo del proyecto.

C.2. Diseño de datos

Para la obtención de datos, he utilizado 4TU.ResearchData.

4TU.ResearchData Este es un repositorio internacional de datos para ciencia, ingeniería y diseño. Ofrece servicios de curaduría, compartición, acceso a largo plazo y conservación de conjuntos de datos de investigación, disponibles para cualquier persona en el mundo. También brinda formación y recursos para ayudar a los investigadores a hacer que los datos de investigación sean localizables, accesibles, interoperables y reproducibles. Alberga miles de conjuntos de datos completos y valiosos de investigaciones técnico-científicas. Estos pueden ser datos crudos, código, datos procesados o datos específicos.[8]

En el repositorio, se ha decidido utilizar el dataset de: Raw data from Yellow Sticky Traps with insects for training of deep learning Convolutional Neural Network. Este dataset contiene imágenes etiquetadas de insectos

recolectados en invernaderos comerciales, lo cual es fundamental para entrenar algoritmos de aprendizaje profundo en tareas de reconocimiento y clasificación. Además, está disponible bajo una licencia CC0 que permite su uso sin restricciones de derechos de autor, lo que facilita la investigación y el desarrollo en el campo de la inteligencia artificial.

Conjunto de datos En cuanto al uso de los datos durante el proyecto, destacan dos fases:

1. Antes del entrenamiento (Pre-entrenamiento)
2. Después del entrenamiento (Post-entrenamiento)

Antes del entrenamiento Antes de proceder con el entrenamiento, es crucial asegurarnos de que todas las imágenes del conjunto de datos tengan las mismas dimensiones. Esto implica redimensionar todas las imágenes a un tamaño uniforme, lo cual es fundamental para garantizar que el modelo pueda procesar los datos de manera consistente y eficiente. Además, será necesario realizar un etiquetado adicional en caso de ser necesario.

Utilizaremos un conjunto de pesos preentrenados, denominado `mask_rcnn_coco.h5`. Este archivo de pesos utiliza Mask R-CNN que ya ha sido entrenado utilizando el dataset MS COCO. Este dataset es ampliamente utilizado para tareas de detección de objetos y segmentación. Los pesos contenidos en este archivo corresponden a la configuración de la red neuronal después de haber sido entrenada con dicho dataset y se utilizan para inicializar el modelo Mask R-CNN para entrenamientos adicionales o para realizar predicciones directamente.

Este archivo de pesos nos servirá como punto de partida, aprovechando el aprendizaje transferido para adaptar el modelo preentrenado a un nuevo conjunto de datos y así no tener que empezar desde un archivo de pesos desde cero.

Después del entrenamiento Después de completar el entrenamiento del modelo, procederemos a evaluar su desempeño y a analizar los resultados obtenidos. Para ello, realizaremos los siguientes pasos:

1. **Impresión de resultados de errores:** Utilizaremos la biblioteca matplotlib para visualizar los errores del modelo durante el proceso de entrenamiento. Esto incluye la representación gráfica de las curvas de pérdida (loss) y precisión (accuracy) a lo largo de las épocas de entrenamiento. Estas gráficas nos permitirán identificar cómo se ha comportado el modelo, detectar posibles problemas de sobreajuste (overfitting) o subajuste (underfitting), y ajustar hiperparámetros en caso de ser necesario.

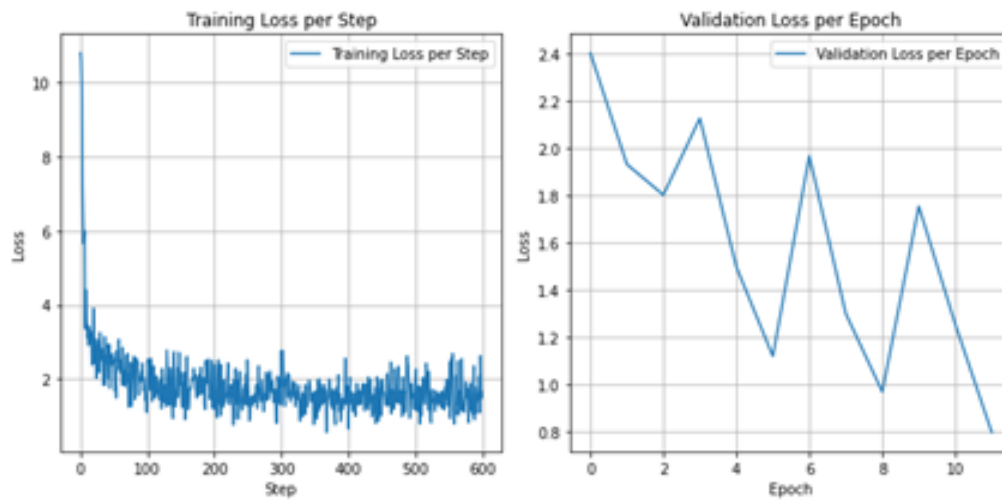


Figura C.1: Ejemplo de resultados producidos durante el entrenamiento

- ### Resultado de la detección



C.3. Diseño procedimental

Esta sección consistirá en el proceso desde que se realiza la petición desde el frontend por parte del usuario hasta que es procesada por el backend y devuelta al frontend a este.

Para ello se ha utilizado SocketIO, el cual ha sido empleado para proporcionar a la aplicación Flask la capacidad de comunicaciones bidireccionales de baja latencia entre los clientes y el servidor [3]. Este servicio ha sido extremadamente útil, ya que permite a los usuarios seguir el progreso del entrenamiento del modelo en tiempo real, visualizando las épocas y los pasos (steps) de dicho entrenamiento.

SocketIO ha sido utilizado en el proceso de la detección y del entrenamiento:

Entrenamiento

Como se ha destacado anteriormente, en el proceso de entrenamiento se ha permitido realizar una comunicación en tiempo real entre el servidor y el cliente. A continuación, se especifican los pasos durante esta comunicación:

1. El cliente realiza una petición POST incluyendo los siguientes parámetros:
 - `filename`: nombre del archivo
 - `algorithm`: algoritmo utilizado
 - `epoch`: épocas
 - `steps`: pasos
 - `proporcionEntrenamiento`: proporción de entrenamiento
 - `proporcionTest`: proporción de prueba
 - `Resnet`: Resnet
 - `validacion`: validación
2. El servidor inicia el proceso de entrenamiento utilizando los parámetros proporcionados y envía los datos iniciales al cliente.

3. El proceso de entrenamiento se ejecuta y el servidor envía actualizaciones periódicas al cliente sobre el estado del entrenamiento:
 - Cuando comienza una nueva época, se envía una actualización con los datos de la época actual y el total de épocas.
 - Durante cada época, se envían actualizaciones de los lotes (batches) procesados, incluyendo el número de lote actual, el total de lotes, y la pérdida (loss) actual.
4. Al finalizar el entrenamiento, el servidor notifica al cliente que el entrenamiento ha sido completado y guarda el modelo entrenado, facilitando también un enlace para su descarga.

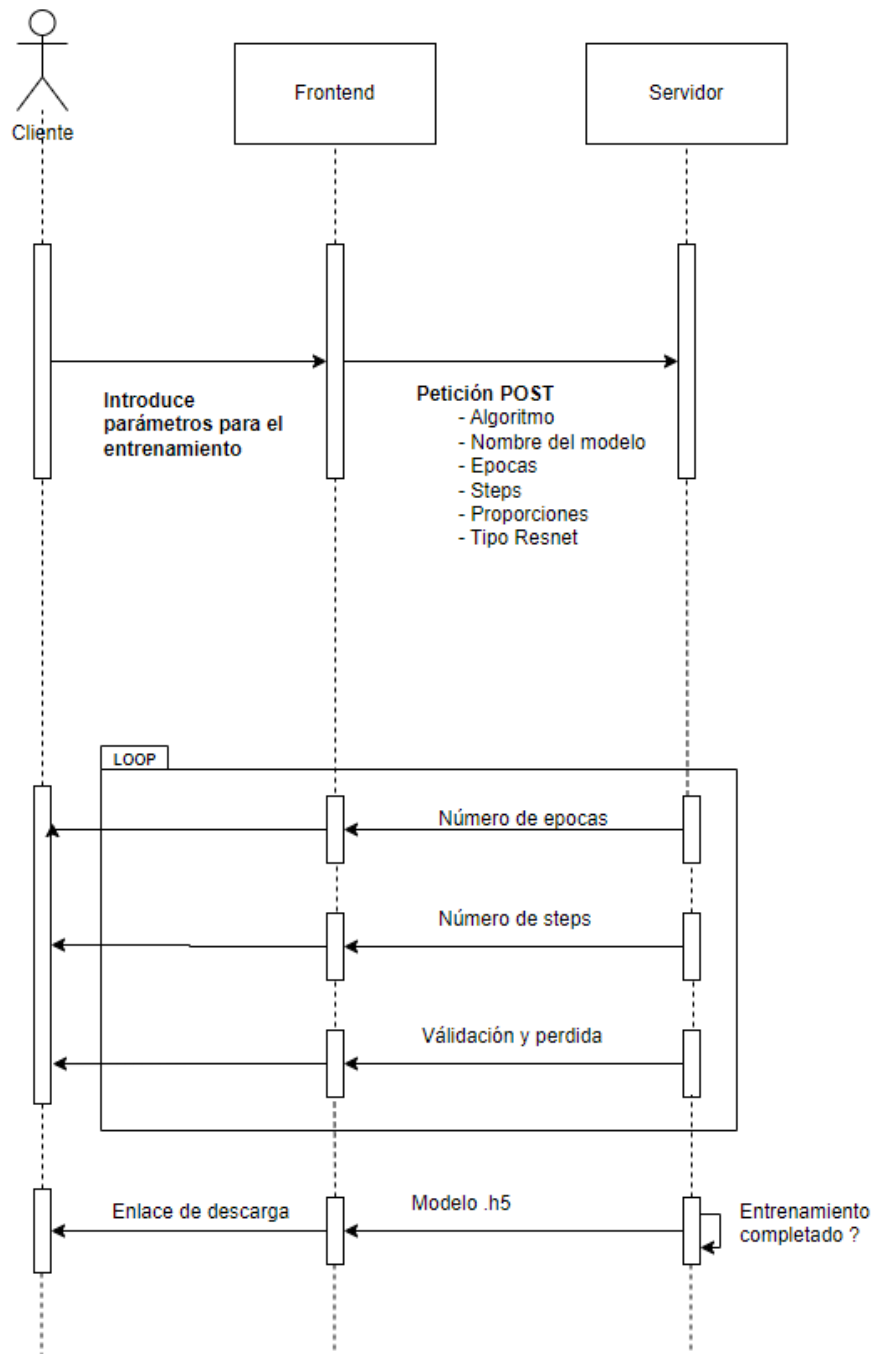


Figura C.3: Comunicación Cliente-Servidor Entrenamiento

Detección

En el proceso de detección se ha permitido la comunicación en tiempo real entre el servidor y el cliente. A continuación se especifican los pasos durante esta comunicación:

1. El cliente realiza una petición POST con la imagen y el modelo a utilizar para la detección. Los parámetros incluyen:
 - `fileInput`: archivo de imagen
 - `modelInput`: archivo del modelo personalizado
 - `model-option`: opción del modelo (personalizado o por defecto)
2. El servidor procesa la solicitud:
 - Verifica y guarda el modelo personalizado si se proporciona uno.
 - Carga el modelo con los pesos adecuados (personalizado o por defecto).
 - Carga y procesa la imagen proporcionada.
3. El servidor realiza la detección de insectos en la imagen y genera una visualización de los resultados:
 - Detecta las regiones de interés (ROIs) y clasifica los insectos presentes.
 - Muestra los resultados y los convierte a una cadena base64 para ser enviados al front.
4. El servidor cuenta las clases detectadas (insectos) y actualiza el estado del proceso.
5. Finalmente, el servidor envía los resultados de vuelta al cliente, incluyendo la imagen procesada con los *bounding boxes* y los conteos de las clases detectadas.

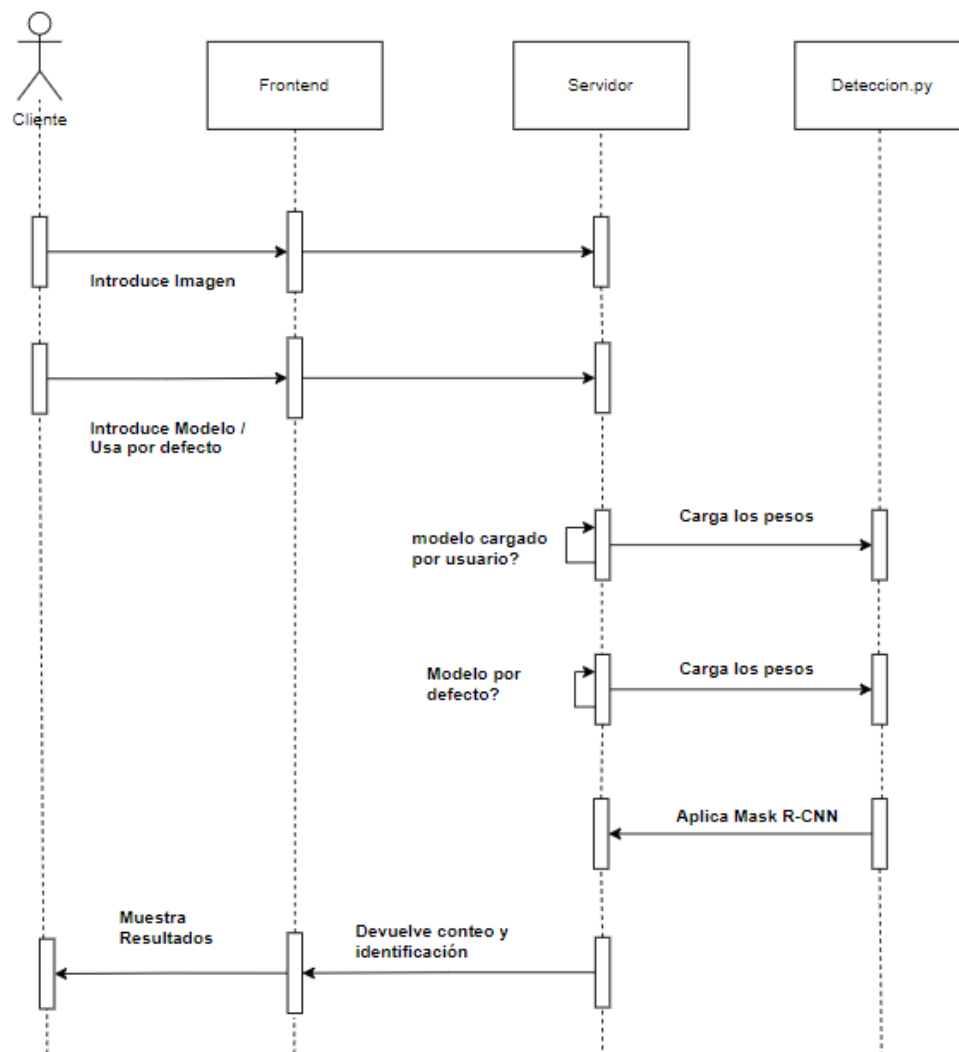


Figura C.4: Comunicación Detección

C.4. Diseño Arquitectónico

En este proyecto, se ha desarrollado mediante una arquitectura monolítica para el diseño y desarrollo de la aplicación implicando que tanto el frontend¹ como el backend² están integrados en una única base de código, facilitando la implementación y el despliegue.

Este enfoque permite una gestión centralizada de los componentes y asegura una comunicación eficiente entre las distintas partes de la aplicación. En las siguientes secciones, se detallará la estructura y funcionalidad tanto del frontend como del backend.

Frontend

En el frontend se destacan las funcionalidades de:

- **Interfaz de usuario (HTML/CSS):** Utilizas plantillas de HTML (mediante Jinja2³) para renderizar las páginas web que el usuario ve e interactúa.
- **Envío de datos al servidor:** Formularios HTML permiten al usuario subir imágenes y modelos (archivos .h5), seleccionar opciones y enviar estos datos al servidor mediante solicitudes HTTP.
- **Actualización en tiempo real:** Utilizas Flask-SocketIO para proporcionar actualizaciones en tiempo real desde el servidor al cliente, especialmente para monitorear el progreso del entrenamiento del modelo.

¹El frontend es la parte de la aplicación que interactúa directamente con el usuario, encargándose de la interfaz y la experiencia del usuario.

²El backend es la parte de la aplicación que maneja la lógica del servidor, el procesamiento de datos y la gestión de la base de datos.

³Motor de plantillas de Flask

Backend

El backend se encarga de la lógica del servidor y el procesamiento de datos. Aquí se maneja la lógica para la carga, procesamiento y análisis de imágenes, así como el entrenamiento del modelo. A continuación se muestra el desglose de sus responsabilidades:

■ Rutas y Controladores:

- `/`: Renderiza la página principal.
- `/deteccion`: Maneja la carga de imágenes y modelos, realiza la detección de insectos y devuelve los resultados.
- `/entrenamiento`: Recibe parámetros para el entrenamiento del modelo, inicia el proceso de entrenamiento en un hilo separado y envía actualizaciones en tiempo real al frontend.
- `{/get_status`: Proporciona el estado actual de un proceso en ejecución, útil para mostrar el progreso al usuario.
- `{/download/<filename>`: Permite la descarga de archivos de peso (modelos entrenados).
- `{/info`: Renderiza una página de información.

■ Procesamiento y Detección de Imágenes:

- Utilizas `skimage` para leer las imágenes subidas por el usuario.
- Mask R-CNN, configurado para inferencia con tu modelo específico, se usa para detectar insectos en las imágenes.
- `matplotlib` se utiliza para visualizar las detecciones y convertir estas visualizaciones en una cadena Base64 que se puede renderizar en el frontend.

■ Entrenamiento del Modelo:

- El entrenamiento se maneja en un hilo separado para no bloquear la aplicación web.
- Utilizas `subprocess` para ejecutar un script de Python que maneja el entrenamiento.
- Flask-SocketIO se usa para enviar actualizaciones del progreso del entrenamiento al frontend en tiempo real.

■ Gestión de Sesiones y Estados:

- Utilizas sesiones de Flask para rastrear el estado del usuario y la progresión de los procesos.
- El estado del proceso se almacena en un diccionario (`process_status`) y se actualiza según sea necesario.

Apéndice D

Documentación técnica de programación

D.1. Introducción

En este apéndice se presenta la documentación esencial para llevar a cabo la correcta programación del proyecto. Esta profundiza sobre la estructura y disposición necesarias para el funcionamiento del proyecto, además del manual del programador para comprender mejor la programación de este. De igual modo, menciona el proceso que se lleva a cabo a la hora de la instalación, ejecución y pruebas correspondientes.

D.2. Estructura de directorios

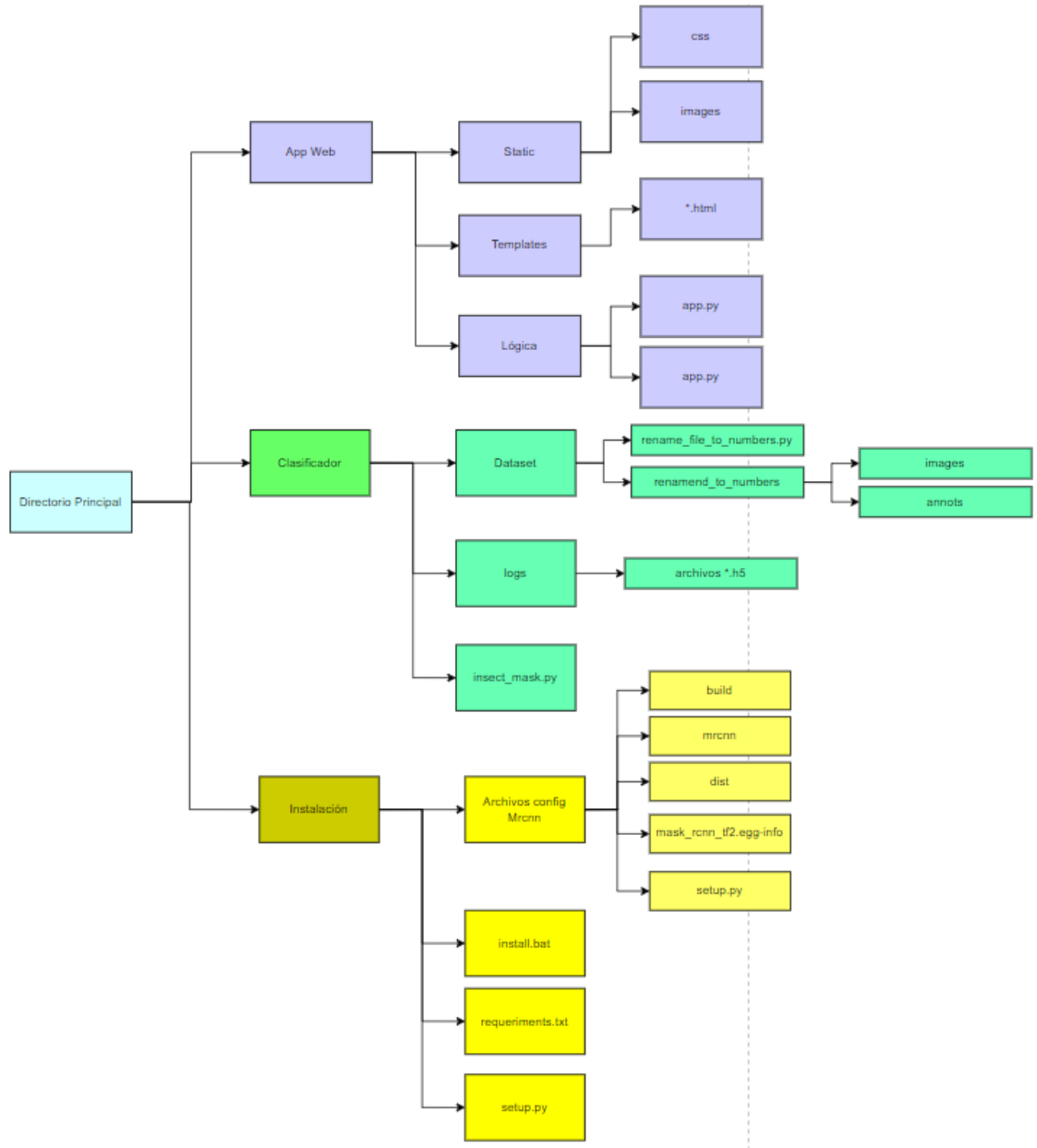


Figura D.1: Diagrama de las estructuras de directorios

El diagrama muestra la estructura y organización del proyecto de detección de insectos en cultivos. A continuación se describe cada sección y su contenido:

Directorio principal

- Es el directorio raíz que contiene todos los subdirectorios y archivos del proyecto.

App Web

- **Static:** Contiene recursos estáticos como CSS e imágenes.
 - `css`: Archivos de hojas de estilo.
 - `images`: Imágenes utilizadas en la aplicación web.
- **Templates:** Contiene los archivos HTML.
 - `*.html`: Plantillas HTML para la interfaz de usuario.
- **Lógica:** Contiene el código fuente de la aplicación web.
 - `app.py`: Archivo principal que maneja la lógica de la aplicación web.

Clasificador

- **Dataset:** Contiene los datos de entrenamiento y prueba.
 - `rename_file_to_numbers.py`: Script para renombrar archivos.
 - `renamed_to_numbers`: Directorio con los archivos renombrados.
 - `images`: Imágenes del dataset.
 - `annots`: Anotaciones del dataset.
- **logs:** Contiene archivos de log generados durante el entrenamiento.
 - `archivos *.h5`: Pesos del modelo entrenado.
- **`insect_mask.py`**: Script principal del clasificador de insectos.

Instalación

- **Archivos config Mrcnn:** Archivos de configuración para Mask R-CNN.
 - **build:** Directorio de construcción.
 - **mrcnn:** Código fuente de Mask R-CNN.
 - **dist:** Distribución del paquete.
 - **mask_rcnn_tf2.egg-info:** Información del paquete Mask R-CNN.
 - **setup.py:** Script de configuración.
- **install.bat:** Script de instalación para Windows.
- **requirements.txt:** Lista de dependencias del proyecto.
- **setup.py:** Script de configuración de Mask R-CNN.

D.3. Manual del programador

En esta sección se especificarán los distintos procesos que se siguieron durante la instalación. El lector (con un nivel medio de programación) ha de ser capaz de seguirla e incluso ampliarla pudiendo llegar a presentar una mejora de este.

Requisitos de hardware Antes de comenzar con la instalación y desarrollo del proyecto, es importante asegurarse de que tu equipo cumple con los siguientes requisitos de hardware:

- **Procesador:** Se recomienda tener un procesador con múltiples núcleos y capacidad para ejecutar tareas intensivas en cómputo.
- **Memoria RAM:** Se recomienda tener al menos 8 GB de memoria RAM, aunque más memoria RAM puede ser beneficiosa para trabajos más grandes y complejos.
- **Tarjeta gráfica NVIDIA (GPU):** Se requiere una tarjeta gráfica NVIDIA con soporte CUDA para aprovechar al máximo las capacidades de procesamiento paralelo y acelerar el entrenamiento de modelos de aprendizaje profundo.

Requisitos de software Además de los requisitos de hardware, también necesitarás cierto software instalado en tu sistema para poder desarrollar y ejecutar el proyecto correctamente:

- **Sistema operativo:** Se recomienda tener un sistema operativo compatible con las herramientas y bibliotecas utilizadas en el proyecto. En mi caso, he utilizado Windows 10 con arquitectura de 64 bits.
- **Python:** Python es un requisito fundamental para el desarrollo del proyecto. Asegúrate de tener instalada una versión de Python compatible, preferiblemente Python 3.7.11 o superior. Más adelante se especifica sobre su instalación.
- **CUDA toolkit:** Si planeas utilizar la GPU para el entrenamiento de modelos de aprendizaje profundo, necesitarás instalar CUDA Toolkit en tu sistema. Más adelante se especifica sobre su instalación.

Instalación

Para poder desarrollar este proyecto se ha de tener previamente instalado *Python*. Para poder instalarlo, acudiremos al siguiente enlace:

<https://www.python.org/downloads>.

Seleccionaremos la versión más apropiada, descargando posteriormente el instalador y siguiendo los pasos recomendados.

Este paso puede ser crucial ya que algunas librerías que utilizaremos posteriormente pueden presentar problemas de compatibilidad entre ellas. En mi caso, utilicé la versión de Python 3.7.11.

Para consultar tu versión instalada, puedes verificarla con el comando:

`python -version`

Utilizaremos la GPU al máximo de sus capacidades para el entrenamiento; para ello, instalaremos CUDA en nuestro equipo, permitiendo así el paralelismo. Para la instalación de este, primero debemos asegurarnos de tener una tarjeta gráfica de NVIDIA y que el sistema operativo sea compatible. Para instalar CUDA Toolkit, iremos al siguiente enlace: <https://developer.nvidia.com/cuda-toolkit>.

Para la instalación de Mask R-CNN, utilizaremos un script llamado *setup.py*, y con el comando `python setup.py install`, podremos realizar la instalación sin problemas.

Además de implementar Mask R-CNN, CUDA y Python, me gustaría resaltar el empleo de GitHub en este procedimiento para administrar el código fuente de nuestro proyecto. GitHub nos proporciona una plataforma para el control de versiones distribuido, permitiéndonos mantener un registro de todos los cambios realizados en nuestro código.

Para acceder al repositorio del proyecto, debemos acudir al siguiente enlace:

https://github.com/arturo1026/Deteccion_Insectos_TFG.

Para la segmentación de instancias en nuestro proyecto, hemos utilizado el repositorio de Matterport [4]. El cual proporciona una implementación inicial en TensorFlow y Keras que incluye todas las herramientas necesarias para el entrenamiento y la inferencia del modelo. Este nos será de gran utilidad ya que podremos tener mrcnn configurado de forma predeterminada (con una arquitectura óptima) permitiendo entre otros una renderización de las máscaras. Para acceder al repositorio acudir al siguiente enlace: https://github.com/matterport/Mask_RCNN.

Entorno para la programación

Para la administración del entorno y de las dependencias, utilizaremos Anaconda, el cual es una distribución muy utilizada en ciencia de datos y aprendizaje automático, incluyendo procesamiento de grandes volúmenes de información, análisis predictivo y cómputos científicos [1].

Para poder descargar la distribución, acudiremos al siguiente enlace:

<https://www.anaconda.com/products/distribution>.

Una vez que lo hemos instalado, y dentro de este, tendremos la siguiente vista:

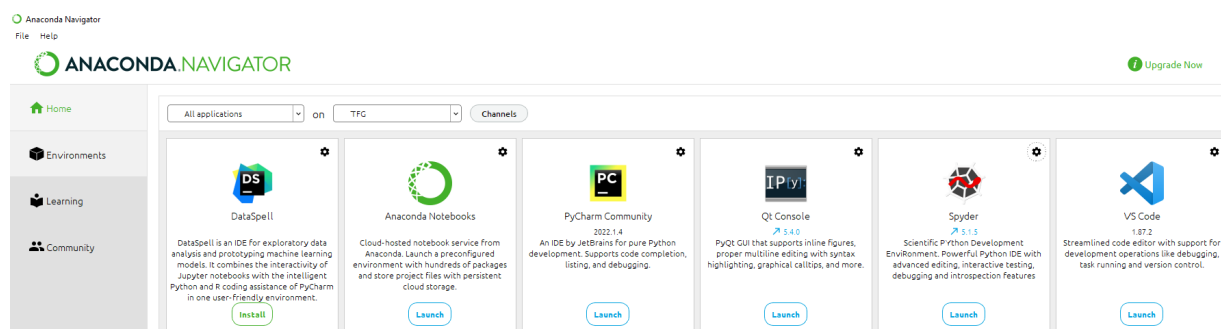


Figura D.2: Vista por defecto de Anaconda

En mi caso, utilizaré el IDE Spyder con la versión 5.1.5.

Funciones y clases

- **load_dataset(dataset_dir, is_train=True)**: Esta función carga el conjunto de datos de insectos, definiendo las clases disponibles en el conjunto de datos y sus ubicaciones. Recorre todas las imágenes en el directorio de imágenes, extrayendo el ID de la imagen y verificando si es parte del conjunto de entrenamiento o de prueba. Luego, agrega la imagen al conjunto de datos junto con su anotación.
- **extract_boxes(filename)**: Esta función extrae las coordenadas de las cajas delimitadoras de una anotación XML dada. Lee el archivo XML y extrae las coordenadas de las cajas delimitadoras de los objetos presentes en la imagen.
- **load_mask(image_id)**: Esta función carga las máscaras correspondientes a una imagen dada. Obtiene los detalles de la imagen a partir de su ID, extrae las cajas delimitadoras y genera las máscaras correspondientes.
- **image_reference(image_id)**: Esta función devuelve la ruta de la imagen correspondiente a un ID de imagen dado.
- **InsectConfig(Config)**: Esta CLASE define la configuración del modelo de Mask R-CNN para el conjunto de datos de insectos. Define el nombre de la configuración, el número de clases y los pasos por época durante el entrenamiento.
- **model.train(train_set, test_set, learning_rate=config.LEARNING_RATE, epochs=x, layers='heads')**: Esta función entrena el modelo de Mask R-CNN utilizando los conjuntos de datos de entrenamiento y prueba especificados. Configura el modelo con la configuración proporcionada y entrena el modelo durante el número especificado de épocas (epochs).
- **PredictionConfig(Config)**: Esta CLASE define la configuración para realizar predicciones utilizando el modelo entrenado. Especifica el nombre de la configuración y el número de clases, entre otros detalles.
- **Rename_file_to_numbers**: Renombra todos los archivos en una carpeta a números secuenciales. Utiliza módulos como os, glob y random para interactuar con el sistema de archivos, buscar archivos y generar números aleatorios. Luego, aleatoriza la lista de archivos, asigna nombres secuenciales y los renombra en orden ascendente.

D.4. Compilación, instalación y ejecución del proyecto

El lenguaje que utilizaremos es Python, por lo tanto, se trata de un lenguaje interpretado y no compilado, sin generarse ningún tipo de ejecutable. Como tal, interpretaremos los programas a través del IDE especificado anteriormente y teniendo Python instalado. Para la instalación de las dependencias, tendremos que instalarlas mediante el archivo *requirements.txt* que se encuentra en la raíz del proyecto.

- Instalaremos con el comando: *pip install -r requirements.txt*
- Comprobamos la correcta instalación con el comando: *pip list*

Con estos pasos, podemos asegurar la correcta instalación y configuración de las dependencias para la ejecución del proyecto de manera correcta.

D.5. Pruebas del sistema

Podremos ir probando las distintas partes de nuestro proyecto ya que, al utilizar Spyder, nos permitirá ejecutar una porción de código que nosotros hayamos seleccionado o bien podremos agrupar el código mediante celdas e ir ejecutándolas.

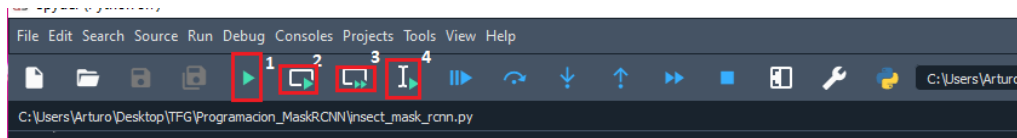


Figura D.3: Debugger en Spyder

1. Ejecución del fichero
2. Ejecución de la celda actual
3. Ejecución de la siguiente celda
4. Ejecución del fragmento de código seleccionado

Debugger

El uso del debugger en Spyder es una herramienta esencial para el desarrollo y la prueba de nuestro proyecto. El debugger nos permite ejecutar el código paso a paso, inspeccionando el estado de las variables y el flujo de ejecución en detalle. Esto es especialmente útil para identificar errores lógicos y comportamientos inesperados en el código

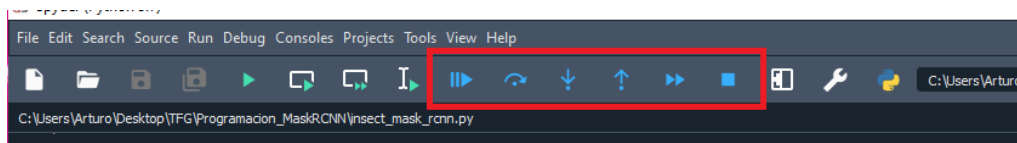


Figura D.4: Debugger en Spyder

Esta versión incluye la mención del uso del *Variable Explorer* para ver la evolución de las variables en tiempo real, lo cual es una característica importante de Spyder para el monitoreo y depuración del código.

Test y pruebas

Para comprobar que el funcionamiento del código es el esperado, se han creado cuatro archivos de prueba en Python utilizando la biblioteca *unittest*. Estos archivos realizan diversas aserciones y pruebas para garantizar la correcta funcionalidad de las diferentes partes del código.

- **testLoadMask.py**: Este archivo se encarga de comprobar la correcta generación de las máscaras de segmentación para cada imagen. Las máscaras son un componente esencial del modelo Mask R-CNN, ya que permiten identificar las regiones específicas de los insectos en las imágenes. Este test verifica que las máscaras se crean adecuadamente y que los identificadores de clase asociados son correctos. Asegurarse de la precisión en la generación de máscaras es fundamental para el rendimiento del modelo en tareas de detección y segmentación de objetos.
- **testConfig.py**: Este archivo se centra en verificar la configuración del modelo. En particular, se asegura de que los atributos definidos en la clase *InsectsConfig* sean correctos. Esto es importante porque una configuración errónea puede llevar a comportamientos inesperados durante el entrenamiento o la inferencia. Este test verifica atributos clave como el nombre de la configuración, el número de clases, y los pasos por época, asegurando que el modelo está correctamente parametrizado antes de su uso.
- **testTraining.py**: Este archivo está destinado a evaluar el proceso de entrenamiento del modelo. Este test carga los pesos preentrenados, configura el modelo con los parámetros especificados, y verifica que el modelo se entrene correctamente sobre el conjunto de datos proporcionado. Además, confirma que los pesos del modelo se guardan adecuadamente después del entrenamiento.
- **testDataSet.py**: Este archivo está diseñado para probar la funcionalidad de la clase *InsectsDataset*, que es responsable de cargar y gestionar las imágenes y sus anotaciones. Este test asegura que las imágenes se cargan correctamente desde el directorio especificado y que las anotaciones correspondientes están presentes y son correctas. Verificar la correcta carga de datos es crucial porque cualquier fallo en esta etapa puede propagarse y afectar negativamente el entrenamiento y la evaluación del modelo.

Apéndice E

Documentación de usuario

E.1. Introducción

En este apartado, se abordan los aspectos esenciales relacionados con los requisitos y procedimientos necesarios para la correcta ejecución y uso del programa desarrollado. Se detallan tanto los requisitos que la aplicación demanda, como las instrucciones para su instalación y utilización por parte del usuario final.

E.2. Requisitos de usuarios

Tendremos que contar con un equipo, ya sea portátil o de escritorio, para llevar a cabo el proyecto informático. Además, será necesario cumplir con los requisitos mínimos tanto de software como de hardware que Python, Spyder y otras herramientas asociadas al proyecto requieran. Esto incluye tener instalado:

- Sistema operativo: Windows XP - Windows Server 2008 - Windows 2003 o versiones superiores.
- Se recomienda tener suficiente almacenamiento de disco duro para el almacenamiento del dataset y la flexibilidad de aumento de este.
- Debe tener como mínimo al menos 8 GB de memoria RAM.
- Permisos de administrador para poder ejecutar el programa.
- Un navegador para poder acceder a la aplicación Flask.
- Entorno de desarrollo para que el usuario pueda programar y realizar ajustes sobre el modelo.
- Se recomienda contar con una GPU con 8 GB como mínimo ya que el programa requiere alrededor de 4,2 GB como mínimo, en caso contrario este requerirá de la CPU realizando entrenamientos más costosos.

Para el funcionamiento de la app el usuario tendrá que lanzar el microservicio Flask, el cual permitirá posteriormente a este interactuar con él y crear su propio modelo.

E.3. Instalación

En caso de ser necesario, se recomienda ver las instrucciones para la instalación de Python y del entorno de desarrollo Spyder en el apartado [D.3](#).

Para la instalación del proyecto, es necesario dirigirse al directorio de instalación en el repositorio. El repositorio contiene todos los archivos y las instrucciones necesarias para configurar y ejecutar Mask R-CNN. Puedes acceder a este directorio mediante el siguiente enlace: https://github.com/arturo1026/Deteccion_Insectos_TFG/tree/main/instalacion.

Una vez en el directorio de instalación, se podrá realizar la instalación paso a paso o mediante la utilización del `install.bat`.

Instalación paso a paso

1. **Clonar el repositorio:** Abre una terminal y clona el repositorio en tu máquina local usando el siguiente comando:

```
git clone https://github.com/arturo1026/Deteccion_Insectos_TFG.git
```

2. **Navegar al directorio de instalación:** Cambia al directorio de instalación:

```
cd Deteccion_Insectos_TFG/instalacion
```

3. **Instalar las dependencias:** Asegúrate de tener todas las dependencias necesarias instaladas. Puedes usar `pip` para instalar las dependencias listadas en el archivo `requirements.txt`:

```
pip install -r requirements.txt
```

4. **Instalar Mask R-CNN:** Finalmente, instala Mask R-CNN utilizando el siguiente comando:

```
python setup.py install
```

Instalación mediante `install.bat`

Para simplificar el proceso de instalación de Mask R-CNN, se proporciona un archivo `install.bat` que automatiza los pasos necesarios. Este archivo es especialmente útil para usuarios de Windows, ya que permite realizar todas las configuraciones e instalaciones con un solo comando.

1. **Clonar el repositorio:** Si aún no has clonado el repositorio, abre una terminal y clona el repositorio en tu máquina local usando el siguiente comando:

```
git clone https://github.com/arturo1026/Deteccion_Insectos_TFG.git
```

2. **Navegar al directorio de instalación:** Cambia al directorio de instalación:

```
cd Deteccion_Insectos_TFG/instalacion
```

3. **Ejecutar el `install.bat`:** Dentro del directorio de instalación, ejecuta el archivo `install.bat` haciendo doble click sobre él o desde la terminal con el siguiente comando:

```
install.bat
```

Este archivo `install.bat` se encargará de ejecutar los siguientes pasos de manera automática:

- Instalar las dependencias necesarias utilizando `pip`.
- Configurar y preparar el entorno para Mask R-CNN.

Siguiendo estos pasos, podrás instalar Mask R-CNN de forma rápida y sencilla utilizando el archivo `install.bat`.

Es posible que se presenten fallos de compatibilidad entre distintas librerías utilizadas en el proyecto debido a sus versiones y que se requiera la instalación de esa librería a mano según la versión adecuada.

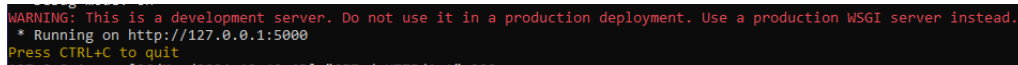
E.4. Manual del Usuario

Una vez instalada la aplicación, el usuario podrá hacer uso de ella siguiendo estos pasos:

1. Diríjase al directorio **webapp** del proyecto en: `Deteccion_Insectos_TFG/webapp/`.
2. Despliegue el servicio Flask utilizando el siguiente comando:

```
python app.py
```

Si se ha podido lanzar el servicio sin problema, el usuario tendrá lo siguiente desde la terminal:

A terminal window with a black background and red and yellow text. The text reads: "WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead." followed by "* Running on http://127.0.0.1:5000" and "Press CTRL+C to quit".

```
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

Figura E.1: Despliegue Flask en dirección 127.0.0.1 y puerto 5000

Como podemos ver, el microservicio se lanza en la dirección 127.0.0.1 y puerto 5000. Accediendo a este mediante un navegador, podremos hacer uso de la aplicación.



Figura E.2: Pantalla Inicial de la aplicación

Una vez dentro de la aplicación, el usuario tendrá varias opciones:

1. Entrenar su propio modelo.
2. Realizar su propia detección, bien con su propio modelo o bien con un modelo predefinido.

Entrenamiento

El usuario puede entrenar su propio modelo basado en Mask R-CNN y modificar las siguientes opciones:

- Proporciones del dataset.
- El backbone de la red, pudiendo elegir entre `ResNet50` o `ResNet101`.
- Número de pasos (*steps*) para la validación.
- Número de pasos (*steps*) para el entrenamiento.
- Algoritmo de optimización, pudiendo elegir entre:
 - SGD
 - Adam
 - RmsProp
- Nombre del modelo.

Los parámetros establecidos en la aplicación permiten al usuario manipular el entrenamiento de manera significativa, determinando su efectividad según los objetivos deseados. Es importante tener en cuenta, como se explica en [127.0.0.1/Info](#), que estos parámetros son considerados de los más trascendentales, pero no son los únicos. Otros factores también influyen en el rendimiento del modelo, como el número de neuronas en la última capa completamente conectada (*Fully Connected*) o la tasa de aprendizaje. Estos parámetros adicionales se pueden modificar manualmente en el archivo `/Programacion_MaskRCNN/mrcnn/config.py`.

Durante el entrenamiento

Durante el proceso de entrenamiento, se muestran en pantalla los parámetros establecidos por el usuario junto con la evolución del entrenamiento. Esto incluye el número de épocas (*epochs*) y los pasos (*steps*) restantes. Esta información permite al usuario monitorear y evaluar el progreso del entrenamiento en tiempo real.

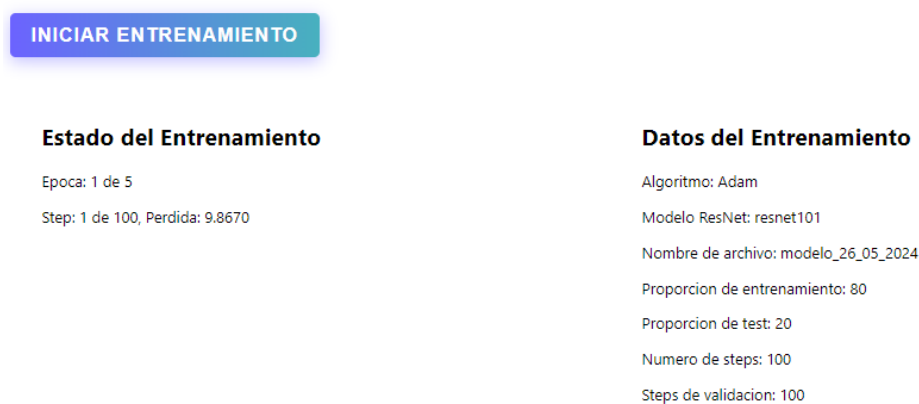


Figura E.3: Información del proceso de entrenamiento

Tras el entrenamiento

Una vez completado el entrenamiento del modelo, aparecerá un enlace en la parte inferior de la información del proceso de entrenamiento. Al hacer clic en este enlace, se iniciará la descarga del modelo entrenado en formato .h5.

- **Formato del modelo:** .h5 es un formato estándar utilizado por Keras para almacenar modelos, incluyendo la arquitectura, los pesos y el estado de entrenamiento.
- **Utilización del modelo:** El modelo descargado puede ser cargado posteriormente en cualquier entorno compatible con Keras o TensorFlow para realizar predicciones o evaluaciones adicionales.

Detección

Este apartado está diseñado exclusivamente para la identificación y conteo de insectos a partir de una imagen proporcionada como entrada.

Selección de la imagen

El usuario deberá proporcionar una imagen en la que se realizará la identificación y conteo de los insectos. Esto puede hacerse arrastrando la imagen desde el explorador de archivos o seleccionándola después de hacer clic en el área designada para la introducción de imagen".

Selección del modelo

El usuario tiene la opción de utilizar un modelo predefinido incluido en el proyecto o cargar un modelo personalizado que haya entrenado por su cuenta. Para utilizar un modelo personalizado, el usuario debe seleccionar la opción "Modelo propio" luego arrastrar el archivo del modelo desde el explorador de archivos o seleccionarlo haciendo clic en el área designada para introducir modelo".

Proceso de detección y conteo

Una vez establecida la imagen y el modelo de entrada, el usuario podrá iniciar el proceso de detección y conteo pulsando el botón de enviar. El proceso se desarrolla en los siguientes pasos:

1. **Carga de la imagen:** La imagen seleccionada se carga en el sistema para su procesamiento.
2. **Identificación de los insectos:** Utilizando el modelo seleccionado, el sistema identifica los insectos presentes en la imagen.
3. **Delineación de los bounding boxes:** Se dibujan cuadros delimitadores alrededor de cada insecto identificado.
4. **Codificación para el frontend:** Los resultados se codifican y se preparan para ser enviados y mostrados en la interfaz de usuario.

Visualización de los resultados

Después de completar los pasos anteriores, el usuario podrá observar la imagen con los cuadros delimitadores (bounding boxes) alrededor de cada insecto identificado. Cada cuadro incluirá una puntuación (Score) y un identificador correspondiente:

- **NC:** *Nesidicoris*.
- **MC:** *Macrolophus*.
- **WF:** Mosca blanca (*White fly*).

Además, el usuario podrá ver el conteo total de cada tipo de insecto identificado en la parte derecha de la imagen. Este conteo proporciona una visión rápida y precisa de la cantidad de cada especie presente en la imagen analizada.

Resultado de la detección



Figura E.4: Ejemplo de visualización de conteo y sus respectivos Bounding Boxes

Apéndice F

Anexo de sostenibilización curricular

Introducción

Este anexo tiene como objetivo reflejar los aspectos de sostenibilidad abordados en el proyecto de detección de insectos utilizando inteligencia artificial. Se incluye una reflexión personal sobre cómo las competencias de sostenibilidad han sido adquiridas y aplicadas.

Conocimiento y conciencia ambiental

Durante el desarrollo del proyecto, se adquirió un conocimiento profundo sobre el impacto ambiental del uso de pesticidas en la agricultura. La tecnología desarrollada tiene un enfoque claro en la sostenibilidad al reducir la dependencia de productos químicos nocivos para el medio ambiente. Mediante la detección precisa y temprana de plagas, es posible aplicar tratamientos fitosanitarios de manera más dirigida y racional, disminuyendo así la cantidad de pesticidas necesarios y minimizando su impacto ambiental.

Innovación y tecnología para la sostenibilidad

El proyecto ha fomentado la innovación tecnológica mediante el uso de redes neuronales convolucionales (CNN) y técnicas de visión por computadora para la detección de plagas. Estas tecnologías permiten una gestión más eficiente y sostenible de los cultivos, contribuyendo a la agricultura de precisión. La implementación de un sistema de monitoreo automatizado

permite a los agricultores tomar decisiones informadas y oportunas, lo que se traduce en prácticas agrícolas más sostenibles.

Integración de sistemas y datos

Se promovió la integración de sistemas de información geográfica (SIG) con los modelos de detección de insectos, lo que facilita el monitoreo y análisis espacial de las poblaciones de plagas. Esta integración es esencial para la gestión sostenible de los ecosistemas agrícolas, ya que permite una visión global y detallada de la distribución de las plagas, mejorando así la toma de decisiones y la planificación de intervenciones.

Responsabilidad social y ética profesional

La elección de una licencia de software libre (MIT) refleja un compromiso con la responsabilidad social y la ética profesional. Esta decisión facilita el acceso y la colaboración en la mejora del software, promoviendo la transparencia y la cooperación entre la comunidad científica y agrícola. Al liberar el software bajo una licencia abierta, se fomenta la adopción y adaptación de esta tecnología en diferentes contextos, contribuyendo así a la sostenibilidad a nivel global.

Aplicación de competencias en el proyecto

Desarrollo Sostenible de Software

El proyecto se desarrolló utilizando metodologías ágiles, específicamente Kanban, lo que permitió una gestión eficiente y adaptable del trabajo. Esta metodología contribuyó a una mejor utilización de los recursos y a la reducción de desperdicios, alineándose con los principios de sostenibilidad en el desarrollo de software.

Optimización y eficiencia energética

La optimización de modelos de aprendizaje profundo para su uso en tiempo real y en dispositivos móviles es otro aspecto clave del proyecto. Al mejorar la eficiencia computacional y reducir la latencia de inferencia, se promueve el uso de tecnologías más eficientes energéticamente, lo que es esencial para la sostenibilidad en el ámbito de la inteligencia artificial.

Reducción del uso de recursos

La utilización de un modelo preentrenado en el conjunto de datos COCO permitió reducir significativamente el tiempo y los recursos necesarios para el entrenamiento del modelo. Esta estrategia no solo optimiza el proceso de desarrollo sino que también minimiza el consumo de energía y recursos computacionales, contribuyendo así a la sostenibilidad del proyecto.

Reflexión personal

A lo largo del desarrollo del Trabajo de Fin de Grado, se ha evidenciado la importancia de integrar la sostenibilidad en los proyectos tecnológicos. La aplicación de tecnologías avanzadas para la detección de plagas no solo mejora la eficiencia y efectividad de las prácticas agrícolas, sino que también promueve un uso más racional de los recursos naturales. Este proyecto ha sido una oportunidad invaluable para comprender y aplicar principios de sostenibilidad en un contexto real, destacando la relevancia de la tecnología en la promoción de prácticas agrícolas sostenibles y responsables.

En conclusión, las competencias de sostenibilidad adquiridas durante este proyecto serán fundamentales para futuras iniciativas profesionales, donde la integración de prácticas sostenibles será crucial para el desarrollo de soluciones innovadoras y responsables con el medio ambiente.

Bibliografía

- [1] Anaconda (distribución de python). [https://es.wikipedia.org/wiki/Anaconda_\(distribuci%C3%B3n_de_Python\)](https://es.wikipedia.org/wiki/Anaconda_(distribuci%C3%B3n_de_Python)). Consultado el día Mes, Año.
- [2] Sarah Damaris Amaro Calderon and Jorge Carlos Valverde Rebaza. Metodologías ágiles. *Universidad Nacional de Trujillo*, 37, 2007.
- [3] Miguel Grinberg. *Flask-SocketIO Documentation*, 2018. <https://flask-socketio.readthedocs.io/en/latest/>.
- [4] Matterport. Mask r-cnn for object detection and instance segmentation on keras and tensorflow. https://github.com/matterport/Mask_RCNN, 2017. Accessed: 2024-06-09.
- [5] Alexander Menzinsky, Gertrudis López, Juan Palacio, M Sobrino, Rubén Álvarez, and Verónica Rivas. Historias de usuario. *Ingeniería de requisitos ágil*, 2018.
- [6] B Molina Montero, H Vite Cevallos, and J Dávila Cuesta. Metodologías ágiles frente a las tradicionales en el proceso de desarrollo de software. *Espiraes revista multidisciplinaria de investigación*, 2(17):114–121, 2018.
- [7] Hubner Janampa Patilla, Edgar Gómez Enciso, José Carlos Juárez Pulache, Jorge Luis Lozano Rodríguez, Eder Solórzano Huallanca, and Yudith Meneses Conislla. Modelo de gestión de desarrollo de software ágil mediante scrum y kanban sobre la programación extrema. *Revista Ibérica de Sistemas e Tecnologías de Informação*, (E43):450–466, 2021.
- [8] TU Delft Library. 4tu.researchdata - TU Delft Library, 2024. Último acceso el 17 de Marzo de 2024.