



**Detección y análisis de datos sobre especies  
exóticas en biomas diferenciados en apoyo  
a la biodiversidad empleando la  
herramienta YOLO y microcontroladores**

**Estudiante:** Xabier Jiménez Gómez

A Coruña, julio de 2024.

*Dedicatoria*

## **Agradecimientos**

A mi familia, en especial a mi madre, el pilar más importante y sin el cual no podría haber llegado hasta donde estoy hoy. Gracias por su apoyo inquebrantable, por su respaldo económico y por ser mi modelo a seguir en mi vida. Le debo mis éxitos, tanto los presentes como los que vendrán en el futuro.

A mis amigos de la infancia, quienes han estado a mi lado desde que comencé mi viaje en la primaria y continúan a mi lado hasta el día de hoy. Sé que siempre puedo y podré contar con ustedes. A pesar de las adversidades, hemos logrado mantener nuestra amistad intacta a lo largo de los años.

A mis compañeros de carrera, ustedes han sido mi brújula, manteniéndome enfocado en el camino a seguir.

Por último, quiero expresar mi gratitud a todos los profesores que han compartido su conocimiento con pasión y entusiasmo. Han demostrado un compromiso y una dedicación inigualables en nuestra educación, especialmente aquellos que nos han llevado más allá de las páginas de los libros.

En palabras de Michael Jordan, 'He fallado una y otra vez en mi vida. Y es por eso que he tenido éxito'. Agradezco a mi familia, amigos y maestros por ayudarme a superar mis obstáculos y alcanzar el éxito en esta travesía que es la vida.

## **Resumen**

En la actualidad, se ha hablado mucho sobre las nuevas tecnologías que están transformando la sociedad y la forma en que vivimos. Entre estas tecnologías se encuentran el Internet de las Cosas, el Big Data, el Cloud Computing y la Inteligencia Artificial. Es fundamental comprender y explorar su potencial para anticipar cómo impactarán en nuestro futuro. En este proyecto, nos centraremos en una de estas tecnologías: la detección de objetos, una rama de la Inteligencia Artificial que busca permitir a las máquinas ver y comprender el mundo de la misma manera que lo hacemos nosotros a través de nuestros ojos.

Nuestro objetivo principal es utilizar la detección de objetos para identificar especies en peligro de extinción o que habitan en biomas de difícil estudio a partir de imágenes de video. Este enfoque tiene un propósito claro: contribuir a la preservación de la biodiversidad, pues el legado que debemos dejar a nuevas generaciones. Implementaremos este sistema en un dispositivo económico y versátil, la Raspberry Pi , que originalmente no fue diseñada con este propósito, para evaluar su rendimiento y determinar su viabilidad. En particular, utilizaremos una versión simplificada de YOLO, uno de los algoritmos más avanzados en detección de objetos en los últimos años, conocido como TinyYOLO. Nuestros objetivos incluyen la instalación y puesta en marcha exitosa del algoritmo, así como la evaluación de su desempeño en aspectos clave de la inteligencia artificial, como el entrenamiento y la inferencia.

A lo largo de esta memoria, exploraremos los fundamentos que rigen el campo de la detección de objetos, desde los principios básicos de la inteligencia artificial hasta el deep learning. Luego, detallaremos el proceso de integración del algoritmo TinyYOLO en la Raspberry Pi . Finalmente, llevaremos a cabo pruebas exhaustivas y análisis de datos que nos permitirán alcanzar las conclusiones y objetivos establecidos en este proyecto, todo con el propósito de avanzar en la conservación de la biodiversidad. En este proyecto, no solo nos limitaremos a implementar la detección de objetos en un dispositivo como la Raspberry Pi , sino que también crearemos un dataset de imágenes propio. Este conjunto de datos contendrá imágenes de las especies que estamos estudiando y en las que hemos realizado detecciones exitosas. Nos enfocaremos en identificar estas especies y recopilaremos datos valiosos sobre su comportamiento y hábitats.

A medida que avancemos en nuestro trabajo, compartiremos estos datos junto con las detecciones realizadas. Esto enriquecerá nuestra comprensión de estas especies, además de que también proporcionará información valiosa que puede ser utilizada en la preservación y conservación de la biodiversidad.

En línea con la filosofía de Michael Jordan, quien dijo: "He fallado una y otra vez en mi vida. Y es por eso que he tenido éxito", consideramos que aprender de nuestros errores y esfuerzos

en este proyecto nos llevará a lograr avances significativos en la protección de las especies en peligro y la comprensión de los ecosistemas vulnerables. El camino hacia la conservación de la biodiversidad comienza con la recopilación de datos precisos y la implementación de tecnologías innovadoras, y eso es precisamente lo que buscamos lograr aquí.

## **Abstract**

Currently, there has been a lot of talk about new technologies that are transforming society and the way we live. Among these technologies are the Internet of Things, Big Data, Cloud Computing and Artificial Intelligence. It is critical to understand and explore their potential in order to anticipate how they will impact our future. In this project, we will focus on one of these technologies: object detection, a branch of Artificial Intelligence that seeks to enable machines to see and understand the world in the same way we do through our eyes.

Our main goal is to use object detection to identify endangered species or species that inhabit difficult-to-study biomes from video images. This approach has a clear purpose: to contribute to the preservation of biodiversity, as the legacy we must leave to new generations. We will implement this system on an inexpensive and versatile device, the Raspberry Pi , which was not originally designed for this purpose, to evaluate its performance and determine its feasibility, in particular, we will use a simplified version of YOLO, one of the most advanced algorithms in object detection in recent years, known as TinyYOLO. Our objectives include the successful installation and implementation of the algorithm, as well as the evaluation of its performance in key aspects of artificial intelligence, such as training and inference.

Throughout this report, we will explore the fundamentals that govern the field of object detection, from the basics of artificial intelligence to deep learning. Then, we will detail the process of integrating the TinyYOLO algorithm on the Raspberry Pi. Finally, we will carry out extensive testing and data analysis that will allow us to reach the conclusions and objectives established in this project, all with the purpose of advancing biodiversity conservation. In this project, we will not only limit ourselves to implementing object detection on a device such as the Raspberry Pi , but we will also create an image dataset of our own. This dataset will contain images of the species we are studying and on which we have made successful detections. Not only will we focus on identifying these species, but we will also collect valuable data on their behavior and habitats.

As we progress in our work, we will share this data along with the detections made. This will not only enrich our understanding of these species, but will also provide valuable information that can be used in the preservation and conservation of biodiversity.

In line with the philosophy of Michael Jordan, who said, "I have failed over and over again in my life. And that is why I have succeeded", we believe that learning from our mistakes and

efforts in this project will lead to significant advances in the protection of endangered species and the understanding of vulnerable ecosystems. The road to biodiversity conservation begins with accurate data collection and the implementation of innovative technologies, and that is precisely what we seek to achieve here.

**Palabras clave:**

**Internet de las Cosas (IoT)**

**Big Data**

**Cloud Computing**

**Inteligencia Artificial (IA)**

**Detección de objetos**

**Biodiversidad**

**Raspberry Pi 3**

**TinyYOLO**

**Deep learning**

**Dataset.**

**Keywords:**

**Internet of Things (IoT)**

**Big Data**

**Cloud Computing**

**Artificial Intelligence (AI)**

**Object Detection**

**Biodiversity**

**Raspberry Pi 3**

**TinyYOLO**

**Deep Learning**

**Dataset**

# Índice general

---

<b>1</b>	<b>Introducción</b>	<b>1</b>
1.1	Introducción . . . . .	1
1.2	Objetivos . . . . .	4
<b>2</b>	<b>Metodología de trabajo</b>	<b>5</b>
2.1	Metodología Iterativa e Incremental . . . . .	5
2.2	Procedimiento del trabajo . . . . .	6
2.3	Estimaciones de planificación del proyecto . . . . .	7
2.3.1	Costes de Desarrollador . . . . .	7
2.3.2	Tiempos Empleados . . . . .	7
<b>3</b>	<b>Fundamento Teórico</b>	<b>10</b>
3.1	Inteligencia Artificial y Machine Learning . . . . .	10
3.2	Redes neuronales . . . . .	11
3.2.1	Redes neuronales convolucionales . . . . .	12
3.3	Métodos de detección clásicos . . . . .	13
3.4	YOLO . . . . .	14
3.5	Microcontroladores . . . . .	16
3.6	Tecnologías y herramientas . . . . .	18
3.6.1	Pycharm . . . . .	18
3.6.2	Labelme . . . . .	19
3.6.3	Json . . . . .	19
3.6.4	MongoDB . . . . .	20
3.6.5	Power BI . . . . .	21
3.6.6	Ultralytics . . . . .	22
3.6.7	Google Colab . . . . .	22

---

<b>4 Desarrollo: Parte 1</b>	<b>23</b>
4.1 Contextualización . . . . .	23
4.1.1 Fantasma del Himalaya: Individuo y Especie . . . . .	25
4.2 Dataset de imágenes . . . . .	30
4.3 Primeras iteraciones Yolov8 . . . . .	32
4.3.1 Primera ejecución . . . . .	32
4.3.2 Segunda ejecución . . . . .	35
4.3.3 Tercera ejecución . . . . .	36
<b>5 Desarrollo: Parte 2</b>	<b>39</b>
5.1 Hapalochlaena maculosa . . . . .	39
5.2 Panthera Tigris Tigris . . . . .	40
5.3 Ailuropoda melanoleuca . . . . .	41
5.4 Ejecución del dataset final . . . . .	42
5.5 GPU vs CPU . . . . .	42
5.6 Pruebas finales . . . . .	43
5.7 Resultados finales, validación y discusión . . . . .	49
<b>6 Análisis de datos</b>	<b>55</b>
6.1 Integración de MongoDB . . . . .	55
6.2 Distinción de ruido y falsas detecciones . . . . .	57
6.3 Almacenamiento de geoposiciones . . . . .	57
6.4 Análisis de información . . . . .	57
<b>7 Prueba de campo</b>	<b>61</b>
7.1 Microcontrolador . . . . .	62
7.2 Ejecución en entorno real . . . . .	63
7.2.1 Consideraciones . . . . .	65
<b>8 Conclusiones</b>	<b>66</b>
<b>9 Trabajo Futuro</b>	<b>68</b>
<b>A Material adicional</b>	<b>73</b>
A.1 Reconocimiento recursos visuales . . . . .	73
<b>B Códigos utilizados</b>	<b>74</b>
B.1 Detección sin MongoDB . . . . .	74
B.2 Detección con MongoDB . . . . .	75
B.3 Prueba para imágenes . . . . .	76

---

B.4	Paso de base64 a imagen . . . . .	77
B.5	Rutas para el dataset . . . . .	78
B.6	Algoritmo en imágenes de un directorio . . . . .	78
B.7	Código final . . . . .	79
<b>C</b>	<b>Otros</b>	<b>82</b>
C.1	Presupuesto . . . . .	82
C.2	Histórico de pruebas de entrenamiento . . . . .	82
C.3	Datasets de acceso gratuito . . . . .	83
<b>Lista de acrónimos</b>		<b>86</b>
<b>Bibliografía</b>		<b>88</b>

# Índice de figuras

---

1.1	Attenborough en una de sus primeras expediciones . . . . .	3
2.1	Metodología Iterativa e Incremental . . . . .	6
3.1	Red neuronal artificial estándar . . . . .	12
3.2	Algoritmo You Only Look Once . . . . .	14
3.3	Comparativa versiones . . . . .	16
3.4	Esquema Raspberry Pi . . . . .	18
4.1	. . . . .	23
4.2	Detección y segmentación . . . . .	24
4.3	Panthera uncia . . . . .	25
4.4	Hábitat del leopardo de las nieves . . . . .	26
4.5	Ciclo actual del hábitat . . . . .	27
4.6	Distribución en el territorio . . . . .	27
4.7	Análisis envoltura climática . . . . .	28
4.8	Imagen de cámara trampa de individuo en movimiento . . . . .	28
4.9	Identificaciones propias del pelaje . . . . .	29
4.10	Ejemplo Labelme . . . . .	31
4.11	Segundo Ejemplo Labelme . . . . .	31
4.12	Comparación métricas de Yolov8 . . . . .	32
4.13	Algunos de los resultados de entrenamiento que exporta Yolov8 . . . . .	34
4.14	Ejecución de un video descargado de internet . . . . .	35
4.15	Métricas generales de la segunda iteración . . . . .	36
4.16	Resultado tercera iteración . . . . .	37
4.17	Métricas generales de la tercera iteración . . . . .	38
5.1	Hapalochlaena maculosa . . . . .	40

5.2	Panthera Tigris Tigris captado por camara trampa en la India . . . . .	41
5.3	Ailuropoda melanoleuca . . . . .	42
5.4	Entorno de escritorio remoto en Google Colab . . . . .	43
5.5	Grafica iteración Google Colab . . . . .	44
5.6	Grafica resultados Google Colab . . . . .	44
5.7	Primera predicción . . . . .	45
5.8	Segunda predicción . . . . .	45
5.9	Tercera predicción . . . . .	46
5.10	Ejemplo de detección en vídeo del leopardo de las nieves . . . . .	46
5.11	Ejemplo de detección en vídeo del tigre de bengala . . . . .	47
5.12	Ejemplo en imagen . . . . .	47
5.13	Nuevos resultados . . . . .	48
5.14	Primera predicción . . . . .	48
5.15	Segunda predicción . . . . .	49
5.16	Tercera predicción . . . . .	49
5.17	Métricas para evaluar el desempeño de un modelo . . . . .	50
5.18	F1-Confidence Curve . . . . .	51
5.19	Precision-Confidence Curve . . . . .	52
5.20	Precision-Recall Curve . . . . .	53
5.21	Resultados por época . . . . .	53
5.22	Factores que pueden generar errores en la clasificación . . . . .	54
6.1	Datos guardados en MongoDB . . . . .	56
6.2	Informe Visual creado con Power Bi . . . . .	58
6.3	Configuración en Mongo Atlas . . . . .	59
6.4	Informe Visual para el lince ibérico . . . . .	60
7.1	Naturebytes wildlife technology kit . . . . .	61
7.2	Raspberry Pi en la que se han ejecutado las pruebas . . . . .	62
7.3	Fuente de energía . . . . .	63
7.4	Cámara de 5MP . . . . .	63
7.5	Menú de configuración . . . . .	64
9.1	Boceto del cuaderno de bitácora de Percy Fawcet . . . . .	69
9.2	Piloto controlando en directo un ROV . . . . .	69
9.3	Ejemplo de ROV . . . . .	70
9.4	. . . . .	71

# Índice de tablas

---

2.1	Estimación de Costes de Desarrollador . . . . .	9
2.2	Tiempos Empleados en el Proyecto . . . . .	9
3.1	Comparativa mediante ChatGPT4 . . . . .	15
3.2	Comparativa versiones Raspberry Pi . . . . .	17
C.1	Presupuesto estimado Total=80.17€ . . . . .	82
C.2	Histórico de entrenamientos . . . . .	84
C.3	Bases de datos de libre acceso con información ambiental . . . . .	85

# Capítulo 1

## Introducción

---

ESTE primer capítulo sirve como breve introducción al proyecto de Detección de Objetos con TinyYOLO. Se justifica su elección e interés, y se describen resumidamente los objetivos y la finalidad de éste, centrándose primero en los conceptos teóricos del proyecto, y finalizando con las elecciones concretas del algoritmo y la placa/microcontrolador a estudiar. A continuación, el segundo capítulo se encargará de desarrollar el fundamento teórico detrás de la principal tecnología del proyecto: la visión artificial basada en inteligencia artificial y redes neuronales. Al ser todos estos conceptos relativamente modernos, se tratará de explicar cada uno de la forma más precisa posible, pero a la vez siendo concisos y enfocándonos en la tecnología final utilizada y en el campo en el que se aplica. Finalmente, los capítulos posteriores tratarán sobre la implementación del modelo elegido, y los resultados obtenidos de los diferentes experimentos, así como una conclusión y breve esquema de los pasos que se podrían seguir en futuras investigaciones.

### 1.1 Introducción

Desde tiempos ancestrales, la humanidad ha demostrado una inclinación innata hacia la superación y la búsqueda constante de sobrepasar sus límites. La invención de herramientas y utensilios para simplificar las tareas cotidianas es un testimonio claro de esta inclinación hacia la eficiencia. Sin embargo, lo que nos distingue como especie es nuestra capacidad para llevar esta búsqueda al siguiente nivel: la automatización y superación de nuestras propias habilidades.

El siglo XX presenció una transformación significativa en este sentido, con la llegada de la informática y las computadoras, desencadenando una vertiginosa carrera tecnológica en la que cada día parecíaemerger una innovación sorprendente. El siglo XXI es una continuación natural de este impulso imparable. Con un poder de procesamiento informático inimaginable hace solo unas décadas, tecnologías destinadas a revolucionar nuestro presente y futuro se

vislumbran en el horizonte, al igual que lo hizo la informática en el siglo pasado.

Conceptos como el Internet de las Cosas, el Big Data, la Computación en la Nube o la Inteligencia Artificial se alzan como los protagonistas indiscutibles de esta era. En este proyecto, exploramos el fascinante mundo de la visión artificial, centrándonos en particular en la detección de objetos.

La visión artificial se define como la capacidad de una máquina para ver e interpretar su entorno de manera similar a un ser humano a través de sus ojos. A lo largo de su evolución, se han explorado diversos enfoques, desde el procesamiento digital de imágenes hasta el reconocimiento de patrones. No obstante, el enfoque que abordamos aquí es aquel que se vale de la inteligencia artificial y el aprendizaje automático. En esencia, no buscamos simplemente programar una máquina para que compare imágenes y proporcione un porcentaje de similitud, sino que aspiramos a enseñarle a ver. Después de que aprenda a reconocer objetos en unas pocas imágenes, nuestra meta es que pueda identificar los mismos objetos en cualquier otro contexto.

La utilidad de la visión artificial es innegable, abarcando desde la vigilancia inteligente hasta la conducción autónoma, pasando por la medicina con diagnósticos inteligentes y la detección de enfermedades, así como la investigación científica, como la relacionada con la pandemia de COVID-19 en 2020. La visión desempeña un papel fundamental en la vida humana, y una máquina con la capacidad de ver y comprender el mundo como nosotros supone una auténtica revolución tecnológica.

Sin embargo, alcanzar este objetivo no es tarea sencilla y requiere avances tecnológicos significativos. En este proyecto, nos enfocamos en uno de los componentes clave de la visión artificial: la detección de objetos. Esto implica que una máquina sea capaz de identificar y reconocer objetos de diversas clases (como personas, vehículos o señales de tráfico) en vídeos o imágenes digitales, con la máxima precisión y seguridad posible. Para lograrlo, emplearemos la familia YOLO (acrónimo de "You Only Look Once"), un conjunto de sistemas de detección de objetos de última generación, que exploraremos en detalle más adelante.

Además de la detección de objetos, abordaremos otro desafío importante relacionado con la inteligencia artificial: el procesamiento computacional. Estos sistemas requieren una gran cantidad de operaciones matemáticas, a menudo superando la capacidad de las computadoras convencionales. Para enfrentar este desafío, implementaremos nuestra solución en un dispositivo computador de propósito general, accesible y fácil de utilizar: una placa computadora, también conocida como "single-board computer" [1]. En el contexto de este proyecto, es importante destacar cómo la tecnología puede desempeñar un papel fundamental en la preservación de la biodiversidad y la protección de nuestro planeta de una manera ecológica. La aplicación de la tecnología en la conservación ambiental ha demostrado ser un factor crucial para abordar los desafíos ambientales a los que nos enfrentamos.

David Attenborough Figura 1.1, en su conmovedor documental sobre la tecnología en la naturaleza, nos revela cómo las innovaciones tecnológicas pueden ayudar a monitorear y proteger la biodiversidad de nuestro planeta de manera más eficiente y precisa que nunca antes. Proyectos como la utilización de drones para la vigilancia de poblaciones de vida silvestre o el uso de sensores remotos para el seguimiento de ecosistemas frágiles han demostrado ser herramientas valiosas en la conservación de especies en peligro de extinción y la preservación de hábitats críticos.



Figura 1.1: Attenborough en una de sus primeras expediciones

La combinación de la visión artificial, la inteligencia artificial y el procesamiento de datos puede proporcionar soluciones innovadoras para abordar los desafíos ambientales [2]. Mediante el análisis y el muestreo de datos, podemos obtener información valiosa sobre el comportamiento de las especies, los patrones de migración y la salud de los ecosistemas. Esta información es esencial para la toma de decisiones informadas en la conservación y la gestión sostenible de nuestros recursos naturales.

En este proyecto, no solo exploramos la tecnología en el contexto de la visión artificial y la detección de objetos, sino que también reconocemos su potencial para contribuir a la

biodiversidad y la sostenibilidad de nuestro planeta. La capacidad de utilizar la tecnología para monitorear y proteger la vida silvestre y los ecosistemas es una herramienta poderosa en nuestra lucha por preservar la belleza y la diversidad de la naturaleza en la que todos dependemos.

## 1.2 Objetivos

Los objetivos de este trabajo se enfocan en la preservación y conservación de la vida silvestre y el medio ambiente mediante el uso de la inteligencia artificial y la tecnología avanzada para monitorizar y proteger la biodiversidad a nivel global. Además, busca desarrollar herramientas y soluciones innovadoras para combatir la caza furtiva, la deforestación y la degradación del hábitat, al tiempo que promueve la conciencia pública sobre los problemas ambientales y la importancia de conservar la vida silvestre.

Los objetivos se centran en:

- Verificar la correcta implementación de TinyYOLOv en un microcontrolador.
- Crear un conjunto de datos propio para el proyecto.
- Evaluar el rendimiento y optimizar la eficacia del sistema.
- Realizar un estudio detallado y trabajar con los datos relacionados con las especies.
- Llevar a cabo pruebas de campo para validar el sistema en un entorno real.

## Capítulo 2

# Metodología de trabajo

---

**E**n esta sección, se detalla la metodología utilizada para llevar a cabo el desarrollo del proyecto. Se ha adoptado una Metodología Iterativa e Incremental [3] para asegurar un enfoque sistemático y progresivo en cada etapa. Esta metodología se alinea estrechamente con los objetivos delineados en los diferentes capítulos del trabajo, proporcionando un marco estructurado para la ejecución de las tareas planificadas.

El Capítulo 2, centrado en la metodología de trabajo, se estructura en tres partes principales: Metodología Iterativa e Incremental, Procedimiento del trabajo y estimaciones de planificación del proyecto .

### 2.1 Metodología Iterativa e Incremental

En el desarrollo de este proyecto, se adoptará una Metodología Iterativa e Incremental [3] para asegurar un enfoque sistemático y progresivo en cada etapa. Esta metodología (Figura 2.1) se alinea estrechamente con los objetivos delineados en los diferentes capítulos del trabajo, proporcionando un marco estructurado para la ejecución de las tareas planificadas.

En el Capítulo 3, centrado en evaluar el rendimiento y optimizar la eficacia del modelo para la detección de leopardos de las nieves, se implementará la primera iteración del proceso. Esto implicará la recopilación y etiquetado de un conjunto inicial de imágenes específicas de esta especie, seguido de la adaptación y entrenamiento del modelo YOLO. La retroalimentación obtenida de esta iteración inicial informará los ajustes necesarios para perfeccionar la precisión del modelo.

El Capítulo 4 dará paso a la siguiente iteración, donde se ampliará la prueba del modelo para incluir otras especies. Este enfoque incremental permitirá evaluar la capacidad del modelo para generalizar la detección a diferentes contextos, mejorando así su versatilidad y aplicabilidad en escenarios más amplios.

El Capítulo 5 se centrará en realizar un estudio detallado de los resultados obtenidos,

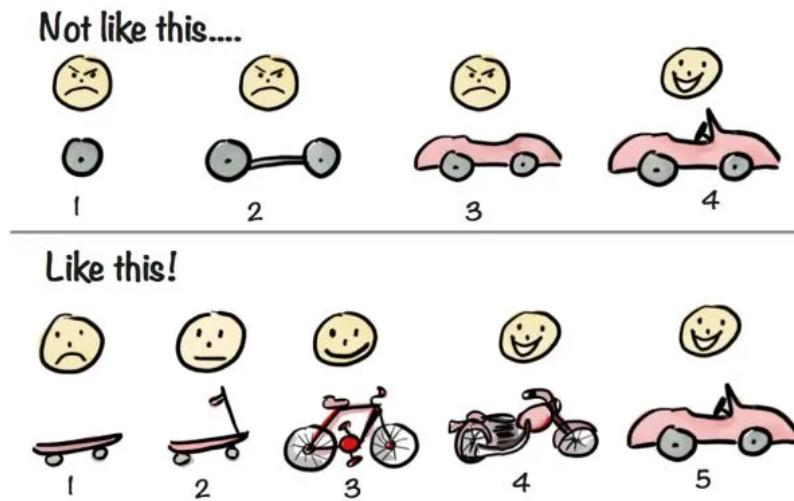


Figura 2.1: Metodología Iterativa e Incremental

utilizando diferentes formas de análisis, incluyendo la creación de una base de datos. Esta iteración proporcionará una comprensión más profunda de las capacidades y limitaciones del modelo, permitiendo ajustes específicos según sea necesario.

A medida que avanzamos en las iteraciones, los resultados y aprendizajes obtenidos se integrarán en el Capítulo 8, donde se presentarán las Conclusiones y el Trabajo Futuro. Esta metodología iterativa garantizará una evolución constante del proyecto, maximizando la eficiencia y la efectividad en la consecución de los objetivos establecidos.

La elección de la Metodología Iterativa e Incremental se justifica frente a alternativas como la Metodología en Cascada. A diferencia de la rigidez lineal de la Metodología en Cascada, la Iterativa e Incremental se destaca por su flexibilidad y adaptabilidad, permitiendo ajustes continuos a medida que se obtienen nuevos conocimientos y se enfrentan desafíos específicos. La Metodología en Cascada, al seguir una secuencia lineal de fases, carece de la capacidad de proporcionar retroalimentación temprana y ajustes oportunos, lo cual es esencial en proyectos dinámicos como la detección de objetos. Su enfoque de entregables al final del proyecto y respuesta limitada a cambios en los requisitos del usuario hacen que la Metodología Iterativa e Incremental sea la elección más adecuada para maximizar la adaptabilidad y eficacia en la consecución de los objetivos establecidos en este proyecto.

## 2.2 Procedimiento del trabajo

El presente proyecto se encuentra estructurado en seis capítulos, a continuación, se detalla un pequeño resumen de su contenido:

- Capítulo 1 Introducción: En él se describe motivación y objetivos del proyecto, así

como la estructura del mismo.

- Capítulo 2 Estado del arte: Se hace un repaso de los métodos y aplicativos utilizados, indagando en concreto en la detección de objetos con Yolo.
- Capítulo 3 Tema base: Evaluar el rendimiento y optimizar la eficacia del modelo en el sujeto del cual se basa el proyecto, el leopardo de las nieves del cual se ha generado un dataset de imágenes.
- Capítulo 4 Pruebas: Realizar pruebas incluyendo otras especies.
- Capítulo 5 Análisis de datos: Realizar un estudio detallado de los resultados mediante diferentes formas, incluyendo una base de datos.
- Capítulo 6 Hardware y software complementario: Componentes y microcontroladores.
- Capítulo 7 Prueba de campo: Llevar a cabo pruebas de campo para validar el sistema en un entorno real.
- Capítulo 8 Final: Conclusiones y trabajo futuro.

## 2.3 Estimaciones de planificación del proyecto

En este apartado se presentan las estimaciones de planificación del proyecto. Este proyecto incluye desarrollo de código en Python, almacenamiento de datos en una base de datos MongoDB, y generación de informes en Power BI. Además, el dataset se ha creado de forma propia utilizando Google Colab. Las estimaciones abarcan tanto los costes de los desarrolladores como los tiempos empleados.

### 2.3.1 Costes de Desarrollador

Para la estimación de los costes de desarrollador, se ha considerado una tarifa horaria estándar para desarrolladores con experiencia en Python, manejo de bases de datos NoSQL (MongoDB), y uso de herramientas de análisis y visualización de datos (Power BI).

### 2.3.2 Tiempos Empleados

A continuación, se detallan los tiempos empleados en cada fase del proyecto, considerando tanto la planificación como el desarrollo y la implementación de cada componente del sistema.

- **Diseño y Planificación (30 horas):** Incluye reuniones iniciales, definición de requisitos, y elaboración del plan de trabajo. - **Desarrollo de Código en Python (100 horas):** Programación de scripts para la detección de especies, integración con YOLOv8, y desarrollo

de funciones de almacenamiento en MongoDB. - **Configuración de Raspberry Pi (15 horas):** Instalación y configuración del sistema operativo, configuración de la cámara trampa, y pruebas de conectividad. - **Implementación de YOLOv8 (50 horas):** Entrenamiento del modelo, ajuste de hiperparámetros, e integración con el sistema de detección. - **Creación y Gestión del Dataset (60 horas):** Recolección de imágenes, etiquetado de datos, y gestión del dataset en Google Colab. - **Configuración y Mantenimiento de la Base de Datos MongoDB (30 horas):** Diseño del esquema de la base de datos, configuración del servidor, y mantenimiento. - **Generación de Informes en Power BI (40 horas):** Creación de dashboards interactivos, visualización de datos y elaboración de informes. - **Pruebas y Validación (25 horas):** Verificación del sistema, pruebas de rendimiento, y validación final del proyecto.

El proyecto contempla un total de 350 horas de trabajo (Tabla 2.2), con un coste total estimado de €16,100 EUR (Tabla 2.1). Este presupuesto cubre todas las fases del proyecto, desde la planificación inicial hasta la entrega final de los informes en Power BI.

Fase del Proyecto	Horas Estimadas	Coste Total Estimado (EUR)
Diseño y Planificación	30	€1,380
Desarrollo de Código en Python	100	€4,600
Configuración de Raspberry Pi	15	€690
Implementación de YOLOv8	50	€2,300
Creación y Gestión del Dataset	60	€2,760
Configuración y Mantenimiento de DB	30	€1,380
Generación de Informes en Power BI	40	€1,840
Pruebas y Validación	25	€1,150
<b>Total</b>	<b>350</b>	<b>€16,100</b>

Tabla 2.1: Estimación de Costes de Desarrollador

Fase del Proyecto	Horas Empleadas
Diseño y Planificación	30
Desarrollo de Código en Python	100
Configuración de Raspberry Pi	15
Implementación de YOLOv8	50
Creación y Gestión del Dataset	60
Configuración y Mantenimiento de DB	30
Generación de Informes en Power BI	40
Pruebas y Validación	25
<b>Total</b>	<b>350</b>

Tabla 2.2: Tiempos Empleados en el Proyecto

## **Capítulo 3**

# **Fundamento Teórico**

---

**E**n este capítulo se presenta el fundamento teórico necesario para comprender los conceptos clave relacionados con la detección de animales mediante cámaras trampa. Se exploran las bases de la visión por computadora, el aprendizaje automático, la inteligencia artificial e introducciones a las herramientas utilizadas.

### **3.1 Inteligencia Artificial y Machine Learning**

La inteligencia artificial (IA) y el aprendizaje automático (Machine Learning, en inglés) representan dos campos interrelacionados que han revolucionado la forma en que las computadoras procesan la información y toman decisiones. La IA se refiere a la capacidad de las máquinas para realizar tareas que normalmente requieren inteligencia humana, como el razonamiento, la planificación y el reconocimiento de patrones. Dentro de la IA, el aprendizaje automático es una subdisciplina que se centra en desarrollar algoritmos y modelos capaces de mejorar su rendimiento a medida que se enfrentan a más datos.

Uno de los enfoques fundamentales en el aprendizaje automático es el aprendizaje supervisado, donde los modelos se entrena utilizando conjuntos de datos etiquetados, lo que significa que se proporciona información de entrada junto con la salida deseada. Por otro lado, el aprendizaje no supervisado implica trabajar con datos no etiquetados, permitiendo que el modelo descubra patrones y relaciones por sí mismo. Estos enfoques han dado lugar a avances significativos en áreas como reconocimiento de voz, visión por computadora y procesamiento del lenguaje natural.

La adopción de algoritmos de aprendizaje profundo ha impulsado aún más el desarrollo de la IA. Estos algoritmos, inspirados en la estructura del cerebro humano, utilizan redes neuronales profundas para procesar información de manera jerárquica, permitiendo la extracción de características complejas. Esto ha llevado a mejoras significativas en la capacidad de las máquinas para comprender imágenes, traducir idiomas y predecir patrones en grandes

conjuntos de datos.

Sin embargo, la implementación exitosa de la inteligencia artificial también plantea desafíos éticos y sociales. Se deben abordar cuestiones como la privacidad, la discriminación algorítmica y la toma de decisiones automatizada. La investigación continua y el desarrollo ético son fundamentales para garantizar que la inteligencia artificial y el aprendizaje automático beneficien a la sociedad en su conjunto, sin comprometer los valores humanos y los derechos fundamentales. En resumen, la combinación de la inteligencia artificial y el aprendizaje automático está transformando profundamente la forma en que interactuamos con la tecnología y plantea preguntas importantes sobre su impacto futuro en la sociedad.

En el contexto del aprendizaje automático, los algoritmos de inteligencia artificial se estructuran para aprender patrones y realizar tareas específicas a través de la exposición a conjuntos de datos. Uno de los paradigmas más utilizados es el aprendizaje supervisado, donde el modelo se entrena con datos previamente etiquetados, ajustando sus parámetros para minimizar la diferencia entre las predicciones y las salidas reales. Este proceso implica la optimización iterativa de funciones de pérdida mediante algoritmos como el descenso de gradiente. Además, el aprendizaje no supervisado se centra en descubrir patrones intrínsecos en datos no etiquetados, utilizando técnicas como la reducción de dimensionalidad y el clustering.

El aprendizaje profundo, una rama del aprendizaje automático, se destaca por el uso de redes neuronales artificiales con múltiples capas (redes neuronales profundas). Estas redes permiten la representación jerárquica de características, lo que facilita la extracción de información compleja. Los modelos de aprendizaje profundo, entrenados con grandes conjuntos de datos, han demostrado notables avances en tareas como reconocimiento de imágenes, procesamiento del lenguaje natural y juegos estratégicos. Sin embargo, es fundamental abordar los desafíos asociados, como la interpretabilidad de los modelos y la necesidad de conjuntos de datos éticos para evitar sesgos y discriminación.

## 3.2 Redes neuronales

Las redes neuronales, también conocidas como redes neuronales artificiales o NNs (por sus siglas en inglés, neuronal networks), representan un modelo computacional que se inspira parcialmente en las complejas redes neuronales biológicas presentes en el cerebro de los animales. Esta aproximación se ha revelado como una estrategia acertada para emular el proceso de aprendizaje del cerebro humano, entre otras funciones.

Una red neuronal artificial está compuesta por una serie de elementos fundamentales, las neuronas artificiales, que están interconectadas de manera que forman una red capaz de abordar problemas específicos. Lo más destacado es que estas redes tienen la capacidad de aprender a resolver dichos problemas con el tiempo. La estructura básica de una red neuronal

artificial se puede visualizar en la figura 3.1.

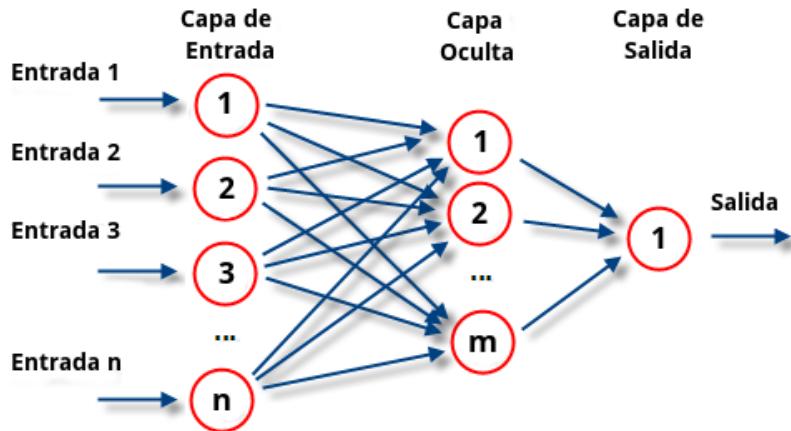


Figura 3.1: Red neuronal artificial estándar

### 3.2.1 Redes neuronales convolucionales

Las Redes Neuronales Convolucionales [4] ([Convolutional Neural Network \(CNN\)](#)) son una clase especializada de redes neuronales profundas diseñadas para procesar y analizar datos de tipo gráfico, como imágenes. Estas redes han demostrado ser extremadamente efectivas en tareas de visión por computadora, donde la interpretación de patrones visuales es esencial. La arquitectura de las [CNN](#) se inspira en la organización del sistema visual biológico, utilizando capas convolucionales para aprender automáticamente características jerárquicas y complejas de las imágenes.

Una característica fundamental de las [CNN](#) es su capacidad para realizar convoluciones, operaciones matriciales que permiten detectar patrones locales en una imagen. Estas convoluciones se aplican a través de varias capas, extrayendo gradualmente características de mayor nivel a medida que la red se profundiza. Además, las capas de agrupación (pooling) reducen la dimensionalidad espacial, preservando las características más relevantes. Esta combinación de convoluciones y capas de agrupación permite a las [CNN](#) aprender representaciones cada vez más abstractas y significativas de los datos visuales.

En el contexto de la memoria, las [CNN](#) se han destacado en la tarea de reconocimiento de objetos, clasificación de imágenes y detección de patrones visuales complejos. La capacidad de generalización de las [CNN](#) les permite reconocer objetos en nuevas imágenes, incluso en condiciones variables. Sin embargo, debido a su profundidad y complejidad, las [CNN](#) también requieren una cantidad significativa de recursos computacionales y datos de entrenamiento para alcanzar su máximo potencial.

La arquitectura de red neuronal convolucional ([CNN](#)) es fundamental para el funcionamiento de YOLO. En su versión original, YOLO utiliza una [CNN](#) para dividir la imagen de entrada en una cuadrícula y predecir las cajas delimitadoras (bounding boxes) y las clases de objetos dentro de cada celda de la cuadrícula. Cada caja delimitadora está asociada con un conjunto de parámetros que definen su posición y tamaño, así como las probabilidades de las clases de objetos presentes en la caja.

Esta capacidad de realizar la detección en una sola pasada y la eficiencia en términos de velocidad hacen que YOLO sea popular para aplicaciones en tiempo real, como la detección de objetos en vídeo.

### 3.3 Métodos de detección clásicos

Los métodos clásicos en detección de objetos antes de la era de las Redes Neuronales Convolucionales ([CNN](#)) se basaban en técnicas tradicionales y algoritmos que no requerían el entrenamiento de modelos complejos. Uno de estos métodos ampliamente utilizados es el Histograma de Gradientes Orientados (HOG), que se centraba en describir la distribución de gradientes en una imagen. HOG divide la imagen en celdas y calcula los gradientes en cada celda, generando un descriptor que representa la forma y la textura de la imagen.

Otro enfoque clásico involucraba el uso de descriptores de características, como el ([Scale-Invariant Feature Transform \(SIFT\)](#)) y ([Speeded Up Robust Features \(SURF\)](#)). Estos métodos identifican puntos clave o características distintivas en la imagen y los describen mediante información como la orientación y la escala, permitiendo la comparación y el emparejamiento entre imágenes.

En términos de técnicas de clasificación, los clasificadores basados en máquinas de soporte vectorial ([Support Vector Machine \(SVM\)](#)) eran comunes en la etapa posterior a la extracción de características. Estos clasificadores se entrenaban para distinguir entre las características de los objetos de interés y los fondos.

La detección de objetos también se abordaba mediante el enfoque de ventanas deslizantes, donde una ventana de tamaño fijo se desplaza sobre la imagen, y un clasificador determina si esa región contiene un objeto de interés. Este método, aunque efectivo en algunos casos, puede ser computacionalmente costoso debido a la evaluación exhaustiva de múltiples regiones.

Además, las técnicas de segmentación, como la umbralización, se utilizaban para dividir la imagen en regiones basadas en propiedades de píxeles, facilitando la identificación de objetos.

Estos métodos clásicos eran eficaces para ciertos casos, pero su rendimiento podía verse limitado en situaciones con variabilidad en las condiciones de iluminación, escala y orientación. Con la llegada de las [CNN](#) y el aumento en la disponibilidad de datos de entrenamiento, se produjo un cambio hacia enfoques más avanzados y automatizados que superaron muchas

de las limitaciones de los métodos clásicos en términos de precisión y robustez en la detección de objetos.

OpenCV (Open Source Computer Vision Library) [5] ha sido una herramienta clave en el desarrollo de aplicaciones de visión por computadora y detección de objetos. OpenCV proporciona una amplia variedad de funciones y algoritmos que facilitan la implementación de métodos clásicos y modernos en el campo de la visión artificial.

En el contexto de la detección de objetos, OpenCV ofrece funciones para realizar operaciones fundamentales como la lectura de imágenes, manipulación de matrices, y procesamiento de imágenes. También incluye algoritmos para la detección de bordes, filtrado, y transformaciones geométricas que son esenciales en el preprocesamiento de imágenes para tareas de detección.

### 3.4 YOLO

YOLO (You Only Look Once) [6] es un sistema de detección de objetos que ha revolucionado la visión artificial. A diferencia de otros enfoques, YOLO es excepcionalmente rápido y preciso. Lo que lo distingue es su capacidad para detectar y clasificar múltiples objetos en una sola pasada de la imagen, en lugar de requerir múltiples análisis. Utiliza una red neuronal convolucional para dividir la imagen en una cuadrícula y asignar cada cuadro a un objeto detectado, proporcionando información sobre su posición y clase. Esto lo hace extremadamente eficiente en términos de velocidad de procesamiento, lo que lo convierte en una elección popular para aplicaciones de detección de objetos en tiempo real, como la conducción autónoma y la seguridad como se puede observar en la figura 3.2.

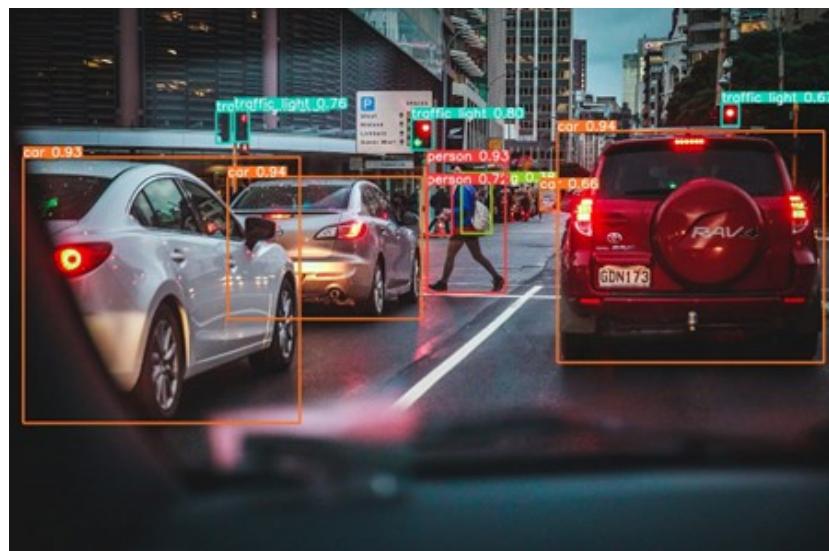


Figura 3.2: Algoritmo You Only Look Once

Las distintas versiones de YOLO [7] ofrecen diferentes combinaciones de rendimiento y precisión las cuales se encuentran resumidas en la tabla 3.1. YOLOv1, la versión original, logró un buen rendimiento en tiempo real, aunque su precisión era moderada. Con YOLOv2, se mejoró la precisión sin sacrificar demasiada velocidad en comparación con la versión anterior. YOLOv3, por su parte, ofreció una mayor precisión, aunque a expensas de una ligera reducción en la velocidad de detección. YOLOv4 mantuvo un rendimiento sólido y mejoró aún más la precisión en comparación con YOLOv3. Finalmente, YOLOv8 representa una versión avanzada y optimizada que se esfuerza por mantener un rendimiento robusto y una alta precisión, aprovechando las últimas mejoras y optimizaciones disponibles en la tecnología de detección de objetos.

Versión	Rendimiento	Precisión	Velocidad	Razonabilidad
YOLOv1	Bueno	Moderado	Alto	Velocidad aceptable
YOLOv2	Bueno	Moderado	Alto	Mejora en precisión
YOLOv3	Bueno	Buena	Moderado	Precisión a costa de velocidad menor
YOLOv4	Bueno	Buena	Moderado	Rendimiento mejorado y precisión sólida
YOLOv8	Bueno	Buena	Moderado	Optimizaciones y rendimiento sólido

Tabla 3.1: Comparativa mediante ChatGPT4

La elección entre estas versiones dependerá de las necesidades específicas de tu aplicación y tus prioridades en cuanto a rendimiento y precisión.

La elección de YOLOv8 se basa en una serie de razones sólidas. Esta versión representa la culminación de varias iteraciones y mejoras en el algoritmo YOLO. YOLOv8 ofrece un equilibrio sobresaliente entre rendimiento y precisión. Aprovechando las últimas optimizaciones y mejoras disponibles en la tecnología de detección de objetos, esta versión busca mantener un rendimiento robusto en tiempo real sin comprometer significativamente la precisión. Es una elección atractiva si buscas lo último en rendimiento y precisión, lo que la convierte en una

opción sólida para una variedad de aplicaciones de detección de objetos.

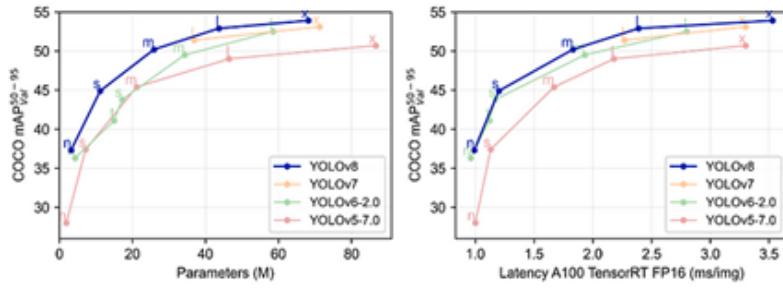


Figura 3.3: Comparativa versiones

### 3.5 Microcontroladores

Por otro lado, la Raspberry Pi [8] es una placa computadora de bajo costo y tamaño reducido que ha ganado una gran popularidad en el ámbito de la informática de proyectos y la educación. A pesar de su pequeño tamaño, la Raspberry Pi está equipada con un procesador, memoria RAM y puertos de entrada/salida que la convierten en un dispositivo versátil para diversas aplicaciones. Su facilidad de uso y asequibilidad la hacen ideal para proyectos de visión artificial y otras aplicaciones relacionadas con la inteligencia artificial. La Raspberry Pi, Figura 3.4, puede ejecutar software de detección de objetos como YOLO y, al mismo tiempo, puede interactuar con otros sensores y dispositivos, lo que la convierte en una opción popular para proyectos de visión artificial accesibles y de bajo costo.

Para la tarea de detección de especies en campo utilizando cámaras trampa con Raspberry Pi, es crucial seleccionar la versión adecuada [9] que equilibre rendimiento y precisión.

El Raspberry Pi 3 Model B, aunque es una opción económica, tiene un rendimiento limitado, lo que puede dificultar la ejecución de aplicaciones de detección en tiempo real, limitando su capacidad para capturar imágenes de animales de manera efectiva. En contraste, el Raspberry Pi 4 Model B ofrece un rendimiento sólido, lo que facilita el procesamiento de cargas de trabajo más intensivas de detección de objetos. Esto lo convierte en la elección más sólida, ya que brinda mejores oportunidades para una detección precisa y en tiempo real de animales en el campo. El Raspberry Pi Zero, a pesar de ser económico, presenta un rendimiento muy limitado, lo que podría comprometer la efectividad de la detección. Por último, el modelo de 8 GB de RAM del Raspberry Pi 4 Model B puede ser la elección ideal si tienes previsto manejar grandes conjuntos de datos, ya que la memoria adicional facilita la gestión eficiente de datos en proyectos de detección de animales en campo. En resumen, para obtener los mejores resultados en términos de rendimiento y precisión, el Raspberry Pi 4 Model B es la elección recomendada para proyectos de detección de animales en campo con cámaras trampa.

Versión	Rendimiento	Precisión	Razonabilidad
Raspberry Pi 3 Model B	Limitado	Variable	Adecuado para proyectos simples, pero no en tiempo real
Raspberry Pi 4 Model B	Bueno	Variable	Mejora en precisión
Raspberry Pi Zero	Muy limitado	Variable	Económico, pero rendimiento insuficiente para tiempo real
Raspberry Pi 4 Model B (8 GB)	Bueno	Moderado	Rendimiento mejorado y precisión sólida

Tabla 3.2: Comparativa versiones Raspberry Pi

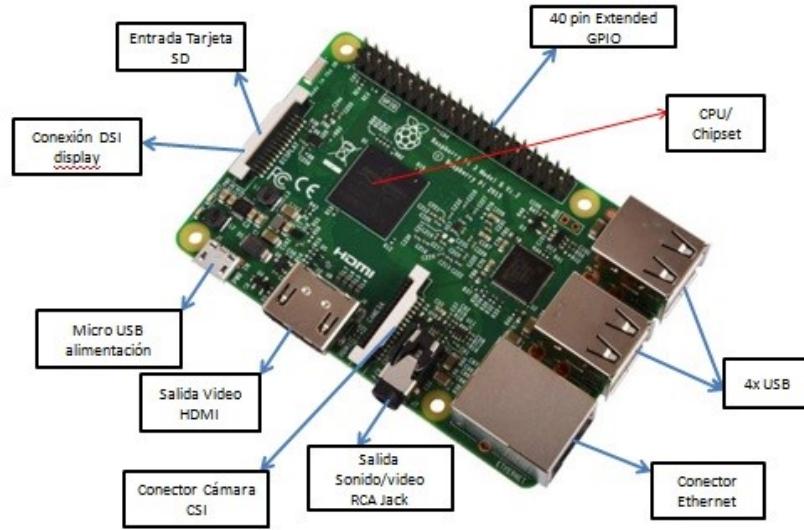


Figura 3.4: Esquema Raspberry Pi

Si bien no se han encontrado proyectos de preservación de vida silvestre usando Raspberry Pi, en algunos proyectos de monitoreo de la fauna, se ha utilizado Arduino, un microcontrolador de código abierto y fácil de programar. Se han implementado sensores de movimiento y cámaras infrarrojas conectadas a placas Arduino para capturar imágenes y videos cuando se detecta actividad. El microcontrolador maneja la captura de datos y, en algunos casos, puede enviarlos a un servidor remoto.

## 3.6 Tecnologías y herramientas

### 3.6.1 Pycharm

Pycharm [10] es un entorno de desarrollo integrado ([Integrated Development Environment \(IDE\)](#)) diseñado específicamente para el lenguaje de programación Python. Desarrollado por JetBrains, PyCharm ofrece una amplia gama de características y herramientas destinadas a facilitar el desarrollo, depuración y mantenimiento de proyectos Python. Entre sus características notables se incluyen la autocompletación de código, el análisis estático, el depurador integrado, la gestión de entornos virtuales y la compatibilidad con frameworks populares como Django y Flask.

Este entorno de desarrollo busca mejorar la productividad de los desarrolladores Python al proporcionar un conjunto integral de herramientas y una interfaz de usuario intuitiva. PyCharm se ha convertido en una opción popular entre los programadores Python debido a su funcionalidad avanzada y su enfoque en la eficiencia del flujo de trabajo.

PyCharm se seleccionó por su gratuidad, la posibilidad de utilizar la pantalla de comandos, la ejecución de comandos Python y su interfaz intuitiva, lo que lo convierte en una herramienta idónea para nuestros propósitos.

### 3.6.2 Labelme

Además de PyCharm, hemos incorporado la herramienta Labelme en nuestro flujo de trabajo. LabelMe es una aplicación que simplifica el proceso de anotación de imágenes, lo que resulta de suma utilidad en la tarea de etiquetar nuestras imágenes del leopardo de las nieves. A continuación, explicaremos detalladamente la utilidad de LabelMe y cómo la hemos empleado en nuestro proyecto:

LabelMe [11] es una herramienta de código abierto utilizada para la anotación de imágenes, especialmente diseñada para aplicaciones de aprendizaje automático y visión por computadora. Desarrollada por el Computer Science and Artificial Intelligence Laboratory (CSAIL), LabelMe permite a los usuarios etiquetar y anotar objetos en imágenes de manera precisa, generando conjuntos de datos anotados que son esenciales para entrenar algoritmos de reconocimiento visual.

Las características principales de LabelMe incluyen la capacidad de etiquetar objetos mediante polígonos, rectángulos y puntos, así como la posibilidad de añadir descripciones y metadatos a las anotaciones. Además, LabelMe permite la colaboración en línea, facilitando la creación colaborativa de conjuntos de datos. Características clave de LabelMe incluyen:

- **Interfaz de Usuario Intuitiva:** Labelme proporciona una interfaz fácil de usar que permite a los usuarios etiquetar objetos de manera eficiente y precisa.
- **Compatibilidad con Formatos Estándar:** La herramienta es compatible con formatos de anotación ampliamente utilizados, como COCO, Pascal VOC y otros, lo que facilita la integración de los datos anotados en flujos de trabajo de aprendizaje automático.
- **Personalización:** Labelme permite a los usuarios definir categorías y atributos personalizados para adaptarse a las necesidades específicas de su proyecto.
- **Exportación de Datos:** Los datos anotados se pueden exportar en formatos compatibles con múltiples bibliotecas y marcos de trabajo de aprendizaje automático, lo que simplifica la incorporación de los datos etiquetados en nuestros algoritmos de entrenamiento.

### 3.6.3 Json

JavaScript Object Notation (JSON) [12], que significa "JavaScript Object Notation" (Notación de Objetos de JavaScript), es un formato de intercambio de datos liviano y fácil de leer.

Diseñado originalmente para ser utilizado con JavaScript, **JSON** se ha vuelto omnipresente en el desarrollo de software y es independiente del lenguaje de programación.

Este formato utiliza una estructura de pares clave-valor, que permite organizar y representar datos de manera jerárquica. Los datos en **JSON** pueden ser simples, como números y cadenas de texto, o estructuras más complejas, como matrices y objetos anidados. La simplicidad y la legibilidad de **JSON** han llevado a su adopción generalizada en la transmisión y almacenamiento de datos.

Una de las ventajas clave de **JSON** es su capacidad para representar datos de manera compacta y estructurada, lo que facilita su comprensión y manipulación tanto por humanos como por máquinas. Además, su independencia del lenguaje lo hace ideal para la comunicación entre sistemas heterogéneos.

En el contexto del desarrollo web, **JSON** se utiliza comúnmente para intercambiar datos entre el cliente y el servidor. También es ampliamente utilizado en la configuración de aplicaciones y en la representación de datos en servicios web y APIs (Interfaces de Programación de Aplicaciones).

La simplicidad y versatilidad de **JSON** han contribuido a su estatus como un formato de elección en el desarrollo de software moderno, y su fácil integración con una variedad de lenguajes y tecnologías lo convierte en una herramienta valiosa para el intercambio y almacenamiento de datos estructurados.

### 3.6.4 MongoDB

MongoDB [13], desarrollado por MongoDB Inc., se destaca como un sistema de gestión de bases de datos no relacional, orientado a documentos y de código abierto. Su popularidad en la comunidad de desarrollo de software radica en su flexibilidad y escalabilidad, rompiendo con el paradigma de las bases de datos relacionales al almacenar datos en documentos **Binary JSON (BSON)** en lugar de tablas.

El modelo de datos flexible de MongoDB se basa en documentos **BSON**, estructuras de pares clave-valor anidados que permiten una representación dinámica de los datos. Esta flexibilidad es clave para adaptarse a esquemas cambiantes y facilitar la evolución de las aplicaciones a lo largo del tiempo.

Una característica destacada de MongoDB es su capacidad de escalar horizontalmente, lo que implica agregar más servidores para manejar volúmenes crecientes de datos y tráfico. Esto lo hace especialmente adecuado para entornos donde la demanda de datos puede aumentar de manera significativa.

Además, MongoDB ofrece eficientes índices y un lenguaje de consulta potente, facilitando la búsqueda y recuperación de datos. Su soporte integrado para datos geoespaciales lo posiciona como una elección sólida para aplicaciones que requieren funcionalidades relacionadas

con la ubicación.

La comunidad activa de desarrolladores y la documentación completa contribuyen a la adopción y desarrollo fluido con MongoDB. Su aplicación se extiende a diversos contextos, desde sistemas de gestión de contenidos hasta aplicaciones de comercio electrónico y plataformas analíticas, donde la flexibilidad en el esquema de datos y la escalabilidad son esenciales.

En resumen, MongoDB se presenta como una opción robusta para aquellos que buscan un sistema de gestión de bases de datos no relacional que ofrezca flexibilidad, escalabilidad y una aproximación moderna al manejo de datos.

### 3.6.5 Power BI

Power BI [14], desarrollado por Microsoft, es una herramienta de análisis y visualización de datos que permite a los usuarios transformar datos sin procesar en información valiosa mediante informes interactivos y cuadros de mando dinámicos. Este software de inteligencia empresarial se destaca por su facilidad de uso, integración con múltiples fuentes de datos y capacidades avanzadas de visualización.

Uno de los aspectos más destacados de Power BI es su capacidad para conectarse a una amplia variedad de fuentes de datos, desde archivos Excel y bases de datos Structured Query Language (SQL) hasta servicios en la nube como Azure y Google Analytics. Esta versatilidad permite a las organizaciones unificar sus datos en una única plataforma y obtener una visión completa de sus operaciones.

El motor de Power BI está impulsado por el lenguaje de consulta DAX (Data Analysis Expressions) y el modelo de datos en memoria VertiPaq. DAX permite realizar cálculos y agregaciones complejas, mientras que VertiPaq optimiza el rendimiento de las consultas, asegurando respuestas rápidas incluso con grandes volúmenes de datos.

Las capacidades de visualización de Power BI son amplias y personalizables. Los usuarios pueden crear gráficos interactivos, mapas geográficos, y tablas dinámicas, entre otras visualizaciones, que facilitan la interpretación de los datos y la toma de decisiones informadas. Además, Power BI ofrece la posibilidad de compartir informes y dashboards con otros miembros de la organización, promoviendo una cultura de datos colaborativa.

Otra característica importante es la capacidad de actualización automática de los datos. Power BI puede programar actualizaciones de datos en tiempo real, asegurando que los informes y dashboards siempre reflejen la información más reciente. Esto es crucial para las empresas que necesitan tomar decisiones basadas en datos actuales.

Power BI también se integra perfectamente con otras herramientas de Microsoft, como Excel, Azure y Teams, lo que facilita su adopción en entornos que ya utilizan soluciones de Microsoft. Además, la comunidad activa y los recursos educativos disponibles, como tutoriales y documentación detallada, apoyan a los usuarios en el aprendizaje y aprovechamiento de la

plataforma.

### 3.6.6 Ultralytics

Ultralytics [15], una empresa líder en el ámbito de la inteligencia artificial y la visión por computadora, es reconocida principalmente por desarrollar el modelo YOLO (You Only Look Once). La popularidad de Ultralytics se debe a su enfoque innovador y a la eficiencia de sus modelos.

Además de crear YOLO, Ultralytics proporciona herramientas adicionales que mejoran la experiencia de desarrollo e implementación de algoritmos de visión por computadora. Estas herramientas incluyen gráficos detallados y visualizaciones que permiten a los desarrolladores entender mejor el rendimiento de los modelos y optimizar sus aplicaciones.

Ultralytics ha facilitado la adopción de sus tecnologías mediante implementaciones accesibles y bien documentadas, disponibles en sus repositorios de código abierto en GitHub. La empresa ofrece tutoriales y recursos educativos, apoyando a la comunidad de desarrolladores en la integración y uso efectivo de sus modelos avanzados.

### 3.6.7 Google Colab

Google Colab [16], desarrollado por Google Research, es una plataforma gratuita basada en la nube que permite a los usuarios ejecutar y compartir notebooks Jupyter. Colab es especialmente popular entre los científicos de datos, investigadores y desarrolladores de inteligencia artificial debido a su accesibilidad y potencia.

Una de las principales ventajas de Google Colab es su capacidad para proporcionar acceso gratuito a recursos de computación en la nube, incluidos [Graphics Processing Unit \(GPU\)](#) y [Tensor Processing Unit \(TPU\)](#), lo que permite a los usuarios ejecutar tareas computacionalmente intensivas sin necesidad de hardware especializado. Esto es particularmente útil para el entrenamiento de modelos de aprendizaje profundo y la experimentación con grandes conjuntos de datos.

Colab permite a los usuarios escribir y ejecutar código Python directamente en el navegador, y sus notebooks pueden incluir texto, código ejecutable y visualizaciones, lo que facilita la documentación y la presentación de resultados. Además, Colab se integra perfectamente con Google Drive, lo que simplifica el almacenamiento y la gestión de archivos.

## Capítulo 4

# Desarrollo: Parte 1

---

**E**n esta sección, abordaremos varios aspectos clave relacionados con nuestro proyecto de identificación y monitoreo del leopardo de las nieves utilizando técnicas avanzadas de visión por computadora.



Figura 4.1

## 4.1 Contextualización

La aplicación de sistemas basados en Inteligencia Artificial (IA) ha revolucionado la forma en que se procesan y analizan datos visuales, especialmente en áreas como la detección y clasificación de entidades en imágenes o videos. Estos sistemas utilizan algoritmos especializados para detectar entidades, como personas, autos o animales, en una imagen o video, y luego clasificarlas según diversos criterios. La detección implica identificar la presencia y ubicación de una entidad específica en una imagen, mientras que la clasificación implica asignar una etiqueta o categoría a la entidad detectada.

Por otro lado, la segmentación (Figura 4.2) es un proceso más avanzado que implica dividir una imagen en regiones significativas para identificar y comprender mejor los objetos

presentes. Mientras que la detección identifica la presencia de un objeto en una imagen, la segmentación va un paso más allá al delinear los contornos precisos de cada objeto dentro de la imagen.

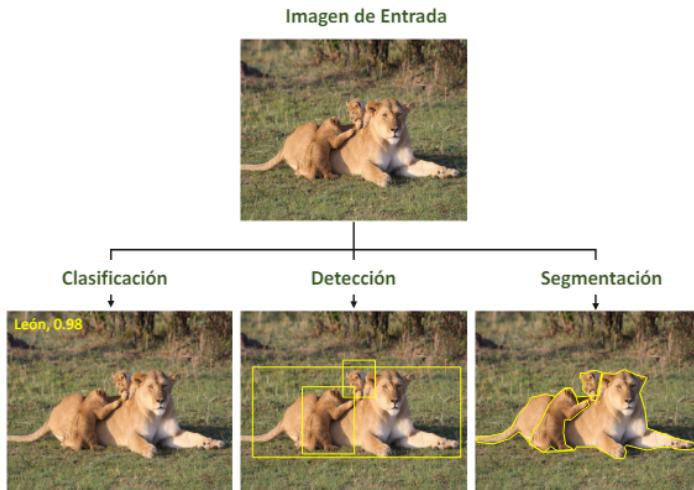


Figura 4.2: Detección y segmentación

En el contexto de la investigación de animales silvestres en ecosistemas terrestres, las cámaras trampa son herramientas fundamentales. Estos dispositivos electrónicos se activan por movimiento o calor, capturando secuencias de imágenes que pueden contener a los animales objeto de estudio. Sin embargo, las cámaras trampa también registran imágenes vacías debido al movimiento de la vegetación o a condiciones ambientales adversas como nieve o niebla, lo que puede dificultar la detección precisa de los animales.

La revisión manual de estas imágenes por parte de especialistas es un proceso laborioso y costoso en términos de tiempo y recursos humanos. Los expertos deben descartar las imágenes vacías, determinar la presencia de un animal y luego clasificarlo en función de su especie. Esta metodología limita la escalabilidad y productividad de la investigación.

Para abordar estos desafíos, la automatización del proceso de clasificación mediante métodos automáticos basados en visión artificial y Machine Learning ofrece una solución prometedora. Estos sistemas pueden procesar grandes volúmenes de imágenes de manera rápida y eficiente, identificando automáticamente la presencia de animales y clasificándolos en función de criterios predefinidos. Al eliminar la necesidad de revisión manual, se mejora la escalabilidad y la productividad de la investigación, permitiendo un análisis más exhaustivo y detallado de la vida silvestre en su hábitat natural.

### 4.1.1 Fantasma del Himalaya: Individuo y Especie

En sus tierras nativas, el leopardo de las nieves (Figura 4.3) es conocido con muchos nombres, incluido “wāwrīn prāng”, “shan” (Ladakhi), “zigsa” (tibetano), “irves” (mongol: ирвэс). , “bares” o “barys” (kazajo: барыс [‘barəs]), “ilbirs” (kirguís: Илбирс), “barfānī chītā”, palang-e barfi y “él tendua”. Su nombre científico es *Panthera uncia* (anteriormente: *Uncia uncia*) cuyo origen se deriva de la antigua palabra francesa once, que se utilizó originalmente para el lince europeo. Al leopardo de las nieves todavía se le llama ocasionalmente onza.



Figura 4.3: *Panthera uncia*

Los leopardos de las nieves han evolucionado para vivir en algunas de las condiciones más duras de la Tierra. Su espeso pelaje blanco grisáceo salpicado de grandes rosetas negras combina perfectamente con las altas montañas escarpadas y rocosas de Asia. Debido a su increíble camuflaje natural, que los hace casi invisibles en su entorno, a los leopardos de las nieves a menudo se les llama el "fantasma de las montañas".

Durante milenios, este magnífico felino fue el rey de las montañas. Las montañas eran ricas en presas como la oveja azul, la oveja salvaje argali, el ibice, las marmotas, las picas y las liebres. El hábitat del leopardo de las nieves se extiende por las regiones montañosas de 12 países de Asia: Afganistán, Bután, China, India, Kazajstán, República Kirguisa, Mongolia, Nepal, Pakistán, Rusia, Tayikistán y Uzbekistán. El área total (Figura 4.4) cubre un área de cerca de 772,204 millas cuadradas, y el 60 por ciento del hábitat se encuentra en China. Sin embargo, más del 70 por ciento del hábitat del leopardo de las nieves permanece inexplorado. El tamaño del área de distribución puede variar desde 4,6 a 15,4 millas cuadradas en Nepal hasta más de 193 millas cuadradas en Mongolia. Y la densidad de población puede oscilar entre <0,1 y 10 o más individuos por 38,6 millas cuadradas, dependiendo de la densidad de presas y la calidad del hábitat. Pese a esto, es muy probable que la población del leopardo de las nieves esté disminuyendo.

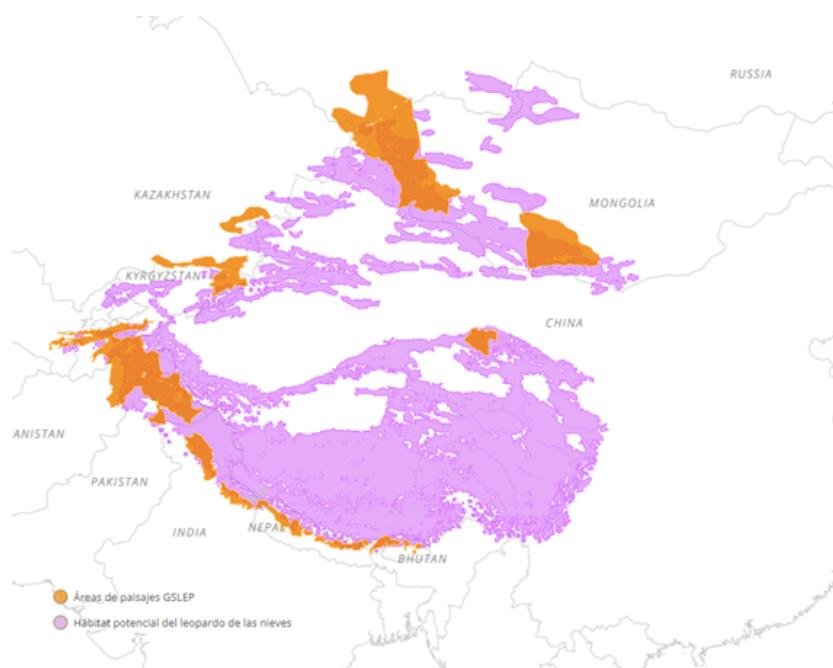


Figura 4.4: Hábitat del leopardo de las nieves

Los científicos estiman que puede que sólo queden entre 3.920 y 6.390 leopardos de las nieves en estado salvaje. Los leopardos de las nieves desempeñan un papel clave como depredador superior (Figura 4.5), un indicador de la salud de su hábitat de gran altitud y, cada vez más, un indicador importante de los impactos del cambio climático en los entornos montañosos. Si los leopardos de las nieves prosperan, también lo harán innumerables otras especies y las mayores reservas de agua dulce del planeta.

La caza furtiva, la pérdida de hábitat, la disminución de las especies de presas naturales y las matanzas en represalia resultantes del conflicto entre humanos y vida silvestre son las principales razones por las que este gran felino está amenazado. El otro impacto importante en la supervivencia del leopardo de las nieves es la crisis climática [17]. El aumento de las temperaturas globales puede afectar gravemente la productividad del hábitat alpino, lo que a su vez puede afectar la disponibilidad de presas y agua dulce en el duro entorno montañoso. La crisis climática plantea quizás la mayor amenaza a largo plazo para los leopardos de las nieves. Los impactos de un planeta que se calienta podrían provocar una pérdida de hasta el 30 por ciento del hábitat del leopardo de las nieves sólo en el Himalaya.

El terreno accidentado y el duro clima de gran parte del hábitat del leopardo de las nieves hacen que la investigación sea particularmente exigente, una de las razones principales por las que vastas secciones del hábitat del leopardo de las nieves siguen siendo territorio inexplorado para los investigadores. Mediante ThirdPoleGeolab, (Figura 4.6) podemos observar un estudio de los diferentes hábitats y zonas de movimiento para relacionar datos y conclusiones de esta

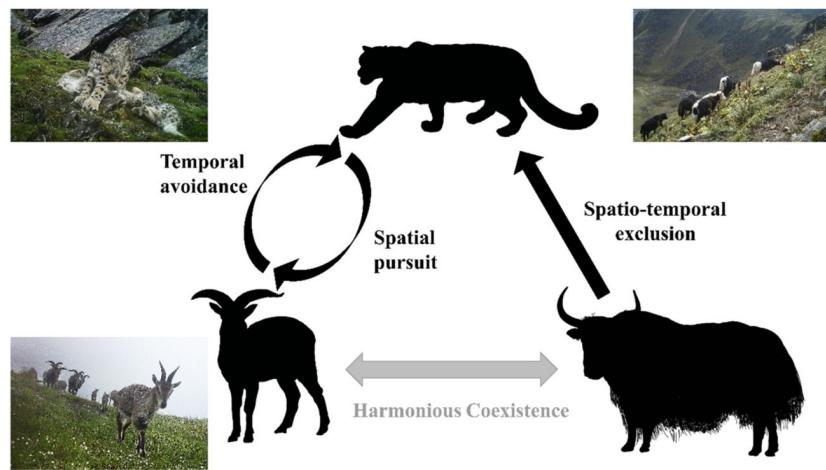


Figura 4.5: Ciclo actual del hábitat

especie de la cual se guía el proyecto y de la cual tenemos extensa información de carácter público sin ánimo de lucro.

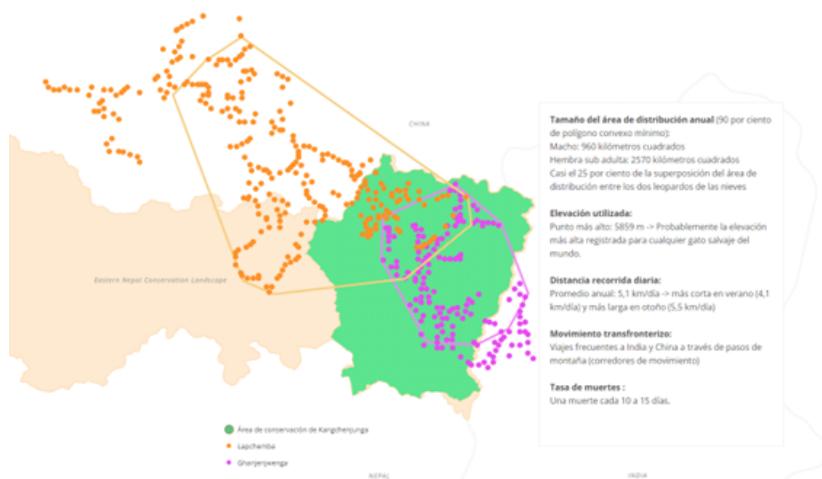


Figura 4.6: Distribución en el territorio

Los investigadores de campo de Snow Leopard Trust [18] realizan estudios durante todo el año para obtener información sobre el hábitat del leopardo de las nieves, las especies de presas silvestres y los propios gatos. Las cámaras de investigación fotografían a los leopardos de las nieves salvajes mientras se mueven por sus áreas de distribución, mientras que los collares GPS nos brindan la oportunidad de rastrear los movimientos de un leopardo de las nieves individual durante todo un año. Los estudios ecológicos se utilizan para comprender mejor el paisaje y el papel de los humanos y la vida silvestre por igual, y la investigación genética ofrece la oportunidad de establecer la salud y la diversidad de una población de leopardos de las nieves.

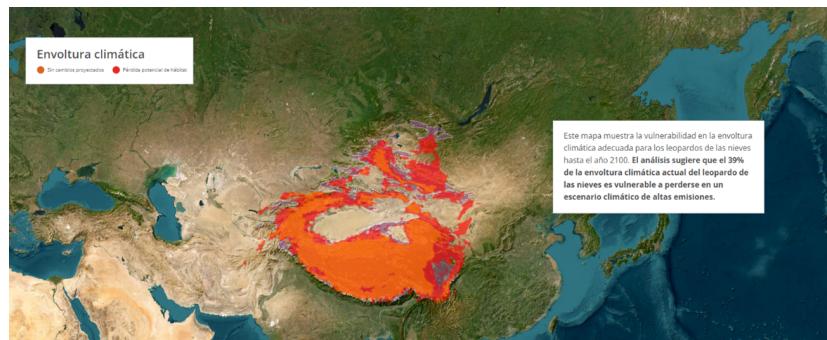


Figura 4.7: Análisis envoltura climática

Las cámaras, equipadas con sensores de calor y movimiento, tienen sus problemas (Figura 4.8). A veces quedan sepultados por avalanchas, arrastrados por inundaciones o derribados por animales. Otras veces, toman fotografías de animales “equivocados”, como cabras, camellos o caballos que se plantan frente a una lente y rumian durante horas, o pasan en hordas. O las cámaras se activan cuando una brizna de hierba se balancea cerca de una roca calentada por el sol.



Figura 4.8: Imagen de cámara trampa de individuo en movimiento

Luego está el abrumador volumen de fotografías. Cada estudio con cámara dura varios meses, cubre un área de 400 a 500 millas cuadradas y genera entre 200.000 y 300.000 imágenes de 30 a 60 cámaras. Clasificar las imágenes (en fotos con leopardos de las nieves y fotos sin ellos) ha sido tradicionalmente una tarea manual tediosa que requiere cientos de horas humanas para la organización sin fines de lucro.

Una nueva solución de inteligencia artificial de Microsoft, Snow Leopard Trust está acelerando el proceso, con un modelo de aprendizaje automático que puede identificar leopardos de las nieves y clasificar automáticamente cientos de miles de fotografías en cuestión de mi-

nutos. Los leopardos son clasificados por ejemplar en una base de datos según el patrón de manchas de su pelaje (Figura 4.9) que los distingue, ya que al cumplen la función de la huella en los humanos.

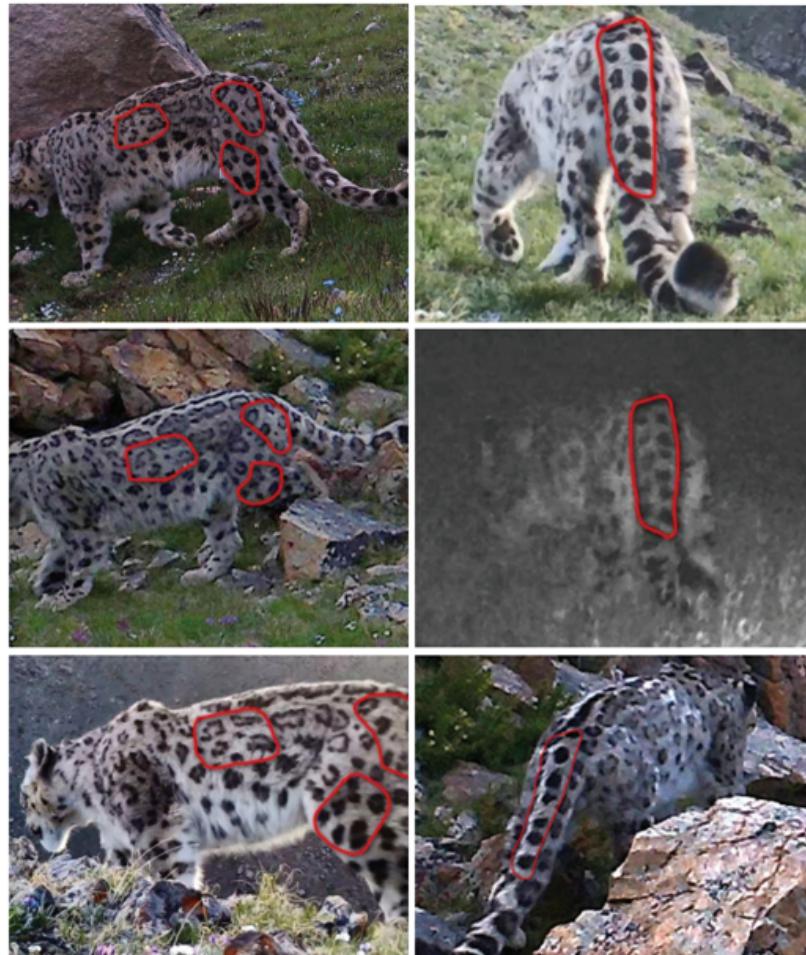


Figura 4.9: Identificaciones propias del pelaje

Nuestro análisis partirá de abordar un trabajo similar al del Snow Leopard Trust, una organización líder en la conservación de leopardos de las nieves y su hábitat. En un esfuerzo por comprender y adaptar sus prácticas exitosas, exploraremos en detalle sus estrategias de monitoreo, educación pública y colaboración con las comunidades locales. Iterativamente, comenzaremos a desglosar las diversas aproximaciones y fases del proceso, con el fin de identificar las mejores prácticas y adaptarlas a nuestro contexto particular, contribuyendo así a la conservación de la vida silvestre y la promoción de la coexistencia armónica entre las especies y las comunidades humanas. Dar las gracias a la organización que nos ha cedido el permiso para utilizar tanto datos como imágenes de sus expediciones.

## 4.2 Dataset de imágenes

En el proceso de desarrollo de nuestro proyecto, se considera primordial la selección de una serie de imágenes del espécimen a estudiar, en este caso el leopardo de las nieves. Estas imágenes constituyen la base fundamental para llevar a cabo un entrenamiento inicial y sencillo de nuestro sistema.

La elección de estas imágenes se llevó a cabo siguiendo criterios para garantizar calidad y diversidad en los datos con los que alimentaremos nuestro modelo. En total, se seleccionaron 20 imágenes de buena resolución que capturan al felino en su hábitat natural.

La mayoría de estas imágenes provienen de diversas fuentes confiables, como fotógrafos de vida silvestre, organizaciones de conservación y recursos académicos, lo que garantiza su autenticidad y valor científico. Destacar que, para la elaboración de este proyecto, la organización pionera en este campo sin ánimo de lucro Snow Leopard Trust nos ha concedido libertad en el uso de su contenido visual y estadístico.

El propósito de esta selección es iniciar un proceso de entrenamiento sencillo de nuestro modelo de reconocimiento de leopardo de las nieves. Estas imágenes servirán como punto de partida para enseñar al algoritmo a identificar y distinguir esta especie de felino en futuras tareas de clasificación y detección. Durante esta fase inicial, el sistema aprenderá a reconocer patrones de color, forma y contextos geográficos asociados con el leopardo de las nieves.

Una vez descargadas las imágenes, el siguiente paso crucial en nuestro proceso es etiquetar la especie del leopardo de las nieves en cada imagen. Este proceso de etiquetado es esencial para permitir un entrenamiento efectivo de nuestro modelo de reconocimiento de esta especie en particular. Para llevar a cabo este procedimiento, hemos optado por utilizar la plataforma PyCharm como nuestro entorno de desarrollo.

En nuestro proyecto, hemos empleado Labelme para etiquetar cada imagen del leopardo de las nieves. Esto implica dibujar contornos precisos alrededor de los leopardos y asignarles la categoría correspondiente (Figuras 4.10 y 4.11). Estas anotaciones proporcionarán información esencial al modelo durante el proceso de entrenamiento, permitiéndole aprender a identificar y distinguir esta especie en futuras tareas de clasificación y detección.

Después de completar el proceso de etiquetado de las imágenes, el siguiente paso es transformar los archivos [JSON](#) resultantes en el formato YOLO (You Only Look Once). La transformación implica la conversión de las coordenadas de las etiquetas y las clases asignadas en archivos [JSON](#) a un formato compatible con YOLO. Cada etiqueta se convierte en un archivo de texto independiente que contiene información sobre la clase del objeto y las coordenadas normalizadas de su ubicación en la imagen.

Las imágenes etiquetadas se dividirán tres categorías que permiten evaluar el rendimiento del modelo en diferentes etapas del proceso de entrenamiento y garantiza que sea capaz de

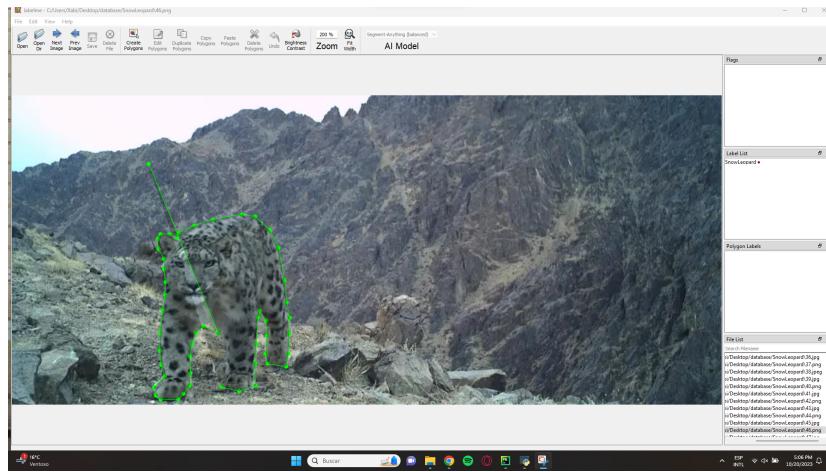


Figura 4.10: Ejemplo Labelme

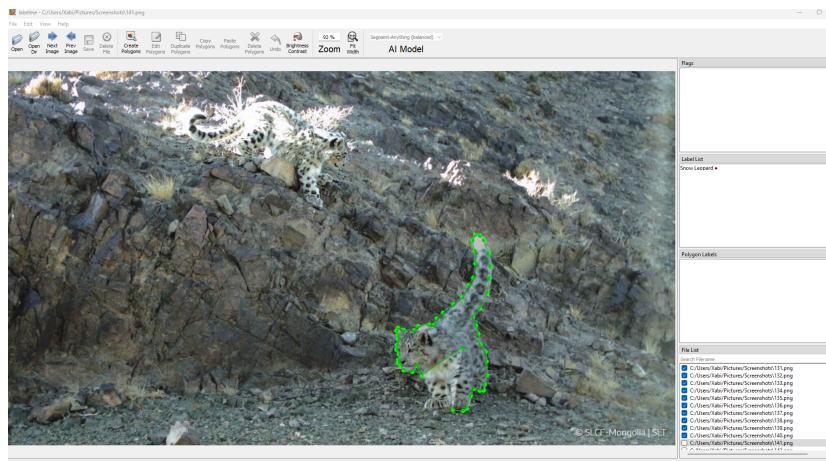


Figura 4.11: Segundo Ejemplo Labelme

generalizar adecuadamente a nuevas imágenes. A continuación, se explica la organización de los datos en estas carpetas:

1. Carpeta de Entrenamiento (train): En esta carpeta se almacenan los archivos YOLO correspondientes a las imágenes utilizadas para entrenar el modelo.
2. Carpeta de Validación (valid): Los archivos YOLO de esta carpeta se utilizan para evaluar el rendimiento del modelo durante el entrenamiento.
3. Carpeta de Prueba (test): En esta carpeta se almacenan los archivos YOLO correspondientes a imágenes que el modelo no ha visto durante el entrenamiento ni la validación.

## 4.3 Primeras iteraciones Yolov8

### 4.3.1 Primera ejecución

Una vez que hemos completado con éxito la organización de nuestros datos etiquetados en formato YOLO en carpetas de entrenamiento, validación y prueba, el siguiente paso crítico en nuestro proyecto es el entrenamiento de nuestro algoritmo YOLO (You Only Look Once).

A continuación, compararé las métricas de los modelos YOLOv8 (Figura 4.12), es decir, YOLOv8n, YOLOv8m y YOLOv8x, enumerando sus características principales:

Model	size (pixels)	mAP <sub>val</sub> 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n	640	18.4	142.4	1.21	3.5	10.5
YOLOv8s	640	27.7	183.1	1.40	11.4	29.7
YOLOv8m	640	33.6	408.5	2.26	26.2	80.6
YOLOv8l	640	34.9	596.9	2.43	44.1	167.4
YOLOv8x	640	36.3	860.6	3.56	68.7	260.6

Figura 4.12: Comparación métricas de Yolov8

En las diferentes configuraciones de YOLOv8, se observan distintas características que se adaptan a diversas necesidades. YOLOv8n, en su modalidad normal, destaca por su modelo liviano y rápida velocidad de inferencia, aunque con una precisión de detección ligeramente inferior. Este enfoque lo hace idóneo para aplicaciones en tiempo real con recursos limitados, como la vigilancia y aplicaciones móviles. Por otro lado, YOLOv8m, con un tamaño de modelo mediano, logra un equilibrio entre velocidad y precisión, siendo apropiado para aplicaciones que requieren una buena precisión sin sacrificar demasiado la velocidad, como el seguimiento de objetos en video.

La versión eXtra-rápida, YOLOv8x, se destaca por su modelo grande que permite una velocidad de inferencia muy rápida. Aunque su precisión de detección es inferior en comparación con modelos más pequeños, resulta ideal para aplicaciones donde la velocidad es la prioridad principal, como la detección en tiempo real en entornos de alto tráfico. Por otro lado, YOLOv8l, en su configuración grande, ofrece una alta precisión de detección, aunque con una velocidad de inferencia más lenta. Este modelo encuentra su aplicación en escenarios que demandan una precisión excepcional, como la detección de objetos médicos o la investigación científica.

Finalmente, YOLOv8s, en su versión pequeña, destaca por su tamaño compacto que posibilita una rápida velocidad de inferencia. Aunque su precisión de detección es inferior a

la de YOLOv8l, supera a YOLOv8x en este aspecto. Este modelo es idóneo para aplicaciones en tiempo real con recursos muy limitados, donde la precisión es crucial, pero la velocidad también desempeña un papel esencial.

En un proyecto como el Snow Leopard Trust, que se enfoca en la conservación de leopardos de las nieves, la elección del modelo YOLO dependería de la necesidad de obtener un equilibrio entre precisión y velocidad. Dado que se trata de un proyecto de conservación de vida silvestre, la precisión en la detección de estos felinos es fundamental para recopilar datos precisos y tomar decisiones informadas. En este contexto, un modelo como YOLOv8m (Medio) podría ser la mejor opción, ya que ofrece un buen equilibrio entre precisión y velocidad, lo que permite detectar a los leopardos de las nieves con suficiente precisión y, al mismo tiempo, realizar un seguimiento en tiempo real para obtener datos valiosos sobre su comportamiento y ubicación. Esto facilitaría la labor de monitoreo y conservación de esta especie en peligro de extinción.

El entrenamiento de un modelo YOLO implica la optimización de una red neuronal convolucional ([CNN](#)) para la detección de objetos. En nuestro caso, el modelo se enfoca en reconocer y localizar al leopardo de las nieves en imágenes. El proceso de entrenamiento se desarrolla de la siguiente manera:

1. Configuración de Hiperparámetros: Antes de comenzar el entrenamiento, definimos los hiperparámetros esenciales, como la tasa de aprendizaje, el número de épocas y el tamaño del lote (batch size). Estos valores son críticos para el rendimiento del modelo y se ajustan en función de la complejidad de la tarea y el tamaño del conjunto de datos.
2. Inicialización del Modelo Preentrenado: A menudo, se inicia el modelo con pesos pre-entrenados en una tarea relacionada, como el reconocimiento de objetos en imágenes generales. Esto permite al modelo aprovechar el conocimiento previo y acelerar el proceso de entrenamiento.
3. Propagación hacia Atrás (Backpropagation): Durante el entrenamiento, el modelo se alimenta con imágenes de las carpetas de entrenamiento. Luego, se calcula el error entre las predicciones del modelo y las etiquetas reales, y se utiliza el algoritmo de propagación hacia atrás para ajustar los pesos de la red y minimizar este error.
4. Evaluación en Conjunto de Validación: A intervalos regulares durante el entrenamiento, evaluamos el rendimiento del modelo en el conjunto de validación para controlar su capacidad de generalización y ajustar los hiperparámetros si es necesario.
5. Pruebas en Conjunto de Prueba: Una vez que el modelo ha alcanzado un nivel deseado de rendimiento en el conjunto de validación, lo probamos en el conjunto de prueba, que contiene datos que el modelo nunca ha visto antes. Esto nos proporciona una medida

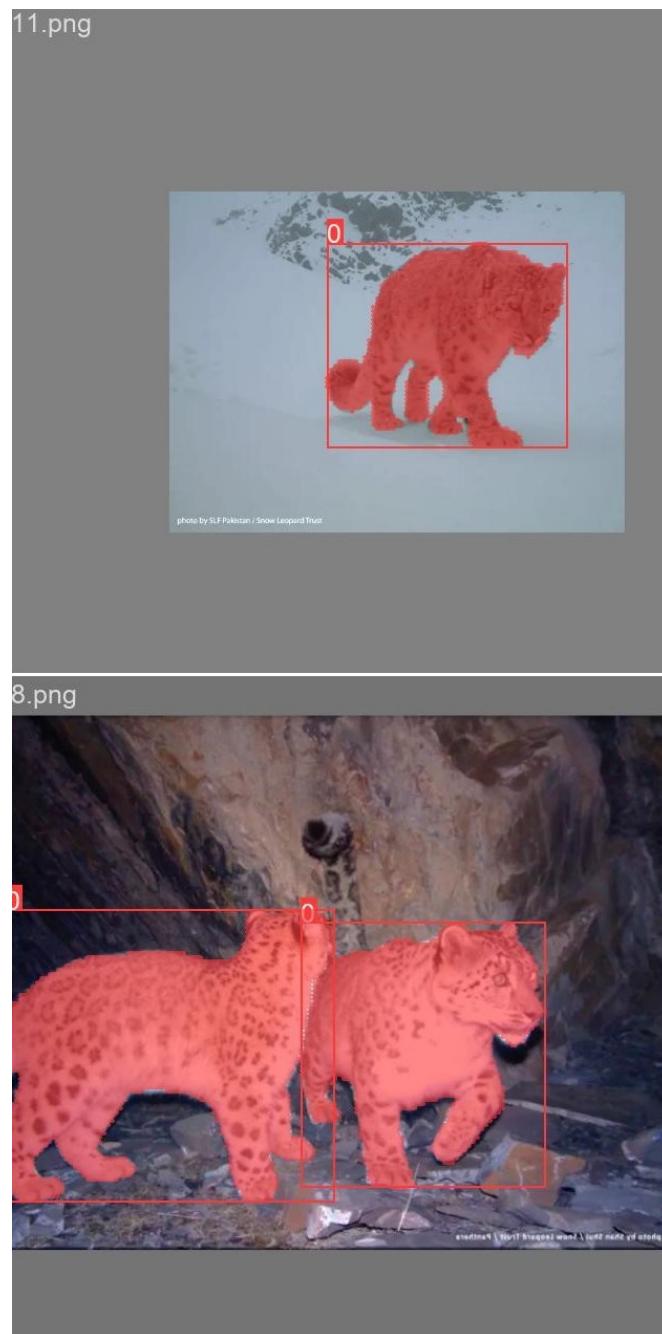


Figura 4.13: Algunos de los resultados de entrenamiento que exporta Yolov8

realista de su capacidad para detectar leopardos de las nieves en situaciones del mundo real.

6. Afinamiento (Fine-Tuning): En función de los resultados en el conjunto de prueba, es posible realizar ajustes adicionales en el modelo para mejorar su rendimiento.

7. Despliegue en Aplicaciones Prácticas: Finalmente, una vez que estamos satisfechos con el rendimiento del modelo, estamos listos para desplegarlo en aplicaciones prácticas, como la monitorización de leopardos de las nieves en su hábitat natural, la conservación de la especie y el estudio de su comportamiento.

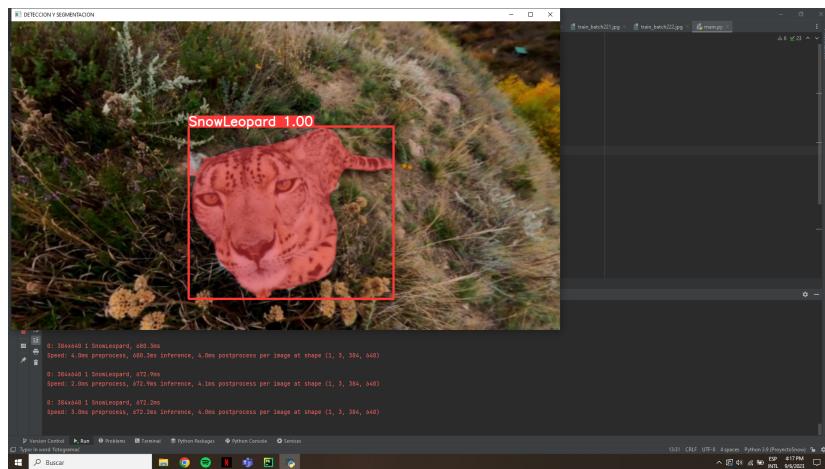


Figura 4.14: Ejecución de un video descargado de internet

Después de llevar a cabo el primer entrenamiento de nuestro algoritmo YOLO para la detección del leopardo de las nieves, hemos logrado obtener resultados favorables en la identificación de esta especie en imágenes (Figura 4.14). Sin embargo, también hemos identificado ciertos errores y desafíos que requieren atención y mejora para lograr un rendimiento óptimo. A continuación, se detallan los errores detectados:

1. Velocidad de Procesamiento Insatisfactoria: El algoritmo no alcanza una velocidad de procesamiento aceptable, especialmente en objetos pequeños. Esto se debe a limitaciones de recursos de la computadora.
2. Reconocimiento a Distancia Limitado: El algoritmo tiene dificultades para reconocer al leopardo de las nieves a distancias significativas. La pérdida de detalles en imágenes de baja resolución es un desafío.
3. Diferenciación de Especies Similares: El algoritmo no distingue de manera efectiva entre el leopardo de las nieves y otros felinos similares. Esto se debe a la necesidad de un conjunto de datos más amplio y diverso.

### 4.3.2 Segunda ejecución

La segunda iteración del proyecto se enfrentó a desafíos notables al incorporar un conjunto de datos más amplio y diverso. La inclusión de numerosas imágenes nuevas mediante

un plugin de descarga de imágenes de la web, junto con la adición de datos de otros animales y un conjunto de 1700 imágenes de Roboflow, resultó en un aumento significativo en el tamaño del conjunto de datos. Sin embargo, esta expansión también reveló problemas críticos en términos de recursos computacionales.

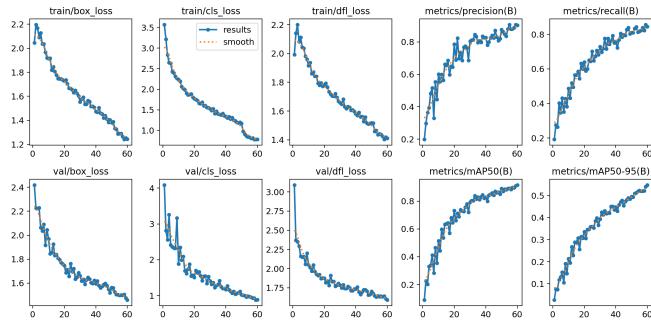


Figura 4.15: Métricas generales de la segunda iteración

La ejecución del algoritmo se volvió prácticamente imposible debido a la insuficiencia de recursos, lo que se tradujo en un tiempo de entrenamiento extremadamente largo, alcanzando hasta 10 horas en el mejor de los casos. La complejidad del nuevo conjunto de datos parece haber afectado negativamente los resultados (Figura 4.15) del modelo, que fueron sorprendentemente peores que en la iteración anterior.

En vista de estos desafíos, se sugiere considerar una reducción del tamaño del conjunto de datos, seleccionando cuidadosamente las imágenes más relevantes. Un análisis detallado de los resultados obtenidos proporcionaría puntos valiosos para identificar áreas específicas que requieren mejoras. Se propone abordar estos desafíos de manera gradual, priorizando las correcciones a los problemas más críticos y avanzando hacia mejoras más complejas.

### 4.3.3 Tercera ejecución

La tercera iteración de nuestro proyecto se basa en un conjunto de datos propio, compuesto por alrededor de 170 imágenes. Durante esta fase, se ha identificado un problema en la configuración de YOLOv8: las imágenes sin etiquetas asignadas (sin detección) no están siendo consideradas por el modelo. Esto plantea dificultades en la detección de otros seres vivos que no están específicamente etiquetados, lo que se convierte en una limitación.

Para abordar esta cuestión, en la segunda fase de nuestro proyecto planeamos introducir una nueva clase de especie y evaluar si YOLOv8 puede realizar una diferenciación adecuada, incluso cuando no existen etiquetas previas. Este enfoque nos permitirá comprobar la capacidad del modelo para adaptarse a la detección de especies no especificadas inicialmente.

En cuanto a la mejora de la velocidad en la ejecución de videos, se ha logrado un incremento al aprovechar más la capacidad de procesamiento de la [Central Processing Unit \(CPU\)](#)

mediante Compute Unified Device Architecture (CUDA). Sin embargo, tras realizar el entrenamiento con 60 iteraciones, se observa que las detecciones en los videos proporcionan resultados ligeramente mejores (Figuras 4.16 y 4.17), pero el porcentaje de detección es considerablemente menor. Aunque el modelo parece ignorar más ruido, lo cual es positivo, la disminución en el porcentaje de detección es un aspecto a tener en cuenta.

Este hallazgo indica un equilibrio delicado entre la supresión de falsos positivos (ruido) y la capacidad de detectar adecuadamente los objetos de interés. La optimización futura deberá buscar un punto medio que permita un rendimiento eficiente en términos de precisión y velocidad. En resumen, la tercera iteración revela nuevos desafíos que requerirán una estrategia específica en la siguiente fase del proyecto.



Figura 4.16: Resultado tercera iteración

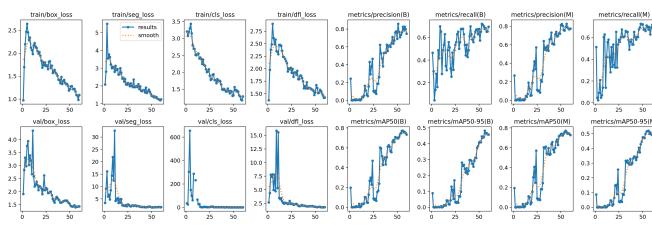


Figura 4.17: Métricas generales de la tercera iteración

## Capítulo 5

# Desarrollo: Parte 2

---

**E**n esta segunda fase del proyecto, se llevará a cabo la integración de una nueva detección en la base de datos existente. El objetivo principal de esta inclusión es evaluar la capacidad del algoritmo para distinguir entre las clases previamente establecidas. No se profundizará demasiado en detalles específicos, ya que el enfoque se centrará en comprobar la viabilidad de la discriminación entre las clases ya definidas.

### 5.1 **Hapalochlaena maculosa**

*Hapalochlaena maculosa* [19] es una especie de pulpo perteneciente a la familia Octopodidae, conocida comúnmente como pulpo de anillos azules. Su nombre científico hace referencia a sus características manchas azules distintivas que adornan su cuerpo. Esta especie, endémica de las aguas del océano Pacífico, se encuentra principalmente en las costas de Australia. Su presencia se extiende desde las aguas costeras hasta las profundidades marinas, y su hábitat natural incluye arrecifes de coral y fondos marinos.

Una de las características más destacadas de *H. maculosa* es su habilidad para cambiar de color y textura, adaptándose camuflajeadamente a su entorno. Este comportamiento, combinado con sus anillos azules brillantes, sirve como una estrategia de advertencia ante posibles depredadores. Además de su capacidad camufladora, este pulpo se caracteriza por su tamaño relativamente pequeño, con longitudes que oscilan entre 5 y 8 centímetros.

Sin embargo, a pesar de su tamaño modesto, *Hapalochlaena maculosa* (Figura 5.1) posee un veneno extremadamente potente. Este veneno contiene neurotoxinas y compuestos bioactivos que le otorgan una capacidad letal, especialmente a sus presas y, en casos raros, a los seres humanos. La presencia de estos anillos azules, que funcionan como una señal visual de advertencia, sugiere un comportamiento aposemático, indicando la peligrosidad de la especie.

A pesar de su fascinante biología, el estudio de *Hapalochlaena maculosa* presenta varios desafíos para los investigadores. En primer lugar, la población de esta especie se enfrenta a

amenazas ambientales, como la pérdida de hábitat debido a la degradación de los arrecifes de coral y el cambio climático dificultando la comprensión de su ecología y comportamiento.

Además, la observación en el entorno natural de esta especie se complica debido a su habilidad para camuflarse y a la dificultad de acceso a ciertas áreas marinas. La baja visibilidad en hábitats marinos y la naturaleza esquiva de este pulpo hacen que el estudio de su comportamiento y biología sea un desafío significativo.



Figura 5.1: *Hapalochlaena maculosa*

## 5.2 *Panthera Tigris Tigris*

El Tigre de Bengala, *Panthera tigris tigris* (Figura 5.2), es un felino de la familia Felidae que habita en el subcontinente indio. Reconocido por su pelaje anaranjado con rayas negras y

blancas en la parte ventral, este majestuoso felino se adapta a diversos entornos, desde selvas densas hasta llanuras abiertas. Su comportamiento solitario y territorial, junto con habilidades de caza excepcionales, lo convierten en una especie emblemática y esencial para el equilibrio ecológico.

A pesar de su estatus emblemático, el estudio del Tigre de Bengala plantea desafíos significativos. La pérdida de hábitat debido a la expansión humana y la deforestación, así como el comercio ilegal de partes del cuerpo, amenazan su supervivencia. La naturaleza evasiva y territorial de estos felinos complica la observación directa, destacando la necesidad de tecnologías de detección, como cámaras trampa, para recopilar datos cruciales sobre su comportamiento y movimiento.



Figura 5.2: *Panthera Tigris Tigris* captado por cámara trampa en la India

### 5.3 *Ailuropoda melanoleuca*

El Panda Gigante, *Ailuropoda melanoleuca* (Figura 5.3), es un mamífero perteneciente a la familia Ursidae que habita en las regiones montañosas de China central. Conocido por su distintivo pelaje blanco y negro, este carismático oso se adapta a los densos bosques de bambú, su principal fuente de alimento. Su naturaleza tranquila y solitaria lo convierte en un símbolo icónico de la conservación de la biodiversidad.

A pesar de su popularidad, la preservación del Panda Gigante enfrenta desafíos significativos. La fragmentación del hábitat debido a la actividad humana, como la tala de bosques y la expansión urbana, así como la caza furtiva, han puesto en peligro su existencia. La reproducción en cautiverio y los esfuerzos de conservación han sido fundamentales para su

supervivencia, pero aún queda mucho por hacer para garantizar su protección a largo plazo. La investigación científica, utilizando tecnologías como la genética y la telemetría, proporciona información crucial para comprender su comportamiento y hábitos de movimiento, lo que es fundamental para su conservación.



Figura 5.3: *Ailuropoda melanoleuca*

## 5.4 Ejecución del dataset final

En esta nueva iteración, hemos tomado medidas para abordar la escasez de recursos físicos y la baja eficiencia observada en los resultados anteriores. Dada la limitación de nuestros recursos locales, hemos optado por una solución innovadora: la virtualización. Para llevar a cabo este enfoque, hemos recurrido a un servicio de pago en la nube que proporciona un equipo remoto con recursos computacionales significativos, específicamente a través de Google Colab [20].

A través de este servicio en la nube, hemos adquirido temporalmente una máquina de escritorio (Figura 5.4) con una capacidad de cómputo suficiente para ejecutar el algoritmo necesario. Este enfoque nos permite superar las limitaciones de nuestros recursos locales y aprovechar la potencia de procesamiento adicional disponible en la nube.

## 5.5 GPU vs CPU

En esta nueva aproximación, disponemos de acceso a una Unidad de Procesamiento Gráfico (**GPU**) de mayor rendimiento a través de Google Colab, lo que nos otorga una capacidad de

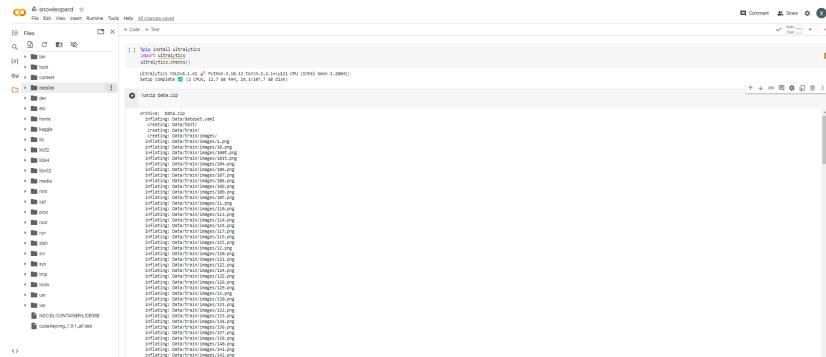


Figura 5.4: Entorno de escritorio remoto en Google Colab

procesamiento notablemente mejorada (recalcar que con **CUDA** ya usábamos **GPU** pero con poco rendimiento) en comparación con la ejecución en la Unidad Central de Procesamiento (**CPU**).

En el ámbito de los sistemas de inteligencia artificial, la elección entre el uso de **CPU** y **GPU** [21] es fundamental debido a las exigencias computacionales de estos sistemas. Mientras que las **CPU**s son eficientes para realizar operaciones complejas en un tiempo reducido, su limitación en la capacidad de realizar múltiples operaciones simultáneas puede resultar en un procesamiento lento de modelos complejos de inteligencia artificial. Por el contrario, las **GPU**s están diseñadas para llevar a cabo un gran número de operaciones simples de manera paralela, lo que las convierte en la opción preferida para el procesamiento de datos en aplicaciones de inteligencia artificial.

## 5.6 Pruebas finales

Para optimizar la precisión y eficacia del algoritmo, hemos construido un conjunto de datos completo que incluye imágenes representativas de cada especie, en las cuales constan las 7 especies, mencionadas anteriormente. Este conjunto de datos se ha dividido adecuadamente en conjuntos de validación, entrenamiento y prueba para garantizar la robustez del modelo. Además, hemos seleccionado la versión más potente del algoritmo de detección YOLO, específicamente la versión "x", para maximizar la precisión en la identificación de las especies en cuestión.

Se llevó a cabo un entrenamiento utilizando un conjunto de datos compuesto por aproximadamente 200 imágenes recopiladas y segmentadas manualmente, con una duración de 60 iteraciones. Se observa que, en el caso de especies como el Panda (*Ailuropoda melanoleuca*), los resultados son menos precisos debido a la escasez de imágenes segmentadas disponibles para el entrenamiento. Por otro lado, en la mayoría de los casos, el leopardo de las nieves, que constituye prácticamente la totalidad de nuestro conjunto de datos, muestra detecciones

satisfactorias. Aunque en algunos casos puede presentar confusiones con otros felinos, estas son considerablemente menos frecuentes en comparación con los resultados (Figuras 5.5 y 5.6) obtenidos en iteraciones anteriores.

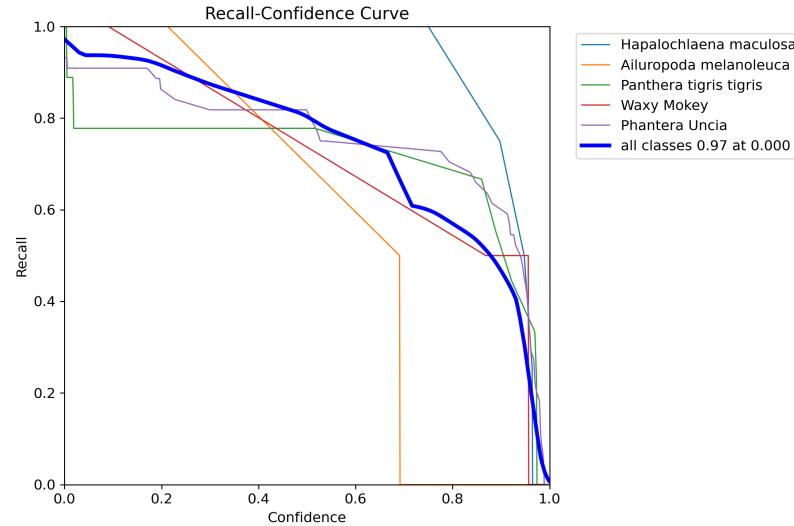


Figura 5.5: Grafica iteración Google Colab

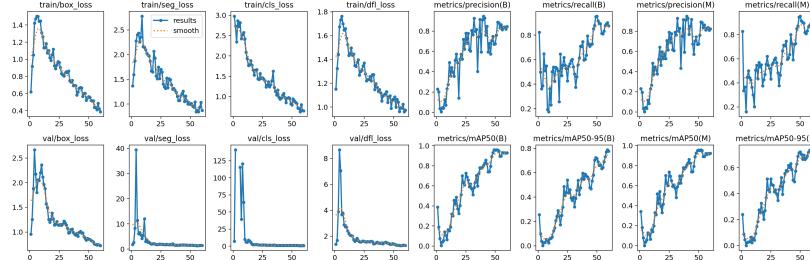


Figura 5.6: Grafica resultados Google Colab

El principal desafío se encuentra al procesar con una alta tasa de Data Frames (Frames), ya que el algoritmo no logra detectar ningún objeto. Sin embargo, en el caso de imágenes o videos con menos Frames, se logran detecciones más precisas. A pesar de las posibles mejoras que podrían implementarse, se destaca que el algoritmo es capaz de distinguir entre diferentes animales presentes en el conjunto de datos. Esto sugiere que podría ser utilizado para una variedad de especies que comparten el mismo hábitat, y además, evidencia un amplio margen para continuar mejorando su desempeño.

En las siguientes imágenes (Figuras 5.7, 5.8 y 5.9) podemos ver las predicciones que hizo el algoritmo en el conjunto de validación, y por consiguiente, algunos de sus errores. Además incluye la especie identificada por un color diferente y porcentaje de acierto.



Figura 5.7: Primera predicción

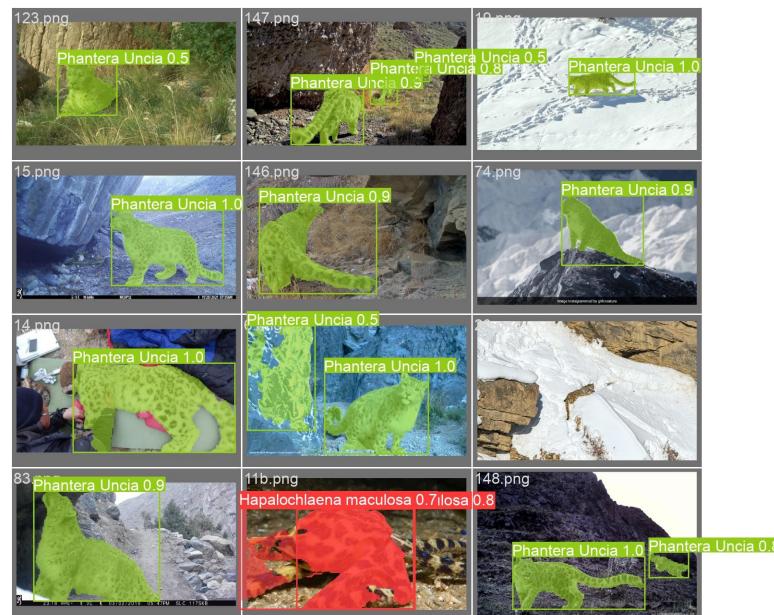


Figura 5.8: Segunda predicción

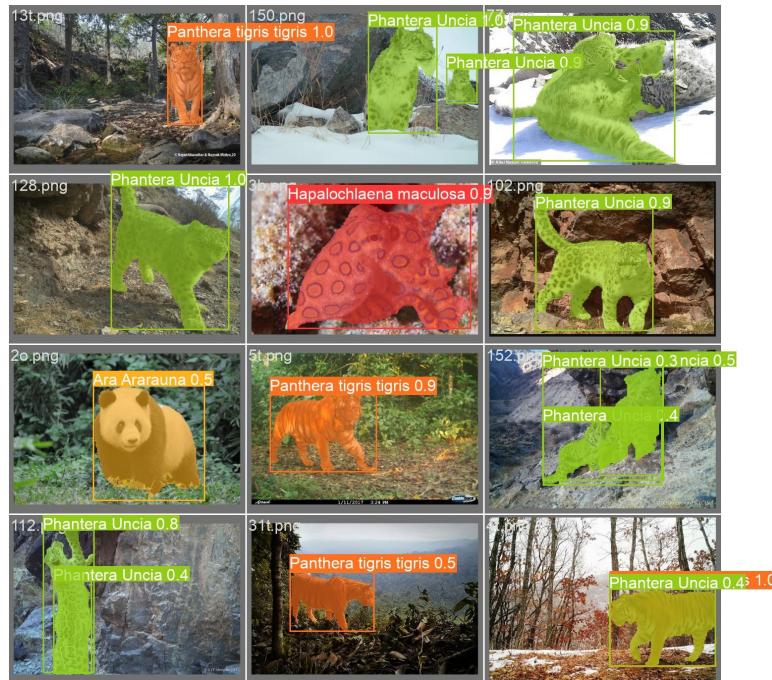


Figura 5.9: Tercera predicción

Algunos ejemplos de detecciones en vídeo en las Figuras 5.10 y 5.11 de diferentes clases:



Figura 5.10: Ejemplo de detección en vídeo del leopardo de las nieves

También se ha utilizado un script diferente al anterior para probar las casuísticas en imágenes (Figura 5.12) nuevas donde conseguimos muchos menos errores que en las reproducciones en vivo.

Se realizó una nueva aproximación al problema de detección de especies utilizando Google Colab. Se ajustaron varios parámetros clave, incluyendo el número de épocas y el tamaño del lote (batch size). Además, se incluyeron imágenes adicionales de especies que anteriormente



Figura 5.11: Ejemplo de detección en vídeo del tigre de bengala

```

/content/ultralytics
from ultralytics import YOLO
from IPython.display import Image, display
# Cargar el modelo entrenado YOLOv8 desde el archivo de pesos
model = YOLO('/content/runs/segment/train4/weights/best.pt')

# Definir la ruta hacia el archivo de imagen
source = '/content/prueba.jpg'

# Ejecutar la inferencia en la imagen
results = model(source)

# Los resultados están en un objeto Results que se puede manipular para mostrar o guardar los resultados.
results[0].show() # Mostrar los resultados anotados en la pantalla
results[0].save() # Guardar los resultados anotados en un archivo

# Asegúrate de que la ruta a la imagen es correcta
display(Image(filename='/content/ultralytics/results_prueba.jpeg'))

```

image 1/1 /content/prueba.jpg: 384x640 1 Phantera Uncia, 66.7ms  
Speed: 1.7ms preprocess, 66.7ms inference, 1.9ms postprocess per image at shape (1, 3, 384, 640)

Figura 5.12: Ejemplo en imagen

causaban conflictos en las detecciones. Se incrementó el número de épocas de entrenamiento de 60 a 100 para permitir una mayor iteración y refinamiento del modelo. El tamaño del lote se aumentó de 4 a 8, lo que potencialmente permitiría un procesamiento más rápido de los datos y una convergencia más estable durante el entrenamiento. Las imágenes se agregaron tanto al conjunto de entrenamiento como al conjunto de validación.

La inclusión de las imágenes adicionales de especies conflictivas resultó en una mejora

significativa en la precisión de las detecciones (Resultados en la Figura 5.13). Se observó una reducción en los errores de clasificación, siendo la especie *Lynx pardinus* la que más conflictos acarrea en esta iteración, debido a su parecido con otros felinos.

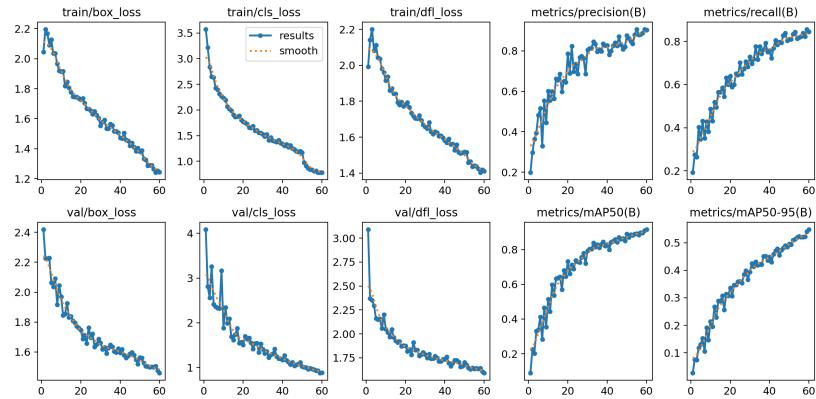


Figura 5.13: Nuevos resultados

Después de observar las imágenes resultantes (Figuras 5.14, 5.15, 5.16) se aprecia un notable aumento en la precisión de la detección, donde el algoritmo reconoce adecuadamente la especie en la mayoría de los casos, con excepciones mínimas. Se concluye que el enriquecimiento del Data Set (Dataset) con un mayor número de imágenes, potencialmente cientos o miles por especie, mejorarían aún más las detecciones. Sin embargo, el objetivo del proyecto no reside en alcanzar la optimización máxima, sino en lograr una aproximación eficiente que pueda integrarse fácilmente con otras tecnologías y entornos.

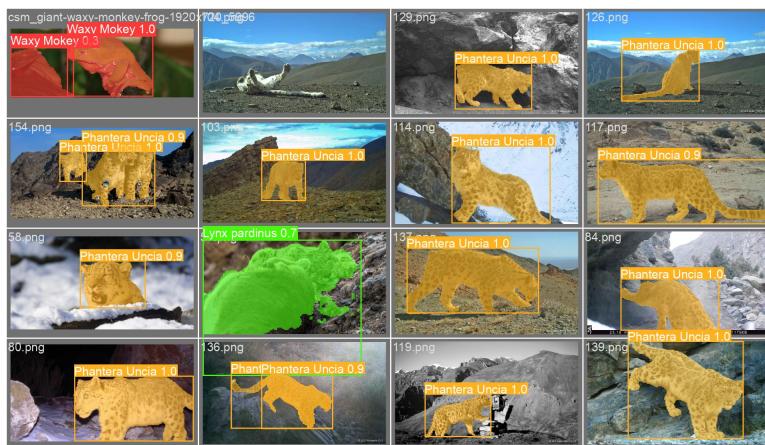


Figura 5.14: Primera predicción

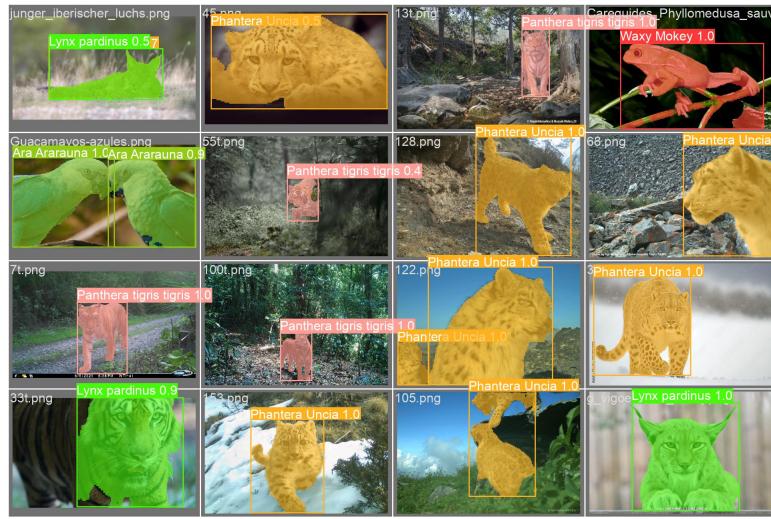


Figura 5.15: Segunda predicción

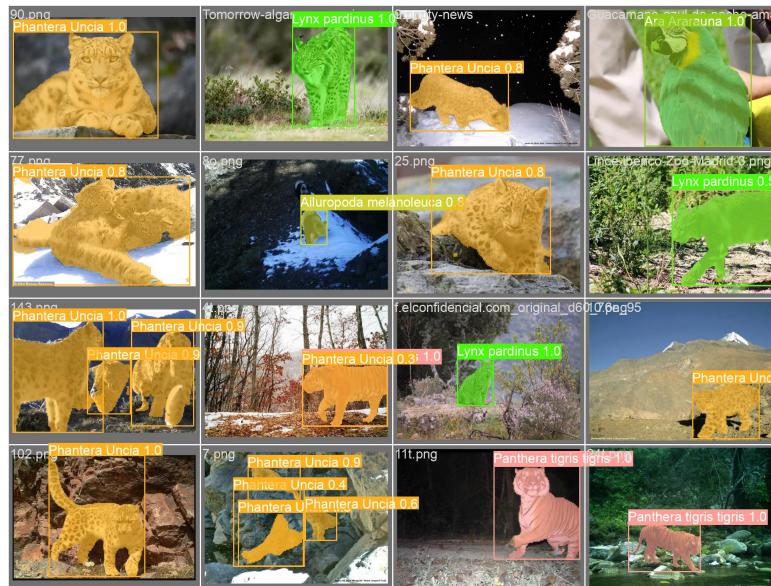


Figura 5.16: Tercera predicción

## 5.7 Resultados finales, validación y discusión

Cuando evaluamos el rendimiento de un modelo de clasificación, es crucial entender cómo se comporta en términos de los datos que predice correctamente y aquellos que no. Aquí es donde entran en juego conceptos (Figura 5.17) como los verdaderos positivos (VP), los falsos positivos (FP), los verdaderos negativos (VN) y los falsos negativos (FN).

- Verdaderos Positivos (VP): Son los casos en los que el modelo predice correctamente que un ejemplo pertenece a la clase positiva.

MÉTRICA	CARACTERÍSTICAS	FÓRMULA DE CÁLCULO
Precisión	Del total de las predicciones positivas, se evalúa cuánto de eso era realmente cierto	$\text{Precisión} = \frac{\text{TP}}{\text{TP} + \text{FP}}$
Recall	Determina la cantidad de predicciones positivas sobre todo lo que era realmente positivo	$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$
F1	Combina las medidas de precisión y recall en un sólo valor un sólo valor	$\text{F1} = 2 \times \frac{\text{precisión} \times \text{recall}}{\text{precisión} + \text{recall}}$
Exactitud	Representa el porcentaje de predicciones en donde el modelo ha acertado sobre el total de datos	$\text{Exactitud} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$

Figura 5.17: Métricas para evaluar el desempeño de un modelo

- Falsos Positivos (FP): Estos son los casos en los que el modelo predice incorrectamente que un ejemplo pertenece a la clase positiva cuando, de hecho, pertenece a la clase negativa.

- Verdaderos Negativos (VN): Son los casos en los que el modelo predice correctamente que un ejemplo pertenece a la clase negativa.

- Falsos Negativos (FN): Estos son los casos en los que el modelo predice incorrectamente que un ejemplo pertenece a la clase negativa cuando, en realidad, pertenece a la clase positiva.

Ahora, hablemos sobre recall y precisión [22]:

- Recall (Recuperación o Sensibilidad): Es la proporción de casos positivos que fueron correctamente identificados por el modelo sobre el total de casos positivos reales. Se calcula como  $\text{TP} / (\text{TP} + \text{FN})$ . En otras palabras, mide la capacidad del modelo para encontrar todos los casos positivos.

- Precisión: Es la proporción de casos positivos que fueron correctamente identificados por el modelo sobre el total de casos que el modelo predijo como positivos. Se calcula como  $\text{TP} / (\text{TP} + \text{FP})$ . La precisión mide la exactitud de las predicciones positivas del modelo.

En resumen, el recall dice la calidad del modelo para capturar todos los casos positivos, mientras que la precisión nos transmite la confiabilidad de las predicciones positivas del modelo.

Al interpretar estas métricas, es fundamental considerar el equilibrio entre ellas. Por ejemplo, un modelo puede tener un alto recall pero una baja precisión, lo que significa que captura la mayoría de los casos positivos pero también clasifica erróneamente muchos negativos como positivos. Por otro lado, un modelo con alta precisión pero bajo recall puede estar ignorando muchos casos positivos.

Para tener una evaluación más completa, a menudo se utilizan medidas que combinan recall y precisión, como el puntaje F1 [23], que es la media armónica de recall y precisión, lo que proporciona una medida única del rendimiento del modelo que tiene en cuenta tanto los

falsos positivos como los falsos negativos.

La "F1-Confidence Curve" es una representación gráfica que muestra cómo varía el puntaje F1 del modelo en función del umbral de confianza utilizado para las detecciones de objetos. Este tipo de gráfico es especialmente útil para comprender cómo el equilibrio entre precisión y recall cambia a medida que ajustamos el umbral de confianza del modelo.

En este gráfico 5.18 el eje X representa el umbral de confianza, que es un valor entre 0 y 1 que determina la probabilidad mínima requerida para que una detección sea considerada válida. El eje Y representa el puntaje F1, que es una medida que combina la precisión y el recall del modelo. Cuanto mayor sea el puntaje F1, mejor será el equilibrio entre la precisión y el recall del modelo.

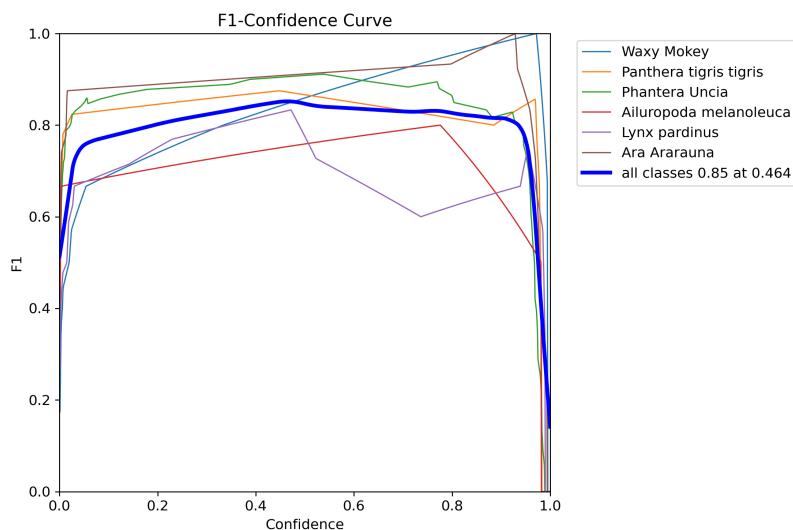


Figura 5.18: F1-Confidence Curve

La interpretación de esta curva implica buscar un punto que maximice el puntaje F1. Este punto ideal generalmente representa el mejor equilibrio entre la precisión y el recall del modelo para la aplicación específica. Sin embargo, la elección del umbral de confianza dependerá de los requisitos específicos del problema. Por ejemplo, si la aplicación requiere una alta precisión, optar por un umbral de confianza más alto, lo que resultará en menos falsos positivos pero posiblemente en un menor recall. Por otro lado, si es más importante capturar la mayoría de los casos positivos, bajar el umbral de confianza, lo que probablemente aumentará el recall a expensas de una menor precisión. Podemos ver que entorno a 0.8 se encuentran los valores con mejor puntaje para F1.

La interpretación de la curva de "Precisión-Confianza" es análoga a la anterior, con la tendencia típica de que a medida que el umbral de confianza aumenta, la precisión también lo hace. Este fenómeno se refleja en la Figura 5.19, donde se observa que un umbral más alto se

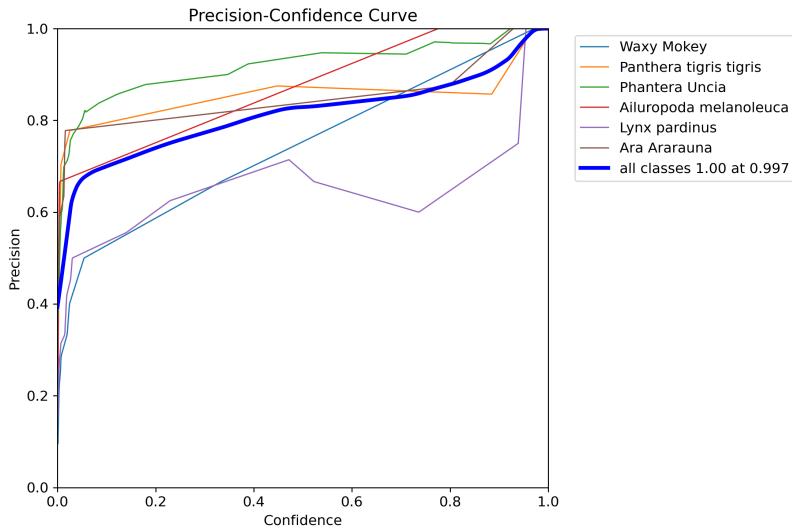


Figura 5.19: Precision-Confidence Curve

traduce en una mayor precisión, aunque con una menor cantidad de detecciones.

En la última tabla "Precision-Recall Curve" (Figura 5.20) podemos ver la relación entre la precisión y el recall. En el contexto de YOLO (You Only Look Once), la métrica "mAP@0.5" significa "mean Average Precision at 0.5 Intersection over Union (IoU)".

"Mean Average Precision" (mAP) es una métrica comúnmente utilizada para evaluar el rendimiento de un modelo de detección de objetos. Esta métrica calcula el promedio de las precisiones en diferentes umbrales de IoU (Intersección sobre Unión) para cada clase. "@0.5" indica el valor de IoU utilizado para calcular la precisión promedio. En este caso, se utiliza un umbral de IoU de 0.5, lo que significa que se considera que una detección es correcta si la relación entre el área de intersección y la unión entre la detección y la verdad fundamental es mayor o igual al 50 por ciento.

Por lo tanto, en la frase "all classes 0.8999 mAP@0.5", "all classes" indica que el mAP se calculó para todas las clases detectadas por el modelo, y "0.8999" es el valor del mAP calculado a un umbral de IoU de 0.5. Esto significa que, en promedio, el modelo logra una precisión del 89.99 por ciento en la detección de objetos en todas las clases consideradas cuando se utiliza un umbral de IoU de 0.5. Pese a que no conseguimos un umbral muy alto, cuanto mayor sea el valor de mAP, mejor será el rendimiento del modelo de detección de objetos.

Por tanto, excepto en la precisión, que cuenta con buenos valores de umbral, en el resto conseguimos los mejores valores con umbrales medianamente bajos. En cuanto a precisión y recall conseguimos buenos resultados en ambos, con los peores resultados en las especies *Ailuropoda melanoleuca* y *Lynx pardinus*. Específicamente, como podemos ver en las Figura 5.21, en la época 96 conseguimos unos valores de precisión del 0.90964 y de recall del 0.80194.

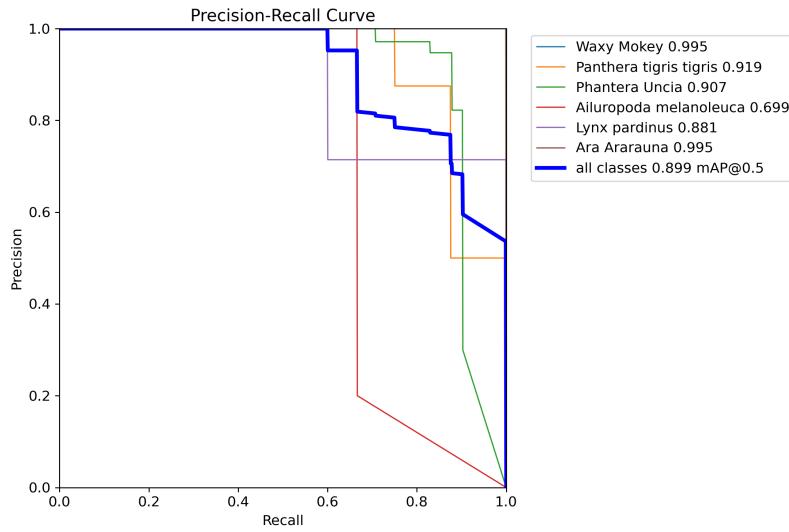


Figura 5.20: Precision-Recall Curve

	epoch	train/box_loss	train/seg_loss	train/cls_loss	train/dfl_loss	metrics/precision(B)	metrics/recall(B)	metrics/mAP50(B)	metrics/mAP50-95	metrics/precision(M)	t
93	92	0.27946		0.63254	0.37055	0.86655	0.74879	0.84796	0.82252	0.69164	0.83205
94	93	0.25653		0.59364	0.34209	0.85921	0.83609	0.83532	0.84521	0.72738	0.86138
95	94	0.26422		0.61881	0.34357	0.86742	0.85023	0.84508	0.86377	0.7384	0.86535
96	95	0.28626		0.61398	0.33485	0.91612	0.8695	0.84583	0.88513	0.735	0.86574
97	96	0.24612		0.57432	0.30938	0.86022	0.90964	0.80194	0.88825	0.75794	0.90964
98	97	0.29447		0.67617	0.37894	0.90093	0.88353	0.82733	0.88777	0.76273	0.88535
99	98	0.23941		0.54109	0.29377	0.84316	0.85899	0.88692	0.8995	0.7731	0.85899
100	99	0.25561		0.57755	0.3008	0.85878	0.82444	0.90237	0.88945	0.76841	0.82444
101	100	0.25333		0.59954	0.34175	0.88503	0.82431	0.90272	0.89927	0.77571	0.82431

Figura 5.21: Resultados por época

Recalcar que desde la perspectiva del animal, pueden surgir varios desafíos en la detección y clasificación de imágenes:

- **Perspectiva del animal:** Cuando el animal está demasiado cerca del dispositivo de captura, es posible que solo una parte de él sea visible, lo que dificulta su identificación completa.
- **Tamaño del animal en la imagen:** La distancia a la que se encuentra el animal al momento de la captura puede afectar su tamaño en la imagen, lo que a su vez puede afectar la capacidad del sistema de clasificación para identificarlo correctamente.
- **Animales interespecie e intraespecie:** Clasificar animales de especies diferentes, como un elefante y un tigre, suele ser más fácil debido a sus diferencias obvias. Sin embargo, el sistema puede tener dificultades para clasificar animales con características similares, como el gato andino y el gato colo colo.
- **Condiciones climáticas y luz:** La presencia de neblina, niebla, lluvia o nieve puede generar ruido en las imágenes, lo que dificulta el proceso de clasificación. Además, la

variación en la cantidad de luz y sombra puede afectar la visibilidad del animal.

- **Calidad de la imagen:** Las imágenes borrosas debido a la velocidad del animal o a condiciones de poca luz pueden dificultar su detección y clasificación.
- **Oclusión y manchado de la cámara:** Si el lente de la cámara está empañado debido a condiciones climáticas adversas o está obstruido por vegetación al capturar la imagen del animal, el sistema puede tener dificultades para clasificarlo correctamente.

Es importante destacar que diversas causas, como las mencionadas en la Figura 5.22, pueden influir significativamente en la calidad de resultados. Estos factores pueden hacer que los resultados sean peores en algunos casos, comprometiendo la fiabilidad de nuestro sistema de clasificación.



Figura 5.22: Factores que pueden generar errores en la clasificación

## Capítulo 6

# Análisis de datos

---

EN este capítulo, se lleva a cabo un análisis de los datos recopilados mediante el sistema de detección de animales utilizando cámaras trampa. Se exploran las técnicas utilizadas para integrar y gestionar los datos en MongoDB, una base de datos [Not Only SQL \(NoSQL\)](#) especialmente adecuada para manejar grandes volúmenes de datos visuales. Además, se detallan los procedimientos implementados para distinguir el ruido y las falsas detecciones, así como para almacenar y analizar la información geoespacial de las detecciones. Por último, se presenta un informe visual generado con Power BI, que permite una interpretación de los resultados obtenidos en el estudio.

### 6.1 Integración de MongoDB

MongoDB es una base de datos [NoSQL](#) diseñada para manejar eficientemente grandes volúmenes de datos en aplicaciones modernas. En lugar de seguir el modelo relacional tradicional, MongoDB utiliza documentos [BSON](#) almacenados en colecciones. Estos documentos, similares a [JSON](#), ofrecen flexibilidad en la estructura de datos y se agrupan en colecciones en lugar de tablas.

En el contexto de la detección de animales mediante cámaras trampa, MongoDB presenta ventajas específicas. Su capacidad para almacenar imágenes de manera eficiente utilizando el tipo de dato Binary en documentos [BSON](#) facilita la gestión de grandes volúmenes de datos visuales. Además, la compatibilidad con índices geoespaciales permite un almacenamiento eficiente y consultas basadas en ubicación, fundamental en la recopilación de datos geolocalizados en la vigilancia de animales.

MongoDB también destaca por su esquema flexible, permitiendo la adaptación sencilla a cambios en los datos o la incorporación de nuevos campos sin requerir una estructura de tabla fija. Esta flexibilidad es crucial en entornos de detección de animales, donde los atributos de los datos pueden variar. Asimismo, las capacidades de consultas complejas y operaciones de

agregación facilitan el análisis de datos recopilados, como contar el número de avistamientos en una región específica durante un período de tiempo.

La replicación y escalabilidad horizontal son aspectos técnicos adicionales relevantes en la implementación de MongoDB para la detección de animales. La replicación asegura la disponibilidad de datos, permitiendo la recuperación en caso de fallos, mientras que la escalabilidad horizontal facilita la gestión eficiente de grandes volúmenes de datos generados por múltiples cámaras trampa.

En comparación con bases de datos relacionales ([SQL](#)), MongoDB supera en entornos donde la estructura de datos puede ser variable y la escala es considerable. En comparación con otras bases de datos [NoSQL](#), la capacidad de MongoDB para consultas y operaciones de agrupación, su soporte para índices geoespaciales y su rendimiento escalable son factores clave que lo destacan. La elección entre MongoDB y otras bases de datos dependerá de las necesidades específicas del proyecto, evaluando la complejidad del esquema, los requisitos de geolocalización y la capacidad de escalabilidad horizontal. En conclusión, MongoDB emerge como una opción sólida para proyectos de detección de animales con cámaras trampa, ofreciendo flexibilidad, eficiencia en el manejo de datos visuales y capacidades avanzadas de consulta.

Durante este proyecto, creamos una base de datos local para registrar especies detectadas, almacenando la hora, ubicación y una imagen segmentada de cada especie (Figura 6.1). Aunque simple, esta base de datos demuestra su potencial para el análisis de datos, a pesar de las problemáticas, como la inserción de las imágenes segmentadas que se han convertido a base 64 para facilitar el almacenamiento.

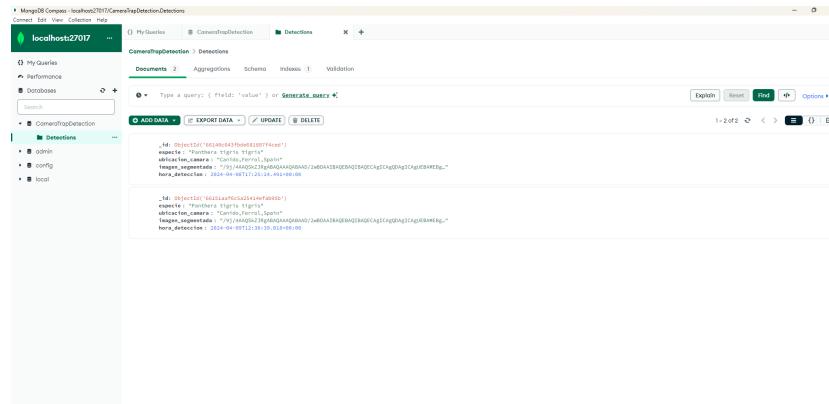


Figura 6.1: Datos guardados en MongoDB

Es importante destacar que la implementación de MongoDB se lleva a cabo localmente. Sin embargo, es posible implementarla en la nube con componentes adicionales. Si las condiciones del hábitat a detectar lo permiten, se asume que en ese escenario el microcontrolador recopilaría datos con la base de datos local.

## 6.2 Distinción de ruido y falsas detecciones

Para mitigar la acumulación de datos irrelevantes en nuestra base de datos, hemos ajustado el umbral de confianza mínimo a 0.7, descartando detecciones con menor certeza. Además, hemos establecido un intervalo de tiempo mínimo entre detecciones del mismo individuo para evitar registros duplicados. Aunque limitar las detecciones a una por minuto puede conllevar la pérdida de algunas detecciones (más de un individuo a posteriori), consideramos que es necesario para mantener la relevancia de los datos almacenados y evitar la saturación de la base de datos con información redundante. Estas medidas de filtrado y limitación garantizan que nuestra base de datos contenga únicamente detecciones significativas, facilitando así su utilidad en el análisis de datos biológicos.

## 6.3 Almacenamiento de geoposiciones

Mediante la localización predefinida previamente de la cámara incluimos coordenadas y mapa terrestre de la detección en la base de datos. También con Power BI haremos mapas coperlóticos [24] para analizar las distancias y hábitats a nivel general (porcentaje de individuos por zonas etc). Sería posible almacenar las coordenadas de posición mediante módulos GPS pero no se ha tenido en cuenta ya que las cámaras no son móviles en este caso.

## 6.4 Análisis de información

En la base de datos almacenamos la segmentación del individuo junto con otros datos como el ID, hora, geoposición, clase de animal y porcentaje de acierto. Además, incluiremos datos que se podrán visualizar superpuestos en un tipo de informe completo en Power BI [25].

En primera instancia, se ha creado un informe utilizando Power BI que incluye diversos elementos visuales para una mejor comprensión y análisis de los datos. Entre los elementos visuales más destacados del informe (Figura 6.2) se encuentran:

- **Total de Población Detectada:** Un gráfico que muestra el número total de individuos identificados en el estudio.
- **Mapa de Calor de los Reconocimientos:** Un mapa interactivo que resalta las áreas con mayor densidad de reconocimientos, proporcionando una visión geográfica de la distribución de los datos.
- **Porcentaje de Población por Zona:** Un gráfico circular o de barras que desglosa el porcentaje de la población en diferentes zonas, facilitando la comparación entre distintas áreas.

- Enumeración de Zonas:** Una lista detallada de las diferentes zonas analizadas junto con su respectivo número de población y otras métricas relevantes.

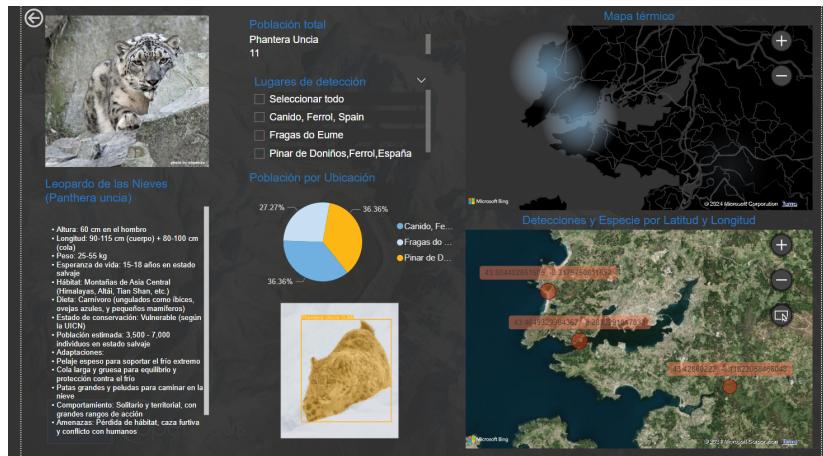


Figura 6.2: Informe Visual creado con Power Bi

Además de estos elementos visuales, el informe incluye un pequeño resumen que sintetiza información de la especie.

Inicialmente, los datos se han importado manualmente desde una tabla Excel, que se puede exportar desde MongoDB. Este enfoque permite una rápida incorporación de los datos pero requiere actualizaciones manuales cada vez que se dispone de nueva información.

Sin embargo, una buena práctica para mejorar la eficiencia y la precisión del informe sería conectar directamente Power BI con la base de datos MongoDB. Esto se puede lograr mediante la configuración de una conexión en tiempo real, lo que permitiría que el informe se actualice automáticamente cada vez que se modifiquen los datos en la base de datos. Este método no solo ahorraría tiempo, sino que también garantizaría que el informe siempre refleje la información más reciente disponible, mejorando así la toma de decisiones basada en datos actualizados y precisos. Recalcar que obviamente las localizaciones son ficticias.

Para sincronizar Power BI con la base de datos, inicialmente fue necesario conectar MongoDB Compass, nuestra herramienta de interfaz local para la base de datos, con MongoDB Atlas (Figura 6.3, una interfaz más actualizada que ofrece opciones tanto gratuitas como de pago [26], incluyendo la sincronización y conectividad con Power BI.

Una vez sincronizado Power BI y MongoDB, podemos organizar las páginas con los elementos visuales por especie mediante filtrados de datos de las columnas obtenidas de la base de datos. Un ejemplo de informe sería el que podemos observar en la figura 6.2.

En este informe de Power BI, se ha logrado identificar y monitorear a 11 ejemplares de la especie Panthera uncia, comúnmente conocida como leopardo de las nieves, en tres zonas diferenciadas. El informe proporciona las coordenadas exactas de las cámaras utilizadas para

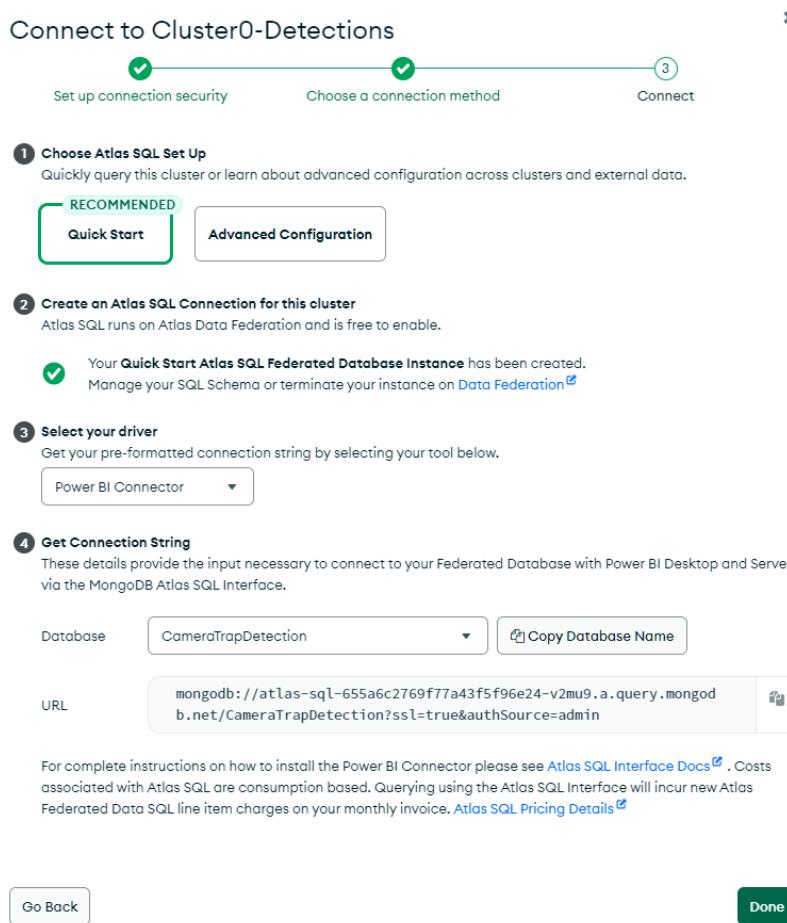


Figura 6.3: Configuración en Mongo Atlas

la detección, lo que permite visualizar un mapa de calor que ilustra la distribución geográfica de estos animales.

Además, se presenta una distribución porcentual de las detecciones, ofreciendo una visión clara de la concentración de ejemplares en las distintas zonas. Un pequeño resumen acompaña estos datos, destacando los puntos más relevantes del estudio.

Un aspecto crucial del informe es la capacidad de Power BI para filtrar la información (Figura 6.4 con otra especie filtrada). Los usuarios pueden seleccionar datos específicos por localización o porcentaje, permitiendo un análisis más detallado y personalizado según las necesidades del estudio.

Con la implementación de localizaciones reales y un análisis más exhaustivo, este informe podría proporcionar información aún más valiosa. Sería posible obtener datos sobre áreas de población más sofisticadas, patrones de migración, el estado de salud y sexo de los ejemplares,

así como el impacto ambiental que tienen en su hábitat.

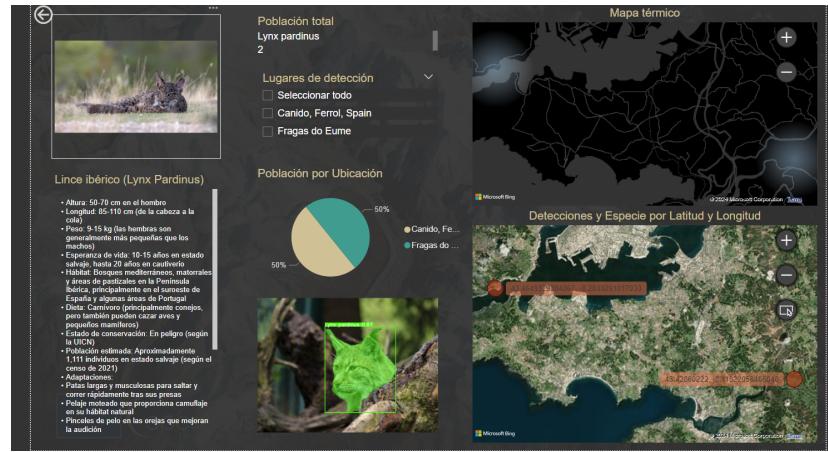


Figura 6.4: Informe Visual para el lince ibérico

## Capítulo 7

# Prueba de campo

---

**E**n este capítulo se detalla la puesta en escena de una prueba de campo para validar el sistema de detección de animales mediante cámaras trampa. A pesar de constituir un punto insatisfactorio del trabajo, debido a limitaciones presupuestarias en el hardware, se presenta cómo sería la realización de esta prueba y se exploran posibles alternativas para su ejecución.



Figura 7.1: Naturebytes wildlife technology kit

## 7.1 Microcontrolador

En el marco de este proyecto, se llevará a cabo la implementación del algoritmo YOLO (You Only Look Once) en un microcontrolador específico, en este caso, el Raspberry Pi 2 (Figura 7.2). El propósito de este dispositivo es la creación de una cámara trampa que consta de una estructura compuesta por una placa base, una cámara infrarroja capaz de grabar tanto de día como de noche, una batería independiente para asegurar la autonomía energética, y un contenedor tipo tupperware que servirá como carcasa protectora para la herramienta. La estructura se ha orientado en aplicaciones reales como naturebytes wildlife (Figura 7.1) que usan un kit con Raspberry Pi.



Figura 7.2: Raspberry Pi en la que se han ejecutado las pruebas

El objetivo principal de esta cámara trampa es realizar grabaciones de manera continua. Cuando el algoritmo YOLO detecte un objeto o sujeto de interés, se activará un proceso de captura y almacenamiento de información relevante. Este proceso incluirá la creación de un paquete de datos que contendrá un identificador de la cámara, información de geolocalización, la imagen segmentada que contiene la detección, la hora en que se realizó la detección, así como la clase de objeto o sujeto identificado.

Para facilitar la gestión y organización de los datos, se establecerá una conexión con una base de datos MongoDB. Esta base de datos almacenará de manera estructurada y accesible la información generada por la cámara trampa. De esta manera, se logrará mantener un registro detallado de las detecciones realizadas a lo largo del tiempo, permitiendo un análisis posterior de patrones de comportamiento, movimientos, y la identificación de sujetos específicos.

Componentes necesarios para el microcontrolador:

- Módulo de placa de expansión de energía o fuente de energía (Figura 7.3).



Figura 7.3: Fuente de energía

-Cámara de ojo de pez compatible (Figura 7.4).

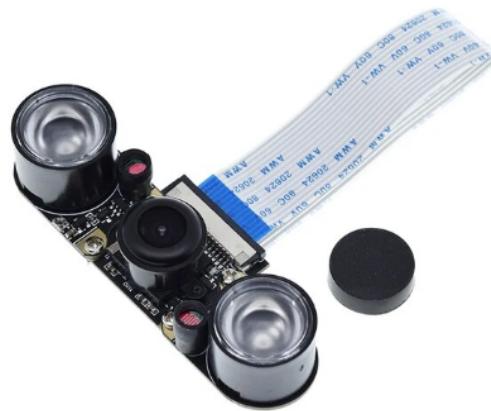


Figura 7.4: Cámara de 5MP

## 7.2 Ejecución en entorno real

Para la ejecución de campo, procedimos a configurar y preparar nuestro dispositivo de detección en tiempo real.

En primer lugar, será necesario instalar el sistema operativo adecuado en una tarjeta Secure Digital ([SD](#)). Se recomienda descargar el Raspberry Pi Imager desde el sitio web oficial y

seguir las instrucciones proporcionadas en <https://www.raspberrypi.org/blog/raspberry-pi-imager-imaging-utility/>. El sistema operativo, en este caso Raspbian [27], requerido estará disponible en el menú de selección de Raspberry Pi OS (other).

Una vez descargado, se debe insertar la tarjeta SD en la Raspberry Pi y conectarla a una fuente de alimentación para encenderla. Para la configuración inicial, se recomienda conectar un monitor y un teclado directamente a la Raspberry Pi. Aunque posteriormente podremos utilizar **Secure Shell (SSH)** para el manejo, en esta etapa inicial el acceso será directo.

Una vez encendida, espera hasta que se solicite el inicio de sesión con la línea "raspberrypi login:". Por defecto, el nombre de usuario es "pi" y la contraseña es "raspberry". Ingresa ambos para continuar.

Dado que estamos utilizando una versión ligera del sistema operativo, necesitaremos configurarla manualmente. Para ello, utilizaremos el comando "raspi-config". Es posible que debas ajustar la configuración del teclado en este punto.

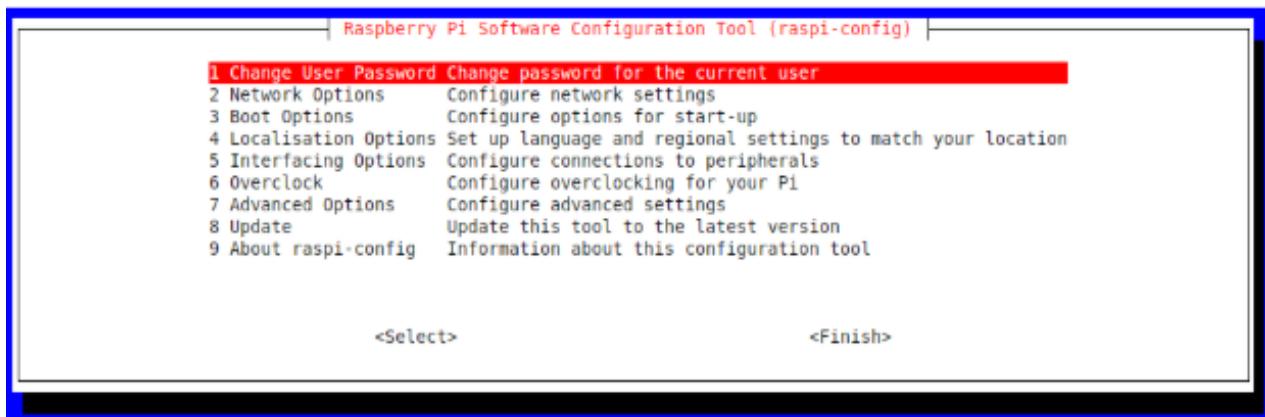


Figura 7.5: Menú de configuración

Después de configurar el teclado, procederemos a conectarnos a una red. Si es una red inalámbrica, selecciona la opción correspondiente y sigue las instrucciones para ingresar el nombre de la red (**Service Set Identifier (SSID)**) y la contraseña.

Una vez conectado a la red, activaremos la interfaz **SSH** para permitir la conexión remota a través de cualquier terminal. Esto se puede hacer desde el menú de "Interfacing Options" y seleccionando la opción **SSH**.

Finalmente, actualizamos los paquetes y las librerías utilizando el comando "sudo apt-get upgrade" y "sudo apt-get update" y instalamos un compilador para python para poder ejecutar nuestro .pt resultante del entrenamiento

Una vez completados estos pasos, la Raspberry Pi estaría lista para la utilización de Yolo, ejecutando con la cámara en vivo de la raspberry. Podemos desconectar el monitor y el teclado y acceder a la Raspberry Pi mediante **SSH** desde un programa compatible como **Windows**

Secure Copy (WinSCP) y Telnet and SSH client (PuTTY).

### 7.2.1 Consideraciones

Para transferir la información recopilada por la cámara trampa a la base de datos MongoDB, es esencial establecer algún medio de conexión. En entornos con acceso a Internet, una opción viable sería emplear un módulo Wi-Fi conectado al Raspberry Pi 2, permitiendo la transmisión directa de datos a través de la red. En situaciones donde no se disponga de Wi-Fi pero haya cobertura celular, se podría considerar la utilización de un módulo de comunicación celular ([Global System for Mobile Communications \(GSM\)](#), 3G, o 4G) para posibilitar la transmisión de datos a través de la red móvil.

Otra alternativa consistiría en almacenar localmente la información relevante en una tarjeta de memoria u otro dispositivo de almacenamiento en la cámara trampa. Posteriormente, cuando se encuentre disponible una conexión a Internet, los datos podrían ser transferidos a la base de datos MongoDB de manera automatizada o manual.

Por otro lado, encontramos que la raspberry pi, por lo menos en la versión que se ha utilizado, cuenta con un procesador que dificulta mucho la tarea y la [SD](#) no tiene almacenamiento suficiente para almacenar todas las funciones deseadas pero ante mayores recursos debería trabajar óptimamente.

Finalmente se llevó a cabo una prueba de funcionalidad del algoritmo bajo condiciones de recursos limitados, utilizando la Raspberry Pi como cámara trampa para la detección de elementos basandonos en una librería movimiento. Se consideró la conveniencia de incluir un sensor de calor para reducir las interferencias de objetos inertes en el proceso de detección.

En el procedimiento, se configuró la Raspberry Pi para capturar imágenes automáticamente cuando se detectara movimiento. Estas imágenes fueron posteriormente transferidas a un PC central donde se ejecutó el algoritmo de detección de objetos en cada una de ellas.

El proceso de detección automática permitió descartar aquellas imágenes que no cumplían con las premisas necesarias o no eran relevantes para el análisis. Las imágenes fueron clasificadas automáticamente según los objetos detectados.

## Capítulo 8

# Conclusiones

---

EN el presente proyecto, hemos logrado implementar YOLOv8 obteniendo resultados óptimos en la detección de varias especies consecutivas, con muy buenos resultados en imágenes y resultados decentes en videos. Este éxito subraya la capacidad de YOLOv8 para adaptarse a diferentes escenarios de detección de especies, destacando su precisión y versatilidad en la identificación de animales en diversas condiciones.

Además, hemos creado un nuevo dataset específico para el proyecto, lo que ha permitido entrenar el modelo de detección con datos relevantes y personalizados. Paralelamente, se ha desarrollado una base de datos que se conecta directamente con las detecciones, permitiendo el almacenamiento automático de los resultados. Esta base de datos no solo almacena las detecciones sino que también facilita la generación de informes dinámicos, ofreciendo ejemplos claros y prácticos de interpretación de datos.

Uno de los logros significativos del proyecto es la demostración de la viabilidad de utilizar una Raspberry Pi como cámara trampa. Este dispositivo, conocido por su bajo coste y versatilidad, se ha mostrado como una herramienta eficaz para la captura de imágenes y videos de especies exóticas, abriendo la puerta a aplicaciones de monitoreo ambiental más accesibles y distribuidas.

El estado actual del medio ambiente en el planeta, junto con los riesgos proyectados en términos de pérdida de biodiversidad y servicios ecosistémicos asociados, subraya la necesidad urgente de desarrollar e implementar nuevos métodos para acelerar la generación de información y conocimiento ambiental. Estos métodos deben contribuir a la implementación de planes de acción y políticas públicas para la conservación de la biodiversidad, basadas en información científica sólida. En este contexto, el uso de técnicas de Inteligencia Artificial, como la Visión Artificial y el Machine Learning, en tareas ambientales se presenta como una alternativa prometedora para optimizar los recursos destinados a la investigación en biodiversidad.

No obstante, es crucial reconocer que cada problema ambiental tiene un contexto, carac-

## CAPÍTULO 8. CONCLUSIONES

---

terísticas y desafíos propios. Por ello, se debe tener precaución al utilizar modelos de Inteligencia Artificial desarrollados en circunstancias y contextos diferentes al problema específico que se desea resolver. En este marco, la colaboración y el trabajo conjunto entre especialistas en Inteligencia Artificial y Medio Ambiente es esencial para generar modelos automáticos robustos. Estos modelos pueden incrementar el conocimiento ambiental sobre diversas especies de animales y plantas, mejorar su estado de conservación y prevenir impactos futuros.

En resumen, el proyecto no solo ha demostrado la eficacia técnica de las herramientas utilizadas, sino que también ha evidenciado el potencial de la colaboración interdisciplinaria para abordar los desafíos ambientales contemporáneos. La integración de YOLOv8, un dataset personalizado, una base de datos eficiente, y el uso de la Raspberry Pi como cámara trampa, representan avances significativos en el monitoreo y conservación de la biodiversidad, contribuyendo al conocimiento científico y la formulación de políticas de conservación más efectivas.

## Capítulo 9

# Trabajo Futuro

---

EN el transcurso de este estudio, se ha evidenciado que la optimización de algoritmos y técnicas para la detección, en muchos casos, no ha alcanzado los niveles deseados debido a la escasez de recursos. Sin embargo, se destaca que en entornos con mayores prestaciones, el rendimiento práctico experimentaría un incremento significativo. Es crucial considerar la posibilidad de ampliar el tamaño de los conjuntos de datos para diversas especies, aprovechando la información disponible en las bases de datos.

En relación a los componentes físicos, es relevante señalar que elementos como el Bluetooth en microcontroladores o dispositivos WiFi podrían presentar ineficiencias debido al entorno y la falta de infraestructura en los lugares de estudio, especialmente en hábitats naturales.

Para el futuro de este proyecto, se sugiere expandir su alcance mediante la incorporación de otros tipos de sensores, no limitándose únicamente al reconocimiento visual. Se pueden explorar técnicas de rastreo y análisis de datos, tomando como ejemplo los sensores utilizados por el Hawai'i Institute for Marine Biology en ballenas. Estos sensores, adheridos mediante ventosas, incluyen capacidades para medir temperatura del agua, velocidad (a través de acelerómetros), presión atmosférica, luminosidad, cámaras en tiempo real y GPS, proporcionando diversas métricas para el estudio de estos majestuosos animales, sus rutas migratorias y el estado ambiental del agua. Esta diversificación en la recopilación de datos puede enriquecer considerablemente la comprensión de los ecosistemas estudiados.

Además de la recopilación de estos datos bioacústicos y las cámaras trampa, se podrían usar imágenes satelitales como la que usa el Proyecto Guacamaya para preservar el Amazonas mediante Inteligencia Artificial o estudios realizados con LiDAR [28] descubrieron que esta misma selva escondía secretos que estaban vendados a los hombres. Como bien pensaba el explorador británico Percy Fawcett, que empezó su búsqueda de la legendaria ciudad perdida de Z en Bolivia a principios del siglo XX, se ha dado a la luz con esta tecnología que existe una gran civilización sepultada bajo la vegetación (Figura 9.1).



Figura 9.1: Boceto del cuaderno de bitácora de Percy Fawcet

La utilización en el ámbito marino de robots adaptados [29] ofrece un gran abanico de posibilidades con los relevantes modelos de tipo [Remotely Operated Vehicle \(ROV\)](#) (Remotely Operated Vehicles) (Figuras 9.2 y 9.3) mediante guía remoto, controlados y alimentados desde la superficie por un operador o piloto a través de un cordón umbilical o usando el control remoto. Existen vehículos con capacidad de control sobre sí mismos durante el cumplimiento de una tarea predefinida; estos son conocidos como [Autonomous Underwater Vehicles \(AUVs\)](#) (Autonomous Underwater Vehicle), los cuales tienen un grado de autonomía superior y no necesitan de guiaje remoto.



Figura 9.2: Piloto controlando en directo un ROV



Figura 9.3: Ejemplo de ROV

En última instancia, es imperativo resaltar la importancia de proyectos como este para la preservación de la diversidad de especies en la Tierra. Estas especies desempeñan un papel crucial en el equilibrio medioambiental, haciendo que nuestro planeta sea único. Es esencial recordar que la belleza se encuentra en la rareza y la singularidad de cada ser vivo, y proyectos de conservación como este son fundamentales para mantener esa diversidad y garantizar un futuro sostenible para las generaciones venideras. Los hombres nos vendamos los ojos a nosotros mismos pero, si aprendemos a mirar, quizás con la ayuda de todos podamos preservar la integridad de la flora y la fauna que es vital para el planeta que nos ha dado la posibilidad de vivir todas las aventuras, experiencias, progresos y recuerdos del ser humano.

En el futuro, una dirección prometedora de investigación sería ampliar y diversificar nuestro conjunto de datos mediante la incorporación de nuevas imágenes. Esto proporcionaría una base más sólida para el reentrenamiento de nuestros modelos, permitiendo que se enfrenten a una variedad más amplia de casos. Esta diversificación facilitaría que nuestras redes neuronales extrajeran características relevantes de manera más efectiva, lo que conduciría a una mejor capacidad de generalización y, en última instancia, a una mejora en los resultados obtenidos.

Además, sería útil desarrollar un programa de interfaz gráfica de usuario (GUI) que permitiera a usuarios no técnicos utilizar fácilmente nuestros modelos para clasificar imágenes. Esta interfaz intuitiva agilizaría el proceso y ampliaría la accesibilidad de nuestra solución. Esto podría tener un impacto significativo en la protección silvestre de los hábitats naturales y especies autóctonas.

CAPÍTULO 9. TRABAJO FUTURO

---



Figura 9.4

# **Apéndices**

## **Apéndice A**

# **Material adicional**

---

**R**ECURSOS visuales que han enriquecido y facilitado la comprensión del presente trabajo.

### **A.1 Reconocimiento recursos visuales**

1. Conservancy AI
2. De Jens Petersen - Trabajo propio, CC BY 2.5
3. Third Pole GeoLab
4. Unsplash
5. Pixabay
6. Snow Leopard Trust

## Apéndice B

# Códigos utilizados

---

**E**N este capítulo se presentan los códigos utilizados en el desarrollo del sistema.

### B.1 Detección sin MongoDB

```
1 # Importamos las librerias
2 from ultralytics import YOLO
3 import cv2
4
5 # Leer nuestro modelo
6 model = YOLO("bestlast.pt")
7
8 # Realizar VideoCaptura
9 #cap = cv2.VideoCapture(0)
10 cap = cv2.VideoCapture('snowleopardvideo2.mp4')
11
12 # Bucle
13 while True:
14     # Leer nuestros fotogramas
15     ret, frame = cap.read()
16
17     # Leemos resultados
18     resultados = model.predict(frame, imgsz = 320, conf = 0.7)
19
20     # Mostramos resultados
21     anotaciones = resultados[0].plot()
22
23     # Mostramos nuestros fotogramas
24     cv2.imshow("DETECCION Y SEGMENTACION", anotaciones)
25
26     # Cerrar nuestro programa
27     if cv2.waitKey(1) == 27:
```

```

28     break
29
30
31
32 cap.release()
33 cv2.destroyAllWindows()

```

## B.2 Detección con MongoDB

```

1 # Importamos las librerias
2 from ultralytics import YOLO
3 import cv2
4 from pymongo import MongoClient
5 import base64
6 from datetime import datetime, timedelta
7
8 # Conexión a MongoDB
9 client = MongoClient('mongodb://localhost:27017/')
10 db = client['CameraTrapDetection']
11 collection = db['Detections']
12
13 # Leer nuestro modelo
14 model = YOLO("bestlast.pt")
15
16 # Realizar VideoCaptura
17 #cap = cv2.VideoCapture(0)
18 cap = cv2.VideoCapture('snow3.mp4')
19
20 # Inicializar la hora de la última detección
21 ultima_deteccion = datetime.now()
22
23 # Bucle
24 while True:
25     # Leer nuestros fotogramas
26     ret, frame = cap.read()
27
28     # Leemos resultados
29     resultados = model.predict(frame, imgsz = 512, conf = 0.7)
30
31
32     # Mostramos resultados
33     anotaciones = resultados[0].plot()
34
35

```

```

36
37     # Mostramos nuestros fotogramas
38     cv2.imshow("DETECCION Y SEGMENTACION", anotaciones)
39
40     # Obtener la hora actual
41     ahora = datetime.now()
42
43     # Verificar si ha pasado un minuto desde la última detección
44     if ahora - ultima_deteccion > timedelta(minutes=1):
45         # Guardar la información en la base de datos MongoDB,
46         # incluyendo la hora de la detección
47         data = {
48             'especie': 'Phantera Uncia',
49             'ubicacion_camara': 'Canido,Ferrol,Spain', # Aquí
50             # deberías obtener la ubicación de la cámara
51             'imagen_segmentada':
52                 base64.b64encode(cv2.imencode('.jpg',
53                 frame)[1]).decode('utf-8'), # Convertir la imagen a base64
54             'hora_deteccion': ahora # Incluir la hora de la
55             # detección
56         }
57         collection.insert_one(data)
58
59     # Actualizar la hora de la última detección
60     ultima_deteccion = ahora
61
62     # Cerrar nuestro programa
63     if cv2.waitKey(1) == 27:
64         break
65
66 cap.release()
67 cv2.destroyAllWindows()

```

### B.3 Prueba para imágenes

```

1 from ultralytics import YOLO
2 import cv2
3
4 # Leer nuestro modelo
5 model = YOLO("best.pt")
6
7 # Ruta de la imagen
8 image_path = 'image.jpg'
9

```

```

10 # Leer la imagen
11 frame = cv2.imread(image_path)
12
13 # Leer resultados
14 resultados = model.predict(frame, imgsz=320, conf=0.5)
15
16 # Mostrar resultados
17 anotaciones = resultados[0].plot()
18
19 # Mostrar imagen con anotaciones
20 cv2.imshow("DETECCION Y SEGMENTACION", anotaciones)
21
22 # Esperar hasta que se presione una tecla para cerrar la ventana
23 cv2.waitKey(0)
24 cv2.destroyAllWindows()

```

## B.4 Paso de base64 a imagen

```

1 import base64
2 import numpy as np
3 import cv2
4 from pymongo import MongoClient
5
6 # Conexión a MongoDB
7 client = MongoClient('mongodb://localhost:27017/')
8 db = client['CameraTrapDetection']
9 collection = db['Detections']
10
11 # Recuperar un documento de la base de datos
12 documento = collection.find_one()
13
14 # Decodificar la imagen segmentada de base64 a formato de imagen
15 imagen_segmentada_base64 = documento['imagen_segmentada']
16 imagen_segmentada_bytes = base64.b64decode(imagen_segmentada_base64)
17 imagen_segmentada_np = np.frombuffer(imagen_segmentada_bytes,
18                                     np.uint8)
18 imagen_segmentada = cv2.imdecode(imagen_segmentada_np,
19                                     cv2.IMREAD_COLOR)
20
21 # Mostrar la imagen
22 cv2.imshow('Imagen Segmentada', imagen_segmentada)
23 cv2.waitKey(0)
23 cv2.destroyAllWindows()

```

## B.5 Rutas para el dataset

```

1 train: Data/train/
2 val: Data/val/
3 test:
4 nc: 7
5 names: ['Hapalochlaena maculosa', 'Ailuropoda melanoleuca',
       'Panthera tigris tigris', 'Ara Ararauna', 'Lynx pardinus', 'Waxy
       Mokey', 'Phantera Uncia']

```

## B.6 Algoritmo en imágenes de un directorio

```

1 from ultralytics import YOLO
2 import cv2
3 import os
4
5 # Función para realizar la detección y segmentación en una imagen y
6 # guardar el resultado
7 def detect_and_segment(model, image_path, output_path):
8     # Leer la imagen
9     frame = cv2.imread(image_path)
10
11     # Realizar la detección y segmentación
12     resultados = model.predict(frame, imgsz=320, conf=0.5)
13
14     # Obtener la ruta de salida para la imagen segmentada
15     output_image_path = os.path.join(output_path,
16                                     os.path.basename(image_path))
17
18     # Guardar la imagen segmentada
19     resultados[0].save(output_image_path)
20
21     print(f"Imagen segmentada guardada en: {output_image_path}")
22
23 # Ruta del directorio de entrada y salida
24 input_dir = 'pruebas_entrada'
25 output_dir = 'pruebas_salida'
26
27 # Crear la carpeta de salida si no existe
28 os.makedirs(output_dir, exist_ok=True)
29
30 # Leer nuestro modelo
31 model = YOLO("bestlast.pt")

```

```

31 # Iterar sobre todas las imágenes en el directorio de entrada
32 for filename in os.listdir(input_dir):
33     # Comprobar si es una imagen
34     if filename.endswith('.jpg', '.jpeg', '.png', '.bmp')):
35         # Obtener la ruta completa de la imagen de entrada
36         image_path = os.path.join(input_dir, filename)
37
38         # Realizar la detección y segmentación y guardar el
39         # resultado
40         detect_and_segment(model, image_path, output_dir)

```

## B.7 Código final

```

1 from ultralytics import YOLO
2 import cv2
3 import os
4 from pymongo import MongoClient
5 from datetime import datetime
6
7 # Función para conectar a MongoDB
8 def connect_to_mongo(uri, db_name):
9     try:
10         client = MongoClient(uri)
11         db = client[db_name]
12         return db
13     except Exception as e:
14         print(f"Error al conectar a MongoDB: {e}")
15         return None
16
17 # Conexión a MongoDB
18 db = connect_to_mongo('mongodb+srv://xabi:9620@cluster0-detections.
19 mcxpoys.mongodb.net/', 'CameraTrapDetection')
20 if db is None:
21     raise Exception("No se pudo conectar a la base de datos")
22
23 collection = db['Detections']
24
25 # Función para realizar la detección y segmentación en una imagen y
26 # guardar el resultado
27 def detect_and_segment(model, image_path, output_path, lat, lon,
28                         ubicacion):
29     # Leer la imagen
30     frame = cv2.imread(image_path)

```

```

30
31     # Realizar la detección y segmentación
32     resultados = model.predict(frame, imgsz=320, conf=0.5)
33
34     # Obtener las especies detectadas
35     especies_detectadas = [model.names[int(cls)] for cls in
36     resultados[0].boxes.cls]
37
38     # Solo guardar en la base de datos si se detecta al menos una
39     # especie
40     if especies_detectadas:
41         especie = especies_detectadas[0] # Usar la primera especie
42         detectada
43         cantidad_individuos = len(especies_detectadas) # Obtener
44         la cantidad de individuos detectados
45
46         # Obtener el identificador único
47         identificador =
48             str(datetime.now().timestamp()).replace('.', '_')
49
50         # Guardar la imagen segmentada con el nombre de especie y
51         # el identificador único
52         output_image_name = f"{especie}_{identificador}.jpg"
53         output_image_path = os.path.join(output_path,
54         output_image_name)
55         resultados[0].save(output_image_path) # Guardar la imagen
56         segmentada
57         print(f"Imagen segmentada guardada en: {output_image_path}")
58
59         # Guardar la información en la base de datos MongoDB
60         data = {
61             'latitud': lat,
62             'longitud': lon,
63             'especie': especie,
64             'ubicacion': ubicacion,
65             'cantidad_individuos': cantidad_individuos, # Guardar
66             la cantidad de individuos detectados
67             'imagen_segmentada': output_image_name, # Guardar solo
68             el nombre de la imagen en el directorio de salida
69             'hora_deteccion': datetime.now() # Incluir la hora de
70             la detección
71         }
72         collection.insert_one(data)
73
74         print(f"Detección guardada en MongoDB: {data}")
75     else:

```

```
65     print(f"No se detectó ninguna especie en: {image_path}")
66
67 # Ruta del directorio de entrada y salida
68 input_dir = 'pruebasE'
69 output_dir = 'pruebasS'
70
71 # Crear la carpeta de salida si no existe
72 os.makedirs(output_dir, exist_ok=True)
73
74 # Leer nuestro modelo
75 model = YOLO("bestlast.pt")
76
77 # Datos adicionales para las detecciones
78 lat = 43.504402651505 # Ejemplo de latitud
79 lon = -8.3179750611532 # Ejemplo de longitud
80 ubicacion = 'Pinar de Doniños,Ferrol,España'
81
82 # Iterar sobre todas las imágenes en el directorio de entrada
83 for filename in os.listdir(input_dir):
84     # Comprobar si es una imagen
85     if filename.endswith('.jpg', '.jpeg', '.png', '.bmp')):
86         # Obtener la ruta completa de la imagen de entrada
87         image_path = os.path.join(input_dir, filename)
88
89         # Realizar la detección y segmentación y guardar el
90         # resultado
91         detect_and_segment(model, image_path, output_dir, lat, lon,
92         ubicacion)
```

## Apéndice C

# Otros

---

### C.1 Presupuesto

PARA finalizar, se incluye el presupuesto que supone la implantación del sistema desarrollado en un emplazamiento real. Este contiene todos los gastos, incluyendo el software y hardware, sin incluir el coste de la mano de obra necesario para su desarrollo:

Ítem	Precio
Tarjeta SD (Mínimo 32 GB)	3,50€
Módulo cámara	8,20€
Raspberry pi 4 Model B (modelo a escoger)	47,22€
Módulo de Batería	4,25€
Carcasa de la cámara de vigilancia	5€
Licencia Pycharm gratuita	-
Licencia Latex gratuita	-
Ordenador Personal	-
Google Collab GPU 100 iteraciones	12€

Tabla C.1: Presupuesto estimado Total=80.17€

### C.2 Histórico de pruebas de entrenamiento

LA tabla a continuación proporciona una visión general de las diferentes configuraciones y resultados obtenidos durante este proceso de experimentación. Cada fila de la tabla

describe las variaciones en los hiperparámetros del modelo, los resultados observados y las notas relevantes sobre cada iteración.

### **C.3 Datasets de acceso gratuito**

**L**A tabla a continuación proporciona un listado de bases de datos gratuitas de las que adquirir datos para modelos de próximas investigaciones.

Hiperparámetros modificados	Resultado	Observaciones
YOLOv8m, épocas=10, batch=2, imágenes=30	Segmentation Fault	Configuración estándar
YOLOv8m, épocas=10, batch=2, imágenes=90	Satisfactorio	No distingue efectivamente
Dataset Roboflow	Fallida	PC colapsa
-	Satisfactorio	Malos resultados
170 imágenes, activamos CUDA, épocas=60	Satisfactorio	Disminución en la detección de falsos positivos pero menor porcentaje de acierto
Imágenes=200, épocas=80	Cancelada	Ejecución sin éxito
Épocas=60	Cancelada	Calentamiento excesivo
YOLOv8x, imágenes con nuevos sujetos, batch=4	Satisfactorio	Mejor rendimiento
Más de 370 imágenes	Segmentation Fault	-
-	Segmentation Fault	-
-	Satisfactorio	Arregladas dependencias de ultralytics
Épocas=100, batch=8	Satisfactorio	Errores concretos pero mejores resultados en precisión y recall
Integración de MongoDB	Segmentation Fault	-
-	Satisfactorio	No almacena imágenes
Conversión de imágenes a base64	Satisfactorio	-
Ejecución sobre directorios en imágenes (no vídeo)	Satisfactorio	Resultados óptimos
Ejecuciones sincronizando Power BI	Satisfactorio	Resultados óptimos

Tabla C.2: Histórico de entrenamientos

Nombre de la base de datos	Tipo de datos	URL
Plant Trait Database	Registros de especies de plantas	<a href="https://www.try-db.org">https://www.try-db.org</a>
Snapshot Serengeti	Imágenes de cámaras trampa	<a href="http://lila.science/datasets/snapshot-serengeti">http://lila.science/datasets/snapshot-serengeti</a>
WCS Camera Trap	Imágenes de cámaras trampa	<a href="http://lila.science/datasets/wcscameratraps">http://lila.science/datasets/wcscameratraps</a>
North American Camera Trap Images	Imágenes de cámaras trampa	<a href="http://lila.science/datasets/nacti">http://lila.science/datasets/nacti</a>
Caltech Camera Traps	Imágenes de cámaras trampa	<a href="http://lila.science/datasets/nacti">http://lila.science/datasets/nacti</a>
HKH Glacier Mapping	Imágenes satelitales de glaciares	<a href="https://wp.nyu.edu/birdvox/codedata/#datasets">https://wp.nyu.edu/birdvox/codedata/#datasets</a>
BirdVox	Registro de audios de aves	<a href="http://www.vision.caltech.edu/registree/publications-and-database.html">http://www.vision.caltech.edu/registree/publications-and-database.html</a>
Pasadena Urban Trees	Imágenes de árboles etiquetadas	<a href="https://www.firebaseio.in">https://www.firebaseio.in</a>
NABirds	Imágenes de aves etiquetadas	<a href="https://d1.allaboutbirds.org/nabirds">https://d1.allaboutbirds.org/nabirds</a>
Stanford Dogs	Imágenes de razas de perros	<a href="https://www.robots.ox.ac.uk/~vgg/data/flowers/">https://www.robots.ox.ac.uk/~vgg/data/flowers/</a>
Animals with attributes	Imágenes de animales con atributos	<a href="https://cvml.ist.ac.at/AwA2/">https://cvml.ist.ac.at/AwA2/</a>

Tabla C.3: Bases de datos de libre acceso con información ambiental

# **Lista de acrónimos**

---

**AUVs** Autonomous Underwater Vehicles. [69](#)

**BSON** Binary JSON. [20](#), [55](#)

**CNN** Convolutional Neural Network. [12](#), [13](#), [33](#)

**CPU** Central Processing Unit. [36](#), [43](#)

**CUDA** Compute Unified Device Architecture. [37](#), [43](#)

**Dataset** Data Set. [48](#)

**Frames** Data Frames. [44](#)

**GPU** Graphics Processing Unit. [22](#), [42](#), [43](#)

**GSM** Global System for Mobile Communications. [65](#)

**IDE** Integrated Development Environment. [18](#)

**JSON** JavaScript Object Notation. [19](#), [20](#), [30](#), [55](#)

**NoSQL** Not Only SQL. [55](#), [56](#)

**PuTTY** Telnet and SSH client. [65](#)

**ROV** Remotely Operated Vehicle. [69](#)

**SD** Secure Digital. [63](#)–[65](#)

**SIFT** Scale-Invariant Feature Transform. [13](#)

**SQL** Structured Query Language. [21](#), [56](#)

**SSH** Secure Shell. [64](#)

**SSID** Service Set Identifier. [64](#)

**SURF** Speeded Up Robust Features. [13](#)

**SVM** Support Vector Machine. [13](#)

**TPU** Tensor Processing Unit. [22](#)

**WinSCP** Windows Secure Copy. [64](#)

# Bibliografía

---

- [1] M. Richardson and S. Wallace, *Getting started with raspberry PI.* ” O'Reilly Media, Inc.”, 2012.
- [2] C. Brito Martínez *et al.*, “Detección y clasificación automática de animales silvestres mediante visión artificial y machine learning.” 2022.
- [3] A. J. González and O. a Objeto, “Ingeniería de software: Metodologías,” *Visitado el*, vol. 8, 2007.
- [4] Á. Artola Moreno, “Clasificación de imágenes usando redes neuronales convolucionales en python,” 2019.
- [5] S. Brahmbhatt, *Practical OpenCV*. Apress, 2013.
- [6] T. Diwan, G. Anirudh, and J. V. Tembhurne, “Object detection using yolo: Challenges, architectural successors, datasets and applications,” *multimedia Tools and Applications*, vol. 82, no. 6, pp. 9243–9275, 2023.
- [7] G. Jocher, A. Chaurasia, and J. Qiu, “Ultralytics yolov8,” 2023. [En línea]. Disponible en: <https://github.com/ultralytics/ultralytics>
- [8] E. L. Aldea, *Raspberry PI fundamentos y aplicaciones*. Ra-Ma Editorial, 2017.
- [9] S. C. Móviles, “Modelo de consumo de energía de raspberry pi 4 b con escalamiento de frecuencias,” Ph.D. dissertation, INSTITUTO POLITECNICO NACIONAL, 2021.
- [10] JetBrains, *PyCharm Documentation*, 2023. [En línea]. Disponible en: <https://www.jetbrains.com/pycharm/documentation/>
- [11] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman, “Labelme: a database and web-based tool for image annotation,” in *International Journal of Computer Vision (IJCV)*, vol. 77, no. 1-3, 2008, pp. 157–173. [En línea]. Disponible en: [https://www.labelme.csail.mit.edu/Release3.0/browserTools/php/matlab\\_toolbox.php](https://www.labelme.csail.mit.edu/Release3.0/browserTools/php/matlab_toolbox.php)

- [12] D. Crockford. (2002) Introducing json. [En línea]. Disponible en: <https://json.org/>
- [13] M. Inc. (2007) Mongodb. [En línea]. Disponible en: <https://www.mongodb.com/>
- [14] Microsoft, “Power bi,” 2024, Último acceso: 30 de mayo de 2024. [En línea]. Disponible en: <https://powerbi.microsoft.com/>
- [15] Ultralytics, “Ultralytics,” 2024, Último acceso: 30 de mayo de 2024. [En línea]. Disponible en: <https://ultralytics.com/>
- [16] Google Research, “Google colab,” 2024, Último acceso: 30 de mayo de 2024. [En línea]. Disponible en: <https://colab.research.google.com/>
- [17] J. R. Khatiwada, M. K. Chalise, and R. C. Kyes, “Survey of snow leopard (*uncia uncia*) and blue sheep (*pseudois nayaur*) populations in the kangchenjunga conservation area (kca), nepal,” *Final Project Report Submitted to International Snow Leopard Trust, Seattle, USA, 13pp*, 2007.
- [18] S. L. Network, “Snow leopard survival strategy,” *Seattle, Washington, USA*, pp. 1–145, 2014.
- [19] P. Morse, S. R. Kjeldsen, M. G. Meekan, M. I. McCormick, J. K. Finn, C. L. Huffard, and K. R. Zenger, “Genome-wide comparisons reveal a clinal species pattern within a holobenthic octopod—the australian southern blue-ringed octopus, *hapalochlaena maculosa* (cephalopoda: Octopodidae),” *Ecology and Evolution*, vol. 8, no. 4, pp. 2253–2267, 2018.
- [20] E. M. BODERO, M. P. LOPEZ, A. E. CONGACHA, E. E. CAJAMARCA, and C. H. MORALES, “Google colaboratory como alternativa para el procesamiento de una red neuronal convolucional,” *Revista Espacios*, vol. 41, no. 07, 2020.
- [21] J. A. Stuart, M. Cox, and J. D. Owens, “Gpu-to-cpu callbacks,” in *Euro-Par 2010 Parallel Processing Workshops: HeteroPar, HPCC, HiBB, CoreGrid, UCHPC, HPCF, PROPER, CCPI, VHPC, Ischia, Italy, August 31–September 3, 2010, Revised Selected Papers 16*. Springer, 2011, pp. 365–372.
- [22] C. Goutte and E. Gaussier, “A probabilistic interpretation of precision, recall and f-score, with implication for evaluation,” in *European conference on information retrieval*. Springer, 2005, pp. 345–359.
- [23] J. Davis and M. Goadrich, “The relationship between precision-recall and roc curves,” in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 233–240.
- [24] I. Despujol Zabala, “Power bi. solucionar problemas ubicaciones en mapas,” 2023.

- [25] A. López Hernández *et al.*, “Presentación de datos odata en power bi,” 2017.
- [26] M. Jundzill, R. Spott, M. Lohde, M. Hölzer, A. Viehweger, and C. Brandt, “Create a mon-godb atlas cluster,” 2023.
- [27] W. Harrington, *Learning raspbian*. Packt Publishing Ltd, 2015.
- [28] X. Liu, “Airborne lidar for dem generation: some critical issues,” *Progress in physical geography*, vol. 32, no. 1, pp. 31–49, 2008.
- [29] R. D. Christ and R. L. Wernli Sr, *The ROV manual: a user guide for observation class remo-tely operated vehicles*. Elsevier, 2011.