



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



**TFG del Grado en Ingeniería
Informática**

**Sistema de visión artificial
para detección de personas en
entornos naturales a partir de
imágenes tomadas desde un
dron**

Documentación Técnica



Presentado por Félix de Miguel Villalba
en Universidad de Burgos — 7 de julio de 2024

Tutor: Daniel Urda Muñoz
Cotutor: Carlos Cambra Baseca

Índice general

Índice general	i
Índice de figuras	iii
Índice de tablas	v
Apéndice A Plan de Proyecto Software	1
A.1. Introducción	1
A.2. Planificación temporal	2
A.3. Estudio de viabilidad	5
Apéndice B Especificación de Requisitos	11
B.1. Introducción	11
B.2. Objetivos generales	12
B.3. Catálogo de requisitos	12
B.4. Especificación de requisitos	14
Apéndice C Especificación de diseño	23
C.1. Introducción	23
C.2. Diseño de datos	24
C.3. Diseño procedimental	26
C.4. Diseño arquitectónico	29
Apéndice D Documentación técnica de programación	37
D.1. Introducción	37
D.2. Estructura de directorios	37
D.3. Manual del programador	40

D.4. Compilación, instalación y ejecución del proyecto	41
D.5. Pruebas del sistema	47
Apéndice E Documentación de usuario	53
E.1. Introducción	53
E.2. Requisitos de usuarios	53
E.3. Instalación	53
E.4. Manual del usuario	54
Apéndice F Anexo de sostenibilización curricular	65
F.1. Introducción	65
F.2. Reflexión sobre la sostenibilidad en el proyecto	65
F.3. Competencias de sostenibilidad adquiridas	66
F.4. Conclusión	67
Bibliografía	69

Índice de figuras

B.1. Diagrama de Casos de uso	14
C.1. Diagrama de Flujo de datos	25
C.2. Diagrama de estados de la aplicación	27
C.3. Diagrama de secuencia	28
C.4. Arquitectura cliente-servidor de la WebApp	30
C.5. Caché en Streamlit [9]	31
D.1. Estructura general del proyecto	38
D.2. Conexión en Pulse Secure	42
D.3. Entorno en WinSCP	43
D.4. Login en Putty	43
D.5. Screen	44
D.6. Entorno	45
D.7. Entrenamiento en ejecución	46
D.8. Almacenamiento de resultados	46
D.9. Archivos de resultados	47
D.10. Apartado de Testing de VSCode	48
D.11. Configurar Tests	48
D.12. Seleccionar pytest como framework de pruebas	48
D.13. Seleccionar el <i>root</i> del proyecto, para aglutinar todas las pruebas	49
D.14. Ejecutar tests	49
D.15. Ejecución de los tests	50
D.16. Salida por consola	51
E.1. Página principal	54
E.2. Selección de modelo	54
E.3. Cargando modelo	55

E.4. Modelo cargado y nuevas funcionalidades	55
E.5. Selección de parámetros	55
E.6. Actualizar parámetros	56
E.7. Selección de imágenes	57
E.8. Selección de imágenes locales	57
E.9. Selección de imágenes locales desde el explorador de archivos . .	58
E.10. Imágenes locales cargadas	58
E.11. Seleccionar colección	59
E.12. Colección cargada	59
E.13. Procesar imágenes locales	60
E.14. Procesar predicción de la colección	60
E.15. Procesando predicción	60
E.16. Imágenes procesadas	61
E.17. Opción de pantalla completa	61
E.18. Resultado de la predicción a pantalla completa	62
E.19. Descargar CSV de resultados	62
E.20. CSV descargado	62
E.21. Vista del CSV de resultados	63
E.22. Volver a inicio	63

Índice de tablas

A.1. Costes calculados para el personal del proyecto.	6
A.2. Costes calculados para el hardware del proyecto.	6
A.3. Costes adicionales del proyecto.	7
A.4. Costes totales del proyecto.	7
A.5. Dependencias del proyecto y sus licencias.	9
B.1. CU-1 Selección de Modelos.	15
B.2. CU-2 Umbral de Decisión de Predicción.	16
B.3. CU-3 Configuración de Solapamiento entre Subimágenes.	17
B.4. CU-4 Configuración de Márgenes.	18
B.5. CU-5a Selección de Colección de Imágenes de Prueba.	19
B.6. CU-5b Selección de Imágenes a Analizar.	20
B.7. CU-6 Realización de la Predicción.	21
B.8. CU-7 Descarga de Resultados.	22

Apéndice A

Plan de Proyecto Software

A.1. Introducción

En este documento se pretende mostrar el plan que se ha seguido para realizar el trabajo fin de grado. El plan de proyecto nos detalla su ciclo de vida completo:

Planificación → Análisis → Diseño →
Desarrollo → Pruebas → Mantenimiento

Dividiremos el plan de proyecto en:

- **Planificación temporal:** se detallará lo que se ha realizado a lo largo de los meses que ha durado el proyecto.
- **Estudio de viabilidad:** estimación de los costes y restricciones asociados al proyecto. Podremos distinguir entre:
 - *Viabilidad económica:* se estudiarán los costes y posibles beneficios del proyecto.
 - *Viabilidad legal:* se analizarán las licencias y los aspectos relativos a la protección de datos, siguiendo la Ley Orgánica de Protección de Datos (LOPD).

A.2. Planificación temporal

El proceso para la gestión del proyecto fue seguir un desarrollo incremental en cascada *Top-Down*, donde se dividió el proyecto en etapas o fases secuenciales. Las etapas no fueron marcadas con restricciones temporales estrictas, si no que a medida que se iban terminando se empezaban con las siguientes.

Para simplificar la explicación, podemos resumir las etapas a lo largo de los **7 meses de duración del proyecto**, entre Diciembre de 2023 y Julio de 2024.

Etapas

A continuacion se detallan las acciones realizadas durante las etapas:

Diciembre

Se comenzó con una reunión con los tutores detallando las bases del proyecto y las líneas de trabajo a realizar. Durante este periodo, la labor se centró en:

- Lectura de artículos relacionados con *Search & Rescue*, clasificación de imágenes y CNNs.
- Descarga y análisis del Dataset [2].
- Inicio de recorte y reetiquetado de imágenes.

Enero

Se hizo una revisión de lo avanzado hasta el momento. Nueva reunión para definir los siguientes pasos:

- Terminar reetiquetado de imágenes.
- Creación del repositorio.
- Análisis bibliográfico sobre visión por computador con aprendizaje profundo [5].

Enero estuvo marcado por los exámenes del primer cuatrimestre, por lo que el avance en el desarrollo del proyecto fue más lento.

Febrero

Verificación de que lo realizado hasta este momento funcionaba sin errores y realización de los siguientes pasos:

- Creación de estructura de directorios para el futuro entrenamiento.
- Partición uniforme en cajas para la validación cruzada.
- Definición del título y descripción del trabajo fin de grado.

Marzo

Una vez que todas las subimágenes, estructura y particionado eran correctos, a través de una nueva reunión se detallaron los siguientes objetivos:

- Compartir el dataset modificado con la nueva estructura a través de OneDrive.
- Subida de scripts realizados a GitHub.
- Acceso al clúster de GPUs y configuración del entorno virtual para lanzar ejecuciones.
- Empezar con el desarrollo de script de entrenamiento de redes neuronales convolucionales.

Abril

Este mes se centró en la experimentación con las CNNs:

- Continuar con el desarrollo del script de entrenamiento.
- Elección de la red (AlexNet [7]).
- Ajuste de la arquitectura de la red.
- Exploración de hiperparámetros.

Mayo

Se continuó con la experimentación y nos centramos en:

- Análisis por terreno con más porcentaje de humanos en lugar del Dataset completo.
- Ajuste de métricas de evaluación.
- Inclusión de umbrales de decisión.
- Comienzo con el desarrollo de la memoria.

Junio

Las reuniones fueron más frecuentes, se terminó con:

- Última experimentación con el dataset completo.
- Cambios de solapamiento y decisiones de reetiquetado.
- Recolección final de resultados.
- Desarrollo de la aplicación web.
- Redacción de la memoria.

Julio

Durante el último mes, se cerraron todas las líneas de trabajo abiertas, dando por finalizado el proyecto:

- Desarrollo de pruebas del sistema.
- Redacción final de anexos.
- Puesta a punto del repositorio.
- Presentación y defensa del proyecto.

A.3. Estudio de viabilidad

La viabilidad de un proyecto de software se refiere a la capacidad del proyecto para ser llevado a cabo con éxito, cumpliendo los objetivos definidos.

Viabilidad económica

Para determinar si este proyecto es viable en términos económicos, analizaremos los costes y beneficios del mismo. A continuación, se detallan los apartados correspondientes:

Costes de personal

En este apartado consideramos los costes asociados a la contratación del personal necesario para el desarrollo del proyecto.

- **Ingeniero junior:** Se ha estimado que el ingeniero junior invertirá aproximadamente 500 horas en el proyecto. Suponiendo un salario bruto anual de 21.496€¹ (lo que equivale a 11,82€ por hora), el coste será:

$$\text{Coste ingeniero junior} = 500 \text{ horas} \times 11,82 \text{ €/hora} = 5,910 \text{ €}$$

Añadiendo los impuestos que se deben pagar (23,6 % contingencias, 5,5 % desempleo, 1,5 % accidentes de trabajo), el coste total será:

$$\text{Coste ingeniero junior total} = \frac{5,910}{1 - (0,236 + 0,055 + 0,015)} = 8,515,85 \text{ €}$$

- **Tutor:** Se ha estimado que el tutor dedicará aproximadamente 50 horas al proyecto. Suponiendo un salario bruto anual de 38.707,62€² (lo que equivale a 21,27€ por hora), el coste será:

$$\text{Coste tutor} = 50 \text{ horas} \times 21,27 \text{ €/hora} = 1,063,5 \text{ €}$$

Añadiendo los impuestos, el coste total será:

$$\text{Coste total tutor} = \frac{1,063,5}{1 - (0,236 + 0,055 + 0,015)} = 1,532,42 \text{ €}$$

¹Salario programador junior: <https://es.indeed.com/career/programador-junior/salaries>

²Retribución Ubu: <https://www.ubu.es/servicio-de-recursos-humanos/pdi/informacion-publica/tablas-de-retribuciones/tabla-de-retribuciones-2023-iguales-para-mujeres-y-hombres>

Por lo tanto, el coste total a nivel de personal se muestra en la siguiente tabla, donde el coste por mes es igual al total entre los 7 meses de duración del proyecto:

Concepto	Coste bruto (€)	Coste total (€)	Coste por mes (€)
Ingeniero junior	5.910	8.515,85	1.216,55
Tutor	1.063,5	1.532,42	218,92
Total	6.973,5	10.048,27	1.435,47

Tabla A.1: Costes calculados para el personal del proyecto.

Costes de hardware

Para la realización del proyecto, se cuenta con un portátil y una GPU de altas prestaciones NVIDIA.

- **Portátil:** El coste medio de un portátil con buenas prestaciones es de 1.000€. Se calcula que será amortizado en 6 años, resultando en un coste amortizado mensual de:

$$\text{Coste amortizado portátil} = \frac{1.000}{6 \times 12} = 13,89 \text{ €/mes}$$

- **GPU de altas prestaciones NVIDIA:** El coste de la GPU se estima en 2.000€. Se calcula que será amortizado en 6 años, resultando en un coste amortizado mensual de:

$$\text{Coste amortizado GPU} = \frac{2.000}{6 \times 12} = 27,78 \text{ €/mes}$$

Por lo tanto, el coste total para los 7 meses de duración del proyecto a nivel de hardware se muestra en la siguiente tabla:

Concepto	Coste (€)	Coste amort. (€/mes)	Coste total (€)
Portátil	1.000	13,89	97,23
GPU NVIDIA	2.000	27,78	194,46
Total	3.000	41,67	291,69

Tabla A.2: Costes calculados para el hardware del proyecto.

Costes de software

En este proyecto, todo el software utilizado es de código abierto y gratuito para uso personal, por lo que no hay costes asociados a licencias de software.

Otros costes

Este apartado incluye gastos adicionales como desplazamientos, consumo eléctrico, etc.

Concepto	Coste Total (€)	Coste amortizado (€/mes)
Desplazamientos	200	28.57
Consumo eléctrico	100	14.29
Total	300	42.86

Tabla A.3: Costes adicionales del proyecto.

Total de costes

En la siguiente tabla se muestra el total de los costes que ha supuesto el proyecto, por meses y para los 7 meses de duración del proyecto:

Concepto	Coste total (€/mes)	Coste total (€)
Costes de personal	1.435,47	10.048,27
Costes de hardware	41,67	291,69
Costes de software	0	0
Otros costes	42,86	300
Total	1.520	10.639,96

Tabla A.4: Costes totales del proyecto.

Beneficios

El proyecto no ha sido desarrollado para obtener beneficios, sino que se ofrece como un servicio para la búsqueda y rescate de personas en entornos naturales. Su objetivo es apoyar a los equipos de rescate y a la comunidad científica en sus esfuerzos por salvar vidas y mejorar las estrategias de búsqueda y rescate en áreas difíciles de acceder.

Viabilidad legal

El proyecto debe cumplir con varias normativas legales, especialmente las relacionadas con las licencias, protección de datos y el uso de drones.

Protección de Datos Personales

Dado que el proyecto involucra la captura y procesamiento de imágenes que podrían contener datos personales, es crucial garantizar el cumplimiento de la Ley Orgánica de Protección de Datos (LOPD) y el Reglamento General de Protección de Datos (GDPR) [1]. Estas regulaciones establecen que:

- **Consentimiento:** La recopilación de datos personales requiere el consentimiento explícito de las personas afectadas. En contextos como operaciones de búsqueda y rescate, puede ser necesario establecer excepciones basadas en el interés vital de las personas involucradas.
- **Minimización de Datos:** Solo se deben recolectar los datos necesarios para el propósito específico del proyecto. En este caso, la identificación de personas para operaciones de búsqueda y rescate justifica la captura de imágenes.

Regulación de Drones

La predicción de la presencia de humanos a partir de imágenes tomadas mediante el uso de drones debe tener en cuenta la regulación de la Agencia Estatal de Seguridad Aérea (AESA) [4] en España, que incluye:

- **Licencias y Permisos:** Los operadores de drones deben contar con las licencias y permisos adecuados. Además, el dron debe estar registrado y cumplir con las especificaciones técnicas y de seguridad.
- **Zonas de Vuelo:** Está prohibido volar drones en áreas restringidas, como cerca de aeropuertos, zonas urbanas densamente pobladas o instalaciones críticas sin la debida autorización.
- **Seguridad y Privacidad:** Es crucial asegurar que el uso de drones no invada la privacidad de las personas ni comprometa la seguridad pública. Esto incluye el respeto a las alturas y distancias de vuelo permitidas y garantizar que las operaciones no representen un riesgo para las personas en tierra.

En nuestro caso, el operador de vuelo sería un agente de policía con licencia y con permiso para volar en espacios abiertos.

Licencias de Software

Para el desarrollo de este proyecto se han utilizado diversas herramientas y bibliotecas de software, cada una con su respectiva licencia. A continuación se detalla la lista de dependencias y sus licencias:

Dependencia	Licencia
Python	OSI-Open Source
Pandas	BSD 3-Clause
Numpy	BSD
Scikit-Learn	BSD 3-Clause
Tensorflow	Apache License 2.0
Keras	Apache 2.0
Pillow	HPND License
Streamlit	Apache 2.0

Tabla A.5: Dependencias del proyecto y sus licencias.

Como se puede observar, todas las herramientas y bibliotecas utilizadas son de código abierto, lo que permite su uso, modificación y distribución bajo ciertas condiciones.

Para asegurar la claridad en la atribución de licencias en las figuras utilizadas en este proyecto, se ha adoptado la práctica de citar siempre la fuente original, aunque no se especifique explícitamente la licencia en las figuras. Es importante destacar que, salvo indicación contraria, se asume que las figuras reproducidas en este trabajo están bajo la misma licencia que la fuente de la cual fueron extraídas.

Apéndice B

Especificación de Requisitos

B.1. Introducción

Esta sección proporciona una visión general del proyecto desde el punto de vista de la aplicación web desarrollada, describiendo el propósito, el público objetivo y el uso previsto del software desarrollado.

Propósito del Producto

El propósito de esta aplicación web es proporcionar una herramienta eficiente para la detección de humanos en imágenes aéreas, facilitando las operaciones de búsqueda y rescate. La aplicación permite a los usuarios cargar imágenes, seleccionar modelos de predicción y configurar parámetros de análisis para obtener resultados precisos y visualizaciones detalladas.

Valor del Producto

Este producto permite mejorar la eficiencia y efectividad de las operaciones de búsqueda y rescate. Al automatizar la detección de humanos en imágenes aéreas, se reduce significativamente el tiempo y los recursos necesarios para identificar personas en áreas de difícil acceso o grandes extensiones de terreno.

Público Objetivo

El público objetivo incluye equipos de rescate, agencias de respuesta a emergencias y organizaciones de búsqueda y rescate que necesitan una

herramienta confiable y rápida para identificar la presencia de humanos en imágenes aéreas.

Uso Previsto

La aplicación está diseñada para ser utilizada por usuarios sin necesidad de registro o autenticación. Los usuarios pueden cargar imágenes, seleccionar modelos de predicción, ajustar umbrales y parámetros, y visualizar los resultados directamente en la interfaz web.

B.2. Objetivos generales

Este trabajo se ha realizado persiguiendo el cumplimiento de los siguientes objetivos:

- Desarrollar un sistema software que permita la detección y reconocimiento de humanos en imágenes aéreas tomadas con drones.
- Implementar un sistema que sea capaz de procesar las imágenes y proporcionar resultados precisos de detección.
- Optimizar el sistema para que pueda manejar un gran volumen de imágenes y funcionar eficientemente en diferentes entornos naturales.
- Integrar el sistema en una aplicación web para un uso más amigable para el usuario, permitiendo la visualización y descarga de resultados.

B.3. Catálogo de requisitos

Requisitos Funcionales

- **RF-1 Detección de Humanos:** El sistema debe ser capaz de detectar humanos en imágenes aéreas.
 - **RF-1.1 Selección de Modelos:** El usuario debe poder seleccionar entre varios modelos .h5 para realizar las predicciones.
 - **RF-1.2 Umbral de Decisión de Predicción:** El sistema debe permitir seleccionar un umbral de decisión para la predicción.
 - **RF-1.3 Configuración de Solapamiento entre Subimágenes:** El sistema debe permitir configurar el solapamiento entre subimágenes.

- **RF-1.4 Configuración de Márgenes:** El sistema debe permitir configurar los márgenes en la imagen original a partir de los cuales se añadirán subimágenes.
- **RF-1.5 Uso de colecciones de imágenes de prueba:** La aplicación debe permitir el uso de imágenes de prueba para usuarios que no poseen imágenes aéreas propias.
- **RF-1.6 Selección de Imagen a Analizar:** El sistema debe permitir seleccionar una imagen o imágenes específicas para su análisis.
- **RF-1.7 Mostrar la Predicción:** El sistema debe mostrar la predicción basada en los datos y configuraciones proporcionados.
- **RF-2 Exportar resultados:** El sistema debe permitir al usuario obtener los resultados en un formato estandarizado.

Requisitos No Funcionales

- **RNF-1 Rendimiento:** El sistema debe poseer unos tiempos de cálculo y carga aceptables para un navegador web.
- **RNF-2 Escalabilidad:** La aplicación debe permitir la adición de nuevas funciones de forma fácil y transparente para el desarrollador.
- **RNF-3 Seguridad:** Los datos privados y sensibles, como las imágenes introducidas, deben ser gestionados de la forma adecuada para garantizar la privacidad.
- **RNF-4 Disponibilidad:** El sistema debe estar disponible para su uso en todo navegador compatible con *HTML5* y conexión a internet.
- **RNF-5 Usabilidad:** La interfaz debe ser intuitiva y *user-friendly*, facilitando el uso a usuarios no técnicos.
- **RNF-6 Mantenibilidad:** El patrón de desarrollo debe permitir su fácil mantenimiento y corrección de posibles errores tras el despliegue.

B.4. Especificación de requisitos

La siguiente figura muestra un diagrama con los **casos de uso** de la aplicación:

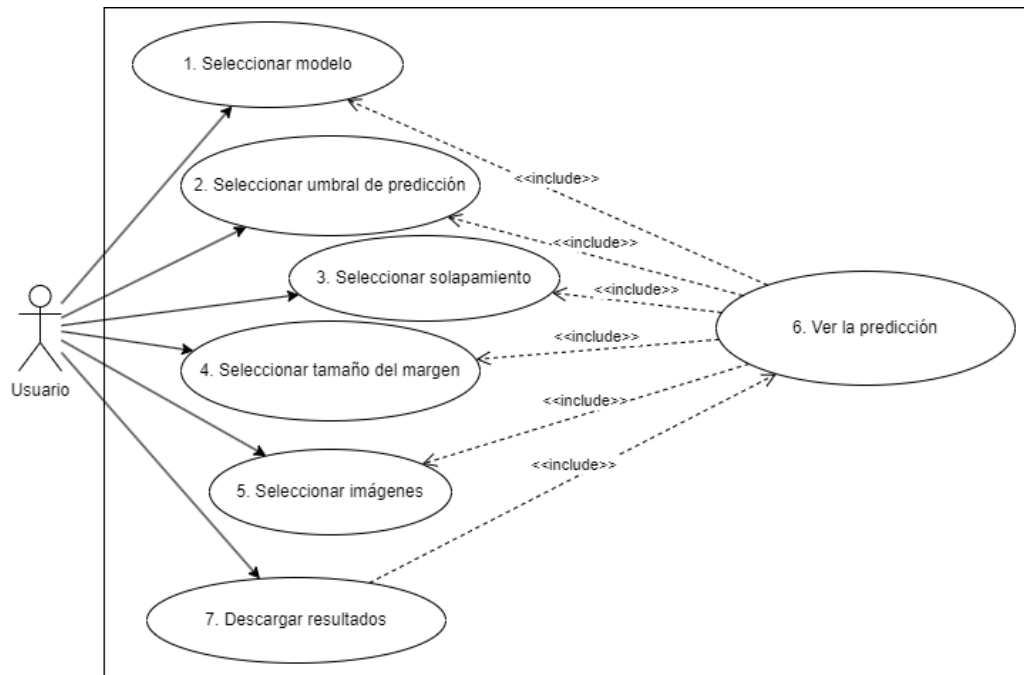


Figura B.1: Diagrama de Casos de uso

A continuación se detallan las tablas de cada caso de uso.

CU-1	Selección de Modelos
Versión	1.0
Autor	Félix de Miguel Villalba
Requisitos asociados	RF-1.1
Descripción	Permite al usuario seleccionar entre varios modelos .h5 para realizar las predicciones.
Precondición	El usuario tiene acceso a la aplicación web.
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona la opción de elegir un modelo. 2. El sistema muestra una lista de modelos disponibles. 3. El usuario selecciona un modelo. 4. El sistema carga el modelo seleccionado.
Postcondición	El modelo seleccionado está cargado y listo para realizar predicciones.
Excepciones	Fallo en la carga del modelo (mensaje de error).
Importancia	Alta

Tabla B.1: CU-1 Selección de Modelos.

CU-2	Umbral de Decisión de Predicción
Versión	1.0
Autor	Félix de Miguel Villalba
Requisitos asociados	RF-1.2
Descripción	Permite al usuario seleccionar un umbral de decisión para la predicción.
Precondición	El usuario ha seleccionado un modelo de predicción.
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona la opción de configurar el umbral de decisión. 2. El sistema muestra un control deslizante para ajustar el umbral. 3. El usuario ajusta el umbral al valor deseado. 4. El sistema guarda la configuración del umbral.
Postcondición	El umbral de decisión está configurado según lo indicado por el usuario.
Excepciones	Fallo en la configuración del umbral (mensaje de error).
Importancia	Media

Tabla B.2: CU-2 Umbral de Decisión de Predicción.

CU-3	Configuración de Solapamiento entre Subimágenes
Versión	1.0
Autor	Félix de Miguel Villalba
Requisitos asociados	RF-1.3
Descripción	Permite al usuario configurar el solapamiento entre subimágenes.
Precondición	El usuario ha seleccionado un modelo de predicción.
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona la opción de configurar el solapamiento. 2. El sistema muestra controles para ajustar el solapamiento en filas y columnas. 3. El usuario ajusta el solapamiento a los valores deseados. 4. El sistema guarda la configuración del solapamiento.
Postcondición	El solapamiento está configurado según lo indicado por el usuario.
Excepciones	Fallo en la configuración del solapamiento (mensaje de error).
Importancia	Media

Tabla B.3: CU-3 Configuración de Solapamiento entre Subimágenes.

CU-4	Configuración de Márgenes
Versión	1.0
Autor	Félix de Miguel Villalba
Requisitos asociados	RF-1.4
Descripción	Permite al usuario configurar los márgenes.
Precondición	El usuario ha seleccionado un modelo de predicción.
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona la opción de configurar los márgenes. 2. El sistema muestra controles para ajustar los márgenes horizontal (a la derecha) y vertical (abajo) de la imagen original. 3. El usuario ajusta los márgenes a los valores deseados. 4. El sistema guarda la configuración de los márgenes.
Postcondición	Los márgenes están configurados según lo indicado por el usuario.
Excepciones	Fallo en la configuración de los márgenes (mensaje de error).
Importancia	Media

Tabla B.4: CU-4 Configuración de Márgenes.

CU-5a	Selección de Colección de Imágenes de Prueba
Versión	1.0
Autor	Félix de Miguel Villalba
Requisitos asociados	RF-1.5
Descripción	Permite al usuario seleccionar una colección de imágenes para pruebas.
Precondición	El usuario ha seleccionado un modelo de predicción.
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona la opción de elegir una colección de imágenes. 2. El usuario selecciona la colección deseada. 3. El sistema carga las imágenes de la colección. 4. El sistema muestra las imágenes seleccionadas.
Postcondición	La colección de imágenes está lista para ser analizada. Las imágenes son mostradas al usuario.
Excepciones	Fallo en la selección de la colección de imágenes (mensaje de error).
Importancia	Alta

Tabla B.5: CU-5a Selección de Colección de Imágenes de Prueba.

CU-5b	Selección de Imágenes a Analizar
Versión	1.0
Autor	Félix de Miguel Villalba
Requisitos asociados	RF-1.6
Descripción	Permite al usuario seleccionar imágenes específicas para su análisis.
Precondición	El usuario ha seleccionado un modelo de predicción.
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona la opción de elegir imágenes. 2. El sistema abre el explorador de archivos. 3. El usuario selecciona las imágenes deseadas. 4. El sistema carga las imágenes seleccionadas. 5. El sistema muestra las imágenes seleccionadas.
Postcondición	Las imágenes seleccionadas están listas para ser analizadas. Las imágenes son mostradas al usuario.
Excepciones	Fallo en la selección de imágenes (mensaje de error).
Importancia	Alta

Tabla B.6: CU-5b Selección de Imágenes a Analizar.

CU-6	Ver de la Predicción
Versión	1.0
Autor	Félix de Miguel Villalba
Requisitos asociados	RF-1.7
Descripción	Permite al usuario ver la predicción basada en las configuraciones seleccionadas.
Precondición	El usuario ha seleccionado una o varias imágenes, bien localmente bien procedentes de una colección de prueba.
Acciones	<ol style="list-style-type: none">1. El sistema procesa las configuraciones y realiza la predicción.2. El sistema muestra los resultados de la predicción.
Postcondición	Los resultados de la predicción son mostrados al usuario.
Excepciones	Fallo en la realización de la predicción (mensaje de error).
Importancia	Alta

Tabla B.7: CU-6 Realización de la Predicción.

CU-7	Descarga de Resultados
Versión	1.0
Autor	Félix de Miguel Villalba
Requisitos asociados	RF-2
Descripción	Permite al usuario descargar los resultados de la predicción en un formato estandarizado.
Precondición	El usuario ha realizado una predicción y los resultados están disponibles.
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona la opción de descargar los resultados pulsando un botón. 2. El sistema prepara los resultados en un formato estandarizado. 3. El usuario selecciona mediante el explorador de archivos el nombre del archivo y dónde descargar los resultados.
Postcondición	Los resultados de la predicción son descargados en un formato estandarizado en el dispositivo del usuario.
Excepciones	Fallo en la descarga de los resultados (mensaje de error).
Importancia	Alta

Tabla B.8: CU-7 Descarga de Resultados.

Apéndice C

Especificación de diseño

C.1. Introducción

En esta sección nos centramos en la especificación de diseño de la aplicación web desarrollada.

La especificación de diseño es un apartado fundamental del desarrollo de un proyecto de software que detalla cómo debe ser construido un sistema para satisfacer los requisitos previamente establecidos. Esta sección actúa como un puente entre la fase de análisis y la fase de implementación, proporcionando una guía clara y precisa sobre cómo se estructura y comporta el sistema.

Para la aplicación se ha elegido **Streamlit**, plataforma diseñada para Python, con un enfoque principal en el análisis de datos y el aprendizaje automático. Dado que el objetivo consiste en desarrollar una sencilla aplicación web a modo de prueba de concepto, donde la funcionalidad es más importante que la estética, Streamlit es una opción ideal.

A continuación, se detallarán los diferentes aspectos del diseño de esta aplicación, incluyendo el diseño de datos, el diseño procedimental y el diseño arquitectónico.

C.2. Diseño de datos

Datos de entrada

Las entradas de datos en esta aplicación a través de la interfaz interactiva son:

- las **imágenes** que se van a analizar para detectar la presencia de humanos. Estas imágenes pueden estar en formato `jpg/jpeg` o `png`. Se cargarán como una lista de imágenes.
- El **modelo H5** de TensorFlow que almacena la red neuronal en formato *HDF5* (Hierarchical Data Format).
- Además, el usuario puede proporcionar parámetros adicionales, los valores numéricos para el **umbral de decisión de predicción** (real) y los parámetros de **solapamiento** (enteros) y **márgenes** (enteros). Para estos parámetros se ofrecen valores por defecto.

Datos de salida

La estructura de los datos de salida incluye:

- Una **representación visual de las imágenes** originales, marcando con un rectángulo aquellas subimágenes de la imagen original donde se ha detectado la presencia de un humano
- Un **archivo .csv** que contendrá una fila con información relevante por cada imagen de entrada en la que se haya encontrado al menos un humano.

Flujo de Datos

El flujo de datos en la aplicación sigue los siguientes pasos:

1. El usuario selecciona un modelo de predicción.
2. El usuario selecciona los parámetros de predicción: umbral de decisión, solapamiento y márgenes. El usuario puede utilizar los facilitados por defecto.

3. El usuario carga una o más imágenes en la aplicación web.
4. La aplicación procesa las imágenes utilizando el modelo seleccionado
5. La aplicación genera una visualización de los resultados de la predicción y los muestra al usuario.
6. El usuario puede optar por descargar los resultados en un archivo .csv.

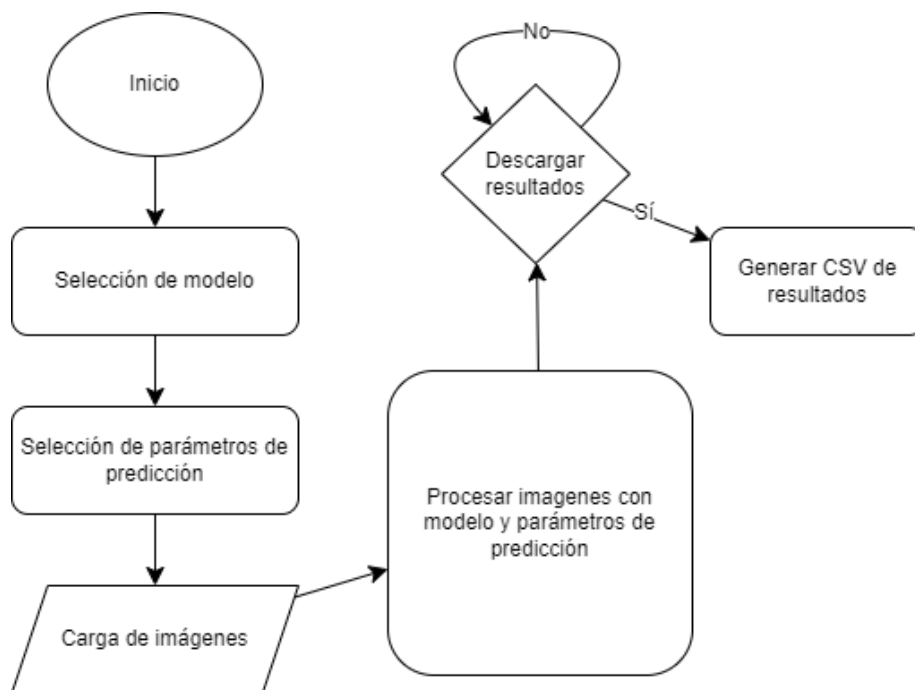


Figura C.1: Diagrama de Flujo de datos

C.3. Diseño procedimental

Esta sección describe el flujo de procedimientos de la aplicación web y cómo se realiza cada funcionalidad importante, incluyendo la interacción del usuario con la aplicación y el manejo de errores.

Para la realización de esta parte del diseño se utilizó una **máquina finita de estados (MFE)**. Una máquina finita de estados permite modelar el comportamiento secuencial de un sistema. Los componentes principales de una *MFE* son:

- **Estados**, que representan las diferentes condiciones en las que el sistema puede estar.
- **Transiciones**, que definen cómo y cuándo el sistema cambia de un estado a otro en respuesta a eventos o condiciones.
- **Eventos**, que son las condiciones que provocan las transiciones entre estados.

Los estados principales de la aplicación web son los siguientes:

- **inicial**, donde se carga la página web y se inicializan las variables internas que configuran la *sesión*.
- **hayModelo**, donde el usuario ha elegido un modelo de predicción pero aún no ha seleccionado imágenes.
- **hayImágenes**, donde el usuario ha cargado una lista de imágenes para ser procesada.
- **imagenesProcesadas**, se ha realizado la predicción para cada una de las imágenes seleccionadas.
- **excepcion**, estado que conlleva la finalización de la ejecución de la aplicación como consecuencia de la captura de alguna excepción insalvable.

Las transiciones más relevantes de la sencilla *MFE* implementada se ajustan a los procedimientos fundamentales de la aplicación web:

- **cargaModelo**, modela la transición **inicial** → **hayModelo**. Tras seleccionar el usuario un modelo en la interfaz, se ejecuta el procedimiento que **carga un modelo de predicción**.

- **cargaImágenes**, modela la transición **hayModelo** \rightarrow **hayImágenes**. Tras seleccionar imágenes localmente o de una colección de prueba, se realiza el procedimiento de **carga de una lista de imágenes**.
- **predice**, modela la transición **hayImágenes** \rightarrow **imagenesProcesadas**. Tras pulsar el botón, se activa la acción de **procesar imágenes**, utilizando los ajustes seleccionados.

Una transición que no tiene asociada una funcionalidad de procesamiento de datos es la siguiente:

- **nuevasPredicciones**, modela la transición **imagenesProcesadas** \rightarrow **hayImágenes**. Tras pulsar el botón correspondiente, se muestra de nuevo la página principal de la aplicación web.

Los eventos que generan *auto-transiciones* son:

- La **selección de un nuevo modelo** cuando se realiza en los estados **hayModelo** o **hayImágenes**
- La **selección de ajustes**, que solo puede realizarse en los estados **hayModelo** o **hayImágenes**.
- La **carga de nuevas imágenes** cuando se realiza en el estado **hayImágenes**
- La **descarga de los resultados** en un archivo .csv, que solo puede realizarse en el estado **imagenesProcesadas**

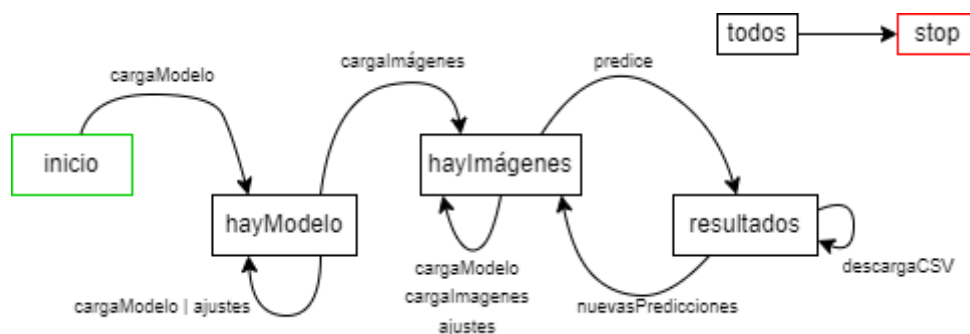


Figura C.2: Diagrama de estados de la aplicación

Diagrama de Secuencia

Se muestra a continuación un diagrama de secuencia completo de la interacción del usuario con el sistema:

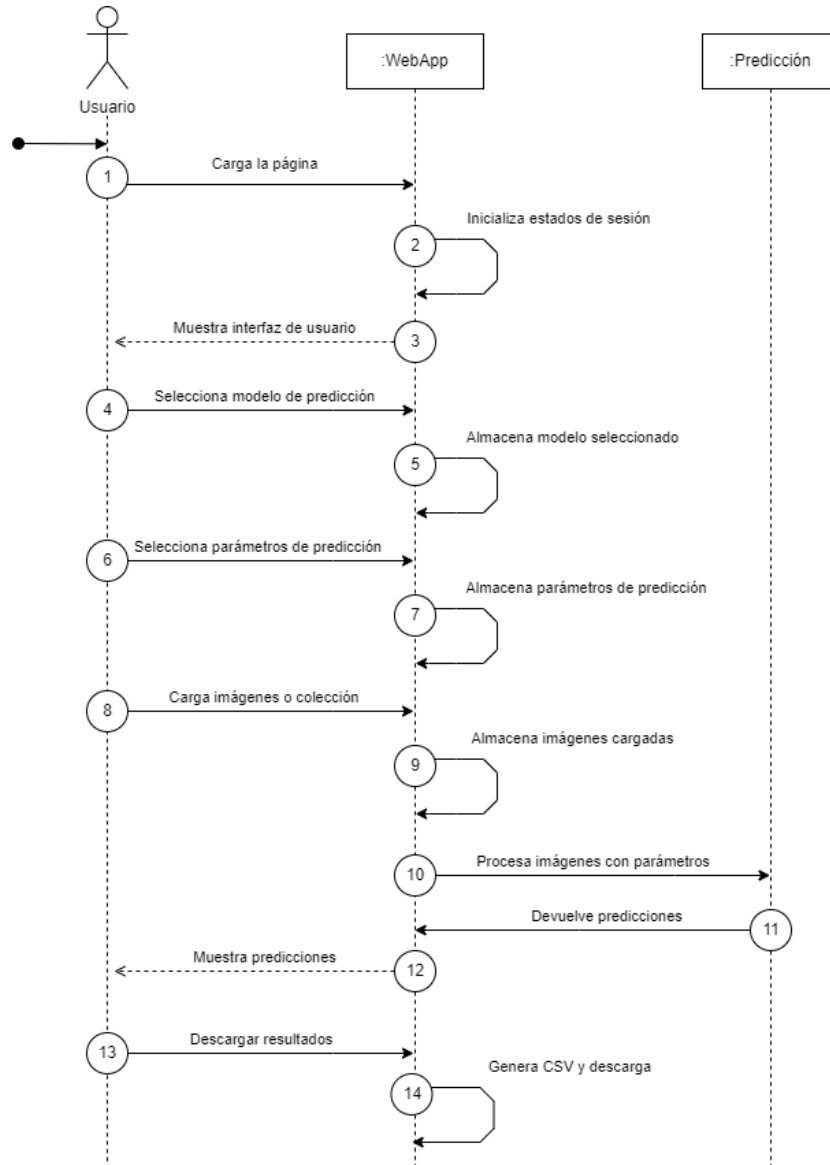


Figura C.3: Diagrama de secuencia

C.4. Diseño arquitectónico

Introducción

Un modelo arquitectónico clásico para realizar el diseño de una aplicación web es el **Modelo-Vista-Controlador (MVC)**. Este patrón proporciona una separación clara entre la lógica de negocios, la interfaz de usuario y el control de flujo, facilitando la escalabilidad y el mantenimiento en aplicaciones complejas.

Por otro lado, **Streamlit** es una herramienta de creciente popularidad para crear aplicaciones web interactivas y de visualización de datos en Python.

Aunque el Modelo-Vista-Controlador es un patrón de diseño arquitectónica ampliamente utilizado en el desarrollo de aplicaciones web tradicionales, hay razones específicas por las que **no se ha contemplado** el uso del patrón **MVC** en la aplicación web conjuntamente con **Streamlit**:

- **Streamlit** está diseñado para ser simple y rápido de usar. Su objetivo es permitir a los desarrolladores construir y desplegar aplicaciones rápidamente con una cantidad mínima de código. **MVC**, por otro lado, introduce una estructura compleja que puede ralentizar el desarrollo inicial.
- La arquitectura interna de **Streamlit** es reactiva, donde los cambios en los datos y en el código se reflejan automáticamente en la interfaz de usuario. El código se ejecuta de arriba hacia abajo cada vez que hay una interacción del usuario. Este modelo secuencial complica la separación de responsabilidades estricta de **MVC**.
- **MVC** implica la creación de módulos y estructuras adicionales para separar las capas de la aplicación, lo que aumenta innecesariamente la cantidad de código. **Streamlit** permite a los desarrolladores centrarse más en la funcionalidad y menos en la estructura.

Arquitectura de Streamlit

La aplicación Streamlit se despliega siguiendo el modelo **cliente-servidor**. El servidor Streamlit ejecuta el *script* de la aplicación y maneja la lógica, mientras que el cliente (navegador web) renderiza la interfaz de usuario y envía las interacciones del usuario de vuelta al servidor.

Algunos aspectos de Streamlit que condicionan el diseño son:

- El servidor debe ser capaz de manejar la carga computacional y de almacenamiento para todos los usuarios concurrentes.
- Actualmente, Streamlit no permite el acceso directo a las rutas de archivos o directorios locales; en su lugar, permite a los usuarios seleccionar y cargar archivos específicos desde su sistema de archivos local.
- Cualquier proceso o programa externo ejecutado por la aplicación se ejecutará en el servidor, no en la máquina del usuario.
- **Streamlit Cloud**, el servidor donde se aloja la aplicación, utiliza **GitHub** para la integración de código. Dado que la versión gratuita de **GitHub** tiene una limitación en el tamaño de los archivos, se ha usado **Google Drive** para almacenar archivos grandes, en concreto, los modelos *HDF5*.

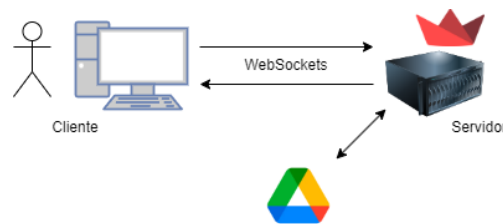


Figura C.4: Arquitectura cliente-servidor de la WebApp

Manejo de los datos

La arquitectura de Streamlit tiene un manejo de los datos especial: cada vez que algo se actualiza en la pantalla, por ejemplo, arrastrar un deslizador, hacer clic en un botón o seleccionar un ítem de un menú desplegable, se vuelve a ejecutar el script completo de Python de arriba a abajo.

Relacionado con lo anterior, cada vez que se actualiza la aplicación guardando el script, Streamlit detecta si hay algún cambio y permite en la aplicación web volver a ejecutar la aplicación. Esto permite un desarrollo interactivo rápido, más si las ventanas del editor de código y el navegador se muestran una al lado de otra: se modifica el código, se guarda, se prueba en vivo, se escribe más código, se guarda, se prueba en vivo, y así sucesivamente. Este bucle que permite codificar y ver los resultados en la aplicación web en vivo es una de las características más relevantes de Streamlit.

Almacenamiento en caché

El **almacenamiento en caché** permite que una aplicación siga siendo eficiente incluso cuando carga datos desde la web, manipula conjuntos de datos grandes o realiza cálculos costosos. Estos datos en caché persisten a lo largo de diferentes ejecuciones, sesiones y usuarios.

La idea básica detrás del almacenamiento en caché es almacenar los resultados de llamadas a funciones costosas y devolver el resultado en caché cuando se vuelven a producir los mismos valores de entrada. Esto evita la ejecución repetida de una función con los mismos valores de entrada cada vez que un usuario interactúa con la aplicación.

Para almacenar en caché la salida de una función en Streamlit se aplica un decorador [6]:

- `@st.cache_data` se utiliza para almacenar en caché las funciones que devuelven datos. Debe tenerse en cuenta que `st.cache_data` crea una nueva copia de los datos en cada llamada a la función, lo que lo hace seguro contra mutaciones y condiciones de carrera.
- `@st.cache_resource` es la forma recomendada de almacenar en caché recursos globales devueltos tras una llamada a una función, como modelos de aprendizaje automático o conexiones a bases de datos. Una llamada a una función con este decorador devuelve el objeto en caché en sí mismo, que se comparte en todas las ejecuciones y sesiones sin copiar ni duplicar.

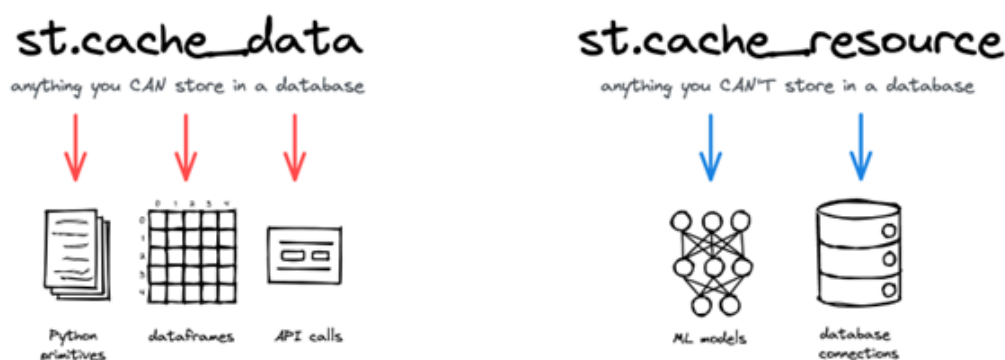


Figura C.5: Caché en Streamlit [9]

En la aplicación se ha usado el almacenamiento en caché de recursos para diversas funciones. Podemos mencionar:

- `cargaModeloH5(nombreModelo, urlModelo)`, que carga el modelo de nombre `nombreModelo` a partir de la url `urlModelo` del archivo en **Google Drive**.
- `cargaParametrosConfiguracion(ficheroConfiguracion)`, que carga el fichero `ficheroConfiguracion`, con formato `yaml`, con todos los parámetros relevantes para la inicialización de la aplicación, tales como los parámetros de ajuste, colores para visualización, etc.
- `cargaColecciones(path)`, que carga la lista de nombres de las colecciones de prueba alojadas en el servidor en el directorio `path`.

La caché de datos se ha utilizado para la siguiente función:

- `creaRectangulos(tamImOriginal, tamSubimagen, solapamiento, margenes)`, que crea una lista de rectángulos dados por las coordenadas del vértice superior izquierdo e inferior derecho. Los parámetros entrada son el tamaño de la imagen original, `tamImOriginal`, el de las subimágenes, `tamSubimagen`, y los parámetros de solapamiento y márgenes.

Estado de la sesión

El **estado de la sesión** `st.session_state` proporciona una interfaz similar a un diccionario que permite guardar aquella información que debe conservarse entre las ejecuciones del script. Actúa, *de facto*, como una variable global.

Para Streamlit, una sesión es una única instancia de visualización de una aplicación. Si se ejecuta una aplicación desde dos pestañas diferentes del navegador, cada pestaña tendrá su propia sesión. Por lo tanto, cada usuario de una aplicación tendrá un estado de la sesión vinculado a su vista específica. Streamlit mantiene esta sesión mientras el usuario interactúa con la aplicación. Si el usuario actualiza la página de su navegador o vuelve a cargar la URL de la aplicación, su estado de sesión se restablece y comienza de nuevo con una nueva sesión.

El estado de la sesión es necesario en el contexto de aplicaciones progresivas como en nuestro caso, donde una nueva interacción depende de la

anterior. El estado de la sesión también se puede usar para evitar recálculos, similar al almacenamiento en caché. Sin embargo, las diferencias son importantes:

- El almacenamiento en caché asocia valores almacenados a funciones y entradas específicas. Los valores en caché son accesibles para todos los usuarios en todas las sesiones.
- El estado de la sesión asocia valores almacenados a claves que solo están disponibles en la sesión única donde se guardaron.

Con ánimo de no ser exhaustivos, se mencionarán los estados más relevantes utilizados:

- `st.session_state['fsm']`: almacena una instancia de la **máquina finita de estados** asociada a la sesión. Es sin duda, la variable más importante de la aplicación.
- `st.session_state['modelo']`: almacena una instancia del modelo de predicción. Nótese que al almacenar el modelo en un estado de la sesión, dos o más usuarios pueden trabajar con modelos diferentes almacenados en la caché de recursos.
- `st.session_state['listaImagenes']`: almacena la lista de imágenes que van a ser o han sido objeto de procesamiento

Otros valores del estado de la sesión se utilizan para guardar los parámetros de ajustes, como solapamiento, márgenes y umbral de predicción, los nombres de los modelos, los nombres de las colecciones, etc.

Los componentes (widgets)

Cada **componente (widget)** de Streamlit tiene asociado un valor en el estado de la sesión. Los componentes devuelven tipos de datos simples de Python, desde una booleana que devuelve el componente botón, `st.button`, hasta un valor entero o real de una entrada de texto numérica, `st.number_input`, pasando por una lista de ficheros cargados usando `st.file_uploader`.

Los componentes llevan asociada una **clave** que permite distinguirlos entre sí, además de crear un medio para acceder y manipular el valor del del componente a través de `st.session_state['claveComponente']`.

Muchos de los componentes también tienen asociado una **función de devolución de llamada (callback)**, lo que permite ejecutar código en respuesta a una interacción del usuario con el componente, como actualizar otros elementos de la interfaz, realizar cálculos adicionales o modificar el estado de la aplicación.

Separación de responsabilidades

En el diseño de la arquitectura, se ha prestado especial atención en separar las responsabilidades de la aplicación web de aquellas correspondientes a la capa del modelo.

La aplicación web tiene básicamente dos módulos:

- `main.py`, donde se localiza:
 - La función principal que, en función del estado de la *MFE*, muestra los componentes.
 - Inicialización de variables de sesión de estado.
 - Las funciones que alteran los estados en función de las interacciones del usuario.
- `widgets.py`, donde se localizan componentes y funciones asociadas que por su tamaño en código disponen de una entidad relevante. El propósito en última instancia de este módulo ha sido aligerar el tamaño de la función principal.

La aplicación web realiza llamadas a funciones de otros módulos correspondientes a la capa del modelo como respuesta a las diferentes interacciones:

- `prediccion.py`, donde se encuentra el código correspondiente a:
 - la carga del modelo H5 de TensorFlow.
 - la predicción para cada una de las subimágenes.
- `entradaSalida.py` en el directorio del proyecto `utils`, donde se encuentra el código correspondiente a:
 - una función que carga archivos de configuración `yaml`.
 - una función que carga archivos de Google Drive a partir de su *url*.

- la carga de una imagen.
- `graficosImágenes.py` en el directorio del proyecto `utils`, donde se encuentra el código correspondiente a:
 - una función que calcula las coordenadas de los rectángulos de las subimágenes.
 - una función que dibuja un rectángulo de una subimagen en la imagen original.

Apéndice D

Documentación técnica de programación

D.1. Introducción

En este apéndice se trata de explicar todo lo relacionado con el código desarrollado para realizar el proyecto. Se detalla la estructura y organización de los archivos de forma detallada, para que sea fácilmente reproducible por un programador.

D.2. Estructura de directorios

Se ha estructurado todo en 3 subproyectos:

- Subproyecto de **preparación de imágenes**
- Subproyecto para el **entrenamiento**
- Subproyecto correspondiente a la **webApp**.

Además se incluye una carpeta de utilidades que sirve de apoyo a los subproyectos.

El repositorio se encuentra subido en GitHub a través de la licencia **Creative Commons Attribution-NonCommercial 4.0 International (CC BY-NC 4.0)** en:

https://github.com/felixdmv/TFG_Search-Rescue

A continuación se muestra la estructura general del proyecto:

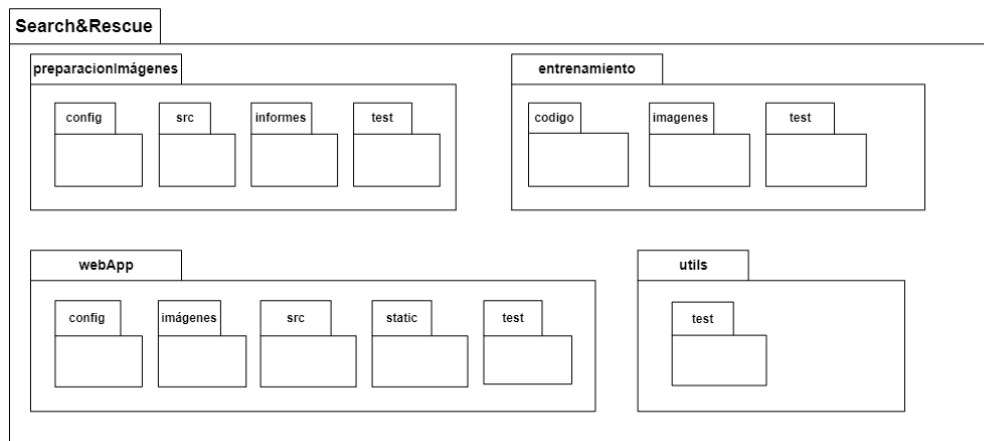


Figura D.1: Estructura general del proyecto

Dentro del **directorio raíz** encontramos en un primer nivel las 4 carpetas principales, seguidas de los siguientes archivos relevantes:

- **README.md**: contiene información general del proyecto; su lectura es importante para comprender los aspectos más relevantes del trabajo.
- **pytest.ini**: configura el framework de pruebas pytest para los subproyectos de pruebas. Cada subproyecto contiene un fichero similar para poder realizar las pruebas de forma independiente.
- **.gitignore**: especifica los ficheros que Git debe ignorar y no incluir en el control de versiones

Estructura del subproyecto **preparacionImágenes**

El subproyecto **preparacionImágenes** de **preparación de imágenes** se divide en:

- **config**: contiene un **.yaml** de configuración con parámetros del recorte, rutas y extensiones.
- **informes**: aquí se almacenarán los análisis estadísticos del dataset antes y después del procesado.

- **src**: scripts de recorte, reetiquetado y creación de CSVs para la validación cruzada. Incluye además un archivo de **settings** donde configura y añade rutas al `sys.path` para asegurar que los módulos se pueden importar correctamente.
- **test**: pruebas del subproyecto.

Estructura del subproyecto **entrenamiento**

El subproyecto **entrenamiento** correspondiente a la parte de **entrenamiento de CNNs** se divide en dos carpetas:

- **src**: scripts de entrenamiento de CNNs y **.yaml** de configuración.
- **imagenes**: subimágenes recortadas y los **.csv** que se leerán para la validación cruzada.
- **test**: pruebas del subproyecto.

Tras ejecutar el script de entrenamiento, se crea automáticamente una adicional al mismo nivel que estos directorios:

- **resultados**: aquí se almacenan los resultados de los entrenamientos, **.log** de las métricas en cada época, **.csv** con las métricas por umbrales, modelo entrenado **.h5** y parametrización elegida en **.yaml** para poder guardarla.

Durante la ejecución, dos carpetas más son creadas, aunque se borrarán al finalizar con el entrenamiento:

- **datos_entreno**: carpeta que se usará para ir separando las imágenes en entrenamiento, validación y prueba durante la ejecución.
- **tmp**: la usará el script para ir guardando ficheros temporales durante el entrenamiento.

Estructura del subproyecto **webApp**

Se divide en cuatro subcarpetas:

- **coleccionesImagenes**: contiene colecciones de imágenes de ejemplo que puede usar un usuario si no dispone de fotos aéreas.

- **config**: contiene un `.yaml` de configuración con parámetros de solapamiento, márgenes y extensiones.
- **src**: contiene los scripts principales de la lógica de la webApp, un `.txt` de requerimientos necesario para poder desplegar la aplicación y una carpeta `.streamlit` con un `.toml` que permite personalizar el comportamiento y apariencia de la web.
- **static**: almacena archivos estáticos que no cambian frecuentemente, en nuestro caso un logo de la app.
- **test**: pruebas del subproyecto.

La carpeta de **utils** contiene un gran número de scripts `.py` de utilidad para los subproyectos de preparación y WebApp. También contiene una carpeta con test de cada función desarrollada.

D.3. Manual del programador

Obtención del código fuente

Para obtener el código fuente del proyecto al completo (con excepción de los modelos), se puede descargar desde GitHub [3].

Entorno de desarrollo

Para la reproducción completa del proyecto, se recomienda (aunque se podría realizar con otras) tener instaladas las siguientes herramientas:

- Anaconda
- Visual Studio Code

De forma opcional, y para mayor comodidad a la hora de desarrollar nuevo código o subproyectos asociados, se puede usar:

- GitHub Desktop
- Pulse Secure, WinSCP, Putty

Estos tres últimos se usarán en caso de tener acceso a un clúster de GPUs al que poder lanzar ejecuciones de entrenamiento.

D.4. Compilación, instalación y ejecución del proyecto

Preparación de imágenes

Para la primera parte del proyecto, relativa al procesamiento de imágenes, reetiquetado y creación de estructuras de directorios, se recomienda crear un entorno virtual y tener instalado Anaconda.

Durante el proyecto se ha trabajado con Visual Studio Code, IDE que nos facilita el trabajo permitiendo crear entornos de manera personalizada. El entorno virtual se ha configurado con *conda* a través de *Anaconda Prompt*. Para poder ejecutar este *subproyecto* de preparación, deberemos especificar un intérprete de Python (para el proyecto se ha usado la versión Python 3.11.9) desde VSCode e instalar los siguientes paquetes:

```
1 # Para abrir el Command Palette de VSCode
2 Ctrl + Shift + P
3
4 # Seleccionamos la opcion de crear un entorno virtual conda.
5
6 # Elegimos un interprete de Python (3.11.9 recomendado).
7
8 # Abrimos Anaconda Prompt.
9
10 # Permite descargar archivos desde Google Drive
11 pip install gdown
12
13 # Biblioteca de procesamiento de imagenes
14 pip install pillow
15
16 # Para analizar y generar archivos YAML
17 pip install pyyaml
18
19 # Visualizacion de datos, mapas de calor
20 pip install seaborn
21
22 # Biblioteca de aprendizaje automatico
23 pip install scikit-learn
24
25 # Para la maquina de estados
26 pip install transitions
27
28 # Para ejecutar un script de Python ubicado en el directorio
   actual
29 python -m script
30
```

```
31 # Para ejecutar un script de Python especificando la ruta  
    completa al archivo.  
32 python "/tu/ruta/script.py"
```

Entrenamiento en clúster de GPUs

Tras tener preparado la estructura de directorios, las subimágenes y la división en particiones manteniendo la proporción de humanos en el `csv`, podemos realizar el desarrollo y la ejecución del script de entrenamiento de nuestra CNN.

Pulse Secure

Para conectarnos al entorno de ejecución, primero utilizamos **Pulse Secure**. Con esta herramienta y la aplicación *Google Authenticator*, ingresamos una clave para autenticarnos y establecer una conexión segura.

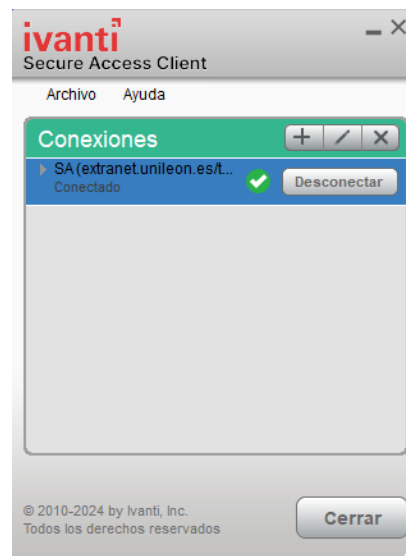


Figura D.2: Conexión en Pulse Secure

WinSCP

Una vez establecida la conexión, accedemos a nuestro entorno virtual utilizando **WinSCP**. Este software nos permite gestionar archivos y directorios en el servidor de forma gráfica.

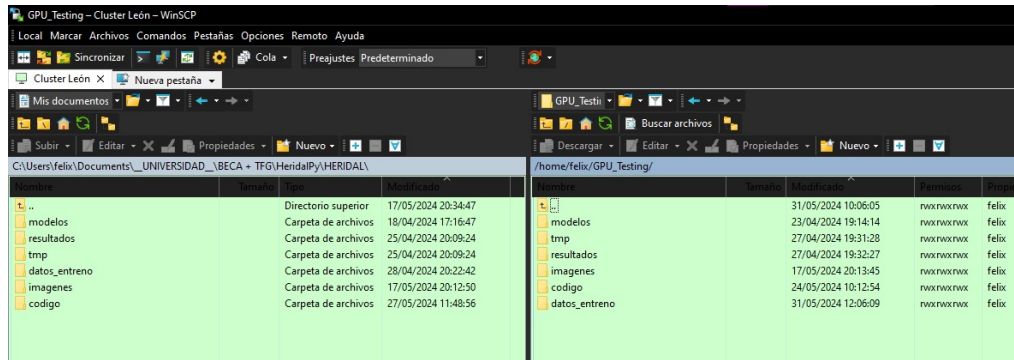


Figura D.3: Entorno en WinSCP

Aquí crearemos la estructura de directorios referente al subproyecto de entrenamiento de CNNs. Basta con copiar aquí las carpetas de código y de imágenes, ya que el resto se generan durante y tras el entrenamiento.

Putty

A continuación, utilizamos **Putty** para conectarnos al servidor.

Login: Hacemos login usando nuestro usuario y contraseña. Esto nos permite establecer una conexión por SSH para trabajar de manera remota.

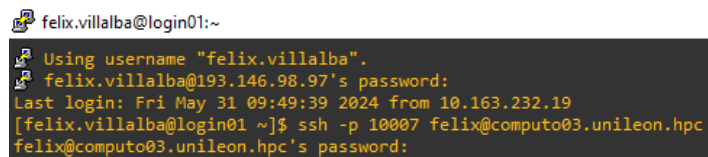
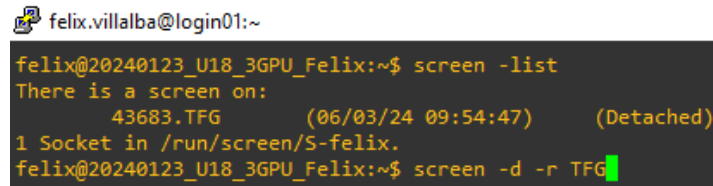


Figura D.4: Login en Putty

Screen: El comando *screen* nos permite crear o acceder a una sesión que hayamos guardado previamente. Esto es útil para mantener nuestras sesiones activas y gestionar múltiples tareas simultáneamente. En casos en los que el entrenamiento dure muchos minutos o incluso varias horas, podemos apagar nuestro equipo sabiendo que el proceso puede ser recuperado más adelante.



```

felix.villalba@login01:~
felix@20240123_U18_3GPU_Felix:~$ screen -list
There is a screen on:
      43683.TFG      (06/03/24 09:54:47)      (Detached)
1 Socket in /run/screen/S-felix.
felix@20240123_U18_3GPU_Felix:~$ screen -d -r TFG

```

Figura D.5: Screen

Configuración del entorno: Instalación de Anaconda

```

1 wget https://repo.anaconda.com/archive/Anaconda3-2024.02-1-
  Linux-x86_64.sh
2 bash Anaconda3-2024.02-1-Linux-x86_64.sh
3 chown -R user:user ./anaconda3

```

Creación de un Entorno para Entrenamiento de Modelos de Clasificación

```

1 # Crear y activar el entorno con Python 3.9.16
2 conda create <<name>> python=3.9.16
3 conda activate <<name>>
4
5 # Instalar cudatoolkit y cudnn
6 conda install -c conda-forge cudatoolkit=11.8 cudnn=8.8
7
8 # Verificar la variable de entorno CONDA_PREFIX
9 echo $CONDA_PREFIX
10
11 # Si no existe, crearla
12 export CONDA_PREFIX=$HOME/anaconda3
13
14 # Verificar la existencia del directorio de activacion
15 echo $CONDA_PREFIX/etc/conda/activate.d
16
17 # Crear y editar env_vars.sh
18 echo 'export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$CONDA_PREFIX/lib
  /' > $CONDA_PREFIX/etc/conda/activate.d/env_vars.sh
19
20 # Verificar el contenido de env_vars.sh
21 cat $CONDA_PREFIX/etc/conda/activate.d/env_vars.sh
22
23 # Desactivar y activar el entorno para cargar las variables
24 conda deactivate
25 conda activate <<name>>
26
27 # Verificar LD_LIBRARY_PATH
28 echo $LD_LIBRARY_PATH
29

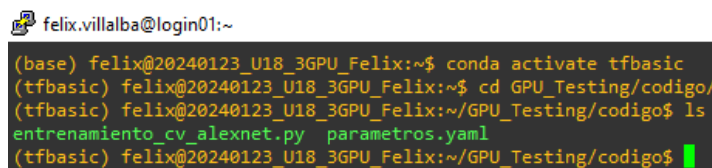
```

```

30 # Instalar TensorRT
31 python -m pip install tensorrt==8.5.3.1
32
33 # Obtener y añadir TENSORRT_PATH a env_vars.sh
34 TENSORRT_PATH=$(dirname $(python -c "import tensorrt; print(
    tensorrt.__file__)"))
35 echo $TENSORRT_PATH
36 nano $CONDA_PREFIX/etc/conda/activate.d/env_vars.sh
37
38 # Añadir la siguiente línea al final de env_vars.sh
39 # export LD_LIBRARY_PATH=/usr/local/cuda-11.2/lib64:/home/user/
    anaconda3/lib/./home/user/anaconda3/envs/<<name>>/lib/
    python3.9/site-packages/tensorrt
40
41 # Desactivar y activar el entorno nuevamente
42 conda deactivate
43 conda activate <<name>>
44
45 # Verificar LD_LIBRARY_PATH
46 echo $LD_LIBRARY_PATH
47
48 # Instalar TensorFlow y verificar la GPU
49 python -m pip install tensorflow==2.13
50 python3 -c "import tensorflow as tf; print(tf.config.
    list_physical_devices('GPU'))"

```

Activamos el entorno virtual previamente creado y configurado para trabajar con GPUs remotas. Esto es esencial para aprovechar la capacidad de cómputo del clúster y acelerar el entrenamiento de la CNN. Accedemos a la carpeta donde se encuentra nuestro código.



```

felix.villalba@login01:~
(base) felix@20240123_U18_3GPU_Felix:~$ conda activate tfbasic
(tfbasic) felix@20240123_U18_3GPU_Felix:~$ cd GPU_Testing/codigo/
(tfbasic) felix@20240123_U18_3GPU_Felix:~/GPU_Testing/codigo$ ls
entrenamiento_cv_alexnet.py  parametros.yaml
(tfbasic) felix@20240123_U18_3GPU_Felix:~/GPU_Testing/codigo$

```

Figura D.6: Entorno

Ejecución: Revisamos la configuración de hiperparámetros del archivo `.yaml` y ejecutamos el script `.py` para iniciar el entrenamiento de la CNN.

```

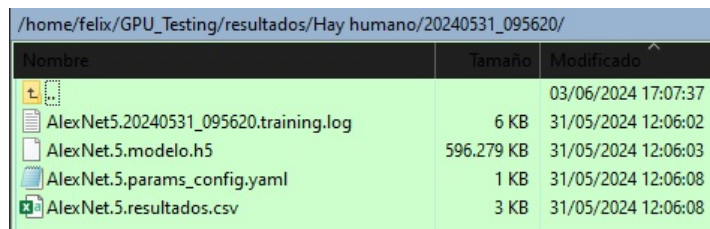
1 python tu_script_entrenamiento.py

```

Figura D.7: Entrenamiento en ejecución

Almacenamiento de resultados: Finalmente, los resultados obtenidos del entrenamiento se almacenan dentro de la carpeta **resultados** de nuestra estructura. Esto incluye el modelo entrenado, **log** de entrenamiento por cada época, un **csv** con las métricas por cada umbral definido y una copia del archivo **yaml** para mantener un registro de la hiperparametrización.

Figura D.8: Almacenamiento de resultados



Nombre	Tamaño	Modificado
AlexNet5.20240531_095620.training.log	6 KB	31/05/2024 12:06:02
AlexNet.5.modelo.h5	596.279 KB	31/05/2024 12:06:03
AlexNet.5.params_config.yaml	1 KB	31/05/2024 12:06:08
AlexNet.5.resultados.csv	3 KB	31/05/2024 12:06:08

Figura D.9: Archivos de resultados

En las figuras D.8 y D.9 se muestran las carpetas de cada iteración del entrenamiento y los archivos que contienen.

Instalación y configuración de Streamlit

Para configurar el entorno de desarrollo, preferiblemente en un entorno virtual [8], se deben seguir los siguientes pasos:

1. Ejecuta: `pip install streamlit`
2. Valida la instalación ejecutando: `streamlit hello`
3. Para iniciar un servidor de Streamlit, agrega unas sentencias básicas a un script de Python y ejecútalo con:

- `streamlit run tu_script.py [- argumentos del script]`
- o en módulo: `python -m streamlit run tu_script.py`

Tan pronto como se ejecuta el script, la aplicación se abrirá en una nueva pestaña de tu navegador web predeterminado.

D.5. Pruebas del sistema

Pruebas unitarias

Para el apartado de pruebas del proyecto se han realizado pruebas unitarias para cada subproyecto. Se han incluido en carpetas de `test` dentro de cada subproyecto, para que fuesen pruebas independientes entre sí. Se han comprobado la mayoría de las funciones realizadas, excluyendo algunas que realizan operaciones internas de las librerías de aprendizaje automático o de la web.

Se ha utilizado el marco de pruebas `pytest` integrado en Visual Studio Code para la ejecución de las pruebas:

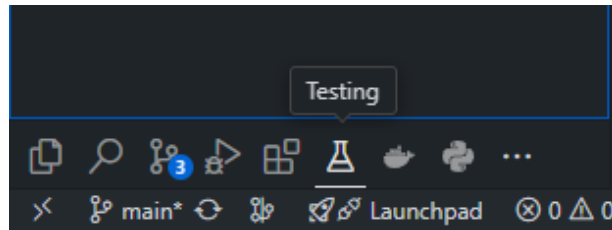


Figura D.10: Apartado de Testing de VSCode

Para la ejecución de las mismas de forma automática es necesario configurar desde el IDE donde se encuentran:

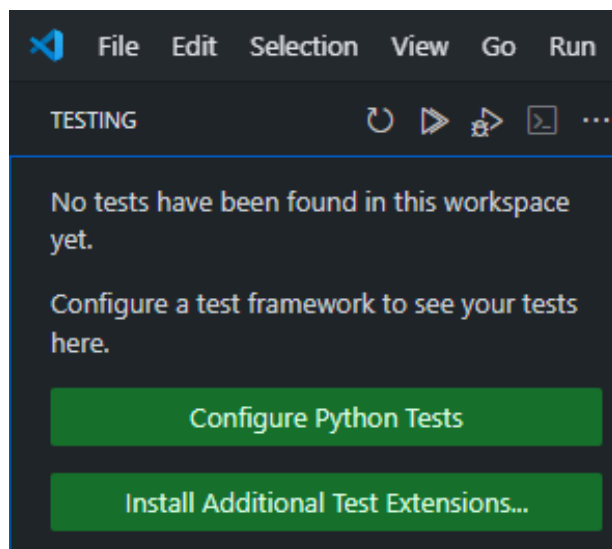


Figura D.11: Configurar Tests

Seleccionamos qué framework de pruebas se va a usar:

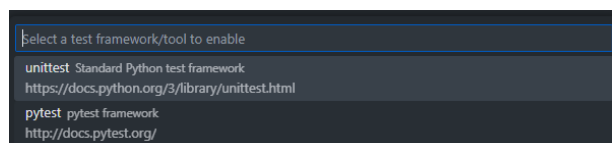


Figura D.12: Seleccionar pytest como framework de pruebas

Seleccionamos desde dónde se deben buscar los tests:

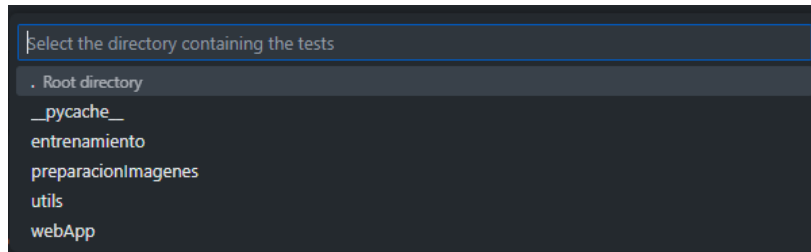


Figura D.13: Seleccionar el *root* del proyecto, para aglutinar todas las pruebas

Ahora ya podemos ejecutar los tests:

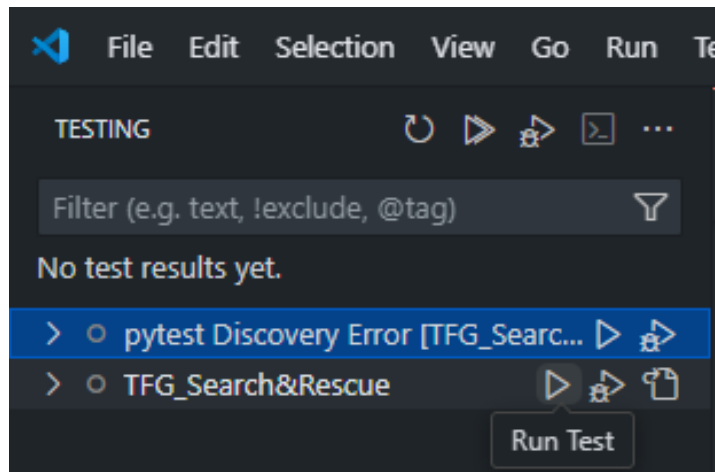


Figura D.14: Ejecutar tests

Se muestra como se van ejecutando uno a uno:

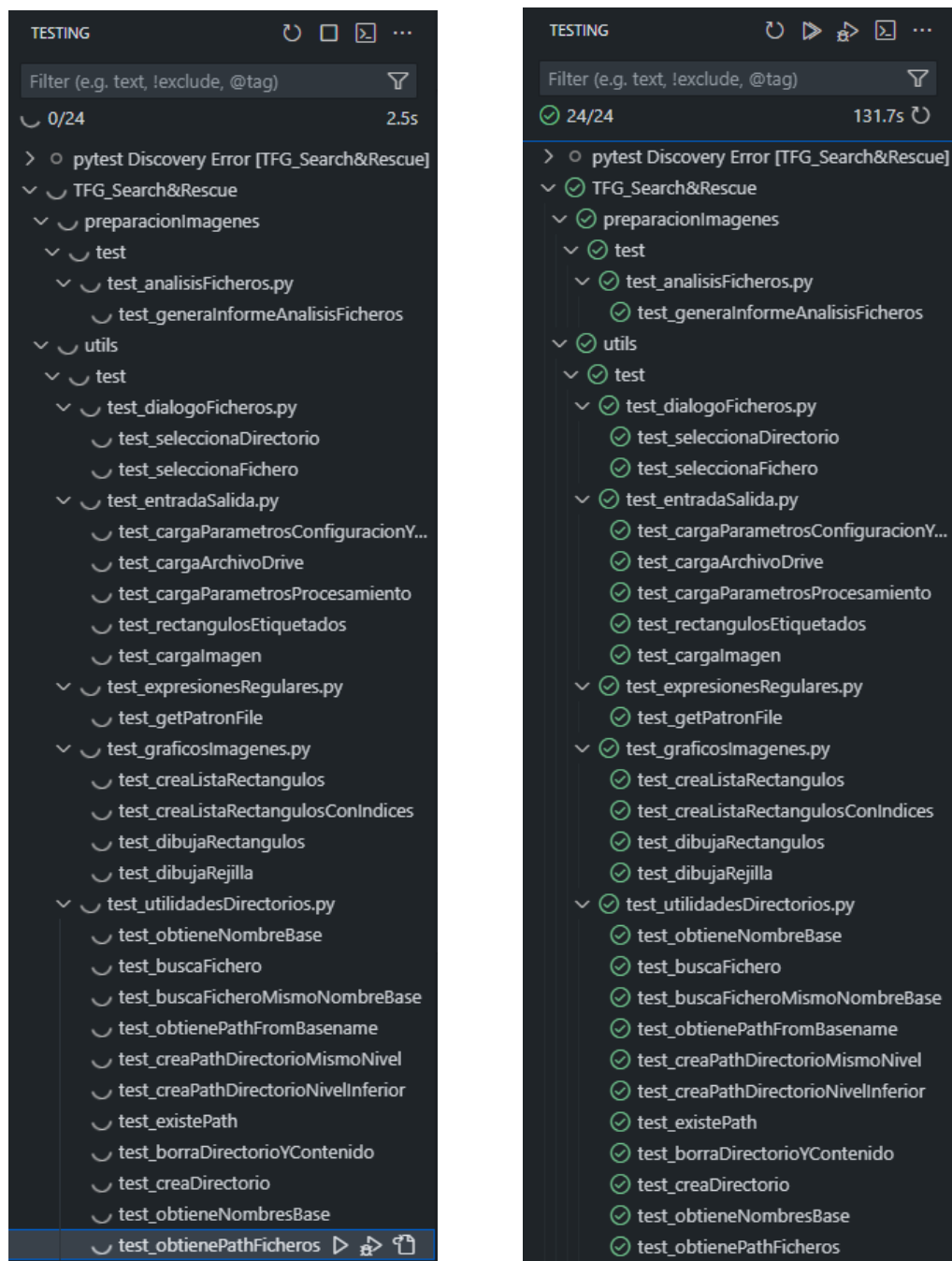
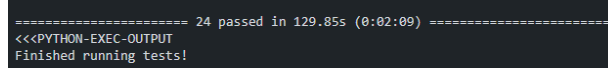


Figura D.15: Ejecución de los tests

También podemos comprobar por consola el resultado de la ejecución:

A screenshot of a terminal window with a dark background. The text displayed is white and consists of three lines: a top line with a long string of equals signs, the text '24 passed in 129.85s (0:02:09)', and another long string of equals signs; a second line starting with '<<<PYTHON-EXEC-OUTPUT'; and a third line that says 'Finished running tests!'.

```
===== 24 passed in 129.85s (0:02:09) =====  
<<<PYTHON-EXEC-OUTPUT  
Finished running tests!
```

Figura D.16: Salida por consola

Pruebas del usuario

Se ha comprobado la usabilidad del sistema enseñando la aplicación a diferentes perfiles de usuario, mostrando cual era el propósito del sistema y los pasos que se debían realizar. Se recogió la opinión de los usuarios tras el proceso, indicando las posibles mejoras de la aplicación en términos de interfaz y navegación.

Como conclusión, se determinó que el sistema era usable, aunque como todo, siempre se puede mejorar.

Apéndice E

Documentación de usuario

E.1. Introducción

En este apartado se explica todo lo relacionado con el usuario final incluyendo una guía completa de como usar la herramienta, concretamente la webApp desarrollada a modo de prueba de concepto.

E.2. Requisitos de usuarios

Para utilizar esta aplicación web, asegúrese de cumplir con los siguientes requisitos:

- **Conexión a Internet:** El usuario debe tener acceso a una conexión a Internet activa.
- **Capacidad de Conexión HTTP:** El navegador utilizado debe ser capaz de realizar conexiones HTTP. Esto incluye la mayoría de los navegadores web modernos como Google Chrome, Mozilla Firefox, Safari, Microsoft Edge...

No se requieren requisitos adicionales de hardware o software específico.

E.3. Instalación

No es necesario instalar nada para hacer uso de la aplicación, más allá de tener un navegador web y un explorador de archivos en el equipo.

E.4. Manual del usuario

Vamos a realizar un recorrido completo por la web, explicando cada funcionalidad desarrollada y cómo proceder para obtener resultados útiles en un proceso de búsqueda y rescate de personas en entornos naturales.

Acceso a la web

Para acceder a la aplicación web, podemos hacerlo desde el siguiente link:

<https://searchandrescue-tfg.streamlit.app/>

Página principal

Nada más acceder, nos encontramos con la pantalla de inicio:

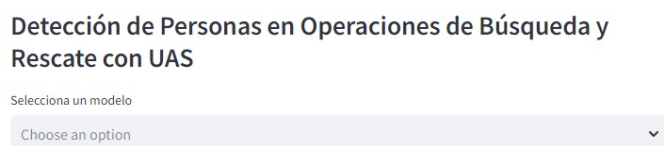


Figura E.1: Página principal

Selección de modelos

Deberemos elegir un modelo para realizar la predicción de las imágenes. La posibilidad de elección de varios modelos radica en tener modelos entrenados para un tipo específico de terreno, por si la búsqueda que se quiera hacer sea enfocada a unas u otras características.

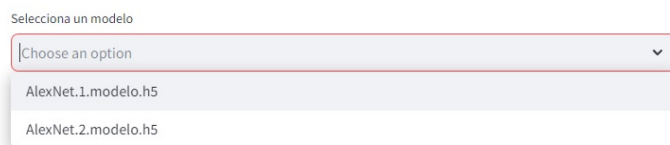


Figura E.2: Selección de modelo

Una vez seleccionado, el modelo se cargará a la aplicación. La primera vez que se carga puede tardar unos segundos, ya que el modelo tiene un

tamaño superior a 100MB. Las siguientes veces que se cargue será más rápido, pues se almacena en caché [9].



Figura E.3: Cargando modelo

Una vez que se ha cargado, aparecen dos nuevas funcionalidades, la selección de ajustes de predicción y la selección de imágenes a predecir.

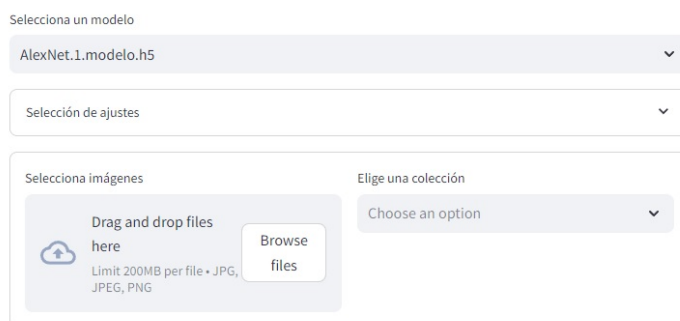


Figura E.4: Modelo cargado y nuevas funcionalidades

Ajuste de parámetros

Al desplegar la selección de ajustes, aparecen todas las opciones configurables para la predicción:



Figura E.5: Selección de parámetros

Umbral de decisión

Este umbral permite filtrar las predicciones del modelo en función de cuánto de seguro está el modelo de encontrar humanos. Un umbral muy bajo como 0.1 puede detectar tanto humanos como otros elementos de la imagen. Un umbral demasiado alto como 0.99 puede pasar por alto ciertos humanos en la imagen. Se recomienda establecer el umbral entre 0.8 y 0.9.

Solapamiento

La configuración del tamaño de solapamiento entre subimágenes es crucial, ya que si este no existiese, el recorte en subimágenes podría provocar objetos entrecortados, lo que afectaría en la predicción. Para una tarea de detección de humanos en imágenes aéreas como el que se ha entrenado la red, se recomienda un solapamiento en ambos ejes no inferior a 50 píxeles.

Márgenes

Los márgenes son los responsables de crear o no una nueva fila o columna al final de ambos ejes. Debido al solapamiento y al tamaño de las subimágenes, suele ocurrir que al final de los ejes haya píxeles sobrantes que no pertenecen a ninguna subimagen, por lo que si esos píxeles contienen un humano, nunca se detectarán. Para evitar que esto ocurra, basta con seleccionar ambos márgenes a 0, esto quiere decir que con que haya un píxel sobrante, se cogerá una nueva fila o columna.

Actualizar parámetros

Tras realizar cambios en los ajustes de predicción, se debe pulsar el botón de actualizar parámetros:

Un botón rectangular con esquinas redondeadas, con un borde rojo y el texto "Actualizar parámetros" en color rojo.

Figura E.6: Actualizar parámetros

Selección de imágenes

Podemos proceder a seleccionar imágenes desde:

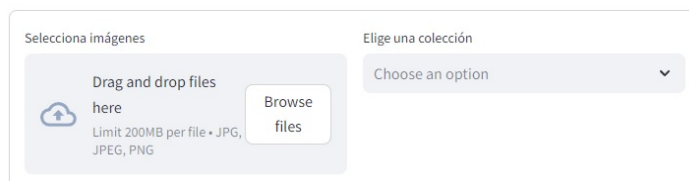


Figura E.7: Selección de imágenes

Podemos elegir la opción de cargar imágenes de nuestro equipo si las tenemos, o la opción de elegir colecciones de imágenes de muestra para ver como funciona el modelo.

Local

Para elegir imágenes desde local:

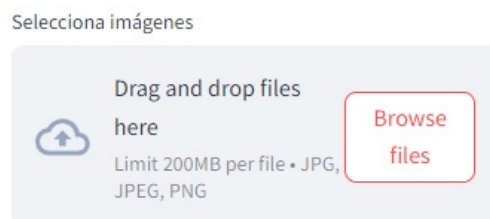


Figura E.8: Selección de imágenes locales

Elegimos desde el explorador de archivos (en este caso de Windows):

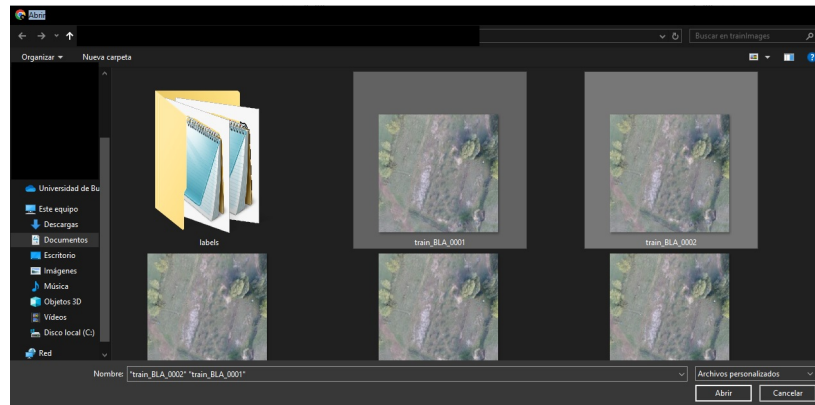


Figura E.9: Selección de imágenes locales desde el explorador de archivos

Se cargan las imágenes seleccionadas en la aplicación:

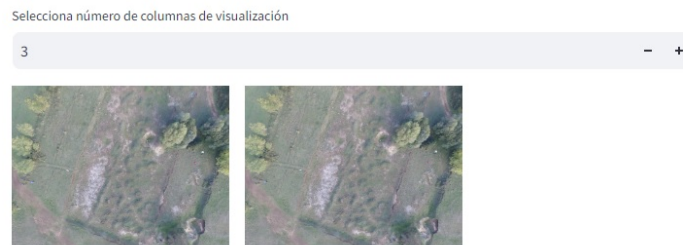


Figura E.10: Imágenes locales cargadas

Colecciones

Para elegir colecciones, desplegamos las distintas opciones y seleccionamos una:

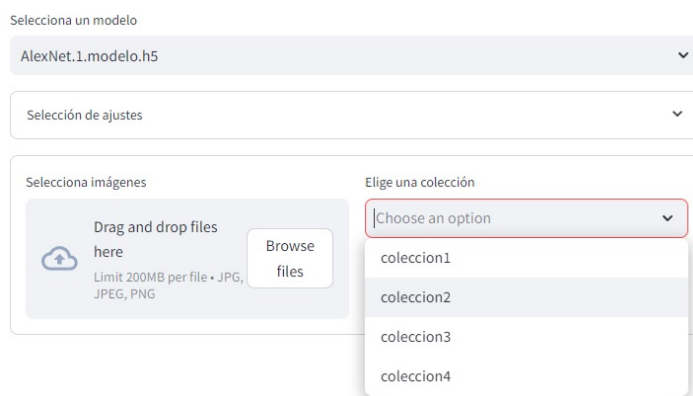


Figura E.11: Seleccionar colección

Una vez cargadas, podemos ajustar el número de columnas en las que se van a distribuir las imágenes cargadas. Esto es útil por si se cargasen muchas imágenes y quedasen muchas filas de imágenes en la interfaz.

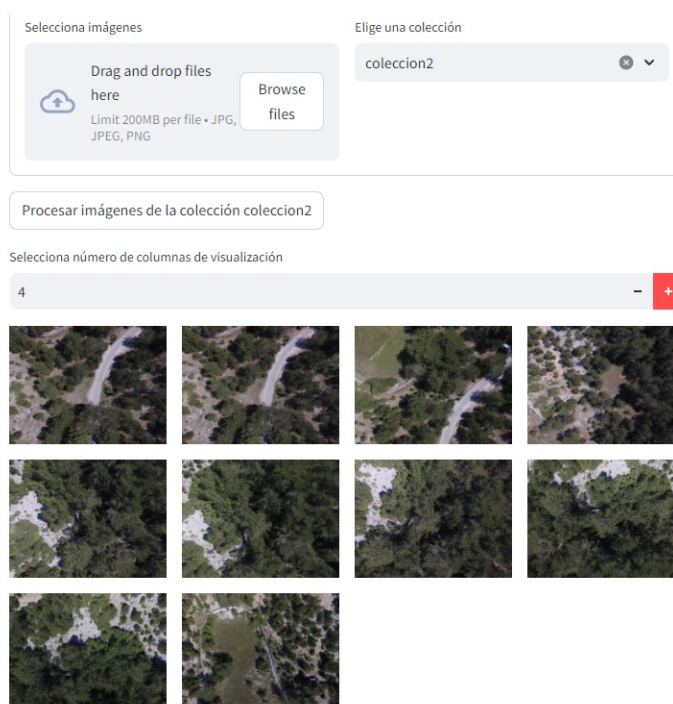


Figura E.12: Colección cargada

Predicción

Una vez que hemos elegido un modelo, se han ajustado los parámetros y se han cargado las imágenes, se puede hacer la predicción.

Para las imágenes en local:

Un botón rectangular con un borde rojo y un fondo blanco, que contiene el texto "Procesar imágenes seleccionadas localmente" en rojo.

Figura E.13: Procesar imágenes locales

Para las colecciones:

Un botón rectangular con un borde rojo y un fondo blanco, que contiene el texto "Procesar imágenes de la colección coleccion2" en rojo.

Figura E.14: Procesar predicción de la colección

La predicción tarda unos segundos por cada imagen. Dependiendo de los solapamientos elegidos tardará más o menos, ya que tendrá que realizar más subimágenes y predecirlas:



Figura E.15: Procesando predicción

Resultados

Cuando la predicción ha terminado, podemos ver los resultados.

Visualización

Podemos ver en recuadros rojos (referentes a las subimágenes) las regiones donde se han detectado humanos por el modelo.



Figura E.16: Imágenes procesadas

Podemos visualizarlo en pantalla completa desde:



Figura E.17: Opción de pantalla completa

Haciendo *click* vemos la imagen ampliada:



Figura E.18: Resultado de la predicción a pantalla completa

Descarga

Una vez visualizados los resultados, podemos descargarlos en formato CSV:

Descargar csv

Figura E.19: Descargar CSV de resultados

Al pulsar, se nos descarga en nuestro equipo:

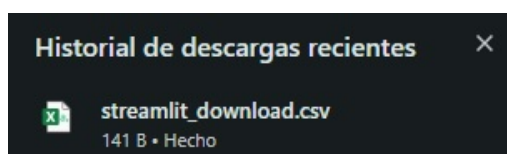


Figura E.20: CSV descargado

Podemos ver el contenido, por ejemplo desde Excel:

	A	B	C	D	E
1	Nombre, Latitud, Longitud				
2	train_BLA_0001.JPG, 43.35450252777778, 17.658015083333332				
3	train_BLA_0002.JPG, 43.35450205555556, 17.658014111111111				

Figura E.21: Vista del CSV de resultados

El archivo de resultados solo incluye aquellas imágenes en las que se ha detectado algún humano, ya que aquellas que no lo contienen no se consideran relevantes.

La inclusión de la latitud y longitud de la imagen es de gran utilidad para los equipos de búsqueda y rescate, pues les permite localizar de forma rápida donde se tomó la imagen.

Volver a inicio

Para volver a realizar una nueva predicción con unos nuevos ajustes, se puede hacer recargando la página o desde el botón:

Volver a página principal

Figura E.22: Volver a inicio

Anexo de sostenibilización curricular

F.1. Introducción

En este anexo se presenta una reflexión personal sobre los aspectos de sostenibilidad abordados en el desarrollo de mi proyecto de clasificación de imágenes con Redes Neuronales Convolucionales (CNNs) aplicadas a imágenes tomadas por un dron. Este proyecto no solo tiene implicaciones tecnológicas, sino que también toca aspectos fundamentales de sostenibilidad desde una perspectiva social, económica y ambiental, conforme a las directrices de sostenibilidad de la CRUE (Conferencia de Rectores de las Universidades Españolas).

F.2. Reflexión sobre la sostenibilidad en el proyecto

Sostenibilidad ambiental

El uso de drones para la toma de imágenes permite reducir significativamente la huella ecológica comparado con métodos tradicionales de recolección de datos, que podrían implicar el uso de vehículos terrestres o aéreos de mayor impacto ambiental. Los drones, siendo eléctricos, generan menos emisiones de carbono, promoviendo así un uso más sostenible de los recursos tecnológicos. Además, el análisis eficiente de imágenes mediante CNNs puede

optimizar el monitoreo de ecosistemas y la gestión ambiental, contribuyendo a la conservación de la biodiversidad y al control de desastres naturales.

Sostenibilidad económica

La implementación de drones y técnicas avanzadas de clasificación de imágenes ofrece una solución costo-efectiva para diversas industrias, desde la agricultura hasta la vigilancia ambiental. Al reducir costos operativos y mejorar la precisión en la recolección y análisis de datos, estas tecnologías pueden democratizar el acceso a herramientas avanzadas de monitoreo y gestión, facilitando el desarrollo económico sostenible en comunidades que de otro modo no podrían permitirse estos recursos.

Sostenibilidad social

El proyecto también aborda la sostenibilidad desde una perspectiva social al enfocarse en la búsqueda y rescate de personas desaparecidas o atrapadas en desastres naturales. Al emplear drones y CNNs, se mejora la eficiencia y rapidez en las operaciones de rescate, aumentando las posibilidades de encontrar y salvar vidas humanas.

F.3. Competencias de sostenibilidad adquiridas

Durante el desarrollo de este proyecto, he adquirido varias competencias clave en sostenibilidad, conforme a las directrices de la CRUE:

Competencia en la contextualización crítica del conocimiento (SOS1)

He desarrollado la capacidad de contextualizar críticamente el uso de tecnologías avanzadas en la problemática social, económica y ambiental. Entender las interrelaciones entre la tecnología y su impacto en el medio ambiente y la sociedad me ha permitido aplicar soluciones más holísticas y responsables en el desarrollo del proyecto.

Competencia en la utilización sostenible de recursos (SOS2)

El proyecto ha enfatizado el uso eficiente y sostenible de recursos tecnológicos, como la implementación de algoritmos eficientes que optimizan el uso de datos y recursos computacionales.

Competencia en la aplicación de principios éticos (SOS4)

He aprendido a aplicar principios éticos relacionados con los valores de la sostenibilidad en mi comportamiento profesional y personal. Esto incluye la responsabilidad hacia el medio ambiente, la equidad social y la toma de decisiones informadas y justas que promuevan un desarrollo sostenible.

F.4. Conclusión

El desarrollo de mi proyecto sobre la clasificación de imágenes con CNNs en imágenes tomadas por drones ha sido una experiencia enriquecedora que me ha permitido integrar conocimientos tecnológicos con principios de sostenibilidad.

Bibliografía

- [1] *Reglamento General de Protección de Datos (GDPR)*, 2016. Reglamento (UE) 2016/679 del Parlamento Europeo y del Consejo de 27 de abril de 2016.
- [2] Dunja Božić-Štulić, Željko Marušić, and Sven Gotovac. Deep learning approach on aerial imagery in supporting land search and rescue missions. *International Journal of Computer Vision*, 2019.
- [3] Félix de Miguel Villalba. Tfg_search-rescue. https://github.com/felixdmv/TFG_Search-Rescue, 2024.
- [4] Agencia Estatal de Seguridad Aérea (AESA). *Normativa sobre el uso de drones*, 2021. Disponible en: <https://www.seguridadaerea.gob.es/es/drones/normativa>.
- [5] Dora María Ballesteros Diego Renza. *Fundamentos de visión por computador utilizando aprendizaje profundo*. redipe, 2023.
- [6] NEELLAPALLI PALLAVI. Convolutional neural network(cnn) in deep learning. <https://medium.com/@pallavipallu37847/convolutional-neural-network-cnn-in-deep-learning-51c39a009898>.
- [7] Alex Punnen. Write and test your cnn with explainability. <https://medium.com/data-science-engineering/how-to-train-your-cnn-a8c84d108a77>, 2018.
- [8] Inc Snowflake. Install streamlit using command line. <https://docs.streamlit.io/get-started/installation/command-line>.

- [9] Karen Javadyan Tim Conkling and Johannes Rieke. Introducing two new caching commands to replace st.cache. <https://blog.streamlit.io/introducing-two-new-caching-commands-to-replace-st-cache/>.