Universidad ORT Uruguay

Facultad de Ingeniería

SCG: Sistema de Conteo de Ganado

Entregado como requisito para la obtención del título de Ingeniero en Electrónica

Manuel Navarrete - 225379

Ramiro Sosa – 243473

Tutor: Ismael Garrido

2024

Declaración de autoría

Nosotros, Manuel Navarrete y Ramiro Sosa, declaramos que el trabajo que se presenta en esa obra es de nuestra propia mano. Podemos asegurar que:

• La obra fue producida en su totalidad mientras realizábamos nuestro Proyecto de Tesis de grado;

• Cuando hemos consultado el trabajo publicado por otros, lo hemos atribuido con claridad;

• Cuando hemos citado obras de otros, hemos indicado las fuentes. Con excepción de estas citas, la obra es enteramente nuestra;

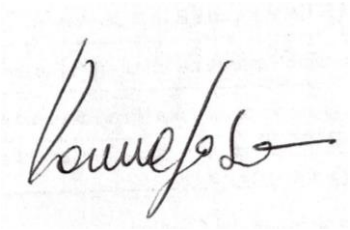
• En la obra, hemos acusado recibo de las ayudas recibidas;

• Cuando la obra se basa en trabajo realizado conjuntamente con otros, hemos explicado claramente qué fue contribuído por otros, y qué fue contribuído por nosotros;

• Ninguna parte de este trabajo ha sido publicada previamente a su entrega, excepto donde se han realizado las aclaraciones correspondientes.

Manuel Navarrete

19 de Septiembre de 2024



Ramiro Sosa

19 de Septiembre de 2024

Abstract

Este documento presenta el desarrollo de un proyecto llamado “Sistema de Conteo de Ganado”, un sistema automatizado para el conteo de ganado en establecimientos rurales mediante el uso de drones y tecnologías de *machine learning*. La motivación surge de la necesidad de mejorar los procesos de gestión en la industria ganadera, donde los métodos tradicionales de conteo resultan ineficientes, costosos y propensos a errores. El sistema propuesto tiene como objetivo ofrecer una solución más rápida, precisa y escalable para el sector agropecuario.

El sistema se basa en el uso de drones equipados con cámaras de alta resolución que realizan recorridos sobre el área de los establecimientos ganaderos, capturando videos del ganado. Estos videos son procesados por modelos de *machine learning*, entrenados específicamente para la detección y reconocimiento de vacas, logrando un conteo automatizado. Los datos generados se almacenan y gestionan en una plataforma web desarrollada con tecnologías como Django, PostgreSQL y Celery, donde los usuarios pueden visualizar información detallada sobre su ganado y revisar el histórico de conteos en sus diferentes lotes y pasturas.

Se evalúa el rendimiento del sistema a través de pruebas en los establecimientos rurales, asegurando su precisión y confiabilidad en diversas condiciones ambientales. Además, se presenta un análisis de las fases de desarrollo, los requerimientos técnicos y los desafíos superados en la implementación.

Palabras clave

Machine Learning, Modelos de detección, Uso de tecnología en el agro, Uso de tecnología en el campo, Reconocimiento de objetos, Detección de instancias, Ganadería de precisión,Conteo de objetos, Aplicaciones Web, *Tracking*, Conteo de ganado automatizado, Django, Celery, Redis, AWS, Docker, Drones, Hugging Face, Yolov8, Detectron 2, DETR, Botsort, ByteTrack.

Índice

1. Introducción

1.1 Motivación

La motivación de este proyecto surge de una problemática real en la industria ganadera de nuestro país, como lo es la ganadería de precisión, actividad en la que realiza su enfoque en el engorde de vacas. La motivación consiste en poder generar datos de valor para que los productores ganaderos puedan tomar decisiones informadas en cuanto a la gestión de su ganado.

Con el objetivo de solucionar estos problemas en la industria nace Ganader IA, un emprendimiento que se centra en ser el cambio en la ganadería y poder brindarles a los productores una manera efectiva, simple y económica de gestionar su ganado y mejorar sus procesos a través de tecnologías innovadoras. Este emprendimiento fue fundado en 2023 siendo los co-fundadores los dos integrantes de este proyecto: Manuel Navarrete y Ramiro Sosa. Otro integrante es Luis Porras, estudiante también de la Universidad ORT Uruguay.

Actualmente incubado en el CIE (Centro de Innovación y Emprendedurismo), este proyecto se encuentra en etapas iniciales de desarrollo por lo cual los tres decidimos que podía ser una buena idea realizar parte del desarrollo del producto cómo proyecto de tesis contando con la experiencia y el apoyo de los profesionales de ORT.

El producto que esta empresa entregaría es el siguiente:

Una plataforma web en la cual los productores del campo pueden visualizar todos los datos sobre su ganado, principalmente la evolución en el peso del animal, su estado de salud, estado de vacunación, etc. Además de esto poder proyectar el progreso del aumento del peso de su ganado a futuro, recibir recomendaciones sobre cuándo vender el mismo obteniendo las últimas actualizaciones sobre el precio del kilo de carne del ganado.

¿Cómo se haría esto?

Se desarrollaría una aplicación que despliegue todos estos datos, pero estos datos serán completados a medida de que se necesite a través de un dron. Este dron realizará recorridas periódicas para recopilar información a la frecuencia se le indique a través de la web.

Por lo cual la idea principal de este emprendimiento es el desarrollo de un dron que sea capaz de reconocer ganado a través de imágenes. Para así poder luego aproximar el peso de esa cabeza de ganado (utilizando *machine learning*) y así ir completando los datos en una base de datos que se despliega en el *frontend* descrito anteriormente. De esta manera el productor y sus peones se desligan de las tareas de conteo y pesaje del ganado y se evitan estresar a las vacas con estas intervenciones, lo que se traduce en una menor pérdida de kilos de carne.

Como primer aproximamiento al objetivo principal de Ganader IA surge el centro de este proyecto de tesis, que consiste en desarrollar un producto capaz de facilitar el conteo del ganado, tarea que a día de hoy es realizada por una persona que recorre toda la superficie del establecimiento rural. De esta manera, el equipo comenzaría a trabajar con drones y *machine learning* y podría tener un punto base del cual partir hacia el producto final.

1.2 Objetivos del proyecto

Se plantean tanto objetivos académicos como prácticos, enfocándose en la creación de una solución funcional y accesible para el sector agropecuario. El producto final deberá ser manejable por personas que trabajan en establecimientos rurales, sin requerir conocimientos avanzados de tecnología.

Objetivo general

Diseñar y desarrollar un sistema automatizado para el conteo de ganado mediante el uso de drones equipados con cámaras y tecnología de reconocimiento de imágenes. El sistema deberá ser fácil de utilizar, preciso en sus resultados y aplicable a escenarios reales en establecimientos rurales.

Objetivos específicos

- Desarrollar un sistema capaz de reconocer vacas y realizar el conteo automático del ganado en un establecimiento rural, utilizando tecnologías avanzadas como *machine learning* y drones.
- Aprender y aplicar técnicas de *machine learning* y entrenamiento de modelos, enfocadas en el reconocimiento de objetos (vacas) en imágenes y videos capturados por drones.
- Evaluar la calidad de los modelos de *machine learning* desarrollados, aplicando métricas y validaciones que aseguren la precisión y eficacia del sistema.
- Desarrollar habilidades para el despliegue de aplicaciones web, integrando la plataforma donde se presentarán los resultados del conteo de ganado y otras funcionalidades clave para el usuario final.
- Solucionar un problema real en el sector ganadero uruguayo, optimizando el proceso de conteo de ganado con una tecnología innovadora, reduciendo el tiempo y los costos operativos.
- Diseñar una solución viable tanto técnica como económicamente, considerando la accesibilidad y la rentabilidad del sistema para los productores rurales, asegurando que el producto sea escalable y sostenible en el largo plazo.
- Desarrollar competencias en el uso y manejo de drones para la captura de imágenes en áreas rurales

Este proyecto no solo busca desarrollar una solución tecnológica innovadora para el conteo automatizado de ganado, sino también generar un impacto significativo en el sector agropecuario del país. A través del uso de drones y *machine learning*, se plantea ofrecer una herramienta eficiente y accesible, que optimice los procesos actuales en los establecimientos rurales, ahorrando tiempo y reduciendo costos. Además, el equipo se enfrenta a nuevos desafíos técnicos y académicos, adquiriendo conocimientos clave en áreas como el desarrollo de productos tecnológicos, la evaluación de modelos de *machine learning*, y el despliegue de aplicaciones web, asegurando que la solución propuesta sea válida no sólo en términos técnicos, sino también económica y operativamente en un entorno real.

1.3 Estructura del documento

Este capítulo explica en qué consiste cada parte del documento.

1. Introducción

Se presenta el contexto general del proyecto, describiendo el área de estudio, la importancia del trabajo, y ofreciendo una visión general de los temas que se abordarán a lo largo del documento.

2. Problema y solución

Este capítulo analiza en detalle el problema principal que el proyecto busca resolver y propone una solución basada en el uso de drones y *machine learning*. Se justifica la elección de esta solución y se define el público objetivo.

3. Requerimientos

Se detallan los requisitos necesarios para el desarrollo del proyecto, abarcando tanto los requerimientos funcionales y no funcionales, como las futuras integraciones, los comerciales, y los relacionados al *testing*.

4. Investigación de drones

Exploramos las características técnicas necesarias para seleccionar el dron adecuado, presentando un análisis comparativo de diferentes modelos de drones y placas programables, y un estudio sobre la cobertura de superficie según la altura del dron.

5. Aplicación web

Explica el desarrollo de la aplicación web, abordando la selección de tecnologías, la arquitectura del sistema y el flujo de trabajo. También se describen las etapas del desarrollo, desde la creación inicial hasta la implementación de funcionalidades clave.

6. Desarrollo modelo de machine learning

Se detalla el proceso de desarrollo del modelo de *machine learning*, desde la obtención de los datos hasta la evaluación de los resultados. Se analizan los aspectos teóricos, se seleccionan los modelos, y se estudian técnicas de entrenamiento y optimización.

7. Conclusiones

Resume los resultados alcanzados a lo largo del proyecto, destacando los hallazgos más importantes y las implicaciones del trabajo realizado. También se plantean posibles futuros trabajos o mejoras.

8. Referencias bibliográficas

Incluye todas las fuentes y referencias utilizadas en la investigación y desarrollo del proyecto, permitiendo rastrear los conceptos y datos empleados.

9. ANEXOS

Proporciona información adicional y documentación complementaria relevante para el desarrollo del proyecto, visitas al campo, entrevistas y casos de uso.

2. Problema y solución

Es importante detallar el problema y el contexto para buscar una solución que cumpla con las necesidades de los productores rurales del país. En este capítulo se detalla sobre la problemática y qué se va a desarrollar para solucionar esto, junto al público objetivo al que se apunta, los beneficios de la solución, el alcance de la misma y su escalabilidad.

2.1 Problema y contexto

Realizando una serie de investigaciones, con personas que trabajan y viven del campo, se conocieron de mejor manera los procesos que realizan periódicamente.

Entre los procesos mencionados, nos explican las alternativas que tienen para realizar conteos de ganado y la periodicidad en la que lo hacen. Los productores realizan conteos regularmente debido a que usualmente suelen escaparse hacia un establecimiento vecino o simplemente por seguridad. También manifiestan que el ministerio o ciertos bancos exigen auditorías de conteo cada cierto tiempo.

La primera manera mencionada para realizar conteos, es la que se ha usado durante mucho tiempo. Esta metodología consiste en recorrer todo el campo realizando un rodeo, y al completar la vuelta sobre el total de la superficie, contar manualmente todos los animales que hay. En esta metodología, no se usa nada de tecnología y se necesitan muchas horas de trabajo. Al ser realizada manualmente, esta cuenta puede estar sometida a errores humanos muy variables.

El segundo método que regularmente utilizan para contar, es realizar el rodeo de todos los animales del campo y llevarlos hacia la manga. La manga es el lugar donde los establecimientos suelen tener su balanza. Es un pasillo muy angosto donde los animales van pasando de a uno. El trabajo que realizan para contar en la manga, es utilizando un bastón que lee los tag RFID que tienen los animales en su caravana brindada por el Ministerio de Ganadería, Agricultura y Pesca.

Cabe mencionar que el conteo en la manga utilizando tecnología RFID, que lee las etiquetas electrónicas de cada animal, no genera conteos duplicados, ya que cada identificación está vinculada a un código único del animal.

Sin embargo, demanda muchas horas de trabajo al necesitar trasladar todo el ganado y tener que realizar lecturas en cada animal, lo cual lo hace un método de baja escalabilidad y altos costos.

Se requiere una solución que no implique el movimiento de todos los animales, y no requiere de una persona contándolas, ya que se puede introducir error en la cuenta.

2.2 Solución

Creemos firmemente que el uso de tecnología avanzada podría generar un fuerte impacto en la eficiencia en la que se manejan los recursos y optimizar los procesos.

Esto puede llegar a tener una relevancia muy considerable si tenemos en cuenta el contexto en el que planteamos esto. En un país que mayoritariamente se dedica a la ganadería, optimizar estos procesos puede ser importante para aumentar la productividad.

Para facilitar la gestión de ganado y dar una mayor eficacia en los conteos que se realizan regularmente en el campo, se ideó un proyecto el cual puede facilitar y solucionar la problemática explicada anteriormente.

Se plantea como solución utilizar un dron con una cámara integrada, el cual mediante reconocimiento en imágenes pueda identificar ganado. En este proyecto, se estudian herramientas de reconocimiento de animales y tecnologías que sirvan para solucionar la problemática mencionada.

A grandes rasgos, este proyecto puede resumirse en un dron que recorre un campo con un área previamente indicada, reconociendo en su camino la cantidad de vacas que se encuentran dentro del establecimiento. Se muestra una imagen tomada desde el dron, posterior al procesamiento (ver imagen 1).

Imágen 1 - Reconocimiento de ganado en una imagen.

De esta manera, simplemente se cuenta la cantidad de ganado reconocido en todo el transcurso del video y se logra saber cuánto ganado hay.

Todo esto es posible de realizar si combinamos las siguientes tres tecnologías.

La primera y fundamental es el uso de drones para realizar una recorrida por todo el campo mientras se hacen grabaciones de los animales. Estas grabaciones son la entrada a la segunda tecnología, que es probablemente la más relevante del proyecto, y es el uso de *machine learning*. Al utilizar *machine learning*, se puede detectar y reconocer objetos, que en este caso son vacas. Reconociendo las vacas, podemos identificar en un video cuántas aparecen.

Finalmente, la última tecnología es una aplicación web. La importancia de esta se debe a que es el lugar donde se centraliza toda la información, y será lo único que ve el usuario. Allí encuentra la información de su establecimiento, historiales de los conteos realizados, con la posibilidad de ver los videos ya procesados, y ciertas estadísticas que le permiten gestionar de mejor manera su campo. De esta manera, se evita que una persona tenga que recorrer el campo, haciéndolo con un dron el cual lo hace de manera más rápida. Y lo más importante, no se tiene que mover el ganado.

2.3 Público objetivo

El público objetivo de este proyecto está compuesto por productores ganaderos, tanto pequeños como medianos, que tienen la necesidad de realizar conteos periódicos de su ganado para fines de control, auditoría y seguridad. Estos productores trabajan en establecimientos rurales alrededor de todo el país (Uruguay).

Principalmente, el público objetivo son los propietarios y trabajadores de establecimientos ganaderos que, como se menciona en el análisis de contexto, actualmente dependen de métodos manuales para contar el ganado. Dichos métodos, como los rodeos tradicionales y el uso de bastones con tecnología RFID, aunque efectivos, son altamente demandantes en términos de tiempo.

Estos productores buscan soluciones que les permitan mejorar la eficiencia en sus procesos de control de ganado, reduciendo el tiempo y los costos asociados, sin perder productividad. Además, la tecnología podría facilitarles el cumplimiento de auditorías exigidas por organismos gubernamentales o instituciones financieras, como el Ministerio de Ganadería, Agricultura y Pesca, o bancos que financian sus actividades.

Este grupo de usuarios puede tener conocimientos limitados en el uso de tecnologías avanzadas como drones o *machine learning*, por lo que la solución debe ser accesible, de fácil uso y con una interfaz simple. Los productores necesitan una herramienta que se integre en sus rutinas diarias sin generar una carga adicional de trabajo o complicaciones técnicas. Por ello, la solución propuesta, con una plataforma web que centraliza la información de manera intuitiva y un sistema de conteo a través de la utilización de un dron, se adecúa a estas necesidades y soluciona sus problemas.

2.4 Beneficios de utilizar drones

Actualmente se está introduciendo la tecnología de drones poco a poco en el sector del agro. Se encuentra en auge el término “Agrotech” que refiere a la tecnología en este sector.

Los drones se están utilizando principalmente en tareas de agricultura, como es la fumigación o la fotogrametría sobre cultivos. El uso de drones en el sector ganadero está transformando la manera en que los productores gestionan sus operaciones diarias, ofreciendo una serie de beneficios significativos en comparación con los métodos tradicionales.

Uno de los beneficios más importantes es el ahorro de tiempo que conlleva el uso de drones para realizar tareas de monitoreo y conteo de ganado. Con el uso de drones, este proceso se puede realizar en una fracción del tiempo, ya que el dron puede cubrir grandes áreas de manera rápida sin la necesidad de desplazar el ganado. Al reducir el tiempo invertido en estas tareas, los productores pueden dedicar más recursos y atención a otras áreas críticas del negocio, mejorando la productividad general del establecimiento.

El uso de drones también contribuye a la reducción de costos operativos. Al disminuir la necesidad de mano de obra para tareas como el rodeo y el conteo de ganado, los productores pueden optimizar sus gastos en personal. Además, se reduce el desgaste de los recursos físicos, como vehículos y equipos, que generalmente son utilizados en los recorridos por el campo. El mantenimiento y operación de un dron es significativamente más económico en comparación con los costos asociados a la movilización del ganado.

El conteo manual de ganado ya sea en el campo o en la manga, está expuesto a errores humanos, como la omisión o el doble conteo de animales. Se estima que los drones en conjunto con tecnologías de reconocimiento de imágenes basadas en *machine learning*, pueden ofrecer una metodología con mayor precisión al identificar y contar automáticamente los animales desde una perspectiva aérea. Esto no solo reduce los errores humanos, sino que también genera reportes más fiables y consistentes para los productores y las auditorías externas. También, se realiza la tarea en un tiempo considerablemente menor, lo que optimiza los procesos del campo.

Otro beneficio clave es la reducción del estrés en los animales. Cuando el ganado es desplazado hacia la manga para ser contado con el método de lecturas en sus *RFID*, implica un proceso invasivo que puede generar altos niveles de estrés en los animales, afectando su bienestar y, en algunos casos, su productividad. Al utilizar drones para realizar el conteo desde el aire, se evita el contacto directo y el movimiento innecesario del ganado, lo que mejora su bienestar general.

2.5 Escalabilidad

Uno de los aspectos más importantes del proyecto es su potencial escalabilidad, tanto a nivel nacional como internacional. La implementación de un sistema de conteo automatizado de ganado basado en drones y tecnología de *machine learning* no sólo responde a las necesidades actuales del sector ganadero en Uruguay, sino que también presenta una solución aplicable a diferentes contextos y geografías, permitiendo su expansión a otros mercados.

Uruguay es un país con una fuerte orientación agropecuaria, siendo la ganadería una de las actividades económicas más relevantes. Actualmente, existen miles de establecimientos rurales de diferentes tamaños que enfrentan el desafío de realizar conteos precisos y regulares de su ganado. La solución propuesta, que integra drones con tecnología de reconocimiento de imágenes, puede adaptarse fácilmente a cualquier establecimiento ganadero en todo el país, independientemente de su ubicación o tamaño.

El sistema ofrece varias ventajas que impulsan su adopción a nivel nacional:

- Aplicación en diferentes escalas: Tanto los pequeños productores como los grandes establecimientos ganaderos pueden beneficiarse de la solución. El sistema es flexible y puede ajustarse para manejar diferentes volúmenes de animales y tamaños de terrenos, lo que lo convierte en una herramienta versátil para todo el sector.
- Adaptabilidad a las normativas locales: El uso del sistema automatizado puede cumplir con las auditorías y regulaciones del Ministerio de Ganadería, Agricultura y Pesca, facilitando los controles periódicos exigidos por el gobierno. Al automatizar el proceso de conteo, los productores pueden realizar auditorías de manera más eficiente y cumplir con los requisitos legales sin complicaciones.
- Reducción de costos y tiempo a nivel nacional: La adopción de esta tecnología en todo el país reduciría los costos operativos relacionados con el conteo de ganado y mejoraría la productividad del sector ganadero en su conjunto. Al ser una solución que no requiere de una infraestructura compleja, su implementación puede realizarse de manera rápida y con una inversión moderada.

- Consideración de los costos de escalabilidad: Si bien la implementación en la nube a través de AWS proporciona una infraestructura altamente escalable, manejar un volumen significativo de videos en alta resolución podría generar costos

considerables. Alternativas como la compresión de videos antes de la subida o el uso de almacenamiento de menor costo podrían optimizar los recursos y hacer el sistema más rentable a largo plazo.

Se plantea que el proyecto también se de en una modalidad de servicio en lugar de un producto ya que esto podría brindar ventajas significativas, como la reducción de costos fijos en infraestructura y el acceso a drones de mayor calidad, lo cual haría el sistema más viable para productores de distintos tamaños. Nuestro sistema es capaz de funcionar con cualquier tipo de dron, aunque se establecen requerimientos mínimos para algunas características del dron como la resolución de imagen y autonomía.

También se podría escalar a nivel internacional ya que la ganadería es una industria global, y los desafíos que enfrentan los productores uruguayos en cuanto al conteo de ganado no son exclusivos del país. Otros países con economías agropecuarias similares, como Argentina, Brasil, Paraguay, Australia y Nueva Zelanda, también podrían beneficiarse de esta tecnología.

Al existir softwares para manejo semi automatizado de los drones, y ejecutar el procesamiento en la nube, cualquier persona puede realizar un conteo con este producto sin importar donde se encuentre, solo necesita un dron para grabar, y conexión para subir el video a procesar.

2.6 Alcance

El alcance de este proyecto se define por los objetivos planteados y las limitaciones dentro de las cuales se desarrollará la solución. Se trata de un proyecto orientado a la creación de una herramienta tecnológica que, mediante el uso de drones y *machine learning*, optimice el proceso de conteo de ganado en establecimientos rurales.

El proyecto tiene un enfoque principalmente tecnológico, desarrollando un sistema que combina hardware y software para resolver la problemática del conteo de ganado. Las tecnologías clave que se utilizarán son drones equipados con cámaras de alta resolución y modelos de *machine learning* para el reconocimiento de imágenes. El alcance técnico del proyecto abarca los siguientes puntos:

- Desarrollo de un sistema de reconocimiento de imágenes: El sistema debe ser capaz de identificar con precisión a los animales (vacas) en videos capturados por el dron, utilizando modelos de aprendizaje automático entrenados específicamente para este propósito.
- Desarrollo de una interfaz web: El sistema incluirá una aplicación web que centralice los datos obtenidos y los presentes de manera accesible al usuario. Los productores rurales podrán consultar los resultados de los conteos, ver los videos procesados y acceder a estadísticas relevantes sobre el estado de su ganado.
- Validación y pruebas del sistema: El sistema deberá ser probado en un entorno real o simulado, garantizando que los modelos de *machine learning* sean precisos y eficientes, y que la solución sea confiable y fácil de usar por los productores.

Las funcionalidades del sistema estarán enfocadas en resolver las necesidades específicas de los productores ganaderos. El proyecto desarrollará las siguientes capacidades:

- Conteo automático de ganado: El sistema permitirá contar de forma precisa y automática el número de vacas en un establecimiento rural, sin necesidad de intervención humana directa.

- Historial de conteos: Los usuarios podrán acceder a un historial de conteos de ganado, con la posibilidad de visualizar datos y videos correspondientes a cada jornada de conteo.
- Interfaz sencilla: Dado que el público objetivo incluye productores rurales con limitado acceso a tecnologías avanzadas, el sistema contará con una interfaz intuitiva y fácil de manejar, asegurando que cualquier persona pueda utilizarlo sin requerir conocimientos técnicos especializados.

Aunque el proyecto tiene un potencial amplio, también presenta algunas limitaciones que se tendrán en cuenta durante su desarrollo:

- Cobertura de los drones: La efectividad del dron dependerá del tamaño del terreno y sus condiciones. En terrenos con obstáculos (como árboles) o pendientes pronunciadas que varían drásticamente la altura, podría ser necesario movilizar mínimamente el ganado para solucionarlo.
- Condiciones climáticas: El funcionamiento del sistema puede verse afectado por las condiciones meteorológicas, como lluvias intensas o vientos fuertes, que podrían interferir con el vuelo de los drones y la calidad de las imágenes capturadas.
- Precisión de los modelos de reconocimiento: Si bien se entrenará modelos de *machine learning* para maximizar la precisión, siempre existirán márgenes de error en la detección y conteo de animales, especialmente en situaciones de superposición o aglomeración de ganado.

2.7 Fases de desarrollo

A lo largo del desarrollo de este proyecto, se llevaron a cabo distintas fases que permitieron la implementación y verificación del funcionamiento, siguiendo una metodología estructurada que abarca desde la investigación inicial hasta el *testing* del sistema.

En primer lugar, se realizó una fase de investigación y definición de requerimientos, donde se identificaron las necesidades específicas de los productores ganaderos y se determinaron los objetivos del proyecto. Durante esta etapa, se investigó la problemática relacionada con el conteo de ganado en establecimientos rurales y se evaluó cómo las soluciones tecnológicas, en particular el uso de drones y *machine learning*, podrían optimizar este proceso. Se definieron los requisitos funcionales y técnicos, considerando aspectos clave como la integración de los drones, el reconocimiento de imágenes y la interfaz de usuario para los productores. También se identifica cuál será el hardware a usar (dron) y qué modelo se ajustaba de manera óptima para el proyecto.

Posteriormente, en la fase de diseño del sistema, se desarrolló la arquitectura tanto del software como de los hardware involucrados. Se entrenó el sistema de reconocimiento de imágenes, incluyendo la selección y configuración del modelo de *machine learning* que permitiría reconocer el ganado en las imágenes capturadas y llevar una cuenta. En paralelo, se construyó un prototipo funcional de la interfaz web, en el que los usuarios podrán visualizar los resultados de los conteos y acceder al historial detallado para facilitar la gestión de su ganado. Durante esta fase, se planteó la integración de todos los componentes, garantizando que tanto el hardware (drones) como el software (modelos de *machine learning* y aplicación web) funcionarán de manera conjunta.

Una vez completado el diseño, se avanzó hacia la fase de desarrollo, en la que se materializaron los componentes previamente planificados. El modelo de m*achine learning* fue

entrenado utilizando un conjunto de datos muy amplio de imágenes del ganado, con ajustes constantes para mejorar su precisión en el reconocimiento de vacas en diversas condiciones. Por otro lado, la aplicación web fue desplegada en un servicio de AWS, asegurando que los usuarios pudieran interactuar con la plataforma, allí pueden subir videos y visualizar resultados con facilidad desde cualquier parte.

Finalmente, se ejecutó una fase de pruebas y validación. Se realizaron pruebas exhaustivas del modelo de *machine learning*, evaluando su precisión, *recall* y otros parámetros en el reconocimiento de ganado. También se prueba el conteo de ganado en diferentes lugares y con aglomeración de animales. La interfaz web también fue probada, enfocándose en la experiencia del usuario y en asegurar que la plataforma fuera accesible y funcional para productores rurales sin conocimientos técnicos avanzados. Por último, se llevó a cabo una validación del sistema integrado, comprobando que el flujo completo, desde la captura del video hasta la visualización de los resultados, se ejecute sin inconvenientes. Estas pruebas permitieron ajustar los últimos detalles y garantizar que el sistema cumpliera con los objetivos planteados.

De esta manera el proyecto fue desarrollado en distintas fases que permitieron una implementación funcional y adaptada a las necesidades y conocimientos de productores ganaderos, garantizando que los objetivos propuestos fueron alcanzados con éxito.

3. Requerimientos

La solución permitirá la creación de establecimientos rurales, la gestión de ganado en lotes y pasturas, y la carga y procesamiento de videos en 4K utilizando *machine learning* para el conteo automático de ganado. El proyecto se desarrollará con la previsión de escalabilidad, dado que el producto está destinado a un mercado global.

A continuación, se detallarán los requerimientos funcionales y no funcionales para el desarrollo del proyecto así cómo los requerimientos comerciales y finalmente los requerimientos para el *testing* del software y la evaluación del modelo.

3.1 Requerimientos para el desarrollo

El objetivo es desarrollar una plataforma web que permita a los usuarios gestionar establecimientos rurales y lotes de ganado, y que cuente con un sistema de procesamiento de video en 4K para contabilizar ganado utilizando un modelo de *machine learning*. Este *MVP (*Mínimo producto Viable*)* será la base para futuras expansiones, como la integración con drones y sistemas de estimación de peso de ganado.

3.1.1 Requerimientos funcionales

El sistema deberá contar con las funcionalidades detalladas a continuación.

Gestión de usuarios

Registro de usuarios: Los usuarios deberán poder registrarse con su email personal.

Iniciar sesión: Deberá tener una vista de inicio de sesión por el cual una vez creada la cuenta el usuario deberá poder ingresar con sus credenciales de registro.

Se tendrán dos roles para el usuario, a asignar en la pantalla de administrador:

- Administrador: Es el superusuario, el mismo tiene control sobre la base de datos directa para todos los usuarios, y puede efectivizar cualquier cambio sobre ella que requiera necesario.

- Miembro: Tiene permisos para visualizar la aplicación, y trabajar sobre toda la gestión de los establecimientos, también puede crear establecimientos.

Gestión de establecimientos rurales

Crear establecimientos: Los usuarios deben poder crear y gestionar varios establecimientos rurales.

Agregar miembros: El creador de un establecimiento puede invitar a otros usuarios para unirse a la gestión del establecimiento.

Gestión de lotes y pasturas

Crear lotes: Los usuarios pertenecientes a un establecimiento deben poder crear lotes de ganado especificando el número de cabezas de ganado y la categoría del ganado dentro de ese establecimiento.

Asignar pasturas: Los lotes pueden estar asociados a pasturas, asimismo los usuarios van a poder mover los lotes por las distintas pasturas.

Carga y procesamiento de videos en 4K

Los videos obtenidos del dron se obtienen de la tarjeta microSD del drón la cual se debe conectar a una PC para luego ser subida al sistema mediante un proceso de carga segura en la nube, utilizando URLs firmadas y fragmentación para garantizar una transferencia estable, incluso en condiciones de conectividad limitada.

Subida de videos: Los usuarios deben poder subir videos en calidad 4K que serán extraídos del dron y utilizados para procesar el conteo de ganado.

Subida en condiciones de baja conectividad: Dado que el usuario principal de esta aplicación estará en áreas rurales, donde muchas veces la conectividad no es la mejor, la plataforma deberá poder soportar la subida que se pueda resumir, para así poder subir archivos pesados sin problemas.

Procesamiento del video: Una vez que se suba el video, el mismo será procesado por un modelo de ML entrenado específicamente para contar el número de cabezas de ganado.

Notificaciones de estado: Los usuarios deben ser notificados cuando el video haya sido subido y cuando el procesamiento haya sido completado.

3.1.2 Requerimientos no funcionales

Rendimiento

La plataforma web tiene que ser capaz de procesar el video en 4k eficientemente, optimizando el tiempo de procesamiento para entregar buenos resultados en un tiempo razonable.

Confiabilidad

El sistema debe ser confiable, tanto para la subida de archivos, asegurando que se pueda realizar la subida a pesar de la mala conectividad. También deberá asegurar la funcionalidad del sistema a pesar del gran manejo de datos y tráfico alto de usuarios.

Escalabilidad

La plataforma debe estar diseñada para ser escalable y soportar un crecimiento global. Esto significa que debe ser capaz de manejar grandes volúmenes de datos y tráfico de usuarios, además de asegurar una experiencia de usuario fluida. Utilizar servicios de infraestructura en la nube como AWS o Google Cloud, con servicios como S3 para el almacenamiento de los videos y EC2 para el procesamiento de los modelos de *machine learning*. De esta manera aprovecha la ventaja de los créditos de AWS que proporciona el CIE para con la empresa.

3.1.3 Requerimientos por integraciones futuras

Estos requerimientos son a partir de funcionalidades que se busca tener a futuro para un mejor producto. Se deberá pensar el desarrollo previendo estas actualizaciones a futuro.

Integración con DJI

En un futuro el sistema deberá poder integrarse con la API de DJI, así poder integrar funcionalidades del dron con la aplicación web. Pudiendo intercambiar datos geográficos entre el dron y la aplicación. Esto permitirá registrar rutas de vuelo, datos geográficos del ganado entre otras cosas.

Estimación de peso de ganado

Se desarrollará una funcionalidad para que, además del conteo de ganado, los videos también puedan ser utilizados para estimar el peso del ganado mediante el uso de modelos de *machine learning* adicionales. El despliegue de este modelo se daría de manera muy similar al de conteo, por lo que su integración no debería dar problemas.

Edge computting

Aunque se descartó el procesamiento en tiempo real a bordo del dron debido a limitaciones en autonomía y eficiencia, reevaluar esta opción en campo para funciones críticas podría ser beneficioso. Esta capacidad permitiría una mayor automatización del sistema, una mayor rapidez en la obtención de los datos y reducir posibles inconsistencias en entornos variables.

3.1.4 Requerimientos modelo de conteo

El modelo entregado debe ser capaz de detectar múltiples vacas en una misma foto. De manera que, si se tiene un video, se puedan procesar sus fotogramas por separado y realizar un conteo del total de instancias que se identificaron. Para esto es importante el uso del *tracking*, que detecta si una instancia es la misma del frame anterior.

También un objetivo importante, es que el modelo debe ser capaz de recibir un video como entrada, aunque se procese dicho video de manera *frame* a *frame*, la caja negra debe recibir como entrada un video y entregar en la salida dicho video procesado, con *bounding boxes* alrededor de las detecciones, junto a información en memoria que contenga la cantidad total de detecciones únicas que hay en el video.

Dada la complejidad del armado de un set de datos de este tipo, el cual requiere imagenes de ganado desde cierto punto de vista (dron a aproximadamente 35 metros de altura), y vistas en cierto ángulo, se considera que un alcance razonable es definir qué se debe cumplir con las cantidades que recomienda Ultralytics en su documentación para entrenamientos óptimos, estas recomendaciones, se describen en el capítulo de Investigación.

En cuanto a la precisión, se plantea que, en condiciones normales, el modelo no puede fallar, es decir imágenes muy claras, de vacas individuales, donde se contrasta notoriamente con el fondo. Se aceptan fallas si son vacas muy apretadas y se solapan entre ellas, o en caso de que no se logre contrastar bien el contorno del animal debido a la iluminación o calidad de las imágenes. Esto se puede resumir en que, la precisión y el *recall* deben ser del entorno de 99%, términos que se explican y detallan en el apartado 6.1.

3.2 Requerimientos comerciales

En concordancia a los objetivos, es un requerimiento del proyecto que se logre un producto comercial el cual brinde un valor mayor a los costos que implica. Para lograr esto, se plantea que cada compra debe ser totalmente justificada, buscando utilizar la menor cantidad de recursos para lograr los objetivos de manera eficaz y eficiente.

Esto implica que, por ejemplo, no se compre un dron caro solamente para tener la mejor cámara y batería, ya que no es justificado debido a que se puede realizar un proyecto así con un dron de gama media. El otro gasto que se tiene en el proyecto es sobre la tecnología *Cloud* donde corren los procesos. Este gasto también debe ser mesurado y no optar por la mejor GPU, ya que no es estrictamente necesario procesar los videos en pocos minutos. Este último gasto, si bien está cubierto por un acuerdo del CIE ORT con AWS, igualmente se considera como un costo del proyecto, eligiendo los recursos computacionales de manera optimizada como si se tuviera que pagar.

3.3 Requerimientos de *testing*

El *testing* de software es fundamental para garantizar que la plataforma web funcione correctamente, que los usuarios puedan interactuar con ella sin problemas, y que las funcionalidades, como la gestión de establecimientos y la subida de videos funcionen correctamente.

3.3.1 *Testing* funcional

Este tipo de *testing* se basa en poder verificar que las funcionalidades que se detallen del sistema operen de acuerdo con los requerimientos funcionales especificados.

- Verificar que el registro, inicio de sesión y cambio de contraseña funcione correctamente.
- Probar que los usuarios puedan crear y gestionar establecimientos, lotes, y pasturas de manera adecuada.
- Asegurar que los videos en 4K se puedan subir sin interrupciones.
- Probar que el sistema maneje correctamente videos que no cumplen con los requisitos (formatos inválidos, resoluciones diferentes, etc.).
- Verificar que, una vez subido el video, el sistema lo procese correctamente utilizando el modelo de *machine learning*.
- Probar que el conteo de ganado y su historial se refleje adecuadamente en la interfaz de usuario.

3.3.2 *Testing* no funcional

Estas pruebas evalúan aspectos como el rendimiento y la usabilidad del sistema.

- Probar que el sistema pueda manejar múltiples usuarios subiendo videos simultáneamente sin afectar el rendimiento.

- Asegurar que el tiempo de procesamiento del video sea razonable y que el sistema pueda soportar la carga de trabajo esperada.

- Asegurar que la interfaz sea intuitiva y fácil de usar, especialmente para usuarios en áreas rurales con menor acceso a tecnología.

3.3.3 Requerimientos de evaluación del modelo de *machine learning*

Para asegurar que el modelo de *machine learning* que cuenta el ganado en los videos alcance el nivel de precisión requerido (99%), es fundamental realizar un proceso riguroso de evaluación y *testing* del modelo.

Evaluación del rendimiento del modelo

El rendimiento del modelo de *machine learning* se mide utilizando varias métricas que permiten cuantificar su precisión y su capacidad para generalizar a nuevos datos.

Precision (Precisión): La precisión mide qué porcentaje de los objetos detectados por el modelo son realmente correctos, minimizando los falsos positivos (es decir, objetos que fueron identificados incorrectamente como ganado).

Objetivo: Alcanzar al menos un 99% de precisión en los datos de prueba.

Recall: Esta métrica mide la capacidad del modelo para detectar todos los animales en el video, minimizando los falsos negativos (es decir, animales que no fueron detectados).

Objetivo: *Recall* alto para asegurarse de que el modelo no se esté perdiendo cabezas de ganado.

Evaluación en diferentes condiciones

Es esencial que el modelo funcione bien en una variedad de condiciones, ya que los videos capturados en el campo pueden variar considerablemente.

Condiciones de luz

Probar el modelo en videos con diferentes niveles de iluminación, como videos tomados al amanecer, en pleno sol o al atardecer.

Ángulos de cámara

Evaluar cómo el modelo maneja videos tomados desde diferentes ángulos y alturas, considerando la variabilidad en los drones o cámaras usadas.

Entorno visual

Probar en diferentes tipos de pasturas o terrenos para asegurarse de que el modelo pueda diferenciar correctamente entre el ganado y el entorno.

Uso de múltiples sets de datos

El uso de distintos conjuntos de datos, en distintas condiciones, asegura que el modelo de detección generaliza y no se ajusta específicamente a un cierto set de validación. Esto es fundamental para asegurarse de que no existe sobre ajuste.

Evaluación de la escalabilidad del modelo

Es importante asegurarse de que el modelo pueda procesar videos de manera eficiente, incluso cuando la carga aumente considerablemente. Esto incluye:

Pruebas de inferencia: Evaluar la velocidad con la que el modelo puede hacer predicciones cuando se le proporciona un video en 4K. Debe haber un equilibrio entre la precisión y la velocidad.

4. Investigación de drones

En este apartado, se investiga y se genera un listado de los requerimientos que debe incluir el dron ideal para este proyecto. Es importante que el dron cumpla todos los requerimientos y además, ser conservadores en la compra, buscando un equilibrio entre grandes capacidades y precio. También, se fundamenta por qué se elige el dron DJI Mini 3 Pro como el dron para llevar a cabo el proyecto, junto a sus capacidades que lo hacen ideal.

4.1 Características predefinidas para la elección del dron

Programabilidad

Es necesario tener la capacidad de indicarle al dron y darle la inteligencia para recorrer un establecimiento. Esto permite que, con programación, se pueda optimizar las rutas que toma el dron para recorrer toda la superficie en el menor tiempo posible. Para esto, también depende de tener un GPS, donde se pueda ver el mapa satelital de la ubicación donde se encuentra.

Rendimiento de la batería

Para este proyecto se estima que se realizan vuelos cortos, aun así, mínimamente se espera que el dron pueda realizar vuelos de 20 minutos ininterrumpidos. De esta manera, se asegura que el dron sea capaz de recorrer superficies de un tamaño considerable.

Cámara

El dron debe contar con una cámara equipada, o la posibilidad de incluir una, con una resolución suficiente para reconocer el ganado. Esta cámara debe contar con las siguientes características para cumplir con los objetivos del proyecto.

Definición suficiente

La calidad de la imagen garantiza que se puede grabar y conseguir buena definición en las imágenes tomadas desde muy alto. Mientras más resolución, más alto se puede volar, lo que implica que el vuelo para recorrer una superficie se de en menos tiempo. Se define como mínimo una calidad Full HD, lo cual es un estándar razonable de buena calidad en términos de resolución de imagen.

Ángulo *FOV* mínimo de 80°

El ángulo *FOV* (*Field Of View*), corresponde al campo de visión de la cámara. Se busca un campo de visión típico en un buen dron. Similar al punto anterior, ante mayor campo de visión, menor tiempo se necesita para cubrir toda la superficie.

Grabación estable

Dado que el conteo de ganado implica grabaciones en movimiento a una velocidad considerable, el dron debe contar con un estabilizador, de manera que las imágenes no sean afectadas por las turbulencias que produce el viento en el dron.

Costo

El dron debe ser económicamente viable, se buscará el mejor dron posible que cumpla los requerimientos que se tienen, buscando reducir los costos de la mayor manera posible.

GPS

Que incluya sistema GPS es importante debido a la utilidad que brinda, ya que las aplicaciones desarrolladas actualmente para drones, que permiten recorridas de superficies inteligentes, dependen de tener un GPS integrado.

Durabilidad

Esta opción es deseable pero no obligatoria, refiere a resistente a caídas, golpes, inclemencias climáticas, etc. En el mejor de los casos el dron incluirá un sistema que lo prepare para dichos casos, pero si no lo incluye no es un impedimento a comprarlo.

4.2 Investigación individual de drones

En este apartado, se investiga individualmente cada dron, examinando sus especificaciones técnicas y se determina si cumple o no con los requerimientos para el proyecto. Se estudian las marcas más relevantes, tanto en Uruguay como internacionalmente. Estas marcas son DJI, Cheerson y Parrot.

Se estudiaron aproximadamente 20 drones en total, en el documento se desarrollan los más importantes.

DJI Mavic 2 Enterprise

Este es un dron de gama alta, de un tamaño mediano (ver imagen 2).

Imagen 2 - Imagen ilustrativa del dron DJI Mavic 2 Enterprise. [1]

Características a favor

- Incluye zoom óptico x2: Lo cual puede ser importante para grabar a largas distancias.
- Vuelo hasta 8km de distancia con 30 minutos de batería.
- Cámara: *FHD* (*Full HD* 1080p), 12MP y grabaciones en 4k, 60/30 fps.
- Sistema de anticolisiones de 260 grados.
- Muy veloz, hasta 70km/h (a una altura del nivel del mar, sin viento) o a grandes alturas a 50km/h (sin viento).
- Memoria interna de 24GB.
- Incluye dos cámaras, lo cual permitiría estimar volúmenes de mejor manera.

Características en contra

- Precio (Importación).

El DJI Mavic es una buena opción, tiene una buena cámara, permite grabaciones en 4k. La batería no es muy buena, pero existe una opción plus de batería que nos puede alargar un poco más el vuelo. Una gran ventaja es su velocidad, esto nos permitiría recorrer largas extensiones de campo y llegar al ganado de manera muy fácil. El precio ronda los 3000 dólares, es algo caro aún para poder hacerlo escalable.

DJI Matrice 100

Dron de la gama más alta de DJI, tamaño grande (ver imagen 3).

Imagen 3 - Imagen ilustrativa del dron DJI Matrice 100. [2]

Características a favor

- Dron totalmente programable, es compatible con *On Board SDK*.

- Distancia de hasta 5 km.
- Batería para un vuelo de 20 minutos y espacio para una batería externa.
- Sistema anticolisiones vertical y horizontal.
- No incluye cámara integrada, se debe comprar por separado.
- Puede incluir distintos sensores, como sensores térmicos, (Podrían ayudar a reconocer ganado).
- *RTSP* para transmisión de video, inclusive infrarrojas o térmicas.
- Sistema GPS.

Características en contra

Tiene un precio bastante elevado. Alrededor de U$S 10.000

- No incluye zoom óptico, aunque se podría comprar un *aerial gimbal* de DJI pero aumentaría alrededor de U$S 3000 el precio.

Este dron es muy bueno, pero la batería es muy corta ronda los 20 minutos, es un dron grande y pesado y tiene la habilidad de utilizar hasta dos *gimbals* en donde se puede colocar dos cámaras de distintas índoles desarrolladas específicamente para DJI, por ejemplo, cámara de infrarrojos o hiperespectral, sin embargo estos son aparte, así que más allá del precio del dron que ya es muy caro, se le agregaría el precio de los *gimbals*.

Cheerson CX-20

Dron de la marca Cheerson (ver imagen 4).

Imagen 4 - Imagen ilustrativa del dron Cheerson CX-20. [3]

Características a favor

- Autonomía de 20 minutos.

- Es *open source.*
- Compatible con placa ardupilot (C/C++).
- No incluye cámara, trae un *slot* diseñado para cámaras similares a GoPro.
- Funcionalidades como *back home*, alcance hasta 5km.
- Software Auto Pathfinder para programar vuelos.
- Dron muy barato.

Características en contra

- No incluye zoom óptico.
- Viene equipado con un soporte para agregar una GoPro desde la versión 9 hasta la 11. No viene con cámara.

El CX-20 es un dron programable con la placa ardupilot esto nos permitiría realizar cierto procesamiento en vuelo. Sin embargo, no viene con una cámara integrada.

DJI Tello

El dron DJI Tello es un dron ideal para proyectos pequeños de *computer vision*. Es un dron barato con el cual se suele iniciar en la programación de drones (ver imagen 5).

Imagen 5 - Imagen ilustrativa del dron DJI Tello. [4]

Características a favor

- Costo económico bajo de aproximadamente 220 dólares en Uruguay.
- Cuenta con SDK a bordo, lo cual facilita la programación del dron y no requiere una placa externa.

Características en contra

- Baja autonomía, 13 minutos.
- No cuenta con GPS a bordo.
- La resolución de la cámara es de grabaciones a 720p y fotos a 5 MP.
- Es controlable solamente en una distancia de hasta 100 metros.

DJI mini 3 Pro

Dron con prestaciones profesionales, de pequeño tamaño y alto rendimiento (ver imagen 6).

Imagen 6 - Imagen ilustrativa del dron DJI Mini 3 Pro. [5]

Características a favor

- Precio 1190 USD.
- Batería: 34 minutos con expansión a 47 con batería plus.
- Cámara: 48 MP, 4k 60 fps, zoom x2, FOV 82,1°.
- Velocidad: 16 m/s en modo veloz - 10 m/s en modo normal.
- Compatible con Mobile SDK.

El dron DJI Mini Pro-3 es un dron que a simple vista ya nos llamó la atención. A un muy buen precio para un dron de estas prestaciones pudiendo cumplir todos los ítems que se mencionaron anteriormente. Permanece cómo una muy buena opción.

4.3 Placas programables

Se investiga también la posibilidad de utilizar un dron que no sea de capacidad programable y poder agregarle una placa programable para el procesamiento, el plan de vuelo y distintas *features* para agregarle a el proyecto.

Estas placas deben ser livianas en comparación con el dron por razones aerodinámicas, también se deben considerar los requerimientos de alimentación que conlleva establecidos. Sin dificultar que se mantenga un mínimo de 30 minutos de vuelo del dron. Las placas deben brindar la posibilidad de guardar vídeo o procesarlo en el momento.

Placa Ardupilot

Ardupilot es un proyecto *open source* que brinda hardware para el manejo de distintos dispositivos aéreos, entre ellos, drones.

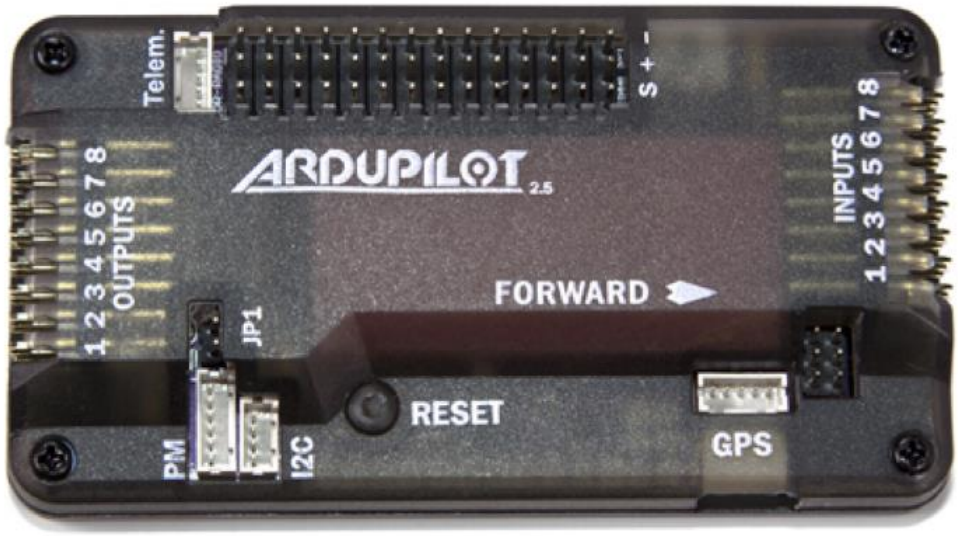


Imagen 7 - Imagen ilustrativa de la placa Ardupilot. [6]

- Esta placa trae un procesador con dos núcleos, con los que controla la estabilización del dron, la planificación de vuelos por GPS, el control de la cámara, etc. (ver imagen 7).
- De esta manera, permite convertir un dron no programable en uno programable, siendo una opción económica.
- Compatible con el IDE de Arduino.

Placa genérica

Otra posibilidad, es utilizar una placa con un cómputo bajo, pero con conexión a internet a bordo. Utilizando ingeniería inversa, se podrán interceptar las señales que envía la placa del dron a sus motores para el movimiento, y de esta manera, con una placa de por medio, poder comandar de la manera que deseamos.

Por lo que es posible realizar este proyecto con una placa como las siguientes:

- Raspberry PI 4
- ESP32 o ESP 8266

- Placas de la familia ATMEGA.

Las características a favor que brinda esta opción son las siguientes:

- Costo económico bajo.
- Es adaptable a cualquier tipo de dron.

Las características en contra son las siguientes:

- Agrega un peso externo al dron, lo cual afecta en su aerodinámica y en rendimientos de batería.
- Es probable que las señales sean únicas para cada dron, por lo que el trabajo se puede adaptar a otros modelos o marcas, pero no será repetible a corto plazo
- Hay problemas para acceder a los esquemáticos de los circuitos de los drones, por lo que al intervenir en ellos podrían generarse inconvenientes.

4.4 Conclusiones del análisis sobre drones y placas programables

Para tomar la decisión de qué dron adquirir, se tienen en cuenta todos los factores analizados anteriormente. También se realiza una entrevista al Ing. Ignacio López, quien tiene experiencia trabajando en desarrollo de software para drones y nos asesora sobre los distintos drones existentes. Esta entrevista está disponible en el Anexo 4.

Sabemos que no se podrá procesar las imágenes a bordo, ya que para esto necesitamos un *Onboard SDK* que se escapa del presupuesto. Además, es poco eficiente en términos de batería, ya que un procesamiento a bordo exigirá al máximo las capacidades del dron. La opción de utilizar alguna placa programable también se valoró, pero se terminó descartando por varias razones. Por un lado, no encontramos drones compatibles con estas placas que cumplan con los requerimientos de calidad establecidos, ya sea por su cámara, por no tener cámara, o por el corto vuelo.

Por otro lado, pensando en la escalabilidad del proyecto a largo plazo no se puede esperar un retrabajo sobre hardware de plaza para poder incluir una placa como la Ardupilot analizada. Por lo cual se busca algo en plaza que cumpliera con grandes partes de los requerimientos y que nos brinden el factor de escalabilidad.

Los drones DJI, son absolutamente controlables utilizando un dispositivo móvil, inclusive se pueden realizar vuelos inteligentes con ruteos optimizados, y el pasaje de un video grabado en la tarjeta SD del dron hacia una computadora, utilizando un software para transferencias de archivos diseñado por la marca.A su vez, DJI es la única marca de renombre con una tienda oficial en Uruguay, a su vez sus drones cuentan con grandes prestaciones aptas para el proyecto.

Se apunta en un principio al dron DJI Mini 3 Pro, debido a que este dron permitirá migrar todo lo desarrollado a un dron más grande de la misma marca, como puede ser el Mavic 3, que brindaría un gran salto en calidad si es deseable para el usuario final. De todas formas, el DJI Mini 3 Pro ya brinda considerables prestaciones como ser compatible con un SDK Mobile y tener una muy buena cámara integrada. Además, este dron sirve para hacer procedimientos

como la fotogrametría, lo cual permitiría tomar medidas de longitud, ancho y altura, para posteriormente realizar modelados 3D de un animal.

También se adquiere un kit de 3 baterías inteligentes con el presupuesto que brinda la universidad para estos proyectos, lo cual expande la autonomía del dron a 47 minutos y nos da la posibilidad de realizar vuelos considerablemente largos, pudiendo recorrer grandes superficies.

4.5 Estudio de superficie cubierta respecto a la altura

Como se mencionó anteriormente, una característica importante de los drones, es que tengan una resolución de imagen suficiente para realizar vuelos a gran altura. Por lo que se debe analizar cuánta superficie cubre ante determinada altura y también tener en cuenta que, a mayor altura, tendré una menor cantidad de pixeles para cada animal, entonces el modelo de predicción también va a perder precisión. Se debe buscar optimizar esto, logrando un punto en el cual la resolución sea suficientemente buena para detectar animales, y se vuele a una altura que permita una recorrida en menos tiempo. Dado que el campo de visión de nuestro dron es 82,1°, se puede calcular el ancho de visión que cubre y luego calcular la superficie de un círculo, aproximándose a la superficie que se verá en imagen.

$$Ancho_{Total} = 2 \cdot h \cdot tan(\frac{FOV}{2})$$

$$Área = \frac{\pi}{2} \cdot (h \cdot tan(\frac{FOV}{2}))^2$$

Teniendo el ángulo de campo de visión del dron, se puede realizar una buena aproximación aplicando trigonometría, dejando la altura variable. Obteniendo un gráfico de superficie cubierta respecto a la altura (ver imagen 8).

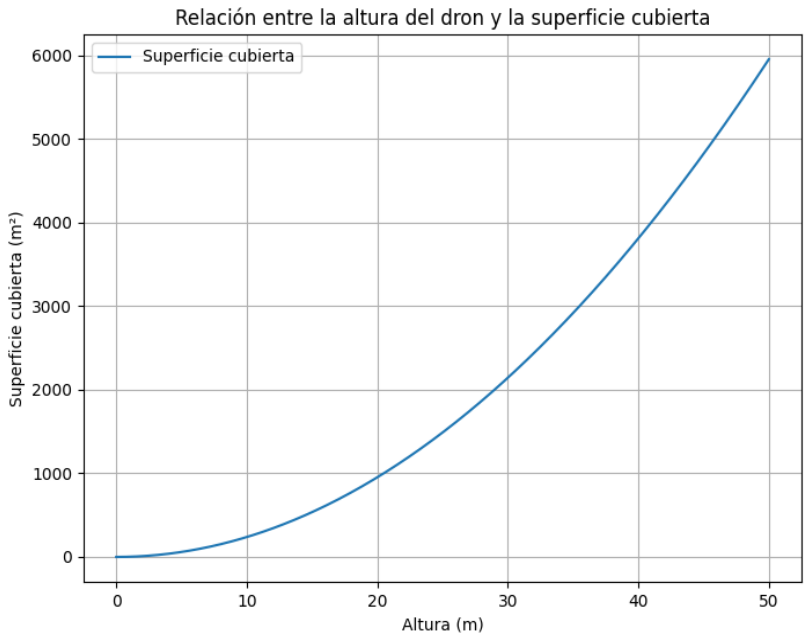


Imagen 8 - Gráfico de superficie cubierta respecto a la altura.

Donde claramente, al incrementar la altura, aumenta exponencialmente la superficie cubierta. Desde este punto de vista, lo mejor es ir lo más alto posible, sin embargo, esto es limitado por la calidad de imagen del dron, el funcionamiento del modelo de detección y la altura máxima a la cual puede ascender el dron.

Además de viajar a la mayor altura posible que permita una buena detección, es fundamental planificar rutas de vuelo que minimicen puntos ciegos para asegurarnos de tener una cobertura completa del área de pastoreo. También se debe evitar lugares con obstáculos como árboles. Aunque drones como el seleccionado cuentan con detección de obstáculos y no van a chocar, tener obstáculos no va a permitir visualizar el ganado y no podremos contarlo.

5. Aplicación web

Este capítulo describe todo lo referente a la aplicación web. Desde los conocimientos que se tuvieron que adquirir para el desarrollo, hasta el procedimiento realizado y la evaluación de la aplicación.

5.1 Marco teórico y elección de *stack* tecnológico

En este apartado se realizan las investigaciones pertinentes al proyecto, en estas investigaciones se basa el posterior desarrollo del proyecto. Es la primera, y la de mayor importancia de las etapas. Es fundamental realizar una buena investigación para tomar decisiones que no solo impactan en los resultados del proyecto, sino que también en costos.

Para definir el *stack* de tecnologías de la aplicación web se partió de los requerimientos funcionales y no funcionales detallados en el capítulo 3.

5.1.1 Framework de desarrollo

Un *framework* de desarrollo [7] es un conjunto de herramientas, librerías y buenas prácticas diseñadas para facilitar y estructurar el proceso de creación de aplicaciones. Su propósito principal es proporcionar soluciones preconstruidas para tareas comunes.

Dados los requerimientos de este proyecto, analizaremos dos frameworks, Flask y Django.

Flask es un *framework* de desarrollo web *WSGI* (*Web Server Gateway Interface*) ligero. Está diseñado para que empezar el desarrollo sea muy rápido y simple, con la habilidad de escalar a aplicaciones complejas. Flask permite organizar el código sin seguir una estructura rígida, por más de que provee sugerencias del mismo. Flask también provee funciones muy interesantes cómo el enrutamiento de URLs, gestión de solicitudes y respuestas, el soporte para extensiones, y manejo de base de datos. Está principalmente enfocado al desarrollo de prototipos, aplicaciones web de pequeña escala o proyectos que requieren un rápido desarrollo.

Por otro lado, Django es un *framework* de alto nivel en Python. Django tiene un patrón de diseño *MTV* (*Model-Template-View* - Modelo, Plantilla, Vista) lo que hace que la separación de responsabilidades sea clara, y facilita el mantenimiento y escalabilidad de la aplicación. Django utiliza su *ORM* (*Object-Relational Mapping*) para interactuar directamente con la base de datos a través de su modelo, sin tener que escribir SQL. Está diseñado para crecer junto con la aplicación. Cuenta con una arquitectura modular, que permite el uso de cacheo, y la posibilidad de integrar bases de datos distribuidas o múltiples instancias de servidores, siendo capaz de manejar grandes volúmenes de tráfico y datos. Además, su capacidad para integrarse con soluciones como Celery para tareas asíncronas y servicios de almacenamiento como Amazon S3, lo convierte en una opción ideal para aplicaciones de gran escala. Por estas razones y su robusto ecosistema se eligió Django para ser el *framework* de desarrollo de esta aplicación web. Comparado con Flask (que requiere más configuración manual), Django facilita el desarrollo rápido de aplicaciones complejas y presenta mayor escalabilidad.

5.1.2 Base de datos relacionales

Una base de datos relacional [8] es un sistema de almacenamiento de información que organiza los datos en tablas, que a su vez están divididas en filas y columnas. Este modelo se basa en un enfoque estructurado y orientado al esquema, donde las tablas pueden estar relacionadas entre sí. Se denominan "relacionales" ya que los datos de diferentes tablas pueden estar conectados mediante relaciones a través de *primary* y *foreign keys*. En este caso, se consideraron para el manejo de bases de datos, PostgreSQL, MySQL, MariaDB y SQLite.

PostgreSQL es un *ORDBMS* (*Object-Relational Database Management System*), esta característica hace que PSQL se destaque por la capacidad de almacenar datos más complejos que las bases de datos tradicionales cómo lo pueden ser objetos, *arrays* o archivos multimedia. PSQL es un sistema de gestión de bases de datos robusto y confiable, ideal para aplicaciones que requieren manejo avanzado de datos y consultas complejas. Una de las grandes ventajas de PostgreSQL es su capacidad para manejar grandes volúmenes de datos y soporte para extensiones como PostGIS [9], que proporciona funcionalidades de geolocalización. Es una opción ideal para aplicaciones con altas exigencias en el rendimiento y la integridad de los datos.

MySQL y MariaDB, por su parte, son alternativas más ligeras y de fácil implementación. Son reconocidos por tener buen rendimiento para operaciones más simples de lectura y escritura, lo que los convierte en una buena opción para aplicaciones web de escala media. MySQL y MariaDB son ampliamente utilizados en la industria debido a su simplicidad, facilidad de uso y disponibilidad de soporte en numerosos servicios de alojamiento web. Aunque no cuentan con tantas características avanzadas como PostgreSQL, ofrecen buena escalabilidad y flexibilidad, haciéndolos ideales para proyectos que requieren una base de datos eficiente pero no necesariamente con todas las funcionalidades avanzadas.

SQLite es una opción más adecuada para proyectos pequeños o aplicaciones embebidas, donde la simplicidad y la facilidad de integración son más importantes que el rendimiento a gran escala. Su rendimiento y escalabilidad son limitados en comparación con sistemas como PostgreSQL o MySQL, por lo que es menos adecuada para aplicaciones que manejan grandes volúmenes de datos o múltiples usuarios simultáneamente.

PostgreSQL fue elegido por su robustez y su capacidad para manejar datos relacionales complejos con un rendimiento excelente. Comparado con el resto de las bases de datos mencionadas, PostgreSQL ofrece mejores capacidades para manejar transacciones y consultas avanzadas, lo cual es crucial en un sistema que puede necesitar consultas complejas sobre los datos de los lotes, pasturas, y resultados de *machine learning*. PostgreSQL también ofrece una gran integración con diversas aplicaciones, entre ellas PostGIS lo cual permite pensar en tomar datos geográficos en un futuro cumpliendo así con los requerimientos para integraciones futuras detallados en el capítulo 3.

5.1.3 Diseño web

Bootstrap [10] es un *framework* de código abierto destinado al diseño de interfaces web. Fue desarrollado internamente por Twitter. Ofrece una colección de herramientas y componentes que simplifican la creación de sitios web y aplicaciones, ya que tiene un *layout* que se adapta a la pantalla del dispositivo usado por el usuario. Su popularidad radica en su facilidad de uso, flexibilidad y en su capacidad para acelerar el proceso de diseño web.

Bootstrap se destaca por tener una amplia gama de componentes predefinidos como botones, formularios y barras de navegación que se integran fácilmente en el código, lo que agiliza el desarrollo web. La estructura modular de Bootstrap también facilita el mantenimiento y actualización de proyectos, permitiendo implementar mejoras con un impacto mínimo en el código existente. Debido a todas estas ventajas, se decide utilizar Bootstrap en el diseño web de nuestra aplicación.

5.1.4 Servidores web

Nginx

Nginx [11] es un servidor web de alto rendimiento que, además de servir archivos estáticos como HTML, CSS o JavaScript, funciona como proxy inverso y balanceador de carga. Nginx se sitúa delante del servidor WSGI para poder procesar las consultas y respuestas del cliente. Como proxy inverso, el servidor envía las consultas al Servidor WSGI que se encarga de procesar la consulta y ejecutar la lógica en el *backend*. Su arquitectura permite manejar un gran número de conexiones concurrentes, lo que lo convierte en una opción ideal para servir aplicaciones web y gestionar el tráfico de red.

Nginx se eligió sobre otros servidores web como Apache debido a su capacidad para manejar múltiples conexiones simultáneas, lo cual es muy importante y un gran requerimiento en la plataforma.

Gunicorn

Gunicorn [12] es un servidor WSGI diseñado para ejecutar aplicaciones web basadas en Python. WSGI es un mediador que se comunica entre servidores web y aplicaciones web Python. Es compatible con el estándar WSGI, lo que lo convierte en una opción robusta y eficiente para gestionar la comunicación entre aplicaciones web Python y el servidor web Nginx. Gunicorn actúa como un intermediario entre el servidor web y la aplicación Python, ejecutando las solicitudes y asegurando una comunicación fluida entre ambos. Gracias a su capacidad para manejar múltiples solicitudes concurrentemente mediante el uso de varios *workers*, Gunicorn ofrece un muy alto rendimiento.

Gunicorn fue elegido frente a otras opciones como servidor WSGI debido a su simplicidad en la configuración y compatibilidad con Django. También ofrece un excelente rendimiento en términos de manejo de múltiples solicitudes simultáneas.

5.1.5 Otras herramientas

Celery

Celery [13] es un sistema de gestión de tareas distribuido que permite ejecutar y programar tareas en segundo plano de manera eficiente. Es especialmente útil para tareas asincrónicas y de larga duración, como el procesamiento de imágenes.

Celery facilita la distribución de tareas a través de múltiples *workers*, lo que incrementa la escalabilidad y mejora la tolerancia a fallos del sistema. Su capacidad para ejecutar tareas de forma asíncrona permite liberar el proceso principal para continuar ejecutando otras tareas, lo que optimiza el rendimiento general de la aplicación. Este sistema es altamente extensible y puede integrarse con una variedad de *brokers* de mensajes, como Redis.

El funcionamiento de Celery se estructura en varios elementos clave, como los *workers*, que son los procesos encargados de ejecutar las tareas; los *brokers* de mensajes, que gestionan la cola de tareas y distribuyen estas entre los *workers*; y el *backend* de resultados, que almacena los resultados de las tareas para su posterior recuperación o análisis.

Comparado con alternativas como RQ (Redis Queue), Celery tiene una mejor integración con Django y soporte para manejar un mayor número de tareas distribuidas. Además, Celery ofrece reintentos automáticos en caso de fallos, lo que es muy importante cuando se trata de tareas laboriosas cómo puede llegar a ser el procesamiento de videos.

Redis

Redis [14] es una base de datos en memoria que actúa como un almacén de datos clave-valor (similar a un Hash). Una de las principales ventajas de Redis es su velocidad, ya que, al operar completamente en memoria, ofrece tiempos de acceso extremadamente rápidos, lo que lo hace ideal para aplicaciones que requieren acceso en tiempo real a los datos. Gracias a su velocidad y eficiencia, Redis es comúnmente utilizado como *broker* de mensajes para sistemas como Celery, lo que optimiza la gestión de colas de tareas y mejora el rendimiento general del sistema.

5.1.6 Docker y Docker Compose

Docker [15] es una plataforma de software que permite crear, desplegar y ejecutar aplicaciones dentro de contenedores. Los contenedores son entornos ligeros y portátiles que agrupan una aplicación y todas sus dependencias (librerías, herramientas del sistema, código, etc.) en un solo paquete. Esto garantiza que la aplicación se ejecute de manera consistente en cualquier entorno, ya sea en la computadora de un desarrollador, en un servidor o en la nube.

Docker Compose es una herramienta para definir y gestionar aplicaciones Docker que constan de varios contenedores interrelacionados. En lugar de iniciar manualmente cada contenedor con su respectiva configuración, Docker Compose te permite definir múltiples contenedores y sus relaciones (como bases de datos, *frontends*, *backends*, etc.) en un solo archivo de configuración llamado docker-compose.yml. Luego, con un solo comando (docker-compose up), se puede desplegar toda la aplicación.

Docker se eligió por la facilidad a la hora de llevar el sistema a distintos entornos ya sea *testing*, producción, o mismo en desarrollo, la capacidad de empaquetar de forma modular la aplicación es la permite esta gran portabilidad.

5.1.7 Subida de archivos

Subida simple [16]

La subida simple es el método más directo y básico para transferir archivos desde el cliente al servidor. En este enfoque, el archivo completo se carga en la memoria del cliente y se envía al servidor en una sola solicitud HTTP. El archivo viaja en el cuerpo de la solicitud, típicamente codificado como parte de un formulario HTML. Este método es adecuado para archivos de tamaño pequeño a mediano, ya que el archivo se carga por completo antes de enviarlo, lo que puede generar problemas con archivos grandes debido a las limitaciones de memoria del cliente o restricciones de tiempo de espera en el servidor. Una vez que el servidor

recibe el archivo, lo procesa y lo guarda en un sistema de almacenamiento (como el sistema de archivos local o un servicio de almacenamiento en la nube).

La subida simple es común en formularios de subida de archivos en aplicaciones web tradicionales, como para subir imágenes o documentos. Sin embargo, este método no ofrece soporte nativo para la reanudación de transferencias si ocurre un fallo durante la carga y tampoco maneja eficientemente archivos muy grandes.

Entonces, para nuestro proyecto no son válidas este tipo de subidas, ya que se van a subir videos con duración de varios minutos y a alta resolución, lo cual implica buscar otro método. A su vez, en el campo se tiene una conexión a internet inestable, por lo que se deben prever interrupciones en la subida de archivos.

Se decidió finalmente por usar Signed URLs en conjunto con *chunking* para la subida de archivos a S3 de acuerdo con los requerimientos del capítulo 3.

Subida directa a S3 con Signed URLs [17]

Con Signed URLs, el archivo se sube directamente a S3 desde el cliente, sin pasar por el servidor de la aplicación. Esto significa que el servidor de la aplicación no tiene que manejar grandes volúmenes de datos, lo cual es muy importante cuando se trata de videos en 4K que pueden ser de gran tamaño. De esta forma, se reduce la carga en el servidor.

Otra razón importante por la cual subir directo los archivos a Amazon S3 sin tener que pasar por el servidor es que Amazon tiene servidores distribuidos globalmente, lo que significa que la subida de archivos grandes (como videos) es mucho más rápida y eficiente al permitir que los clientes suban los archivos a los servidores más cercanos a ellos geográficamente.

Dependencia de conectividad para la subida de datos y opciones de optimización

Dado que la solución depende de una conexión estable para la subida de videos de alta resolución a la nube, esto puede representar un desafío en zonas rurales con conectividad limitada. Si bien el uso de URLs firmadas y fragmentación mejora la fiabilidad de la transferencia en condiciones de conectividad baja, se consideran opciones adicionales para reducir esta dependencia. La compresión de videos antes de la subida, por ejemplo, podría reducir significativamente el tamaño de los archivos y el tiempo necesario para completar la transferencia. También se plantea la posibilidad de realizar algún procesamiento inicial en el dispositivo local antes de la subida para optimizar aún más los datos transmitidos.

Uso de subida por fragmentación o *chunking*

Los videos en 4K pueden ser archivos extremadamente grandes, lo que haría que una subida tradicional (de un solo bloque) sea incluso imposible en conexiones de baja calidad. El *chunking* permite dividir el archivo en fragmentos más pequeños (de 5 MB), lo que permite:

- Si la conexión del cliente se interrumpe, la subida puede continuar desde el último *chunk* subido, en vez de comenzar de nuevo.

- Mejor manejo de memoria, como los archivos se dividen en fragmentos, ni el cliente ni el servidor tienen que manejar grandes volúmenes de datos de una sola vez, lo que reduce el uso de memoria y mejora la eficiencia.

- Mayor fiabilidad en áreas rurales, lo cual es un aspecto clave para esta aplicación, ya que los usuarios (productores rurales) suelen estar en áreas con conectividad limitada.

5.1.8 Storage *buckets* en S3

Amazon S3 [18] es un servicio de almacenamiento en la nube de AWS que permite almacenar grandes cantidades de datos de manera segura, escalable y accesible desde cualquier lugar. S3 es ampliamente utilizado para almacenar todo tipo de datos, desde archivos estáticos (imágenes, videos, documentos) hasta grandes volúmenes de datos generados por aplicaciones web o servicios de análisis.

En Amazon S3, los datos se organizan en *buckets*, que son esencialmente contenedores que agrupan los archivos dentro del servicio de almacenamiento. Cada archivo que subes a S3 se denomina objeto, y se almacena dentro de un *bucket*.

Aunque Amazon S3 es un sistema de almacenamiento plano (no tiene directorios como los sistemas de archivos tradicionales), se puede emular una estructura jerárquica organizando los objetos con prefijos. Por ejemplo, se podría tener objetos con nombres como "fotos/vacas/img001.jpg", lo que da la impresión de una estructura de carpetas, aunque S3 no maneja directorios de forma nativa.

Los costos de Amazon S3 se basan en múltiples factores, como el almacenamiento total utilizado, el número de solicitudes, el tráfico de salida, y las clases de almacenamiento que se eligen para los objetos. S3 también cobra por las transferencias de datos hacia Internet o entre regiones.

Amazon S3 se eligió con el objetivo de brindar un ambiente de almacenamiento de archivos que sea escalable y seguro. Ya que un pilar fundamental de la aplicación converge en el manejo de videos en 4K (archivos pesados) el tener una plataforma donde manejar los archivos que te brinde tanta seguridad, cómo costos de almacenamiento flexibles es muy importante.

Para optimizar la carga y reducir los costos de almacenamiento y procesamiento en la nube, se considera la compresión de videos antes de su subida. Esto permitiría un manejo más eficiente de los datos, especialmente en zonas con conectividad limitada.

5.1.9 Despliegue en AWS

Amazon Lightsail

Amazon Lightsail es un servicio de AWS diseñado para proporcionar una manera sencilla y rápida de lanzar servidores virtuales (instancias) con configuración predeterminada. Está orientado a desarrolladores, pequeñas empresas, o cualquier usuario que busque una solución de *hosting* sencilla sin tener que lidiar con la complejidad y la amplia gama de opciones que ofrece la infraestructura completa de AWS. Es ideal para usuarios que necesitan desplegar aplicaciones básicas rápidamente, como sitios web, blogs, o pequeños proyectos de software. Lightsail provee 8 opciones de instancias cuyas características son fijas, y se detallan en la siguiente imagen (ver imagen 9).

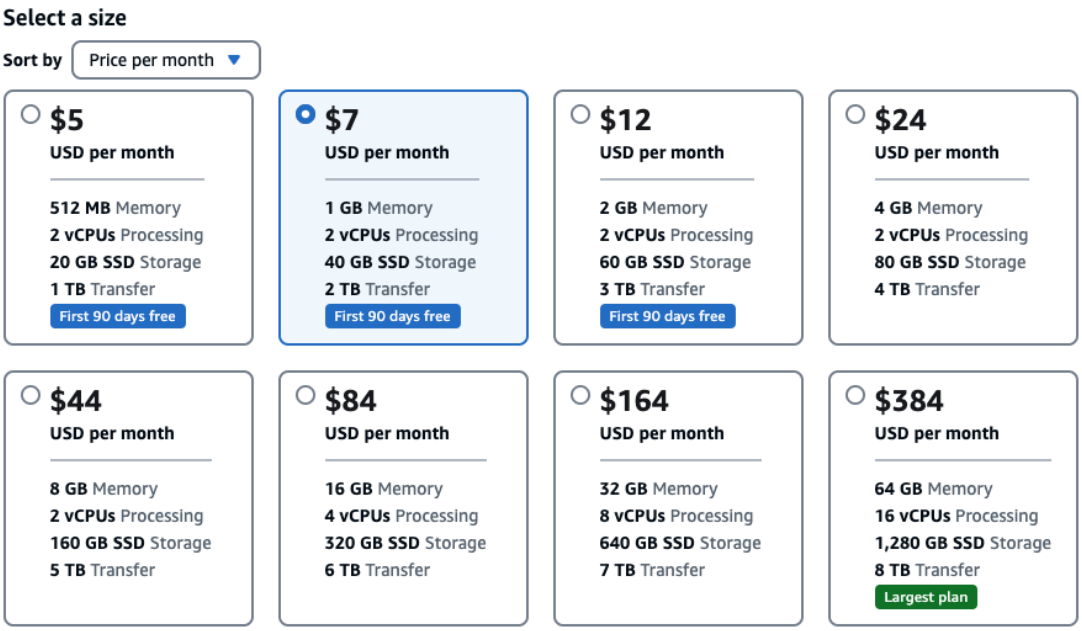


Imagen 9: Detalle de instancias AWS Lightsail.

Amazon EC2

Amazon Elastic Compute Cloud (EC2) es un servicio de AWS que proporciona capacidad de procesamiento escalable en la nube. A diferencia de Lightsail, EC2 está diseñado para usuarios que necesitan control y flexibilidad total sobre su infraestructura en la nube. Es una plataforma robusta que ofrece un alto grado de personalización en la configuración de servidores y es utilizada principalmente por empresas y desarrolladores avanzados para aplicaciones que requieren un gran rendimiento, escalabilidad y control.

Amazon EC2 es la plataforma que se utilizará para el *hosting* de la aplicación. EC2 proporciona la infraestructura necesaria para manejar alto tráfico de usuarios y el procesamiento de video. A pesar de sus altos costos para una etapa temprana de la aplicación, brinda todas las funcionalidades necesarias para poder ejecutarla correctamente, ya que uno puede armar el servidor conforme a sus necesidades. EC2 permite la configuración de instancias inelásticas, esto significa que permite que los servidores escalen según la carga de trabajo, lo que garantiza que la infraestructura pueda adaptarse a picos de tráfico sin intervención. En este caso una de las características más importantes que nos llevó a esta decisión por sobre Amazon Lightsail, es la posibilidad de poder armar nuestro propio servidor, decaemos finalmente por la necesidad de usar tarjetas gráficas dedicadas en nuestro servidor cosa que no se puede utilizar en Lightsail, esto es muy importante para acelerar el procesamiento de los videos sin ocupar espacio de RAM, lo que puede hacer caer la infraestructura en diversas ocasiones.

5.2 Arquitectura

En este apartado se analiza la arquitectura del sistema, partiendo desde un enfoque global hacia lo más específico. Se detalla el diagrama de flujo de la aplicación, y cómo sería un caso de uso normal.

5.2.1 Arquitectura del sistema

El sistema ha sido diseñado para cumplir con los requerimientos detallados en el capítulo 3. La arquitectura que se detalla a continuación soporta una plataforma de gestión ganadera que puede ser escalable, modular y confiable. La misma está diseñada para cumplir con 3 grandes roles: Gestión de usuarios; Gestión de establecimientos, lotes y pasturas y Procesamiento de conteo de ganado (ver imagen 10).

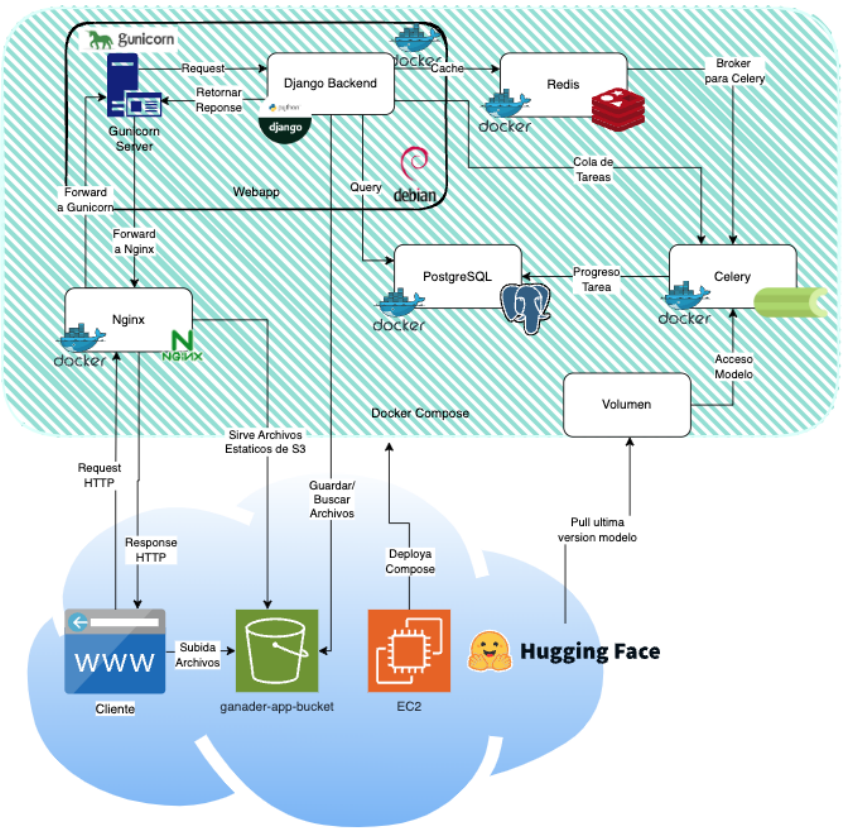


Imagen 10 - Arquitectura del sistema.

5.2.2 Flujo de trabajo

A continuación, se detalla el flujo de trabajo de un usuario que va a realizar un conteo de ganado, este es el caso de uso más común para los usuarios en la plataforma. Se describen tanto las acciones del usuario, cómo lo que reporta el *frontend* y cómo las aplicaciones que se detallan en la arquitectura interactúan entre sí para poder brindarle la respuesta al usuario final.

1. Interacción del usuario (cliente)

- El usuario conecta con la interfaz web a través de un navegador.
- Desde la interfaz, el usuario se dirige hacia su establecimiento, y luego al lote correspondiente.
- Una vez en el lote correspondiente el usuario inicia la subida del archivo (video de conteo).

2. Recepción y almacenamiento del video (Nginx + Amazon S3)

- Nginx, recibe la solicitud HTTP del cliente para subir el video.

- El cliente luego solicita al *backend* las URLs firmadas, lo que permite subir los fragmentos a S3 sin tener que pasar por el servidor.

- El cliente va a partir el archivo en la cantidad de fragmentos que sean necesarios de 5mb cada uno.

- Usando las URLs firmadas los fragmentos del video se suben y almacenan en un *bucket* de Amazon S3 llamado "ganader-app-bucket" aquí cada segmento se sube por separado, pero luego se une el archivo en S3.

- Durante la subida, el cliente hace seguimiento del proceso desplegando en el *frontend* una barra de progreso.

- En caso de fallar un fragmento del archivo, sólo será necesario que el cliente envíe ese fragmento, mejorando así la fiabilidad en condiciones de conectividad mala.

3. Gestión de tareas asíncronas (Django *backend* + Celery)

- Una vez que el video ha sido correctamente subido a S3, el *backend* de Django dispara una tarea para hacer el procesamiento del conteo de ganado. En vez de procesar el video en el *backend*, lo cual podría bloquear el sistema, se delega la tarea a un sistema de tareas asíncronas usando Celery.

- Redis siendo el *broker* para las tareas recibe el pedido y genera una cola esperando que uno de los *workers* de Celery la procese.

4. Procesamiento del video (Celery + Hugging Face)

- Uno de los *workers* de Celery toma la tarea de la cola de Redis y empieza el procesamiento.

- El *worker* utiliza el último modelo de *ML* que se actualizó desde Hugging Face para procesar el video. Este modelo es capaz de detectar y contar automáticamente las cabezas de ganado visibles en el video.

- El modelo procesará los *frames* del video para identificar las cabezas de ganado, ejecutando la inferencia en la GPU de la instancia de EC2.

- También se utilizará un modelo de *tracking*, que seguirá las instancias a través de los *frames*, logrando contar así únicamente una vez cada cabeza de ganado a lo largo del video.

- Mientras se realiza el procesamiento, Celery envía actualizaciones de progreso a la base de datos para desplegar una barra de progreso sobre el procesamiento del video.

5. Almacenamiento de resultados (PostgreSQL)

- Una vez completado el procesamiento del video, Celery envía los resultados del conteo de ganado al *backend* Django.

- Los resultados se almacenan en la base de datos PostgreSQL, vinculados al lote de ganado correspondiente y al video que fue subido. Esto permite un historial completo de conteos de ganado por video y por lote.

- El *backend* Django luego notifica al usuario (a través de la interfaz) que el procesamiento ha sido completado.

6. Visualización de resultados (*frontend* + Nginx)

- El usuario desde la plataforma podrá visualizar los resultados del último conteo procesado así mismo cómo el historial de conteos. Estos datos se toman de la base PostgreSQL.

- Además de los resultados del conteo, el usuario puede acceder al video procesado, visualizando así el modelo en acción utilizando *Bounding Boxes* y un contador sobre el video. Nginx de esta forma sirve los archivos estáticos desde S3 para que el usuario lo pueda ver directamente desde el navegador.

5.3 Etapa de desarrollo de aplicación web

La fase de desarrollo fue crucial para convertir los requisitos funcionales en un sistema de software funcional y escalable. Dado que el equipo tenía experiencia en programación, pero carecía de conocimiento en ciertas tecnologías, como Django, Celery, y AWS, se realizó un proceso de aprendizaje estructurado para adquirir las habilidades necesarias.

En esta sección se detallan los aspectos técnicos del desarrollo, incluyendo el *stack* tecnológico utilizado, la organización del código, las principales funcionalidades implementadas, y el proceso de aprendizaje llevado a cabo para dominar las nuevas tecnologías. Además, se describe el despliegue en la nube, asegurando que el sistema sea escalable y preparado para futuras expansiones.

5.3.1 Proceso de aprendizaje de Django

Al comenzar el desarrollo del proyecto, el equipo necesitaba adquirir los fundamentos de Django, ya que no contaban con experiencia previa en el uso de este *framework*. El primer paso en este proceso fue realizar un tutorial de YouTube [19], que proporcionaba una base sólida para entender los conceptos básicos de Django y comenzar a trabajar en una aplicación web "*template*" que sirvió como punto de partida para el desarrollo del sistema.

El tutorial comienza enseñando cómo configurar el entorno de desarrollo, instalando Django, y luego guiando al usuario a crear un proyecto básico, para poder comenzar a entender la estructura de un proyecto Django. Luego se abordó las configuraciones iniciales, cómo editar el archivo de configuración, explicando cómo manejar la base de datos, el uso de las plantillas y ubicar los archivos estáticos.

Entrando un poco en el patrón *MTV* se demostraba el uso de las vistas y su funcionalidad. Las vistas son funciones que reciben una solicitud que luego ejecuta la lógica del *backend* para devolver una respuesta en forma de una plantilla de HTML. Además, se explicaba cómo asociar estas vistas con rutas específicas, permitiendo que las vistas fueran accesibles desde el navegador y creando las rutas necesarias para diferentes páginas de la aplicación.

En el sistema de plantillas, se introducían las plantillas o "*templates*" HTML para generar contenido dinámico en las páginas web, lo que permitía mostrar información consultada de la base de datos. También se explicaba el concepto de herencia de plantillas, que permitía reutilizar partes de una plantilla (como el *header* y *footer*) en varias páginas.

El tutorial también abordaba la creación y uso de modelos (*models*). Los modelos en Django son clases que representan las tablas de la base de datos. Se introduce el ORM (Object-Relational Mapping) de Django, explicando cómo el mismo interactúa directamente con la base de datos mediante operaciones CRUD (crear, leer, actualizar y eliminar registros).

Finalmente, el tutorial cubría la creación de formularios y validaciones, mostrando cómo crear formularios tanto manualmente como usando los formularios automáticos basados en modelos (*ModelForm*). Además, se enseñaba cómo realizar validaciones de los datos ingresados en los formularios, asegurando que los datos cumplieran con reglas específicas, como verificar que un campo sea obligatorio o que un correo electrónico tenga el formato adecuado.

5.3.2 Estado inicial de la aplicación

En la fase inicial del desarrollo, se implementaron dos aplicaciones clave: la aplicación de usuarios y la aplicación de blog, ambas respaldadas por la base de datos SQLite que viene preconfigurada en Django.

En la estructura de Django se definen aplicaciones, que es la mayor unidad en la cual se puede dividir la estructura. Para cada aplicación se van a definir vistas, modelos, plantillas y el ruteo.

En la aplicación de usuarios, se definen formularios y vistas básicas para el registro e inicio de sesión de los usuarios. También se habilitó una funcionalidad para que los usuarios pudieran cambiar su contraseña, lo cual se maneja a través del sistema integrado de Django que incluye una vista para restablecer contraseñas enviando un enlace por correo electrónico.

Por otro lado, en la aplicación de blog, los usuarios una vez iniciados podrían compartir publicaciones mediante formularios sencillos. Estas publicaciones podrían ser vistas por otros usuarios, ya que la vista recupera la información de cada publicación de la base de datos a través del modelo.

5.3.3 Integración de contenedores de Docker

Después de implementar las funcionalidades básicas de la aplicación (gestión de usuarios y blog), el siguiente paso fue la integración con Docker para mejorar la modularidad y escalabilidad del proyecto. La idea es dockerizar cada servicio y permitir una configuración más flexible y fácil de desplegar.

Celery fue implementado en un contenedor específico para manejar las tareas asíncronas de la aplicación, como el procesamiento de videos. Se configuró para interactuar con el *backend* Django y delegar tareas largas a través de colas de mensajes. Redis actuaba como

intermediario entre Celery y Django, gestionando las tareas pendientes y coordinando su ejecución en segundo plano, lo que resultaba esencial para manejar el procesamiento intensivo requerido por la aplicación.

Se integró el servidor Gunicorn como servidor WSGI para servir la aplicación Django. Gunicorn se encargaba de manejar las solicitudes HTTP y facilitaba la comunicación entre los usuarios y la aplicación. Su configuración se realizó dentro del contenedor del *backend* donde se encuentra la aplicación en sí, basado en una imagen de Alpine.

En cuanto a Nginx, se añadió como un contenedor separado para actuar como proxy inverso. Nginx distribuye el tráfico entrante entre los servicios de *backend*, sirviendo como el primer punto de contacto para las solicitudes HTTP de los usuarios y redirigiéndolas a Gunicorn. Además de esta función, Nginx también maneja los archivos estáticos, como CSS, JavaScript e imágenes, descargando esta responsabilidad de Gunicorn y optimizando el rendimiento del servidor.

Para mejorar la capacidad de manejo de datos, se agregó un contenedor para PostgreSQL, reemplazando la base de datos SQLite utilizada en las primeras etapas del proyecto. Django fue configurado para conectarse a PostgreSQL como su base de datos principal, lo que implicó ajustar las configuraciones para incluir las credenciales y parámetros de conexión correctos.

Finalmente, se utilizó Docker Compose para orquestar todos los contenedores del proyecto, incluyendo Django, Celery, Redis, Gunicorn, Nginx, y PostgreSQL. La principal ventaja de Docker Compose fue permitir la definición y gestión de todos estos servicios en conjunto, lo que simplificó tanto su configuración como su ejecución. Con un solo comando, todos los servicios podían levantarse y conectarse entre sí, facilitando así el despliegue de la aplicación en distintos entornos.

5.3.4 Implementación de funcionalidades: establecimientos, pasturas y lotes

Se prosiguió con la implementación de las funcionalidades clave para la gestión del ganado. Se comenzó con el pilar central, que es la creación y manejo de establecimientos. El usuario podrá crear establecimientos rurales y gestionarlos en la plataforma. El modelo de establecimientos contaba con atributos clave como el nombre del establecimiento, que es un campo de texto, y el propietario, que se asigna automáticamente al usuario que crea el establecimiento. Además, se incluyó una funcionalidad para que el propietario pueda agregar miembros al establecimiento, lo que permite a otros usuarios ver y gestionar el contenido relacionado, como las pasturas y los lotes de ganado. Para garantizar la seguridad de los datos, se implementó una gestión de permisos, asegurando que solo los miembros del establecimiento puedan acceder y modificar su contenido.

Dentro de cada establecimiento, se agregó la opción de crear pasturas, que representan divisiones de tierra donde se asignan los lotes de ganado. El modelo de pasturas incluye un nombre que las identifica dentro del establecimiento y un campo para definir su tamaño en hectáreas. Cada pastura está asociada a un establecimiento específico en el momento de su creación, estableciéndose una relación uno a muchos entre los establecimientos y las pasturas, lo que permite que un establecimiento tenga múltiples pasturas para organizar de manera eficiente la gestión de sus terrenos.

También se implementó la funcionalidad para crear lotes de ganado. El modelo de lotes incluye el número de cabezas de ganado, un campo de comentarios donde los usuarios pueden

agregar notas relacionadas con el lote, la fecha de creación, y la categoría de ganado (sean lotes de terneros, novillos, etc). Además, los lotes pueden ser asignados dinámicamente a una pastura o desasignados según sea necesario, permitiendo a los usuarios gestionar de manera flexible los lotes dentro de sus pasturas. La interfaz de usuario permite visualizar, modificar y asignar los lotes.

5.3.5 Creación de la aplicación de conteo

Originalmente, el modelo de lotes incluía un campo manual para ingresar la cantidad de ganado. Sin embargo, esta funcionalidad fue reemplazada por una aplicación de conteo más avanzada, que permite realizar conteos automáticos a partir de vídeos y también aceptar conteos manuales. El campo de "cantidad de ganado" se sustituyó por un objeto de conteo que puede almacenar tanto registros manuales como resultados obtenidos a través del procesamiento de videos. Ahora, cada lote de ganado puede tener múltiples registros de conteo asociados, lo que permite llevar un historial de conteos incluyendo la fecha del conteo, la pastura en la que estaba el ganado cuando se realizó el conteo, el método ya sea manual o a través de procesamiento de video y finalmente un agregado de comentarios.

La integración del modelo de *machine learning* fue clave para el procesamiento de vídeos subidos por los usuarios. Este modelo, entrenado con datos específicos de ganado, utiliza técnicas de *computer vision* para detectar y contar automáticamente las cabezas de ganado en los videos. El *pipeline* de procesamiento comienza con la subida del video por parte del usuario, seguido por la delegación del procesamiento a una cola de tareas manejada por Celery y Redis, lo que permite que el sistema procese los videos en segundo plano sin afectar el rendimiento.

5.3.6 Implementación de la barra de progreso para tareas

Dado que algunas tareas, como el procesamiento de videos para el conteo de ganado, pueden tomar un tiempo considerable, se implementó una barra de progreso para mejorar la experiencia del usuario. Esto permite a los usuarios visualizar el estado de las tareas largas, reduciendo la incertidumbre sobre los tiempos de espera.

Para implementar esta funcionalidad, se utilizó la librería "celery-progress", diseñada específicamente para crear barras de progreso en tareas gestionadas por Celery. Esta herramienta funciona creando una tabla en la base de datos donde se almacena el estado y progreso de cada tarea, incluyendo detalles como el porcentaje completado, el estado actual (pendiente, en progreso, completado), y la tarea específica en ejecución. A medida que Celery avanza en la tarea, los registros en la tabla se actualizan en tiempo real, reflejando el progreso.

En el *frontend*, se integró código en JavaScript que interactúa con el *backend* para consultar el estado actual de las tareas en curso. Este código toma los datos almacenados en la tabla de Celery y los presenta como una barra de progreso visual. Mediante *fetch API*, el *frontend* solicita periódicamente el estado de la tarea, actualizando la barra de progreso de manera dinámica hasta que la tarea se complete, cómo vemos en la siguiente imagen (ver imagen 11)

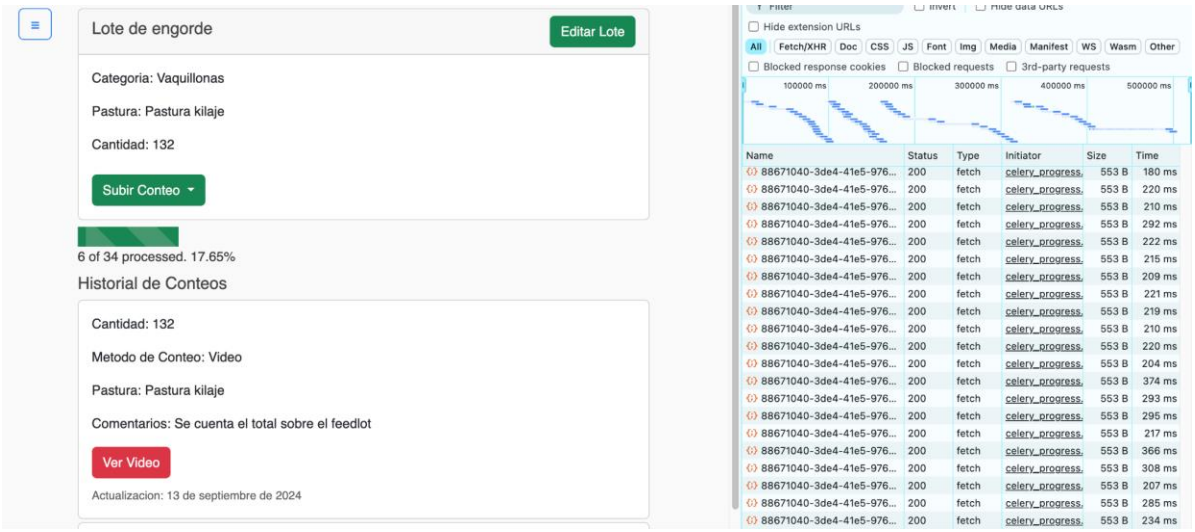


Imagen 11 - Barra de progreso tarea y fetch API.

5.3.7 Implementación inicial de subida de archivos a S3

La primera implementación de la funcionalidad de subida de archivos a Amazon S3 fue diseñada específicamente para manejar videos subidos por los usuarios. En esta fase inicial, los archivos eran subidos directamente desde el *backend* de la aplicación, sin utilizar URLs firmados, lo que significa que los archivos pasaban por el servidor antes de ser almacenados en el *bucket* de S3.

Para llevar a cabo la subida sencilla, se configuró un *bucket* en Amazon S3 destinado a almacenar los archivos de vídeo. El *backend* de la aplicación estaba configurado con las credenciales de AWS, lo que permitía que la aplicación interactuara directamente con S3 y gestionará la subida de archivos. El flujo de trabajo consistía en que, cuando el usuario seleccionaba un archivo para subir, este era enviado al *backend* de la aplicación, el cual, usando las credenciales de AWS, subía el archivo al *bucket* de S3. Después de una subida exitosa, el *backend* almacenaba la URL del archivo en la base de datos, para que pueda ser referenciado más adelante. En esta primera versión, no se incluía una barra de progreso para hacer seguimiento de la subida, ya que la funcionalidad estaba limitada a subidas simples.

5.3.8 Visualización del video procesado desde el historial de conteos

Se añadió una nueva funcionalidad en el historial de conteo que permite a los usuarios visualizar los vídeos procesados. En el historial, cuando un conteo de ganado ha sido realizado mediante el método de procesamiento de video, aparece un botón de reproducción que dirige a una nueva vista donde el usuario puede ver el video procesado. Esta mejora facilita a los usuarios revisar los conteos automáticos realizados a través de videos.

Sin embargo, se detectó un problema de reproducción web en esta nueva funcionalidad. Los videos procesados inicialmente no podían ser reproducidos correctamente en muchos navegadores debido a problemas de compatibilidad de códecs. Muchos navegadores modernos no soportan los códecs utilizados en los archivos generados por el sistema durante el procesamiento de conteo.

Para resolver este inconveniente, se implementó una solución utilizando el paquete ffmpeg para reencodear los videos procesados. El objetivo de este reencodeo fue convertir los

archivos de video a un formato más compatible con los navegadores modernos, MP4 con el códec H.264. Durante el proceso de reencodeo, se ajustaron parámetros clave como la resolución y los FPS para garantizar la compatibilidad y calidad del video, asegurando que los usuarios pudieran reproducir los videos sin problemas en cualquier navegador (ver imagen 12).

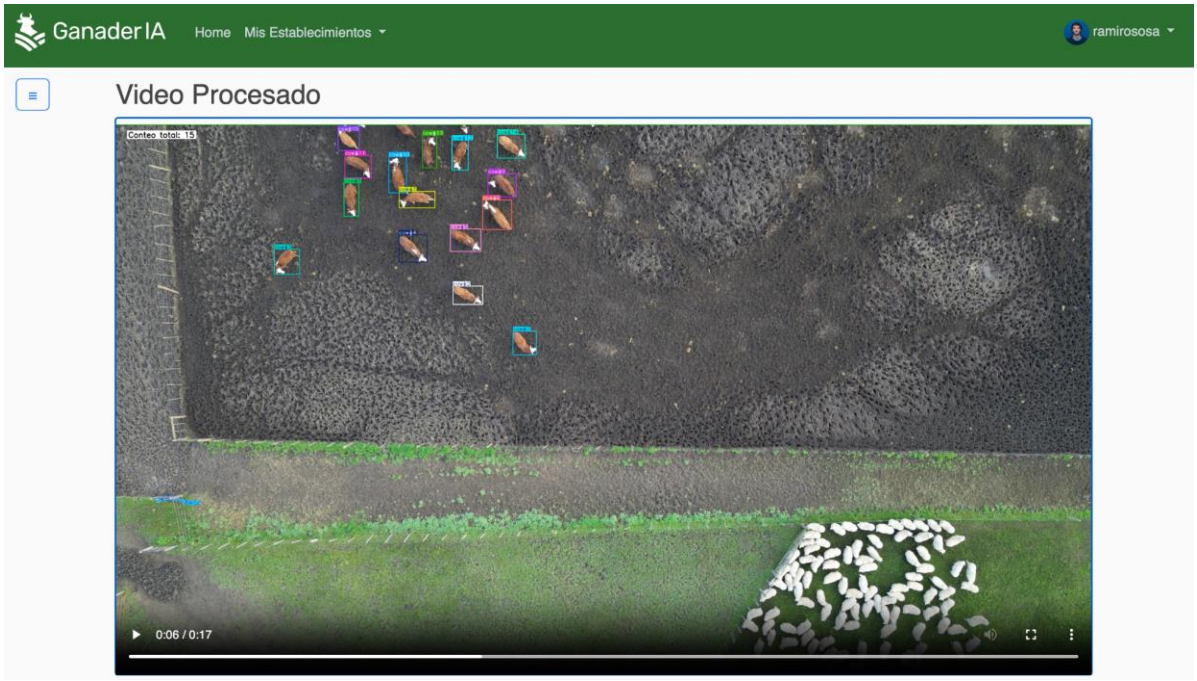


Imagen 12 - Vista de video procesado.

5.3.9 Subida de archivos por fragmentación y URLs firmadas

Después de implementar la subida sencilla, surgieron problemas al manejar archivos grandes, como videos en 4K, debido a los límites de tamaño y tiempos de espera. Para abordar estos problemas, se implementó la subida de archivos por fragmentación, una técnica que permite dividir los archivos en partes más pequeñas y subirlas de forma independiente. Esto resolvió los problemas de *timeout* o interrupciones durante la subida, y en caso de una falla en la conexión, solo es necesario reintentar los fragmentos que fallaron, en lugar de reiniciar todo el proceso.

La implementación de la subida de archivos por fragmentación, parte el archivo en fragmentos de 5MB del lado del cliente, en el *backend* se genera una URL firmada para conectar la subida directamente a S3. Cada fragmento se comienza a subir de forma independiente utilizando un *script* de Javascript desde el *frontend*. Una vez subidos todos los fragmentos, el *backend* los ensambla para formar el archivo completo.

Entre los beneficios de la implementación, se destaca la resiliencia del sistema. Si alguna parte de la subida falla, el proceso puede reanudarse desde el último fragmento subido correctamente, en lugar de reiniciar la subida completa. Además, esta solución permitió manejar archivos grandes sin comprometer el rendimiento de la aplicación, ni la experiencia del usuario.

5.4 Despliegue aplicación web

En primer lugar, se había decidido hacer un despliegue de la aplicación web en una instancia de Lightsail. Se calculó que en el ambiente de desarrollo la aplicación utiliza un poco menos de 3.5GB de RAM, información proporcionada por Docker Desktop. Se eligió una instancia de Lightsail basada en Debían por su compatibilidad con la imagen base del contenedor del *backend*, y con las siguientes características de *hardware*: 4GB de RAM por las razones ya mencionadas, 80GB SSD de disco para mantener el *filesystem* y archivos pesados que puede tomar de S3 para procesarlos y 2 vCPUs (ver imagen 13).

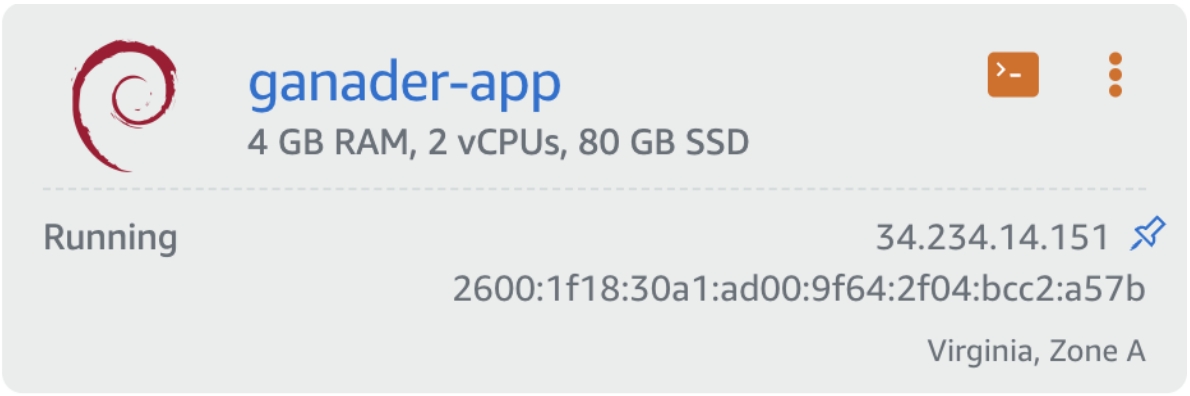


Imagen 13 - Deployment en AWS.

De todas formas, esta instancia se descontinuó y se migró a EC2. Esto ocurrió dado a que la infraestructura no era la suficiente como para mantener las operaciones de la aplicación. Principalmente se observó que la inferencia de cada frame del video tardaba alrededor de 2 segundos. Este tiempo para videos largos puede escalar muy rápidamente haciendo la operación de la aplicación insuficiente, también en videos muy largos, el procesamiento de este comenzaba a ocupar RAM al 100% lo que ocasiona el *timeout* y *crash* de la aplicación (ver imágenes 14 y 15).

Imagen 14 - Timeout aplicación web.

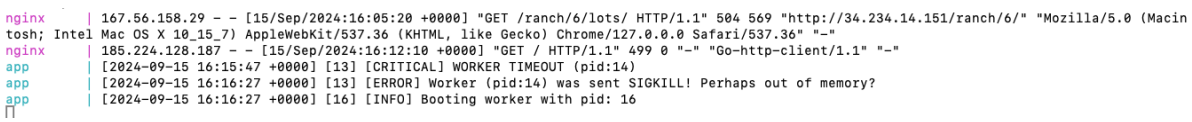


Imagen 15 - Logs de docker indicando el fin del proceso, servidor sin memoria.

Cómo se mencionó anteriormente se finalizó optando por utilizar EC2 y armar un servidor adecuado para el procesamiento que requiere la aplicación.

Para este servidor se utilizó la instancia “g4dn.xlarge” que cuenta con 16GB de RAM y una tarjeta gráfica Nvidia T4. También se tuvo en cuenta de utilizar discos en particiones teniendo un disco dedicado para el booteo de la aplicación y su *filesystem* y otro disco de 128GB que

funcione cómo *storage cache* para guardar los videos momentáneamente durante el procesamiento de este, para luego ser eliminados.

Este servidor tiene un costo aproximado de US$0,526 por hora y entre US$380 y US$400 por mes. Se observó que la instancia no es la más económica, pero se cree que es la adecuada para el contexto de la aplicación, y se tiene en cuenta que el equipo tiene créditos de AWS brindados por el CIE ya que se trata del desarrollo del producto de un emprendimiento.

Con esta infraestructura la aplicación tiene buena *performance* finalmente, el procesamiento de los videos es veloz y se cumple con todos los requerimientos funcionales y no funcionales del desarrollo cómo se detalla en el siguiente capítulo.

5.5 *Testing* aplicación web

Ahora, se desarrolla el *testing* de la aplicación web, mostrando el flujo realizado dentro de la página y los resultados que se obtienen.

Se demuestra con imágenes y explicaciones, cómo se satisfacen los requerimientos para la aplicación web planteados en el apartado 3.3 Requerimientos de testing.

- "Verificar que el registro, inicio de sesión y cambio de contraseña funcione correctamente."

En la siguiente serie de imágenes vamos a poder ver el formulario de registro, el formulario de inicio de sesión y cómo luego el usuario entra a la plataforma, demostrando que estas dos funcionalidades cumplen con los requerimientos (ver imágenes 16, 17 y 18).

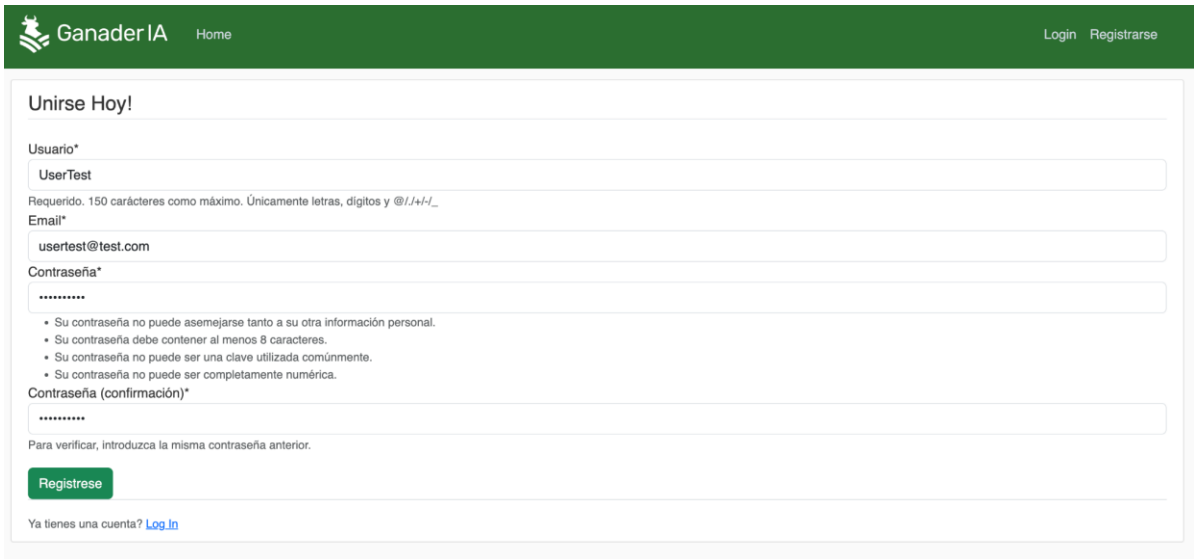


Imagen 16 - Formulario de registro de usuario.

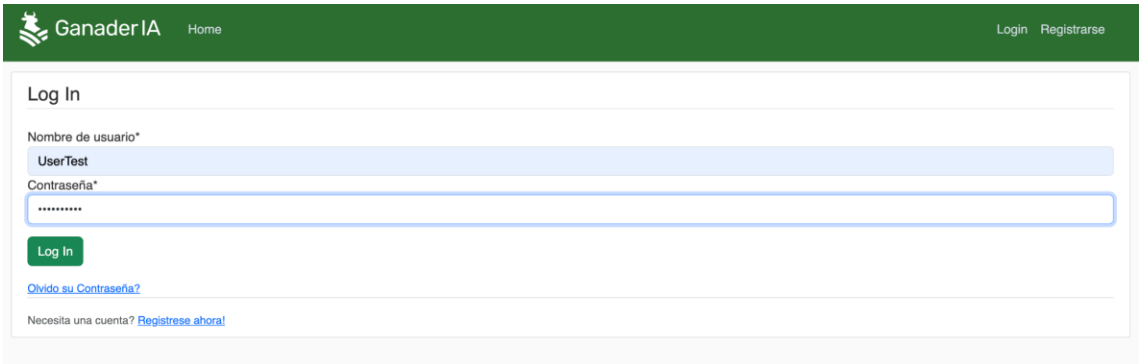


Imagen 17 - Formulario de inicio de sesión.

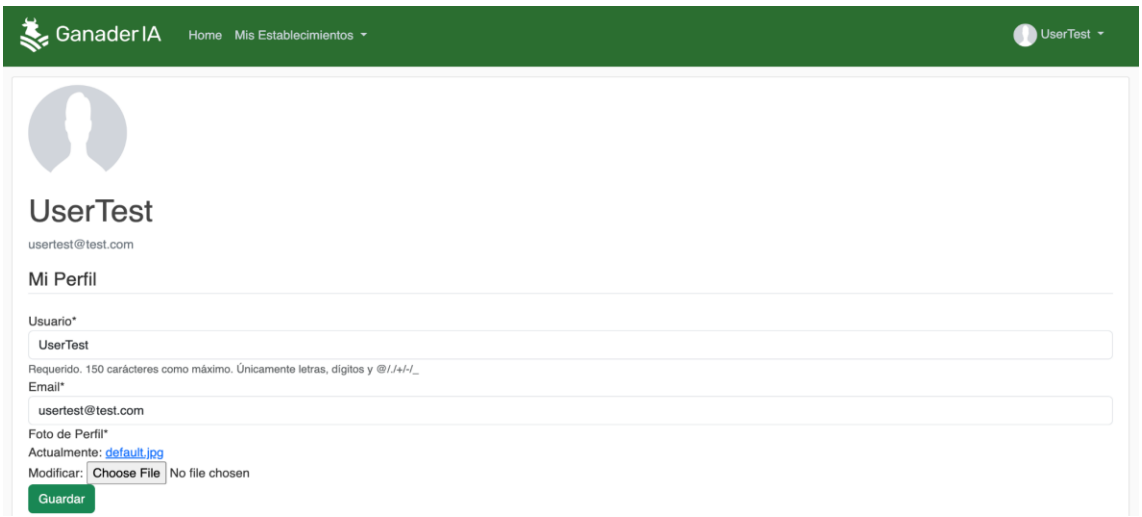


Imagen 18 - Perfil de usuario iniciado.

Luego el usuario decide cambiar su contraseña, recibe un correo, accede al *link* y se despliega el formulario para hacer el reseteo de su contraseña. Lo cual funciona correctamente como podemos ver en esta última imagen (ver imágenes 19, 20 y 21).

Imagen 19 - Recibo de correo desde servidor para cambio de contraseña.

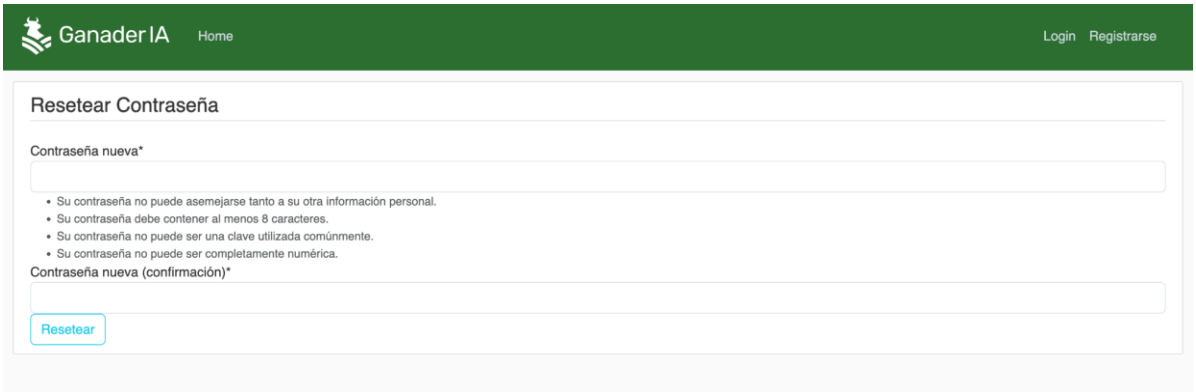


Imagen 20 - Formulario de reset de contraseña.

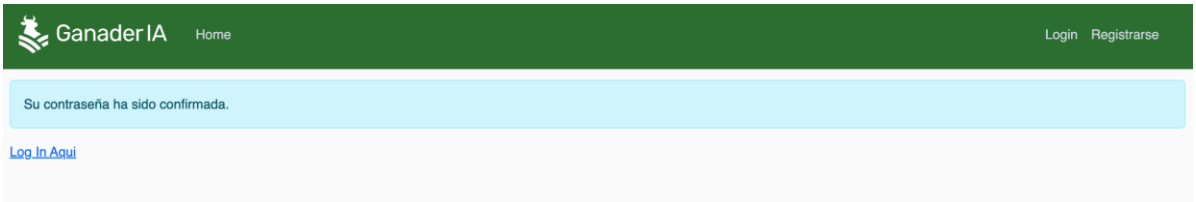


Imagen 21 - Mensaje de que la contraseña ha sido cambiada exitosamente.

- "Probar que los usuarios puedan crear y gestionar establecimientos, lotes, y pasturas de manera adecuada."

En la siguiente pantalla podemos ver cómo el usuario creó un establecimiento, y dentro de ese establecimiento creó un lote y dos pasturas, y asignó ese lote a una de las pasturas, demostrando así un correcto funcionamiento de la gestión del establecimiento rural (ver imagen 22).

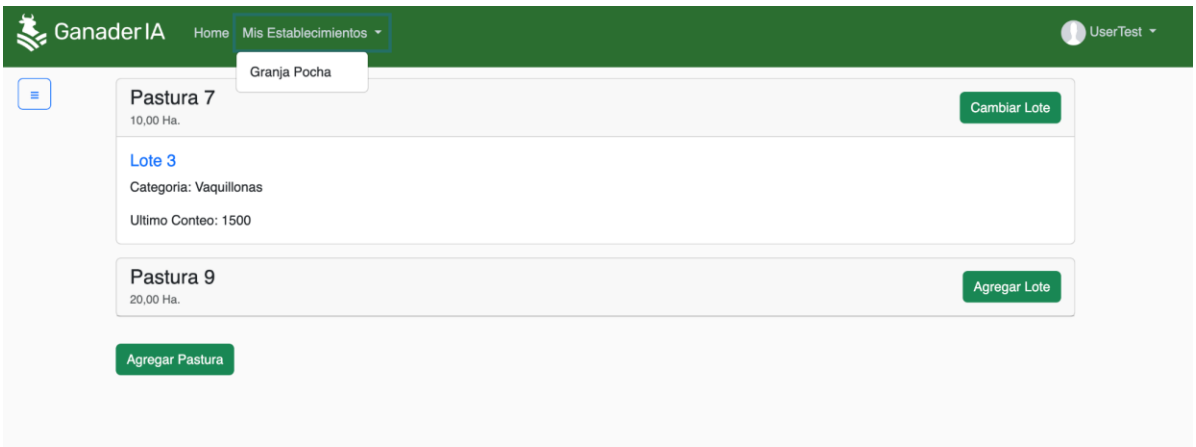


Imagen 22 - Gestión de pasturas.

- "Asegurar que los videos en 4K se puedan subir sin interrupciones."

Se puede ver en la siguiente imagen utilizando las Herramientas de Desarrollo de Chrome en la pestaña de "*Network*" cómo el servidor maneja la subida del archivo a S3 fragmentando el video, y cómo se recibe la respuesta por parte del servidor una vez que la subida es completada (ver imagen 23).

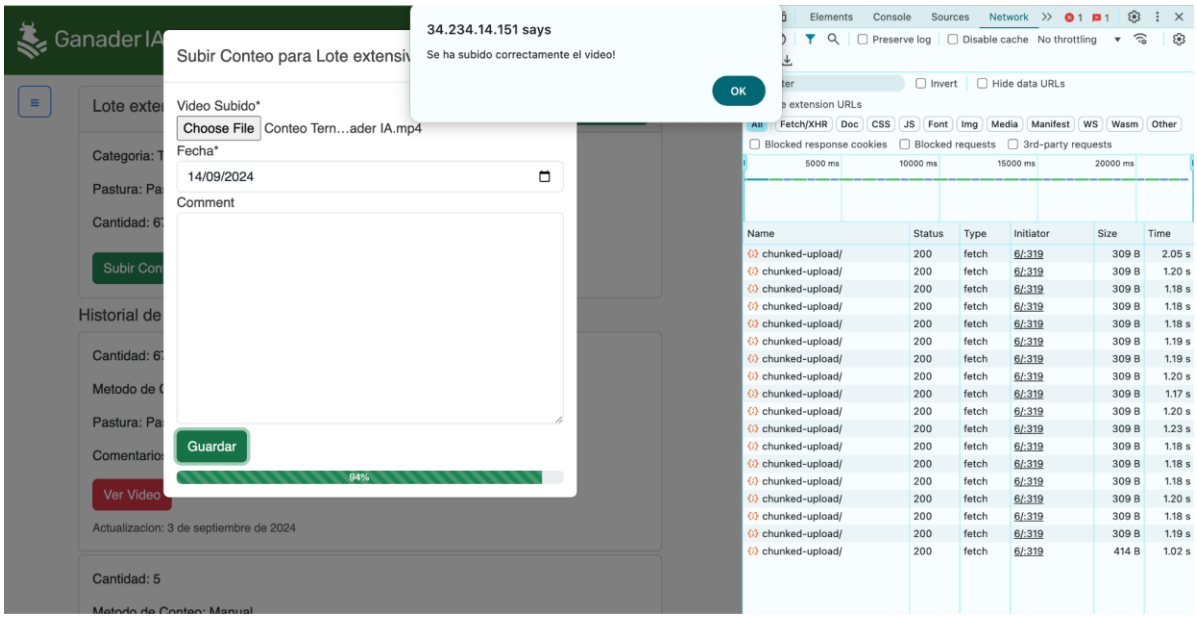


Imagen 23 - Video subido correctamente.

- "Probar que el sistema maneje correctamente videos que no cumplen con los requisitos (formatos inválidos, resoluciones diferentes, etc.)."

El sistema se ajusta a diferentes formatos de vídeo y extrae los parámetros correspondientes a la resolución para ajustar la región en la que se debe contar vacas al ancho y altura de la imagen. En caso de ingresar un formato inválido, por ejemplo subir un video en tipo de archivo PDF, sale una alerta de error.

- "Verificar que una vez subido el video, el sistema lo procese correctamente utilizando el modelo de *machine learning*."

El correcto procesamiento del modelo de *machine learning* lo podemos visualizar en la misma aplicación web, ya que una vez que se procesa el video, podemos ver el resultado del video procesado que despliega todos los *bounding boxes* que encierran a cada cabeza de ganado y un contador que despliega cuantas vacas contó en el video (ver imagen 24).

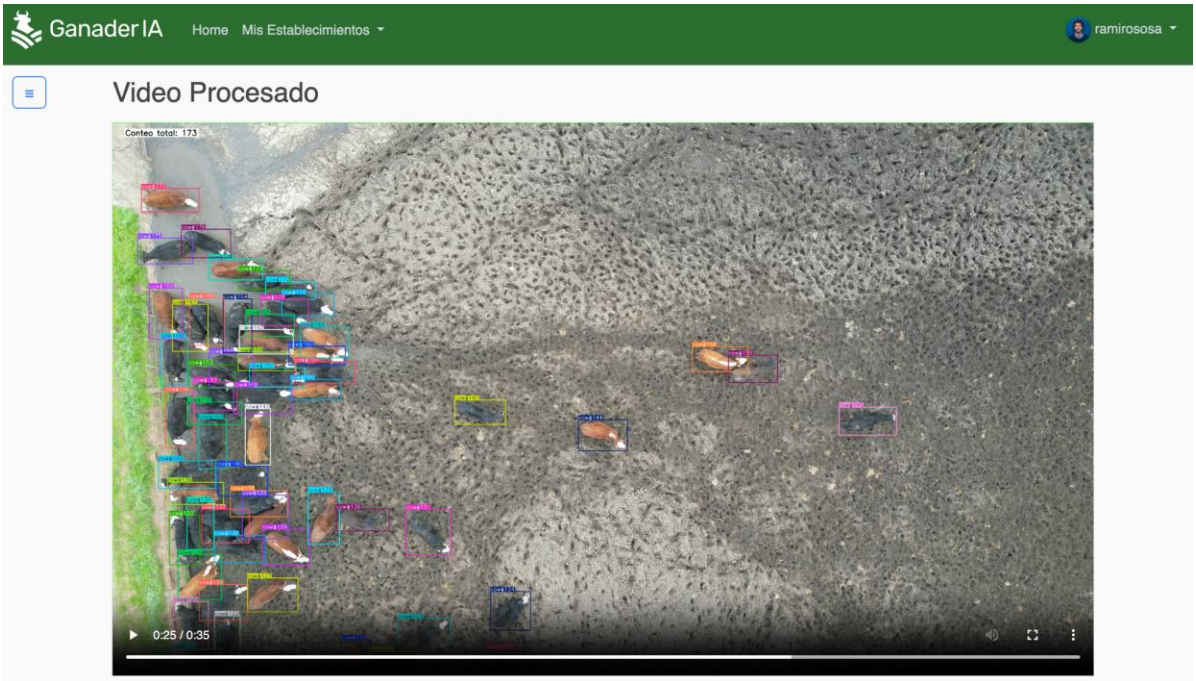


Imagen 24 - Vista con despliegue del video procesado elegido.

- "Probar que el conteo de ganado y su historial se refleje adecuadamente en la interfaz de usuario."

Se puede ver cómo se despliega el historial en la siguiente imagen. El 26 de agosto se creó el lote que indicaba manualmente que contenía 250 cabezas de ganado. El 4 de setiembre se ingresó un video y tras ser procesado se detectó una baja en el conteo de este lote, desplegando las diferencias en el historial, y pudiendo auditar el conteo a través del botón de ver video, para visualizar que todo se haya hecho correctamente (ver imagen 25).

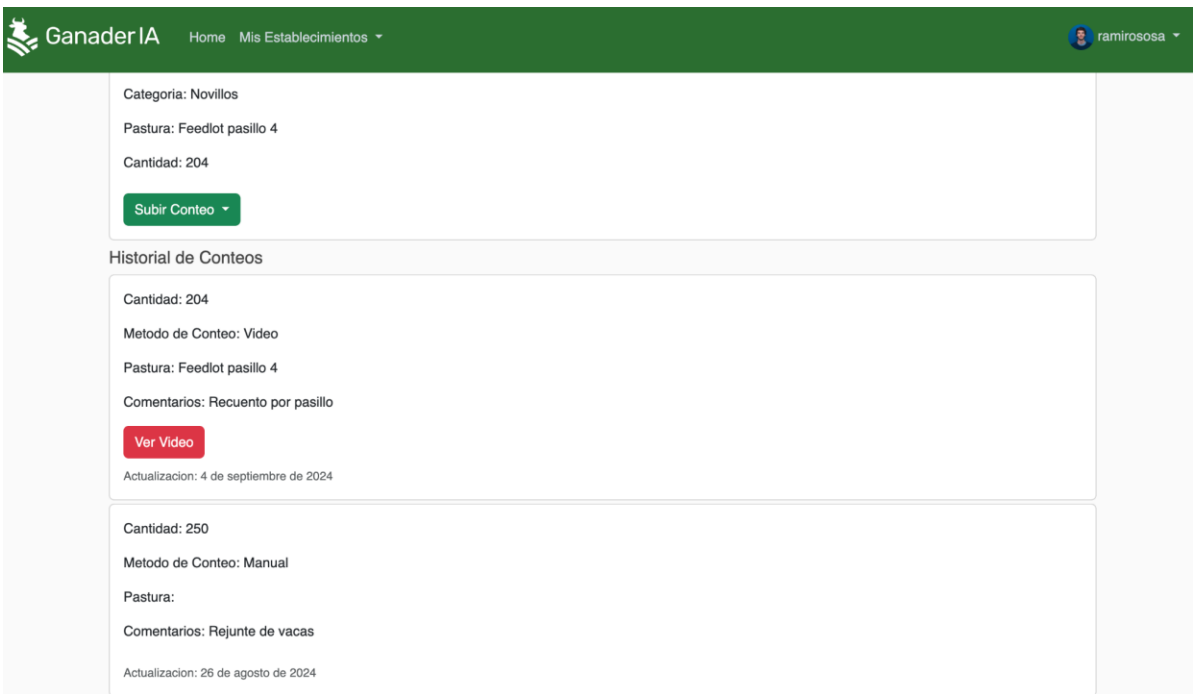


Imagen 25 - Vista del historial de conteos.

- “Probar que el sistema pueda manejar múltiples usuarios subiendo videos simultáneamente sin afectar el rendimiento.”

El sistema puede manejar múltiples usuarios subiendo videos sin afectar el rendimiento ya que la subida de archivos no impacta nunca en el funcionamiento de la aplicación. Esto se debe al método de subida de archivos elegido, cómo el video es subido directamente a S3 a través del cliente sin pasar por el servidor no hay forma que la subida afecte el rendimiento de la aplicación más que cualquier otra operación que el sistema realice.

- “Asegurar que el tiempo de procesamiento del video sea razonable y que el sistema pueda soportar la carga de trabajo esperada.”

Lo que se puede visualizar en la siguiente imagen son los *logs* del contenedor de Celery. Estos *logs* muestran el procesamiento del modelo sobre cada *frame* de video. En promedio cada *frame* tarda alrededor de 60 ms en procesar, por lo cual podemos inferir que el procesamiento con la GPU utilizada hasta el momento permitiría procesar el vídeo en prácticamente el doble del tiempo actual de duración del video. Es decir, por cada minuto de vídeo que se suba, costaría 2 minutos de procesamiento. Consideramos que este tiempo de procesamiento es válido dado que hoy terminamos eligiendo la GPU más económica que teníamos a nuestro alcance, por lo cual la velocidad de procesamiento puede escalar muy rápidamente en cuanto se disponga de mayor capital (ver imagen 26).

```
0: 384x640 4 cows, 70.0ms
Speed: 3.0ms preprocess, 70.0ms inference, 2.0ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 4 cows, 62.2ms
Speed: 2.0ms preprocess, 62.2ms inference, 2.0ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 4 cows, 55.0ms
Speed: 2.0ms preprocess, 55.0ms inference, 1.0ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 4 cows, 54.0ms
Speed: 3.0ms preprocess, 54.0ms inference, 2.0ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 4 cows, 59.5ms
Speed: 3.0ms preprocess, 59.5ms inference, 2.0ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 4 cows, 55.0ms
Speed: 3.0ms preprocess, 55.0ms inference, 3.0ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 4 cows, 57.0ms
Speed: 2.0ms preprocess, 57.0ms inference, 2.0ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 4 cows, 59.0ms
Speed: 3.0ms preprocess, 59.0ms inference, 2.0ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 4 cows, 61.0ms
Speed: 2.0ms preprocess, 61.0ms inference, 2.0ms postprocess per image at shape (1, 3, 384, 640)
```

Imagen 26 - Logs de procesamiento en Celery.

- “Asegurar que la interfaz sea intuitiva y fácil de usar, especialmente para usuarios en áreas rurales con menor acceso a tecnología.”

Para asegurar que la interfaz sea intuitiva y fácil de usar, especialmente para usuarios en áreas rurales con menor acceso a tecnología, se realizan pruebas de usabilidad con usuarios reales. Se involucra a 3 personas de estas zonas, para que interactúen con la página web.

A los participantes se les solicita que realicen tareas dentro de la aplicación, como iniciar sesión, crear un establecimiento y subir un video de conteo. La respuesta a esto es favorable y logran encontrar cada parte de interés.

Además, se realizó un diseño simple para facilitar la usabilidad. Nos aseguramos de que la navegación sea clara y que los botones y enlaces estén bien posicionados y sean fácilmente visibles. El uso de textos y etiquetas es simple, evitando términos técnicos que puedan dificultar la comprensión por parte de los usuarios menos experimentados.

Se concluye de todas estas pruebas, que el funcionamiento de la aplicación web cumple con todos los requerimientos planteados, asegurando que cumple su función y se adapta al tipo de usuarios que van a interactuar con la aplicación.

En el Anexo 3, se incluye un caso de uso con un flujo normal de la aplicación web complementario al *testing* realizado.

6. Desarrollo modelo de *machine learning*

En esta sección, se describe todo lo referente al modelo de *machine learning* implementado. Se incluye un marco teórico con todo lo que se investiga para poder llevar a cabo esta parte del proyecto. También se desarrolla sobre los modelos investigados y cómo se elige cuál modelo es el óptimo para este proyecto. Tras la elección del modelo, se describe cómo utilizándolo se desarrolla un *script* capaz de contar ganado. Por último, se realiza una evaluación del modelo para demostrar cómo el modelo satisface los requerimientos planteados en el capítulo 3.

6.1 Marco teórico

6.1.1 Introducción al *machine learning*

Machine learning [20] es un campo de la inteligencia artificial enfocado en modelos que permiten que un software aprenda y mejore a través de datos. En vez de programar todos los pasos que el software debe realizar se le proporciona datos de entrada para que él mismo aprenda relaciones y patrones proporcionados por los datos de entrada. De esta manera podrá realizar predicciones o tomar decisiones sin haber sido programados específicamente para cierta tarea.

El proceso de *machine learning* generalmente implica tres etapas principales:

Entrenamiento: Durante esta etapa, se le da al algoritmo un conjunto de datos de entrenamiento que contiene ejemplos con sus *features* y los resultados deseados. De esta manera ajusta los parámetros internos al identificar patrones y relaciones entre los datos.

Validación: Una vez entrenado el modelo, se evalúa que tan bien funciona utilizando datos de validación que no se usaron durante el entrenamiento. De esta manera se puede determinar cómo el modelo responde a datos nuevos.

Predicción/Inferencia: Una vez que el modelo ha sido entrenado y validado, se utiliza para hacer predicciones sobre datos nuevos. Esto significa pasar los datos de entrada a través del modelo y obtener una salida que puede ser una clasificación, cómo en nuestro caso sería el nivel de confianza con el cual se identifica una vaca.

Computer vision

El *computer vision* es un campo de la inteligencia artificial que se enfoca en desarrollar software capaz de interpretar, analizar y comprender el contenido de imágenes y videos. El objetivo principal del *computer vision* es permitir a nuestro software "ver" y entender el entorno de manera similar a como lo hacemos nosotros.

Esto significa desarrollo de técnicas que permitan que nuestro software pueda realizar tareas cómo el reconocimiento de objetos, la detección de rostros y el seguimiento de movimientos.

Modelos pre-entrenados

Un modelo pre-entrenado es un modelo de *machine learning* que ha sido entrenado en una gran cantidad de datos para realizar una tarea específica, como reconocimiento de imágenes, procesamiento de lenguaje o cualquier otra tarea.

La idea detrás de un modelo pre-entrenado es que el proceso de entrenamiento puede ser costoso en términos de tiempo, recursos computacionales y datos. Entrenar un modelo desde cero requiere grandes conjuntos de datos con *labels* y tiempo de cálculo.

Los modelos pre-entrenados se pueden utilizar por un lado directamente, para realizar predicciones sobre nuevos datos sin necesidad de entrenarlos, o se pueden utilizar como punto de partida para el entrenamiento en datos específicos. Esta técnica, conocida como *fine-tuning*, permite aprovechar el conocimiento del modelo previo y adaptarlo a problemas específicos con menos datos y recursos computacionales. Esto es exactamente lo que haremos para nuestro software de conteo de ganado.

Set de datos

Los *datasets* son fundamentales en el proceso de entrenamiento y evaluación de modelos de machine learning, ya que proporcionan los datos necesarios para que los algoritmos aprendan patrones y relaciones entre las características de entrada y las salidas deseadas. Además, la calidad del *dataset* es un aspecto crítico para el rendimiento del modelo resultante. Por lo tanto, es muy importante seleccionar cuidadosamente y preprocesar los *datasets* para asegurarse de que sean adecuados para la tarea que se va a realizar.

Usualmente, existen dos sets de datos. Estos son el set de entrenamiento, donde se incluyen imágenes etiquetadas con las cuales se desea que el modelo entrene, y el set de validación, que son imágenes etiquetadas distintas a las de entrenamiento, pero también representativas de la tarea que se desea realizar. Sobre el set de validación es que el modelo va a evaluarse cada vez que finaliza una *epoch* (iteración completa sobre el *dataset* de entrenamiento).

En algunos casos también se incluye un set de datos llamado *test*. Este *dataset*, suele incluir imágenes etiquetadas en condiciones distintas a las que uso tanto para entrenar como validar. El beneficio de usar un set de datos de *test*, es que se puede verificar si el modelo de *machine learning* generalizó lo suficiente como para funcionar correctamente en condiciones no vistas anteriormente.

Fine-tuning

Generalmente, los modelos de *machine learning* vienen pre-entrenados con grandes conjuntos de datos genéricos para aprender representaciones generales de los datos.

El *fine-tuning* implica tomar este modelo pre-entrenado y ajustar sus pesos o parámetros en un conjunto de datos más pequeño y específico relacionado con la tarea que se desea realizar. Por ejemplo, si tenemos un modelo pre-entrenado para la clasificación de imágenes en una amplia variedad de animales, pero deseamos utilizarlo para clasificar imágenes de vacas tomadas específicamente desde arriba, debemos realizar un *fine-tuning* ajustando los pesos del modelo para nuestro conjunto de datos.

Aspectos a tener en cuenta a la hora del entrenamiento

Al momento de entrenar un modelo hay ciertos aspectos importantes a tener en cuenta.

El primero y más importante, es armar un set de datos que contenga las imágenes tipo que yo voy a recibir. Es decir que, en nuestro caso, las imágenes de vacas deben ser en su mayoría tomadas desde un dron. Para que el modelo aprenda a detectar vacas en un plano aéreo.

Segundo, se deben pre-procesar los datos de manera correcta. Este preprocesamiento incluye el etiquetado de las imágenes, donde debemos ser cuidadosos y recortar los recuadros de manera precisa, ya que un mal recorte puede generar confusión en el aprendizaje del modelo.

Tercero, se debe entrenar el modelo con imágenes distintas. Esto implica fotos tomadas desde diferentes ángulos, diferentes razas de ganado, en diferentes posiciones y variando la altura desde donde tomamos la foto. Ya que un entrenamiento con solo un tipo de imagen llevaría a que se espere que una vaca se vea específicamente como se indica en dicha imagen.

Por último y de gran importancia, se debe entrenar lo máximo posible pero sin sobrecargar el modelo de manera innecesaria. Esto se puede lograr realizando las curvas de aprendizaje del modelo a medida que se lo entrena. Cuando observamos en la curva que el modelo converge a un valor, esto significa que seguir entrenando el modelo no brindará grandes cambios y que ya se está lo suficientemente entrenado para el set de datos brindado.

Parámetros de evaluación en un modelo

Generalmente, los modelos de *machine learning* son evaluados según su precisión y rapidez. Estas características pueden ser descritas con los siguientes parámetros.

Precisión

La precisión mide cuántas de las predicciones positivas del modelo son correctas. Es decir, entre todas las instancias que el modelo predijo como positivas (o en una determinada clase), cuántas eran realmente correctas.

$$Precision = \frac{Verdaderos\ Positivos}{Verdaderos\ Positivos + Falsos\ Positivos}$$

Los verdaderos positivos corresponden a los casos donde el modelo predijo correctamente una instancia como positiva. Mientras que los falsos positivos son casos donde el modelo predijo incorrectamente una instancia como positiva.

Una precisión alta indica que el modelo comete pocos errores al predecir positivos.

Recall

El *recall* mide cuántos de los verdaderos positivos han sido detectados por el modelo. De todas las instancias que realmente pertenecen a una clase positiva, cuántas fueron correctamente identificadas.

$$Recall = \frac{Verdaderos\ Positivos}{Verdaderos\ Positivos + Falsos\ Negativos}$$

Los falsos negativos se dan cuando el modelo no detectó una instancia que era verdaderamente positiva.

Un *recall* alto significa que el modelo tiene un buen desempeño identificando la mayor cantidad de positivos.

Relación entre precisión y *recall*

El *recall* está estrictamente relacionado con la precisión. Ya que para tener una cantidad baja de falsos positivos implica considerar un umbral de confianza muy alto, y contrariamente, para tener una cantidad baja de falsos negativos, se debe utilizar un factor umbral de confianza muy bajo. Por lo que se debe tener un balance al elegir dicho umbral, y la precisión está muy relacionada con el *recall*.

Esto se describe gráficamente en la siguiente figura, extraída de un *issue* público de GitHub (ver imagen 27).

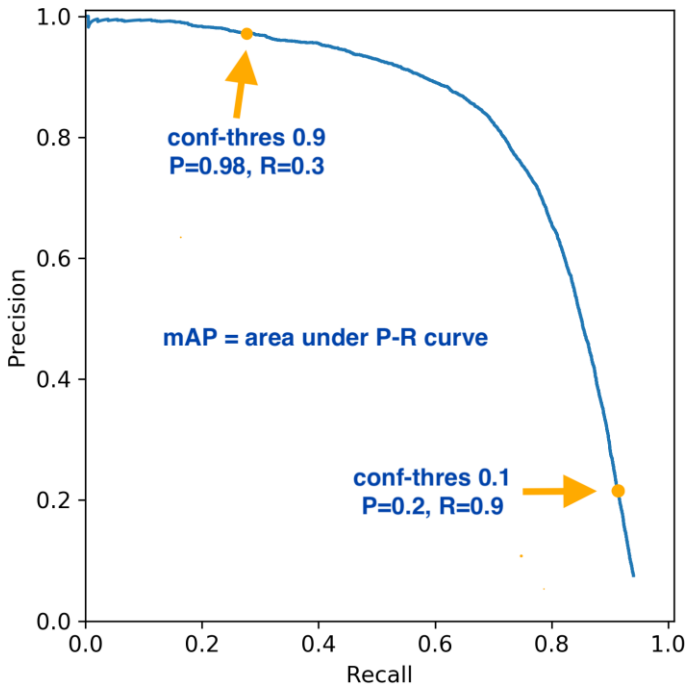


Imagen 27- Curva de precision-recall. [21]

Velocidad de procesamiento

Este parámetro se refiere a la rapidez con la que el modelo procesa los datos y genera predicciones. En tareas de detección de objetos o clasificación en tiempo real, la velocidad de procesamiento es crítica, ya que define cuántos cuadros por segundo (FPS) o cuántas imágenes por segundo el modelo puede manejar.

La velocidad de procesamiento puede verse afectada por la complejidad del modelo, la resolución de las imágenes de entrada, y los recursos de hardware, ya sea que se procesa por GPU o CPU

Comparación de modelos

Al momento de comparar diferentes modelos, es importante tener en cuenta los siguientes puntos:

- Evaluar los modelos con las mismas imágenes de prueba.

- Tener los modelos entrenados de manera equitativa, es decir que todos ya convergen en sus curvas de aprendizaje.

- Haber entrenado los modelos con el mismo set de datos. Sería injusto comparar un modelo entrenado con una mayor cantidad de imágenes.

- Considerar los parámetros anteriormente mencionados: Precisión, *recall* y tiempo de procesamiento.

Si garantizamos dichos puntos, la comparación debería ser equitativa para todos los modelos que comparemos.

Factores importantes a considerar al testear modelos

Más allá de factores técnicos en los que difieren los diferentes modelos para entrenarlos, en este apartado se discuten puntos relevantes a tener en cuenta al momento de testear cualquier modelo.

Es importante que, al testear, se utilicen imágenes similares a las que se utilizaron en el set de entrenamiento, pero no las mismas o muy parecidas. Ya que el modelo puede haber aprendido de memoria dichas imágenes. Al utilizar una similar, se podrá notar cómo funcionó el entrenamiento.

Utilizando imágenes similares, se puede evaluar si se detectan correctamente las detecciones positivas. También debemos evaluar las detecciones negativas, es decir, probar imágenes similares a las de nuestro tipo, pero en vez de mostrar una vaca, utilizar otro tipo de animal u objeto, para así ver el funcionamiento del modelo y corroborar si detecta falsos positivos.

También se debe evaluar la precisión del modelo. Para este proyecto es relevante a qué altura el modelo deja de detectar con un gran factor de confiabilidad a los animales. Ya que recorrer a una mayor altura, nos permite abarcar un mayor campo de visión con la cámara, y por lo tanto, recorrer todo el campo en un menor tiempo. Entonces, se evaluará el funcionamiento del modelo a diferentes alturas y se observará cuándo deja de ser confiable la detección.

6.1.2 *Leakage*

En este apartado se describe una de las problemáticas que puede haber al entrenar y utilizar un modelo de *machine learning*. Esta problemática es el *leakage* [22].

Introducción

El *leakage* es un problema que ocurre cuando se filtra información del conjunto de datos de entrenamiento hacia el modelo, lo que resulta en una evaluación falsa y un rendimiento inflado. Esta situación genera modelos que parecen funcionar muy bien en los datos de entrenamiento o validación, pero que fallan al generalizar en datos nuevos.

Leakage de entrenamiento

El *leakage de entrenamiento* ocurre cuando la información del conjunto de test o validación se utiliza en el entrenamiento del modelo. Esto puede suceder si, por ejemplo, se mezclan datos de entrenamiento y test antes de dividir los conjuntos, o si se seleccionan características basadas en todo el *dataset*, incluyendo los datos de *test*.

Detección del *leakage*

Detectar el *leakage* requiere una revisión exhaustiva del proceso de preparación de datos. Esto incluye tanto el set de entrenamiento como el de validación.

Evaluación de rendimiento

El rendimiento excesivamente alto en los datos de validación en comparación con los datos de test puede ser un indicador de *leakage*. Un análisis exhaustivo de las curvas de aprendizaje y la comparación de los errores entre diferentes conjuntos de datos puede ayudar a identificar problemas.

Estrategias para prevenir el *leakage*

La prevención del *leakage* es esencial para garantizar que los modelos de *machine learning* sean capaces de generalizar a nuevos datos y proporcionar valor en aplicaciones del mundo real.

División correcta de los datos

Es fundamental asegurar una división clara y anticipada de los datos en conjuntos de entrenamiento, validación y *test*, antes de cualquier procesamiento o selección de características. Los datos de *test* deben permanecer completamente fuera del alcance del modelo hasta la fase final de evaluación. En nuestro caso, se generan *scripts* que dividen el *dataset*.

Consideraciones temporales

En problemas temporales, es crucial mantener el orden cronológico en la división de los datos. El conjunto de entrenamiento debe contener datos anteriores en el tiempo a los utilizados para la validación y prueba. Esto evita que el modelo "vea" el futuro durante el entrenamiento.

Conclusión

El *leakage* en *machine learning* es una fuente significativa de error que puede llevar a modelos engañosamente precisos pero ineficaces en la práctica. Identificar y prevenir el *leakage* es crucial para el desarrollo de modelos robustos y confiables que ofrezcan un rendimiento consistente en entornos del mundo real. Mediante la implementación de prácticas adecuadas en la división de datos, selección de características y validación cruzada, los desarrolladores pueden minimizar el riesgo de *leakage* y garantizar la integridad del proceso de modelado.

Particularmente, en este proyecto se tomaron las precauciones de realizar scripts que verifiquen si un archivo está repetido en ambos sets de datos, el de entrenamiento y validación. También se verifica que cambien las características de la imagen verificando que haya pasado una cierta cantidad de *frames* entre imágenes que compongan el set de datos.

6.1.3 *Overfitting*

Ahora se describe lo que es probablemente el problema más común al momento de entrenar un modelo de *machine learning*, este problema es el *overfitting* [23] (sobre ajuste en Español). Este problema puede venir causado por el *leakage* descrito anteriormente e implica resultados engañosos al entrenar un modelo.

Introducción

El *overfitting* es un problema común en el desarrollo de modelos de *machine learning* que ocurre cuando un modelo se ajusta demasiado a los datos de entrenamiento, donde no captura los patrones generales, sino que se ajusta a lo específico del entrenamiento. Como resultado, el modelo muestra un rendimiento excepcional en el conjunto de entrenamiento, pero falla al generalizar en datos nuevos. Este apartado explora los mecanismos para detectar el *overfitting* y las estrategias recomendadas para mitigarlo.

Detección del *overfitting*

Detectar el *overfitting* es esencial para garantizar que un modelo no solo funcione bien en los datos de entrenamiento, sino que también tenga la capacidad de generalizar a otros conjuntos de datos.

Evaluación del rendimiento en conjuntos de datos de validación

Una de las formas más directas de detectar el *overfitting* es comparando el rendimiento del modelo en el conjunto de entrenamiento con su rendimiento en un conjunto de validación. Si el modelo muestra un rendimiento significativamente mejor en el conjunto de entrenamiento en comparación con el conjunto de validación, es probable que esté sobre ajustado.

Curvas de aprendizaje

Las curvas de aprendizaje son herramientas visuales útiles para detectar el *overfitting*. En una curva de aprendizaje, se grafica la precisión o el error del modelo tanto en el conjunto de entrenamiento como en el de validación a lo largo de las épocas de entrenamiento. Un indicador claro de *overfitting* es cuando la curva de entrenamiento sigue mejorando mientras que la curva de validación se estabiliza o incluso empeora.

Estrategias para evitar el *overfitting*

Uso de más datos de entrenamiento: Una de las formas más efectivas de reducir el riesgo de *overfitting* es aumentar el tamaño del conjunto de datos de entrenamiento. Al proporcionar más ejemplos, el modelo puede aprender patrones más generales en lugar de ajustarse a las peculiaridades de un conjunto de datos limitado.

Early stopping: El *early stopping* es una técnica que detiene el entrenamiento del modelo tan pronto como su rendimiento en el conjunto de validación converge. Esto evita que el modelo siga ajustándose a los datos de entrenamiento más allá del punto en el que ha aprendido patrones generales útiles.

Conclusión

El *overfitting* es un desafío importante en el desarrollo de modelos de *machine learning*, ya que puede comprometer la capacidad del modelo para generalizar a datos nuevos.

Sin embargo, mediante el *early stopping* y otras estrategias, es posible desarrollar modelos robustos y efectivos que mantengan su rendimiento en contextos generales.

En este proyecto, se utiliza *early stopping*, se evita el *leakage* y se verifica que los rendimientos sean similares para el set de validación y el de entrenamiento, buscando asegurar que no haya *overfitting* y el modelo generalice.

Casos de *overfitting* en nuestro proyecto y cómo se detecta

En el transcurso del proyecto, principalmente al inicio, se detectó *overfitting* al entrenar modelos, esto se puede notar de las gráficas que devuelve Yolo al finalizar un entrenamiento (ver imagen 28).

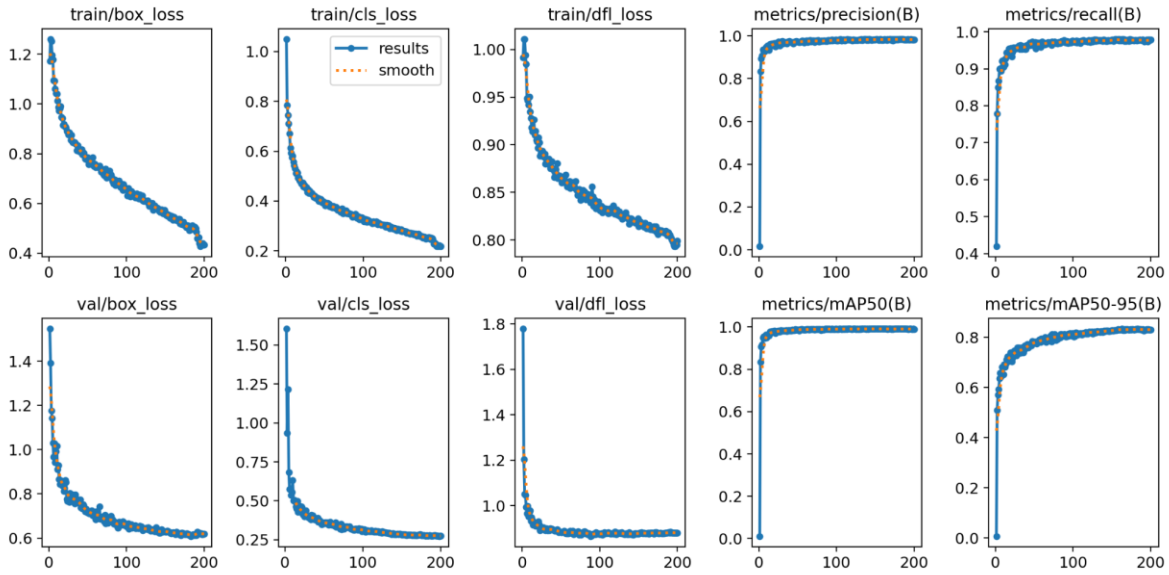


Imagen 28 - Resultados de entrenamiento con overfitting.

Examinando principalmente los primeros tres gráficos, se nota que llega un punto en el que los valores decaen bruscamente. Esto sucede cuando el modelo ya converge en su entrenamiento, y los parámetros del modelo se están ajustando específicamente al set de validación, para arrojar mejores valores. Esto se puede prevenir evitando el *leakage* y utilizando *early stopping* al entrenar, qué fue lo realizado en el caso de este proyecto (ver imagen 29).

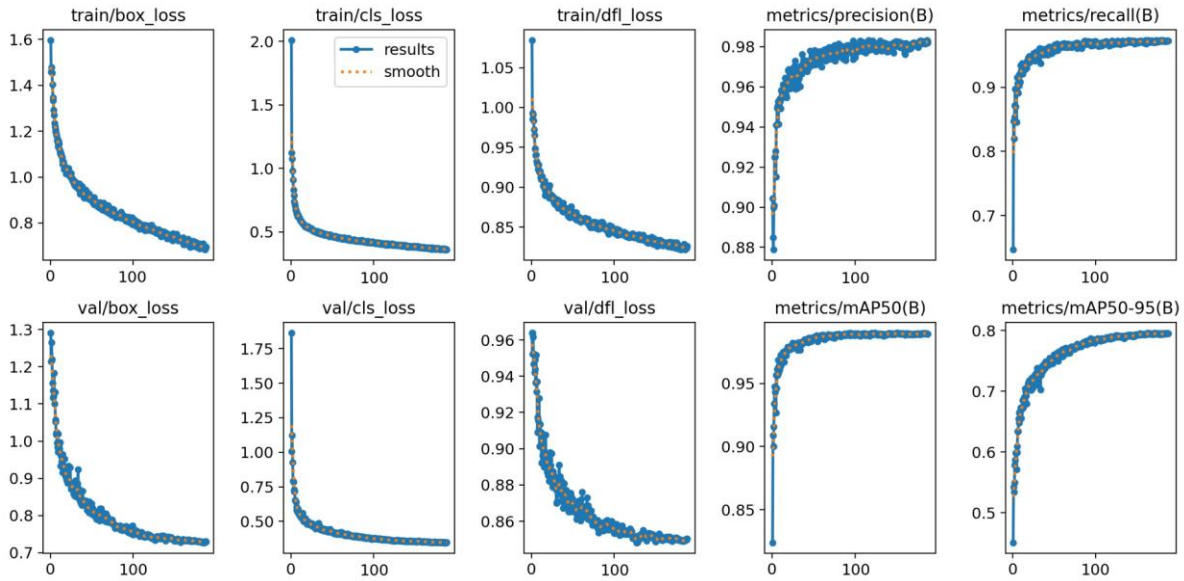


Imagen 29 - Resultados de entrenamiento deseables.

En esta última imagen, se muestran los resultados normales, de un modelo que converge y para de entrenar utilizando *early stopping*. Este modelo asegura que el entrenamiento sirvió para generalizar, y no solamente para adaptarse al set de validación y entrenamiento.

6.1.4 Uso de repositorios: GitHub y Hugging Face

Introducción

GitHub [24] es una plataforma ampliamente utilizada para el desarrollo colaborativo de software, basada en el sistema de control de versiones Git. A lo largo del tiempo, se ha consolidado como una herramienta fundamental para equipos de desarrollo, investigadores y profesionales de software, permitiendo gestionar y alojar código en un entorno seguro y estructurado. A continuación, se expone la importancia de utilizar GitHub para la gestión de proyectos y Hugging Face para el alojamiento de *datasets*.

Colaboración y control de versiones

GitHub facilita la colaboración en proyectos al permitir que múltiples desarrolladores trabajen de manera simultánea en un mismo repositorio. Mediante herramientas como *merge*, *pull requests* e *issues*, los equipos pueden comunicarse y coordinar de forma efectiva, supervisando el progreso y asegurando que todas las contribuciones estén alineadas con los objetivos del proyecto.

También permite un control preciso sobre el historial de cambios en el código. Cada modificación queda registrada, lo que facilita la revisión, comparación y restauración de versiones anteriores en caso de ser necesario. Esta capacidad es clave para asegurar que los errores puedan identificarse y corregirse sin comprometer la integridad del proyecto.

Uso de *branches*

Se pueden utilizar *branches*, que permiten trabajar en nuevas características o correcciones sin afectar la rama principal del código. Esto asegura que el código en producción permanezca

estable mientras los cambios se prueban y validan en ramas separadas. Es muy útil para el desarrollo de *features*.

6.1.5 Importancia de alojar un modelo y su *dataset* en Hugging Face

Hugging Face [25] se ha convertido en una plataforma esencial para la gestión y distribución de *datasets* utilizados en *machine learning*. Alojar el *dataset* en Hugging Face permite acceder a las imágenes y sus etiquetas desde cualquier lugar de forma descentralizada, facilitando el trabajo colaborativo. Además, los colaboradores pueden subir y actualizar etiquetas de manera asíncrona, agilizando el desarrollo.

Al igual que GitHub, Hugging Face nos permite trabajar en conjunto sobre un mismo set de datos, es decir que, podemos subir imágenes etiquetadas nuevas desde diferentes dispositivos, y el otro podrá acceder a ellas de manera rápida y sencilla.

También aquí podemos llevar un historial de versiones. Tanto para el modelo de detección como para el set de datos que se utilizó en cada modelo. Esto es una gran ventaja por si algún modelo presenta una falla en su entrenamiento, o si se genera *leakage* en el set de datos de entrenamiento y validación.

Importancia de desplegar un modelo en una API de Hugging Face[1]

Durante el desarrollo del proyecto, se decide descentralizar el modelo de detección alojando en Hugging Face. Dentro de Hugging Face, existe la posibilidad de generar una API en la cual se pueda subir imágenes o videos y procesarlas rápidamente.

Ventajas

Descentralización: Al estar alojado en una API externa, el modelo es accesible desde cualquier lugar, permitiendo a diferentes usuarios interactuar con él sin necesidad de integración local. Esto aumenta la flexibilidad en su uso, ya que los datos pueden ser enviados de manera asíncrona por ambos integrantes del equipo.

Actualizaciones simplificadas: Cualquier mejora o reentrenamiento del modelo se puede aplicar directamente en la API de Hugging Face, lo que asegura que todas las versiones del sistema estén siempre actualizadas. Solamente hay que actualizar el archivo de nombre "*best.pt*" que contiene los hiperparámetros del modelo entrenado.

Respaldo: Alojar el modelo en Hugging Face, nos asegura que nunca perderemos este, o que la probabilidad es muy baja comparado a los riesgos que conlleva tener los archivos guardados solamente de manera local.

Vinculación al *dataset*: Hugging Face permite una vinculación del modelo con el *dataset* también alojado en la plataforma, lo que facilita el uso y actualización del conjunto de datos para entrenar o validar el modelo. Podemos subir varios modelos entrenados con diferentes sets de datos y siempre conocer en qué *dataset* se entrenó cada modelo.

Sin límite de subida de archivos: La mayor ventaja, o la que se considera mayor, es que a diferencia de otros repositorios, Hugging Face permite subir archivos que demanden mucho

[1] API accesible desde https://huggingface.co/spaces/manunava11/Modelo_De_Deteccion

almacenamiento como lo es nuestro *dataset* y modelos. En otros repositorios era imposible tener un respaldo, y Hugging Face nos permite alojar todo allí.

6.2 Software de etiquetado LabelImg

Para entrenar un modelo, es necesario generar un set de datos con imágenes etiquetadas de la clase que se desea detectar. Existen varios programas para realizar esto, en nuestro caso, se elige LabelImg [26].

LabelImg es una herramienta de código abierto utilizada para etiquetar imágenes para tareas de reconocimiento de objetos en el campo *computer vision*. Este software nos permite marcar y etiquetar objetos dentro de imágenes con cajas delimitadoras, lo que facilita la creación de conjuntos de datos etiquetados para entrenar los modelos de detección de objetos (ver imagen 30).

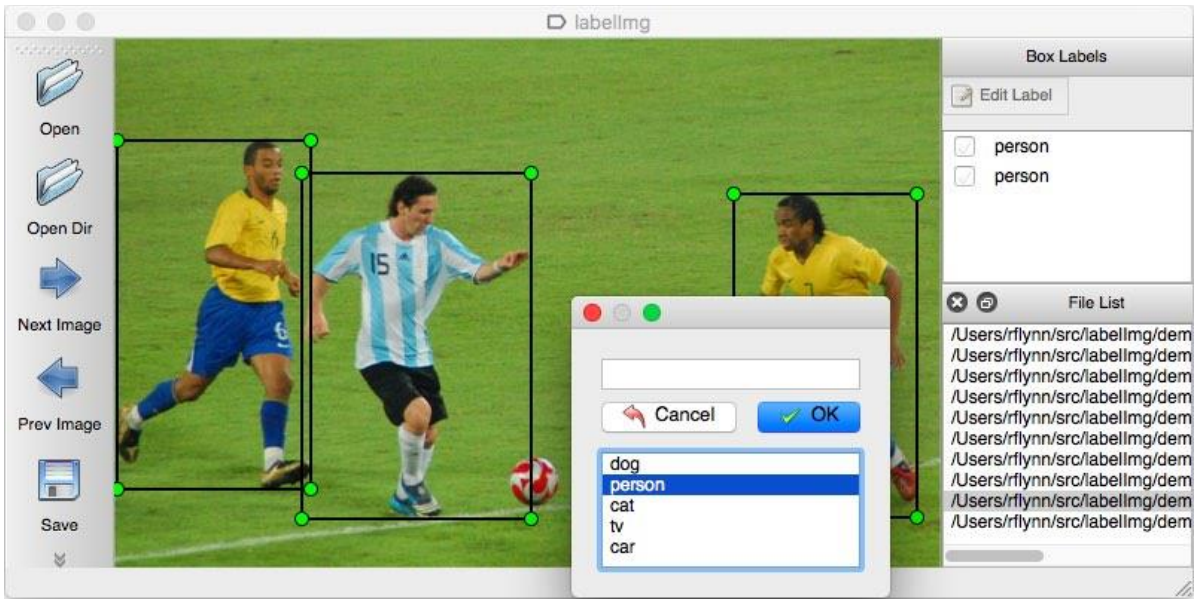


Imagen 30 - Imagen representativa del software.

Este software está dado por *scripts* de Python y es accesible, de código abierto, desde su repositorio oficial.

En nuestro caso, simplemente generamos los recuadros y ponemos la anotación "*cow"* que se refiere a la clase de vacas (ver imágenes 31,32 y 33).

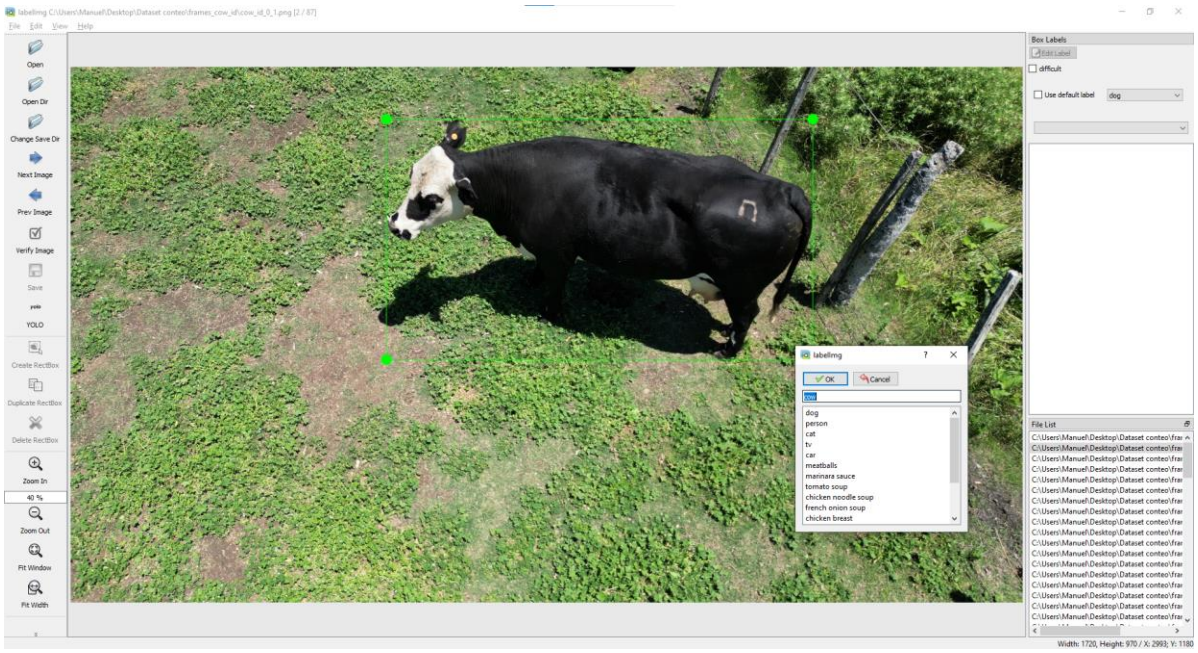


Imagen 31 - Captura de pantalla al etiquetar un recuadro.

Imagen 32 - Captura de pantalla del recuadro hecho.

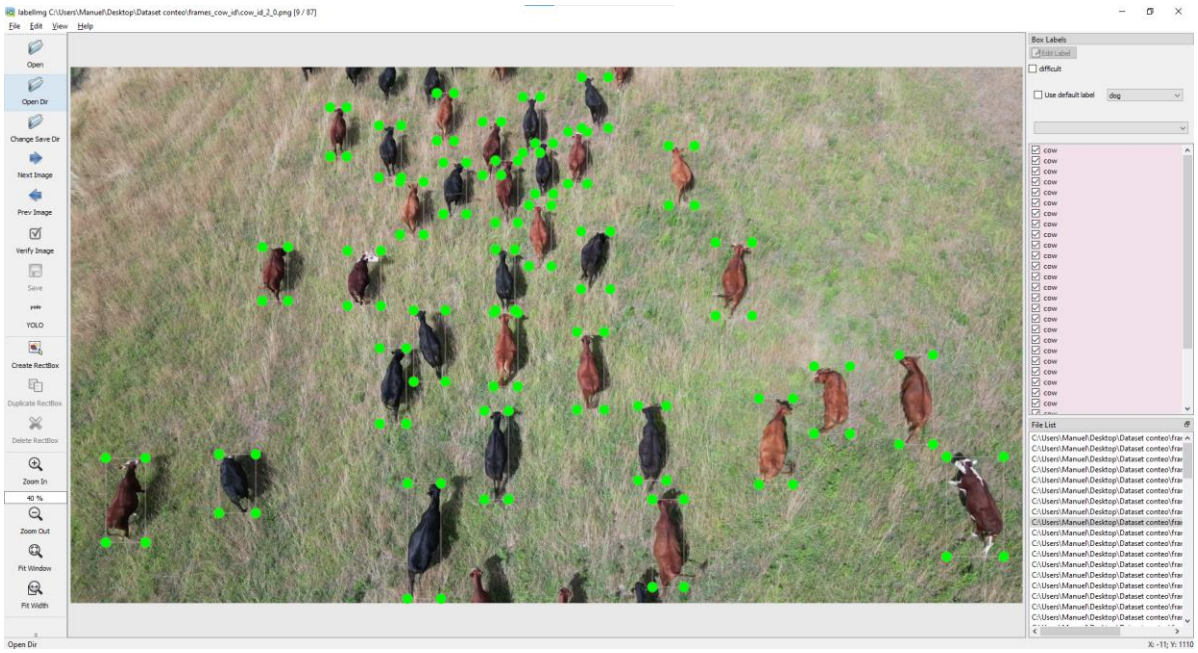


Imagen 33 - Varios recuadros en una misma imagen.

Teniendo las imágenes etiquetadas correctamente, el programa LabelImg genera automáticamente los archivos con el formato necesario para el entrenamiento, un ejemplo de este formato es el siguiente:

0 0.351953 0.334028 0.080469 0.231019

El primer número representa el identificador de la clase, en este caso solo se tiene una clase en el modelo, la clase de las vacas, la cual corresponde al número 0. Posteriormente se indican las coordenadas de la caja delimitadora de cada animal en la imagen.

Labellmg automáticamente genera una carpeta denominada *Labels*, donde aloja todos los archivos de texto generados, y cada archivo de texto, tiene su imagen correspondiente con el mismo nombre en una carpeta *Images*.

6.4 Desarrollo del modelo de *machine learning*

Ahora se presenta una descripción detallada del proceso llevado a cabo para implementar el sistema de conteo automatizado de ganado. Se abarca tanto el desarrollo de la plataforma, como la implementación del modelo de *machine learning* necesarios para la detección y conteo de vacas en imágenes y videos capturados por el dron.

Se cubren las decisiones y desafíos enfrentados y las metodologías empleadas para lograr una solución precisa. Se detallan los componentes del sistema, el proceso de integración entre el modelo de *machine learning* y la aplicación web, así como las pruebas realizadas para garantizar el buen funcionamiento del sistema en un entorno real.

6.4.1 *Machine learning*

En este apartado, se muestra todo lo desarrollado para llegar a un modelo óptimo de detección, desde el armado del primer *dataset* para definir un modelo a utilizar, la metodología empleada para realizar conteos y una evaluación del modelo final.

Obtención del primer set de datos

Para lograr definir qué modelo pre-entrenado de *machine learning* se utiliza, se debe armar un primer *dataset*, el cual debe contener una cantidad suficiente de imágenes para realizar una comparación que permita definir cuál modelo es mejor y se adapta a las necesidades de este proyecto. Por lo que se realizan dos visitas a campos, donde se toman imágenes de ganado teniendo en cuenta todas las precauciones antes mencionadas.

Tras definir un modelo a utilizar, se continúan realizando visitas para entrenar más el modelo definitivo.

Primera visita al campo "Establecimiento El talar" - 22/11/2023

El día 22 de noviembre concurrimos a un campo en Soriano. Este campo pertenece a un productor ganadero que cuenta con varios animales. Entre ellos, y lo que más nos interesa, cuenta con ganado vacuno. Nos dejó a nuestra disposición 120 vacas, con las cuales pudimos tomar fotos y pesarlas.

Pudimos realizar una recorrida en el dron, donde hicimos dos videos de hasta 10 minutos.

Conclusiones:

- La calidad 4K de la cámara permite volar a una gran altura preservando la calidad de las imágenes y el fácil reconocimiento de un animal.
- Al poder viajar a gran altura los animales no sienten molestia alguna por el dron, prácticamente no se escucha su ruido.
- El vuelo bajo puede asustar a los animales y hacerlos correr hacia otro lado.
- Muchas vacas no cuentan con caravanas dado que las pierden
- En este campo, y en la mayoría de Uruguay, había hasta 3 razas diferentes de vacas.
- Había aproximadamente 35 km/h de viento y el dron se mantuvo estable. Se mueve un poco el dispositivo, pero la cámara se mantiene perfectamente estable.

Segunda visita al campo 'Establecimiento Manchega' - 5/12/2023

Visitamos el campo Manchega el día 5/1. Este campo queda ubicado en Maldonado, cerca de la localidad de La Barra. El productor de este campo puso a nuestra disposición 140 vacas. Se pudo realizar una serie de videos sobrevolando el ganado mientras se realizaba el rodeo. También se hicieron tomas individuales y de pocos animales para poder entrenar el modelo de reconocimiento y posteriormente evaluarlo.

Probamos el zoom digital de la cámara del dron que dio muy buenos resultados, se mantiene bastante calidad, no hay movimientos de la cámara a pesar del viento y volando a 12 metros logramos tomas lo suficientemente cercanas.

6.4.2 Elección de modelo

En esta etapa, nos enfocamos en hacer el primer prototipo de nuestro proyecto que cumpla con los objetivos planteados. Para esto, se eligen tres modelos pre-entrenados conocidos y recomendados para tareas de *machine learning* como lo es la detección de objetos.

Se realiza un conteo del ganado con un video grabado desde el dron, para tomar los *frames* de este video, etiquetarlos, y realizar una comparación justa entre los tres modelos pre-entrenados.

El principal objetivo de esta etapa es cumplir con lo mínimo y necesario que garantice el funcionamiento de nuestro proyecto. Es decir, encontrar un modelo pre-entrenado que realice la tarea sin inconvenientes, para posteriormente realizarle un *fine-tuning* (ajuste fino) y aumentar notoriamente la calidad del modelo.

Para la elección de los tres modelos a evaluar, se utilizó el ranking de Hugging Face [27], página donde se encuentran almacenados muchos modelos de *machine learning* y donde se alojan sets de datos. En esta página se encuentran ordenados en un ranking una serie de modelos de detección. Este ranking se arma en función a cómo detectan ciertos objetos de un mismo *dataset*.

De este ranking, se eligieron cuatro modelos. Con estos modelos, se tomó una serie de fotogramas y se observó la precisión con la que cuentan. Posteriormente, se entrenó con un mismo *dataset* estos tres modelos. Una vez que todos los modelos fueron entrenados, nuevamente se los evalúa con los mismos *frames*, para finalmente elegir cuál es el modelo óptimo y de mejor funcionamiento.

Una vez que se seleccionó el modelo óptimo para el proyecto, se comenzó con el desarrollo de un *script* que teniendo como datos de entrada el video pre-grabado, entrega como datos de salida la cantidad de vacas detectadas.

Los pasos que seguir para el entrenamiento de un modelo de reconocimiento en imágenes, es el siguiente:

Preparación de datos

Se debe reunir un conjunto de imágenes en el cual se encuentre el objeto a detectar, es decir, en nuestro caso, imágenes tomadas desde el dron donde se encuentren vacas. Una vez se tienen estas imágenes se las debe etiquetar. Para el etiquetado de imágenes se utilizó un software llamado LabelImg.

Descargar un modelo pre-entrenado

Posteriormente, se debe descargar el modelo pre-entrenado que se desea entrenar. En nuestro caso, estos modelos son: Yolov8, DETR o Detectron 2.

Configuración del entorno y del modelo

Se deben configurar las dependencias necesarias para ejecutar el modelo pre-entrenado. Esto incluye librerías como OpenCV, TensorFlow, PyTorch, etc. También se deben configurar los archivos .cfg de cada modelo para los requisitos específicos del entrenamiento.

Entrenamiento del modelo

En este momento se utilizan los datos etiquetados y el modelo pre-entrenado para mejorar el modelo actual. Eso implica iteraciones de entrenamientos en los datos y ajustar los pesos del modelo para mejorar la precisión de la detección.

Evaluación del modelo

Existen varias herramientas para evaluar un modelo, pero en este caso, se optó por armar un set de datos tomados desde nuestro dron, y verificar cual modelo entrenado muestra un mejor rendimiento.

Primeros entrenamientos y evaluación de modelos

En este apartado se describen los primeros entrenamientos para cada modelo a evaluar. Estos modelos son: Yolov8, Detectron 2 y DETR.

Se explica brevemente el proceso para entrenar cada uno, junto a una evaluación justa, donde se evalúa todos los modelos con la misma cantidad de imágenes y tomando como referencia de los resultados, los mismos parámetros.

Primer modelo: Yolov8

Para realizar el entrenamiento del modelo pre-entrenado de Yolov8, se toma la información de su repositorio oficial.

El entrenamiento es bastante sencillo, para comenzar se debe crear un archivo de tipo Python donde se cree un nuevo modelo desde cero, se cargue un modelo pre-entrenado y se indique la ruta de un archivo de configuración del entrenamiento (ver imagen 34).

```
1   from ultralytics import YOLO
2
3   # Load a model
4   model = YOLO("yolov8n.yaml")  # build a new model from scratch
5   model = YOLO("yolov8n.pt")  # load a pretrained model (recommended for training)
6
7   # Use the model
8   model.train(data="EntrenarYolov8\config.yaml", epochs=300)  # train the model
9   # C:\Users\Manuel\Desktop\Carpeta Visual\Python\config.yaml
10  # coco128.yaml
```

Imagen 34 - Captura de pantalla Visual Studio Code.

Este archivo de entrenamiento, debe ser de tipo ".yaml", y contiene:

- La ruta absoluta hacia la carpeta donde se encuentran el set de datos para entrenar.

- Rutas relativas desde la absoluta que lleven tanto a las imágenes como a los archivos de texto que indique el etiquetado de cada imagen.

- La cantidad de clases (tipos de objetos a detectar) y el nombre de cada clase (ver imagen 35).

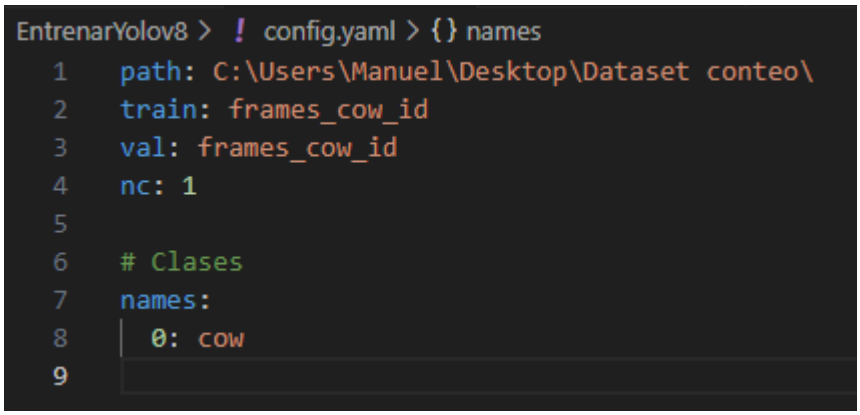


Imagen 35 - Configuración del entrenamiento.

Luego, se debe ejecutar el archivo Python y se comenzará con el entrenamiento. Se realizaron tres ejecuciones, en las cuales se fueron variando la cantidad de *epochs*.

En la primera se realizaron 10 *epochs* y los resultados fueron malos. Cada *epoch*, corresponde a una iteración completa del modelo sobre todo el set de imágenes de validación.

Al realizar la segunda ejecución, utilizando 10 *epochs*, se vieron unos grandes resultados donde la detección es suficientemente precisa para ayudarnos a cumplir nuestros objetivos.

Con el fin de optimizar aún más el modelo, se realizó una última ejecución con la cantidad de 300 *epochs* (ver imagen 36).

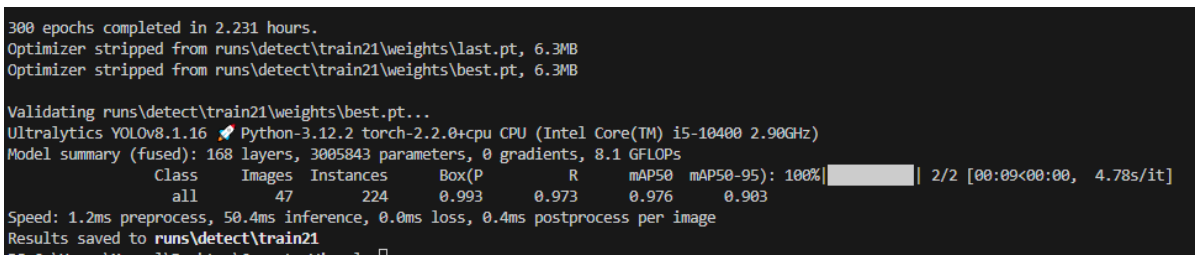


Imagen 36 - Resultados del entrenamiento.

Esta última ejecución de entrenamiento para el modelo Yolo dio los siguientes resultados: (ver imagen 37).

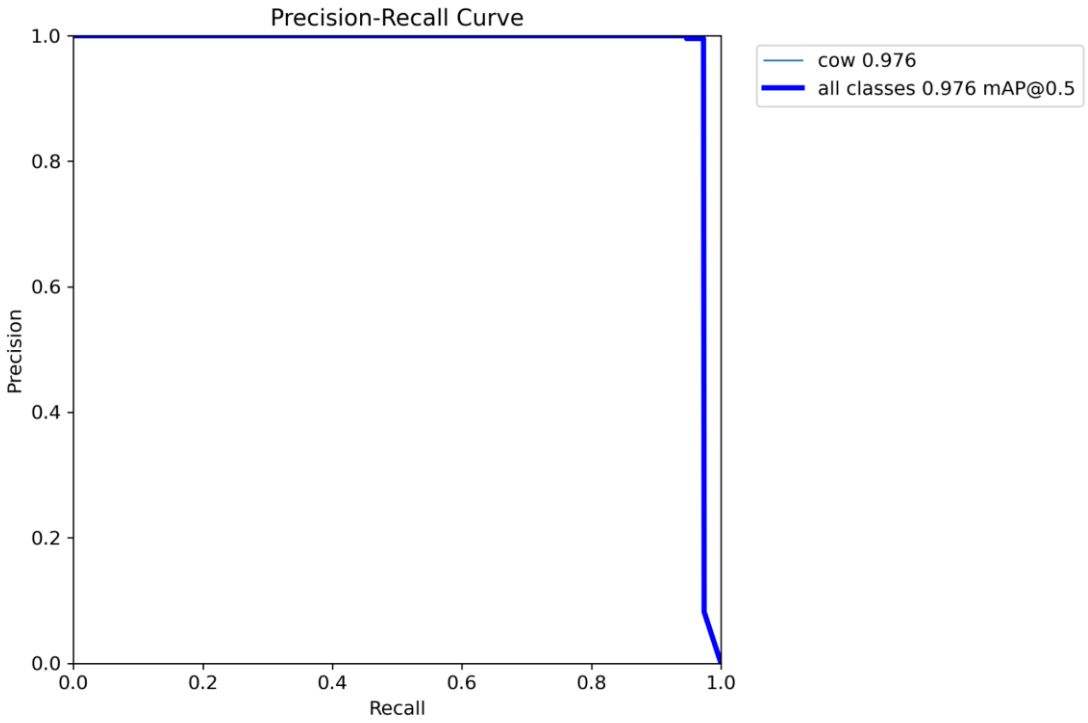


Imagen 37 - Curva de precision-recall.

Interpretación del gráfico *precision-recall*

Esta curva gráfica la precisión respecto al parámetro *recall*, ambos parámetros fueron previamente descritos, pero ahora se los vincula de una forma más detallada. En este gráfico, se indica cuántas veces el modelo fue capaz de identificar correctamente vacas en las imágenes que efectivamente había.

Cuando la precisión vale prácticamente 1 y el *recall* es casi nulo, implica que el modelo está siendo muy preciso en sus predicciones positivas, pero puede estar pasando por alto muchas instancias positivas reales (alto número de falsos negativos).

En contraparte, cuando el *recall* es muy alto, y la precisión muy baja, el modelo está capturando casi todas las instancias positivas reales, pero también clasifica incorrectamente muchas instancias negativas como positivas (alto número de falsos positivos).

Balance entre la precisión y el *recall*

En muchos casos, hay una compensación entre precisión y *recall*, como se ve en los gráficos, cuando uno es alto, el otro suele ser bajo.

Para balancear esto, se puede cambiar el umbral para el cual algo efectivamente se cuenta como detección. Al elegir un umbral muy alto, se pueden dar falsos negativos, por alguna vaca que esté dando baja confianza. Al contrario, si se fija un umbral de confianza muy bajo, se pueden filtrar falsos positivos.

Se sacan las siguientes conclusiones:

Es una curva alta y vertical: Esto indica que el modelo tiene un rendimiento muy bueno ya que logra mantener una precisión alta incluso con un nivel alto de *recall*, hasta que la exhaustividad se acerca al 1 (100% de los verdaderos positivos identificados). Solo en este punto, la precisión cae ligeramente.

mAP desde 0.5 hasta 0.976: El valor mAP significa *Mean Average Precision*, y mide el rendimiento promedio del modelo para detectar objetos correctamente, considerando diferentes umbrales de confianza. Un valor de 0.976 indica que el modelo tiene un rendimiento excelente en términos de detectar correctamente las instancias de la clase "*cow*".

Este gráfico indica que el modelo es altamente efectivo para detectar las vacas. La precisión y *recall* están cerca de 1, lo que significa que el modelo está cometiendo muy pocos errores, tanto en términos de falsos positivos como de falsos negativos.

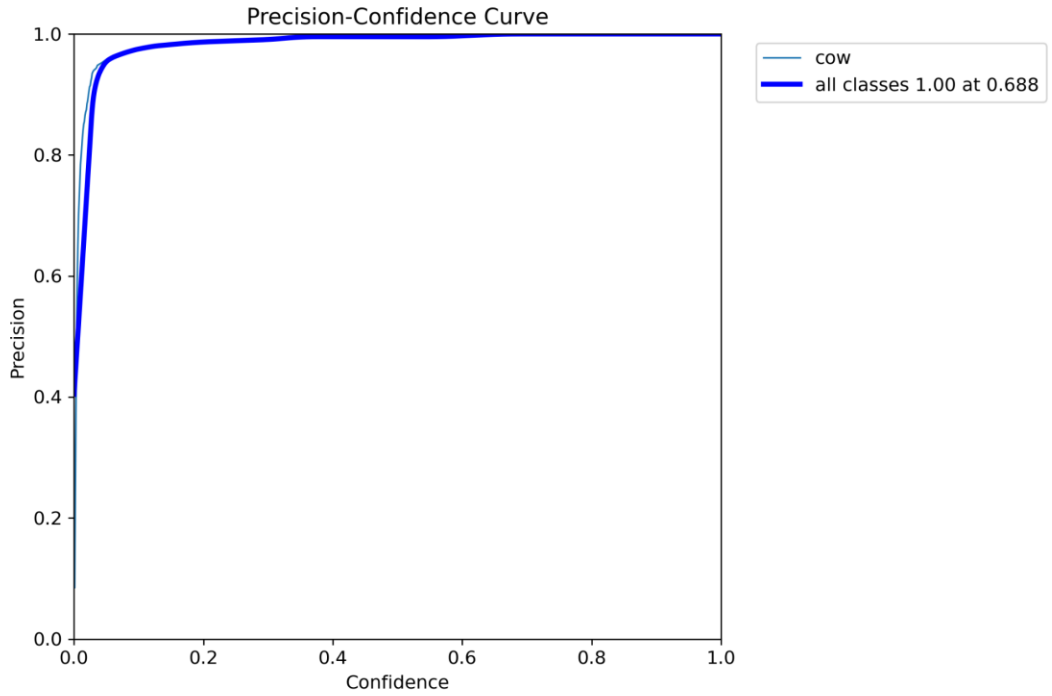


Imagen 38 - Curva de precision y confianza.

Interpretación del gráfico *precision-confidence* (ver imagen 38).

Este nuevo gráfico es una curva de la precisión respecto a la confianza, que muestra cómo varía la precisión del modelo en función de diferentes niveles de confianza en sus predicciones.

Es una curva ascendente y estable: Al principio, con niveles de confianza bajos (izquierda del gráfico), la precisión es relativamente baja, lo que indica que el modelo puede estar haciendo muchas predicciones incorrectas cuando no está muy seguro. A medida que aumenta la confianza, la precisión sube rápidamente y se estabiliza cerca de 1, lo que significa que, cuando el modelo tiene un nivel alto de confianza, sus predicciones son casi siempre correctas.

Funcionamiento para confianzas bajas (<0.2): Con niveles de confianza bajos, la precisión es mucho menor. Esto es esperable, ya que el modelo no está muy seguro de sus predicciones.

Funcionamiento para confianzas medias (de 0.2 a 0.7): La precisión aumenta significativamente en este rango, lo que indica que el modelo mejora mucho cuando tiene un poco más de seguridad en sus predicciones.

Funcionamiento para confianzas altas (mayor a 0.7): A partir de una confianza de aproximadamente 0.7, la precisión se estabiliza en 1. Esto indica que cuando el modelo predice con una alta confianza, lo hace de manera correcta prácticamente todo el tiempo.

Este gráfico muestra que el modelo es muy preciso cuando su nivel de confianza es alto, especialmente a partir de 0.688, donde las predicciones tienen una precisión del 100%. Sin embargo, cuando el modelo tiene niveles de confianza más bajos (por debajo de 0.2), la precisión disminuye significativamente, lo que indica que las predicciones no son tan confiables en estos casos.

Este comportamiento es común en modelos de *machine learning*, y es útil para establecer un umbral de confianza. En este caso, se podría establecer un umbral de confianza de 0.7 o superior para asegurar de que las predicciones sean casi siempre correctas.

Evaluación del modelo Yolov8

Se evalúa el modelo entrenado de Yolov8 con una herramienta de Google Collab, la cual se utilizará también para evaluar los otros modelos, siguiendo los parámetros planteados en el sector de investigación para una correcta evaluación (ver imagen 39).

```
Running evaluation...
100% ████████████████████████████████ 2/2 [00:00<00:00,  2.89it/s]
Accumulating evaluation results...
DONE (t=0.03s).
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.730
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.900
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.830
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.100
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.320
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.950
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.115
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.800
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.625
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.230
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.300
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.900
```

Imagen 39 - Evaluación del modelo entrenado de Yolov8.

Esta imagen muestra la precisión promedio (AP) y *recall* respecto al parámetro IoU (*Intersection Over Union*), siendo que este último parámetro corresponde a la intersección entre la *bounding box* real (imagen etiquetada) y la *bounding box* estimada (imagen procesada por el modelo). Un parámetro IoU alto, indica que se está evaluando las detecciones con una

precisión muy alta de su posición, la caja delimitadora de las vacas debe ser muy exacta cuando se elige un IoU de 0.95 por ejemplo.

El parámetro área, corresponde al tamaño de los objetos detectados respecto a la resolución total de la imagen original. Cuando indica "*all*" es porque existen objetos de varios tamaños en la imagen procesada, mientras que cuando muestra otro tipo de resultado, corresponde al tamaño de detecciones que predomina sobre las otras.

En la evaluación, se toman rangos de IoU distintos, desde 0.50 hasta 0.95. Donde la última columna indica el mAP (*Mean Average Precision*), que se calcula con el promedio de los distintos IoU.

Se considera que el funcionamiento es bueno, donde la precisión y *recall* se mantienen en valores coherentes a los gráficos anteriores para las diferentes áreas y variaciones del parámetro IoU.

Segundo modelo: Detectron 2

Detectron 2 es un modelo de reconocimiento en imágenes desarrollado por Facebook. Al venir pre-entrenado ya tiene un rendimiento aceptable. Pero para ver si es apto para nuestro proyecto, se lo entrena con el mismo set de datos que se entrenaron los otros modelos (ver imagen 40).

Para este entrenamiento, se utilizó un *Notebook* en Google Collab, donde allí se realiza el entrenamiento usando una GPU y CPU externa.

Esto se realiza tras descargar las dependencias necesarias y modificar ciertos parámetros del modelo pre-entrenado para nuestro caso, cambiando las clases a detectar, los nombres de los archivos y la cantidad de iteraciones. También se debería realizar el etiquetado de imágenes previo al entrenamiento, pero esto ya estaba realizado anteriormente.

Entonces desde Google Collab se toman las fotos de la unidad de drive, se instalan las dependencias necesarias para Detectron 2 y por último procede a ejecutar los archivos Python modificados para realizar el entrenamiento en reconocimiento de vacas.

Este entrenamiento se realizó originalmente con 300 *epochs*, aunque el modelo converge antes que esto pase y se termina el entrenamiento con alrededor de 250 *epochs*.

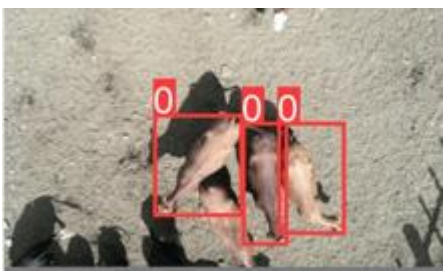


Imagen 40 - Captura de pantalla del modelo detectando vacas.

Evaluación del modelo Detectron 2

Se evalúa este modelo con la misma metodología e imágenes utilizadas para evaluar el modelo de Yolo, donde se muestran los parámetros más relevantes del modelo (ver imagen 41).

```
Accumulating evaluation results...
DONE (t=0.01s).
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.598
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.740
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.709
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.050
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.280
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.812
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.115
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.725
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.725
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.050
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.280
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.740
```

Imagen 41 - Evaluación del modelo Detectron 2.

De esta evaluación, se puede concluir que los resultados son bastante peores que para el modelo de Yolov8, ya que prácticamente todos los parámetros de evaluación redujeron su valor.

La métrica de precisión media (mAP) para un rango de IoU de 0.50 a 0.95 sobre todas las áreas y considerando un máximo de 100 detecciones (*maxDets*), disminuye de 0.730 a 0.598, lo cual indica una caída en la precisión general del modelo. También la precisión a IoU de 0.50, que permite evaluaciones más permisivas, muestra una disminución de 0.900 a 0.740. Esto sugiere que el modelo es menos preciso que el de Yolov8 en identificar correctamente los objetos con una superposición mínima del 50%.

En cuanto a la precisión a IoU de 0.75, que es una evaluación más estricta, también se observa una disminución de 0.830 a 0.709, lo cual indica que el modelo no es capaz de mantener una buena precisión a niveles más altos de superposición entre los cuadros delimitadores predichos y los reales.

En las evaluaciones segmentadas por tamaño de los objetos, la precisión para objetos pequeños desciende de 0.100 a 0.050, lo que indica que el modelo tiene mayores dificultades para detectar objetos de menor área. De manera similar, la precisión para objetos medianos y grandes también se reduce de 0.320 a 0.280 y de 0.950 a 0.800, respectivamente. Esto implica que el modelo ha perdido precisión en la detección de objetos grandes, aunque sigue siendo más efectivo en esa categoría que en las otras dos.

En cuanto a las métricas de *recall* promedio (*average recall*), se mantiene sin cambios la métrica para un máximo de una detección por imagen, con un valor de 0.115. Sin embargo, cuando se permiten hasta 10 detecciones, el *recall* mejora levemente, pasando de 0.800 a 0.812. Esta mejora, aunque pequeña, indica una ligera mayor capacidad del modelo para detectar más objetos correctamente cuando se permite mayor flexibilidad en las detecciones. Por el contrario, el *recall* para un máximo de 100 detecciones desciende de 0.900 a 0.725, lo que refleja una reducción en la capacidad del modelo para detectar correctamente un mayor número de objetos.

El *recall* para objetos pequeños muestra una mejora de 0.230 a 0.280, lo cual es un indicio positivo, pero este es el único punto a favor para este modelo de detección.

En resumen, los resultados del modelo Detectron 2, son bastante peores que los de Yolov8, con caídas significativas en las métricas de precisión y *recall* para la mayoría de las categorías, con la excepción de un leve aumento en el *recall* para objetos pequeños. Estas diferencias pueden deberse a la arquitectura interna de cada modelo de detección o a los hiperparámetros que traen por defecto debido a su pre-entrenamiento.

Tercer modelo: DETR (Detection Transformer)

El modelo DETR, es el que figura como primero en el *ranking* de Hugging Face a fecha de 2 de febrero de 2024, por lo que es un gran candidato junto a los otros dos modelos para ser el que nos de la precisión necesaria para realizar un conteo correctamente. Este modelo también es *open source* y se puede instalar el modelo desde su repositorio.

En este entrenamiento nuevamente se utilizó Google Collab, debido a que los desarrolladores de este modelo otorgan un código ya hecho donde simplemente se debe ajustar a nuestro caso y agregar nuestro *dataset*. Fue necesario subir nuestro *dataset* a un sitio llamado Roboflow, donde se etiquetan las imágenes para el formato CoCo que es el utilizado por DETR (ver imagen 42).

Imagen 42 - Captura de pantalla de la plataforma de etiquetado RoboFlow.

Teniendo esto, se debe exportar las imágenes junto a su etiquetado en el formato CoCo.

Al exportarlas, se genera una *key* para la API de Robo Flow, por lo que se agregan las siguientes sentencias en Google Collabs para acceder al *dataset*: (ver imagen 43).

```
!pip install roboflow

from roboflow import Roboflow
rf = Roboflow(api_key="[redacted]")
project = rf.workspace("manunava11").project("cows-o8e9i")
version = project.version(1)
dataset = version.download("coco")
```

Imagen 43 - Asignación del dataset al entrenamiento en Google Collabs.

Ejemplo del modelo DETR corriendo en una imagen del *dataset* (ver imagen 44).

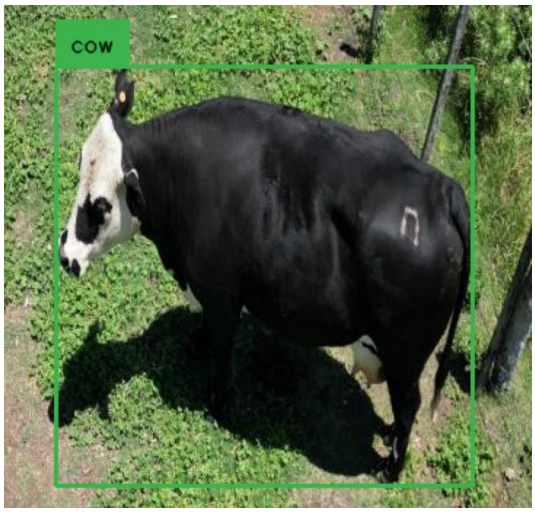


Imagen 44 - DETR detectando una vaca.

Primer entrenamiento (50 *epochs*) (ver imagen 45).

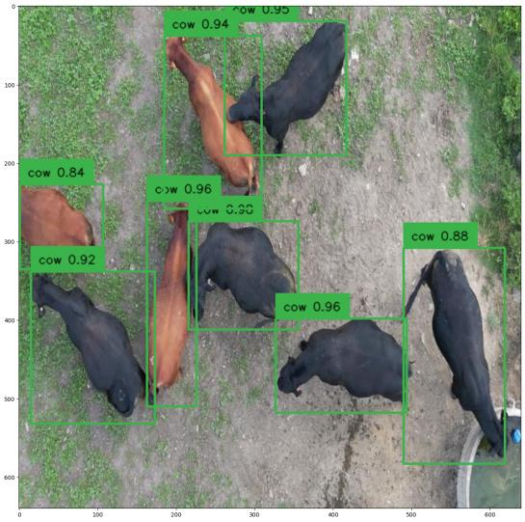


Imagen 45 - DETR detectando varias vacas.

Tras el primer entrenamiento se obtuvieron los siguientes resultados en precisión (ver imagen 46).

```
Running evaluation...
100% 2/2 [00:01<00:00,  1.25it/s]
Accumulating evaluation results...
DONE (t=0.04s).
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.634
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.867
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.674
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.101
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.500
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.703
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.170
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.705
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.705
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.100
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.500
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.788
```

Imagen 46 - Evaluación del modelo de DETR tras el primer entrenamiento.

Es esperable que la precisión mejore si se aumenta la cantidad de *epochs* que se realizan en el entrenamiento, por lo que para llegar a un resultado con mejor entrenamiento se pasará a 300 *epochs* (ver imagen 47).

Entrenamiento de 300 *epochs*:

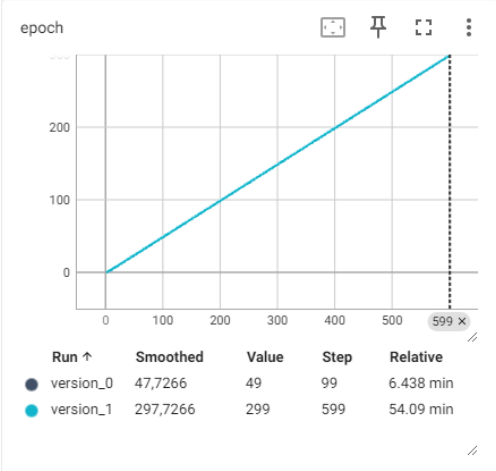


Imagen 47 - Gráfica de epochs en función de pasos.

Por último, se entrena el modelo DETR con el mismo *dataset* pero durante más ciclos. Es un entrenamiento 6 veces mayor al anterior, por lo que los resultados deberían tender a mejorar, aunque sea un poco.

Los resultados que se obtienen son levemente mejores al entrenamiento anterior. Una manera de mejorarlo aún más es agrandando el set de datos. Ya que recurrir a más *epochs*, no brindará el valor que se espera, ya que el modelo ya converge en su entrenamiento y no cambiará como se desea seguir entrenando (ver imagen 48).

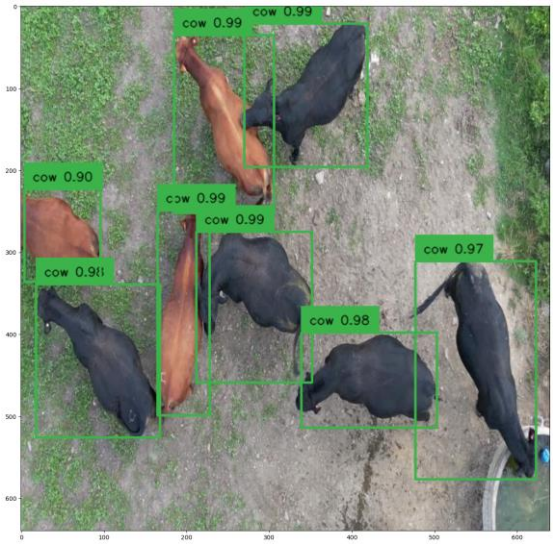


Imagen 48 - DETR detectando vacas con mayor confianza tras el segundo entrenamiento.

Al ejecutar el modelo con este nuevo entrenamiento en la misma imagen que para el ciclo anterior, se obtiene lo siguiente:

Evaluación del modelo DETR

En estos datos se aprecia que hubo una leve mejoría respecto al entrenamiento anterior, lo cual es lógico pero se esperaba una mejoría aún mayor (ver imagen 49).

```
Running evaluation...
100% ████████████████████████████████████████ 2/2 [00:00<00:00,  3.75it/s]
Accumulating evaluation results...
DONE (t=0.01s).
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.627
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.840
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.704
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.050
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.300
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.712
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.135
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.725
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.725
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.050
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.300
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.829
```

Imagen 49 - Evaluación final del modelo DETR entrenado.

6.4.3 Comparativa de los tres modelos

Posteriormente a que se entrenaron los tres modelos considerados más aptos para el proyecto, se debe evaluar cuál cumple de mejor manera con los requerimientos de este trabajo. Los aspectos que evaluar son la precisión, el *recall*, y la facilidad a de manipulación. Los tres modelos son fácilmente manipulables y compatibles con otras librerías útiles para *tracking* de objetos, por lo que en este caso no es importante considerar dicho aspecto.

Se genera una tabla comparativa, donde observar el resultado para cada parámetro y señalar cuál es el mejor de los modelos por parámetro.

Parámetro	Yolo V8	Detectron 2	DETR	Mejor resultado
AP @ IoU=0.50:0.95 | area=all	**0,73**	0,598	0,627	Yolo V8
AP @ IoU=0.50 | area=all	**0,9**	0,74	0,84	Yolo V8
AP @ IoU=0.75 | area=all	**0,83**	0,709	0,704	Yolo V8
AP @ IoU=0.50:0.95 | area=small	**0,1**	0,05	0,05	Yolo V8
AP @ IoU=0.50:0.95 | area=medium	**0,32**	0,28	0,3	Yolo V8
AP @ IoU=0.50:0.95 | area=large	**0,95**	0,8	0,712	Yolo V8
AR @ IoU=0.50:0.95 | area=all | maxDets=1	**0,115**	0,115	0,115	Yolo V8
AR @ IoU=0.50:0.95 | area=all | maxDets=10	0,8	**0,812**	0,812	Detectron 2
AR @ IoU=0.50:0.95 | area=all | maxDets=100	**0,9**	0,725	0,725	Yolo V8
AR @ IoU=0.50:0.95 | area=small	0,23	**0,28**	0,28	Detectron 2
AR @ IoU=0.50:0.95 | area=medium	**0,3**	0,28	0,3	Yolo V8
AR @ IoU=0.50:0.95 | area=large	**0,9**	0,74	0,829	Yolo V8

Tabla 1 - Comparativa de modelos pre-entrenados.

En esta tabla se resumen las evaluaciones de precisión y *recall* para cada modelo, el que mejor se desempeña de los tres modelos, es el de Yolov8.

Es importante recalcar que todos los modelos convergieron en su entrenamiento, es decir llegaron al punto óptimo de entrenamiento, donde seguir entrenando no es redituable. Todos los modelos presentan *'early stopping'* que es una manera de parar el entrenamiento cuando tras pasar varias *epochs* no se logra mejoras en el set de validación.

Probablemente, la diferencia en la precisión y *recall* de los modelos, se deba a la arquitectura interna que tienen, o el preentrenamiento con el que cuentan estos modelos, siendo Yolov8 el que mejor se adaptó a nuestro conjunto de datos.

A partir de estas evaluaciones, se determina que Yolov8 es el modelo a utilizar y el cual se continúa entrenando para lograr los mejores resultados posibles.

6.4.4 Comparativa entre Yolov8x y Yolov8n

Habiendo definido que se va a utilizar un modelo pre-entrenado de Yolov8, se debe elegir qué modelo pre-entrenado utilizar. Existen varias versiones que varían principalmente en el tamaño del modelo y la cantidad de imágenes en las que se entrenaron previamente.

Las versiones recientes de esta serie, Yolov8x y Yolov8n, destacan por ofrecer diferentes equilibrios entre precisión y velocidad. Esta sección aborda las características distintivas de ambos modelos, analizando sus ventajas y desventajas, y justifica la preferencia por Yolov8x en escenarios donde la precisión es el factor determinante.

Yolov8x

Yolov8x representa la variante más avanzada de la serie Yolov8, diseñada para maximizar la precisión en la detección de objetos. Su arquitectura es más compleja, y tiene un mayor número de parámetros comparado a las otras alternativas, lo que permite ofrecer un rendimiento superior en términos de precisión.

Este modelo está optimizado para proporcionar una mayor precisión, lo que resulta esencial en aplicaciones donde se requiere una identificación precisa y confiable. Además, su arquitectura permite un mejor manejo de contextos con múltiples objetos. También se destaca en la detección de objetos pequeños, lo que lo convierte en una opción ideal para casos como este donde los objetos pueden estar a una gran distancia (altura).

Sin embargo, esta mayor precisión implica también requisitos computacionales más elevados, demandando una mayor capacidad de procesamiento y memoria. También, la velocidad de procesamiento tiende a ser mayor en comparación con modelos más ligeros.

Yolov8n

Yolov8n es una variante más ligera de la serie, diseñada para ofrecer un equilibrio entre velocidad y precisión. Su arquitectura menos compleja y un número reducido de parámetros lo hacen ser el modelo más rápido de toda la serie de modelos de Yolov8.

Este modelo ofrece una rápida velocidad de inferencia, lo que lo hace particularmente adecuado para aplicaciones en tiempo real o para sistemas con recursos computacionales limitados. Además, su menor demanda de memoria y procesamiento facilita su implementación en hardware no tan potente.

Sin embargo, Yolov8n presenta una precisión menor en comparación con Yolov8x, especialmente en contextos complejos. Además, puede enfrentar dificultades en la detección eficaz de objetos pequeños o en situaciones con múltiples objetos y oclusiones.

Conclusión

La elección entre Yolov8x y Yolov8n se da debido a las necesidades que tenemos en este proyecto. Yolov8x es la opción más adecuada debido a su capacidad para ofrecer resultados precisos y confiables. Aunque requiere mayor capacidad de procesamiento y presenta una velocidad de inferencia menor, la precisión superior de Yolov8x asegura un alto nivel de exactitud, lo cual es crucial en este proyecto.

También es importante recalcar que se cuenta con los recursos computacionales suficientes como para utilizar un modelo más pesado.

6.4.5 Recomendaciones para el entrenamiento óptimo de modelos Yolov8 [28]

Introducción

El modelo Yolov8, desarrollado por Ultralytics, es una de las herramientas más avanzadas para la detección de objetos en imágenes y vídeos. Su entrenamiento efectivo requiere un enfoque cuidadoso que maximice tanto la precisión como el rendimiento, considerando diversos factores que influyen en la calidad de los resultados obtenidos. Este apartado se basa en la guía de Ultralytics donde se exponen las mejores prácticas recomendadas para optimizar el proceso de entrenamiento de modelos Yolo.

Preparación del *dataset*

La calidad del *dataset* utilizado es fundamental para el éxito del entrenamiento de cualquier modelo de detección de objetos. Ultralytics enfatiza la importancia de disponer de un *dataset* que no solo sea amplio, sino también variado y bien etiquetado.

Cantidad de imágenes y balance de clases

Es recomendable contar con un mínimo de 1500 imágenes por clase para garantizar una representación adecuada de cada categoría en el proceso de entrenamiento. Además, es crucial asegurar que cada clase tenga al menos 10000 instancias etiquetadas, lo que ayuda a mejorar la precisión del modelo en la detección de objetos en diferentes contextos.

Diversidad del *dataset*

El *dataset* debe capturar una amplia variedad de condiciones ambientales, como diferentes horas del día, estaciones y climas. Esto permite que el modelo generalice mejor a situaciones no vistas durante el entrenamiento.

Precisión y consistencia en el etiquetado

La precisión en el etiquetado es crucial. Ultralytics recomienda revisar visualmente las etiquetas al inicio del entrenamiento para detectar posibles errores que podrían afectar negativamente el rendimiento del modelo. La consistencia en las etiquetas, asegurando que los objetos están correctamente delimitados y sin espacios significativos entre los objetos y sus cajas delimitadoras, es esencial para lograr una alta precisión.

Configuración del entrenamiento

Una configuración adecuada del entrenamiento es esencial para maximizar el rendimiento del modelo. Ultralytics sugiere comenzar con los valores predeterminados antes de realizar ajustes, lo que permite establecer una línea base fiable.

Número de *epochs*

Ultralytics recomienda un entrenamiento inicial de 300 *epochs*. Sin embargo, si no se detecta sobreajuste, el entrenamiento puede extenderse hasta 600 o más épocas para mejorar aún más la precisión. Es importante monitorear el proceso para ajustar el número de *epochs* según sea necesario.

Resolución de imagen

El tamaño de las imágenes debe seleccionarse de acuerdo con la naturaleza del *dataset*. Por ejemplo, para el *dataset* COCO, se utiliza una resolución de 640 píxeles. Sin embargo, en casos donde predominan los objetos pequeños, puede ser beneficioso emplear resoluciones más altas para mejorar la detección.

Conclusión

El éxito en el entrenamiento de modelos Yolov8 depende en gran medida de la selección adecuada del *dataset*, la configuración precisa del modelo y la monitorización continua del proceso de entrenamiento. Siguiendo las recomendaciones de Ultralytics, los desarrolladores pueden maximizar la precisión y la eficiencia de sus modelos, asegurando un rendimiento óptimo en una amplia gama de aplicaciones de detección de objetos.

Estas recomendaciones no solo mejoran los resultados del entrenamiento, sino que también optimizan los recursos disponibles, permitiendo implementaciones eficaces tanto en entornos de alta capacidad como en dispositivos con recursos limitados. Para una guía más detallada y actualizada, se recomienda consultar la documentación oficial de Ultralytics.

6.4.6 Entrenamiento del modelo

En este proyecto, uno de los aspectos más relevantes, es tener un modelo correctamente entrenado, que asegure un buen funcionamiento. En este apartado se describe cómo se entrenó el modelo, siguiendo las recomendaciones que se acaban de describir.

Entrenamiento de acorde a las recomendaciones

Como se menciona anteriormente, el modelo de detección utilizado se basa en modelos pre-entrenados de Yolo, elaborados por Ultralytics. Particularmente, se utiliza Yolov8x, el cual es el modelo con más capacidades de la versión 8, el cual cuenta con una mayor cantidad de hiperparametros y entrenamiento.

Ultralytics cuenta con una documentación la cual incluye una serie de recomendaciones al momento de entrenar, entre las que incluye una cantidad de imágenes, cantidad de instancias por clase, variedad y consistencia.

Como cantidad de imágenes recomendadas, Ultralytics considera que se debe tener un mínimo de 1500 por clase. Este número fue ampliamente superado, ya que, en nuestro modelo, se

incluyen 2500 imágenes con vacas. Donde es importante destacar que todas estas imágenes fueron tomadas por el equipo.

Se recomienda un mínimo de 10.000 instancias de clase etiquetadas, el equipo consiguió superar este número en aproximadamente un 200%, ya que contamos con un total de 20.369 instancias de vacas etiquetadas (ver imagen 50).

```
[Running] python -u "c:\Users\Manuel\Desktop\Carpeta Visual\Framesaltura.py"
Total de instancias en el dataset: 20369
```

Imagen 50 - Total de vacas etiquetadas.

También se recomienda variedad en las imágenes, es decir, tomarlas con diferente iluminación, con diferentes fondos, etc. Esto se considera que se cumple con creces, ya que, como se menciona anteriormente, el equipo tomó imágenes en diferentes campos y momentos, lo cual garantiza una gran variedad.

6.4.7 Configuración del entrenamiento

También existe una serie de recomendaciones para el momento en el cual se entrena sobre el set de imágenes, estas recomendaciones incluyen lo siguiente:

Entrenar por un total de 300 *epochs* (número de iteraciones sobre la totalidad del set de datos): Este número es un intermedio que se elige, ya que luego de entrenar, se puede verificar viendo los parámetros de evaluación y los gráficos, y se puede determinar si hubo sobreajuste (*overfitting*). En nuestro caso, al elegir 300 épocas, el modelo realiza un *early stopping* y converge antes de completar las 300 recorridas sobre el set de datos.

Tamaño de lote (*batch-size*): Ultralytics recomienda utilizar el máximo tamaño de lote que se pueda permitir según el poder computacional con el que se cuente. Para este trabajo se utiliza siempre entrenamientos con GPU, ya sea desde una computadora local, donde se utiliza de una tarjeta gráfica doméstica, o a través de plataformas, más particularmente, utilizando una suscripción *Pro* en Google Collab, lo cual nos permite acceder a un entorno con GPU optimizado justamente para este tipo de trabajos.

Procedimiento

Siguiendo todas las recomendaciones mencionadas, se entrena el modelo. En el transcurso de este proyecto, se realizaron una gran cantidad de entrenamientos.

Para garantizar un entrenamiento lo mejor posible, también se toman las siguientes medidas:

Separar el set de datos en un 70% para entrenamiento y un 30% para validación. Al tener una gran cantidad de imágenes, el equipo se permite incluir una cantidad considerable para validación, lo cual no suele ser frecuente debido a la dificultad que conlleva conseguir imágenes y etiquetarlas.

Realizar una API en Hugging Face para probar rápidamente el funcionamiento del modelo ante imágenes. El equipo utiliza Hugging Face para tener cargado el modelo de detección y poder subir imágenes las cuales procesa rápidamente y arroja un resultado. En caso de no tener

resultados satisfactorios, se agrega la imagen al set de datos, también alojado en Hugging Face.

Ciclos de re-entrenamiento con *frames* mal procesados. Como se menciona, el equipo guarda cada *frame* en el cual no se obtienen los resultados esperados, por lo que, en búsqueda de una mejora continua, se genera un ciclo de re-entrenamiento con dichos *frames*, para así lograr un modelo que funcione de la manera esperada en todos los casos.

6.3 *Tracking* para seguimiento de objetos

La herramienta de *trackeo* para modelos Yolo de Ultralytics [29] ofrece una solución que se puede utilizar para el seguimiento de cualquier tipo de objeto y realizar un conteo único sobre ellos. Esta herramienta se destaca por su capacidad para realizar múltiples tareas especializadas, tales como identificación, predicción y segmentación, las cuales permiten un análisis detallado de cada instancia de vaca en el video. Además, la herramienta está diseñada para modelos de la arquitectura de Yolo, modelo el cual entrenamos con nuestro set de datos, lo que garantiza una precisión óptima en el conteo único de cada vaca.

El *tracking* nos permite tener un seguimiento de las detecciones en el video, y mediante *Hashing*, le asigna un ID a cada detección según la posición en la que se encuentra y la disposición de los pixeles, lo cual nos permite realizar un conteo, ya sea por la duración que lleva determinado ID en el campo de visión, por pasar alguna región de la imagen, o simplemente por aparecer. Estas modalidades de conteo se discuten posteriormente en este informe.

Ejemplo de *tracking* en nuestro caso (ver imagen 51).

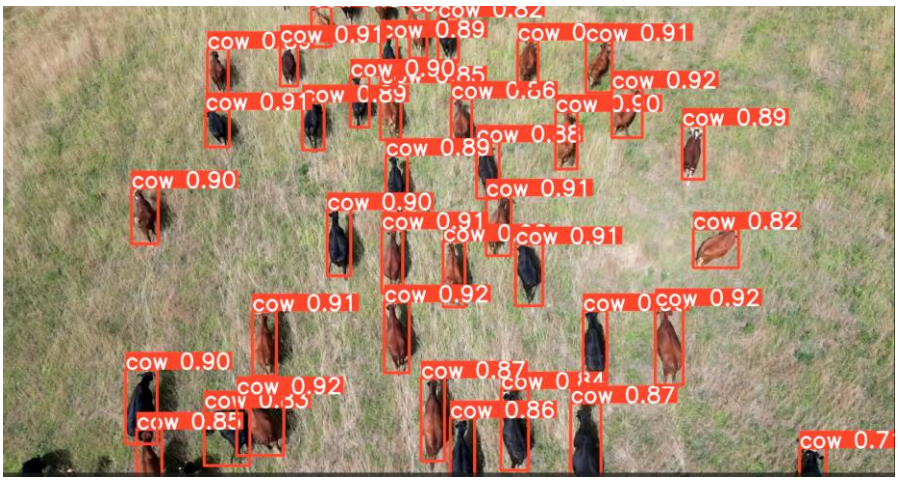


Imagen 51 - Captura de pantalla de un video de tracking.

Esto es un fotograma del video. En cada fotograma del video se ajusta una caja que rodea el animal detectado.

6.3.1 Comparativa entre *tracking* con Botsort y ByteTrack

Introducción

El seguimiento de objetos en videos es algo fundamental para este proyecto, ya que es lo que permite realizar una cuenta. Entre las tecnologías más usadas para esto se encuentran Botsort y ByteTrack. A continuación, se presenta una comparativa entre estos dos sistemas de tracking, analizando sus principales características. [30]

Botsort

Botsort se caracteriza por emplear técnicas avanzadas para el seguimiento de los objetos. Su diseño prioriza la precisión en el seguimiento de objetos, lo que lo convierte en una opción destacada para escenarios que requieren un alto grado de exactitud en la identificación y rastreo de elementos a lo largo de secuencias de vídeo. Este sistema ofrece una precisión elevada, incluso en situaciones complejas como las aglomeraciones de objetos, lo cual es una gran ventaja considerando como se suele amontonar el ganado.

Una de las principales ventajas de Botsort es su capacidad para adaptarse a diferentes tipos de entornos y condiciones de iluminación, gracias a la robustez de su modelo. A su vez, Botsort tiene la capacidad de mantener la identidad de un objeto rastreado incluso cuando este se superpone con otros elementos en la secuencia.

Sin embargo, el uso de Botsort tiene una desventaja en términos de requisitos computacionales, ya que sus modelos complejos requieren una considerable capacidad de procesamiento. Esta demanda de recursos puede limitar su implementación en dispositivos con capacidades más reducidas. Además, su enfoque en la precisión puede afectar la velocidad de procesamiento, sin embargo, en nuestro caso no es inconveniente ya que se tiene un considerable poder de cómputo a través de AWS.

ByteTrack

ByteTrack se presenta como una alternativa eficiente y ligera en términos de consumo de recursos para el seguimiento de objetos. A diferencia de Botsort, este método se enfoca en la velocidad del procesamiento, lo que lo hace adecuado para aplicaciones donde la rapidez es crucial. Este sistema está diseñado para procesar videos a alta velocidad, lo que lo convierte en una solución ideal para entornos en los que el tiempo de respuesta en tiempo real es un factor determinante.

La principal ventaja de ByteTrack radica en la optimización en el uso de recursos, ya que utiliza técnicas optimizadas que permiten su implementación en sistemas con capacidades computacionales limitadas.

Sin embargo, a pesar de sus beneficios en términos de velocidad y eficiencia, ByteTrack puede presentar limitaciones en cuanto a la precisión del seguimiento, especialmente en situaciones de aglomeramiento. Además, este método puede ser menos efectivo en condiciones de iluminación variables o en entornos con alta complejidad.

La elección entre Botsort y ByteTrack debe realizarse considerando las necesidades específicas del proyecto. Mientras que ByteTrack se distingue por su velocidad y eficiencia, Botsort sobresale en escenarios que requieren una alta precisión en el seguimiento de objetos. En aplicaciones como la nuestra donde la exactitud es prioritaria, Botsort resulta más adecuado, a pesar de su mayor demanda computacional. Botsort se posiciona como la opción preferida ya que necesitamos una alta precisión y además contamos con buen cómputo debido a que utilizamos soluciones de AWS.

6.5 Método de conteo de ganado

En el siguiente texto se describe con exactitud la manera en la que se cuentan las vacas, es decir, la inteligencia que se le da al modelo de detección para realizar un conteo, agregando también los diferentes métodos que se usaron y cómo fue evolucionando. Este método de conteo fue variando en base a conclusiones sacadas de visitas al campo, disponibles en el Anexo 1: Visitas al campo.

6.5.1 Primer método de conteo

El primer método que se implementa para realizar un conteo es el de generar una línea en la imagen, donde al pasar las cajas delimitadoras (*bounding boxes*) de cada detección se cuentan como una instancia más. Este método es útil cuando se tiene un modelo que puede percibir falsos positivos, ya que para ser contadas, las falsas detecciones deben persistir hasta pasar la barra. El problema de este método es que si una vaca ingresa al campo de vista de la cámara, pero sale por un costado, no va a lograr alcanzar la línea y no es contada, por lo que no es precisa esta forma de contar (ver imagen 52).

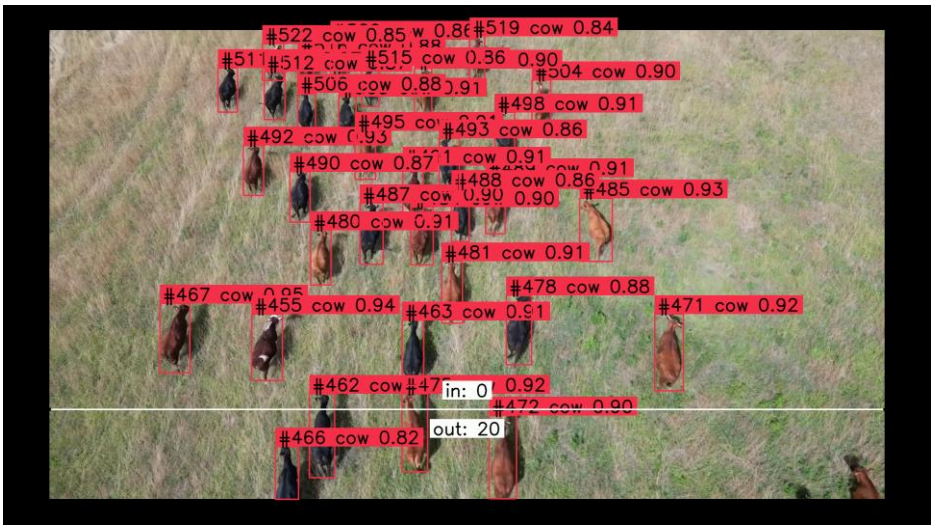


Imagen 52 - Primer metodología de conteo.

En esta imagen, se observa el primer prototipo de conteo de ganado. Este prototipo fue realizado en los inicios del proyecto con el fin de constatar si era válido técnicamente el conteo de ganado utilizando *machine learning*. Como primera iteración, tiene defectos visuales y no representa una buena experiencia para el usuario el cual vea un conteo.

Por lo que, siguiendo la misma metodología, el conteo utilizando una barra, se implementa una nueva versión con mejoras visuales (ver imagen 53).

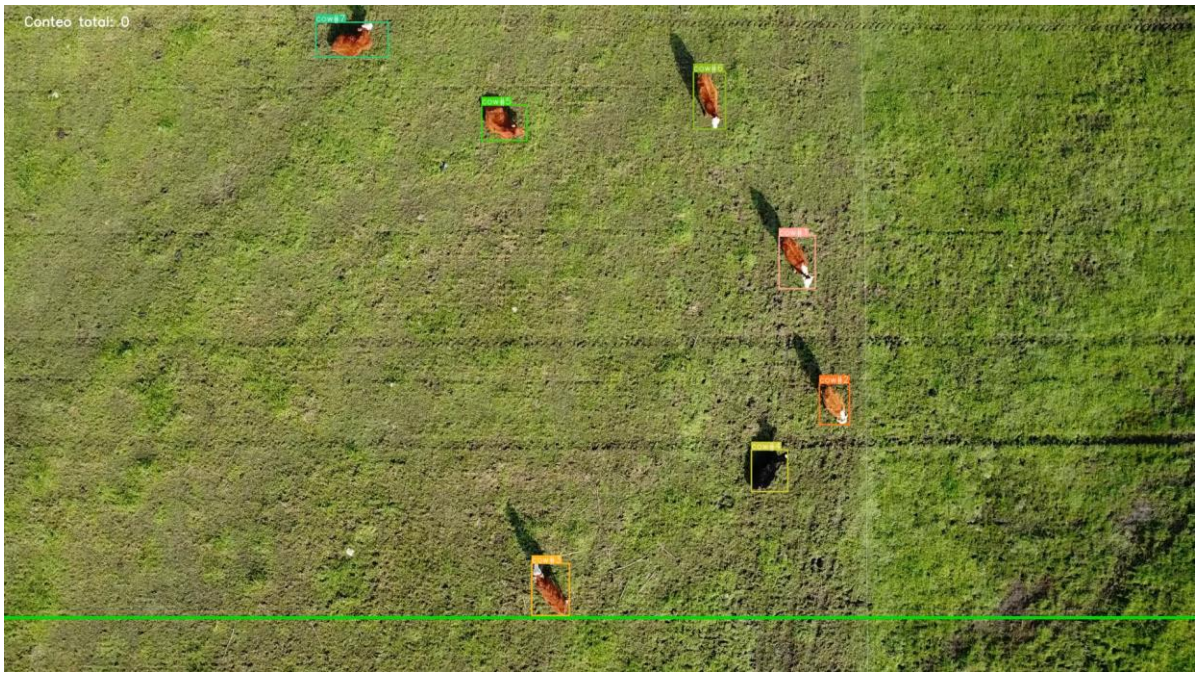


Imagen 53 - Primer metodología de conteo con mejoras visuales.

El conteo se realiza utilizando *tracking* en cada animal detectado. Si este animal es una vaca y su recuadro pasa hacia el otro lado de cierta recta establecida dentro de la imagen, se la cuenta como una más.

Para el correcto funcionamiento, el dron se debe estar moviendo en la misma dirección que el ganado y a una mayor velocidad, ya que si una vaca contada, vuelve a aparecer en el cuadro de la imagen y sobrepasa la recta, puede ser contada nuevamente. Esto no debería ser un problema ya que las velocidades que maneja el dron suelen ser significativamente mayores a las de una vaca, pero es un punto a considerar y en el cual se trabajará a futuro.

6.5.2 Método de conteo final

Tras lograr un modelo de detección mejor, que garantice detecciones efectivas, con menor cantidad de falsos positivos, se puede iterar a un método el cual cuente detecciones en cualquier parte del campo de visión. Teniendo en cuenta que el modelo de detección presenta una precisión excelente, y una cantidad de casos detectados sobre casos positivos muy cercana a 1, esta metodología es la que mejor se adapta y mejores resultados genera para este trabajo. Cualquier detección que aparezca en la imagen, sube a la cuenta del total (ver imágenes 54, 55, 56 y 57).

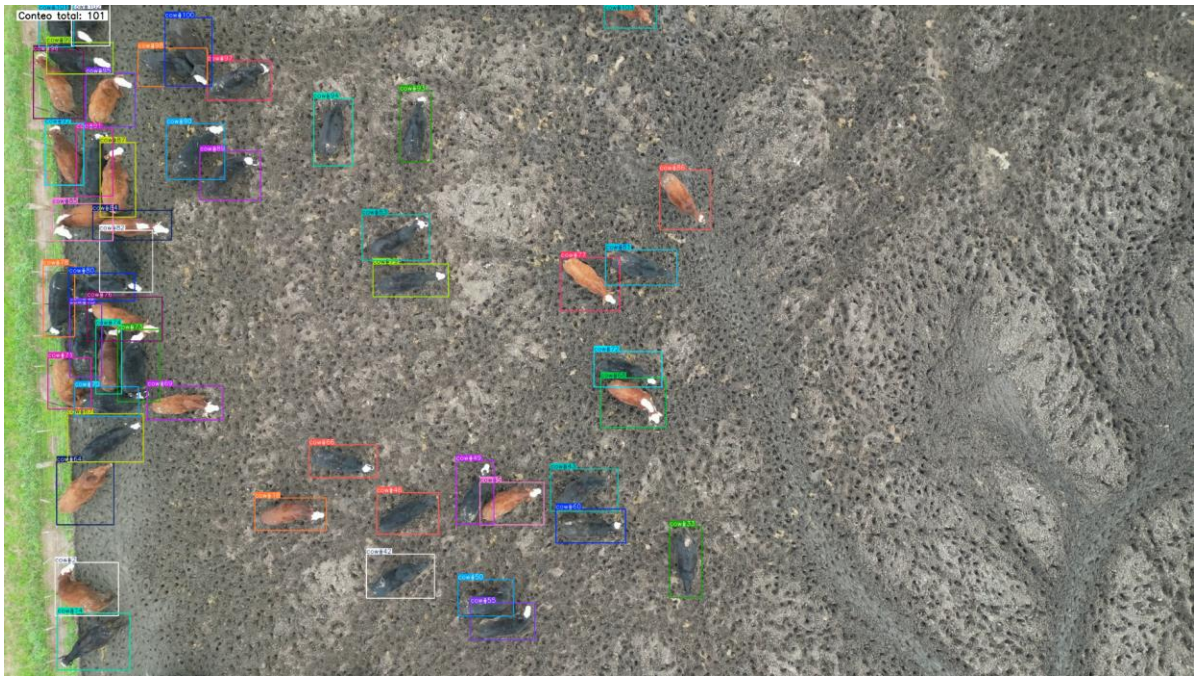


Imagen 54 - Método final de conteo de ganado.

Imagen 55 - Método final de conteo de ganado.

Imagen 56 - Método final de conteo de ganado.

Imagen 57 - Método final de conteo[2] de ganado.

Al igual que el anterior, este método funciona realizando *tracking* sobre el video, a diferencia que este, suma uno a la cuenta cada vez que identifica una nueva instancia de *tracking*, no necesita que pasen por una recta.

Esta iteración asegura que las vacas puedan aparecer en cualquier parte de la imagen, y aunque salgan por algún costado, se cuenten efectivamente.

[2] Se pueden observar más ejemplos de conteo en el Anexo 2: Fragmentos de conteo

Como se menciona anteriormente, el *tracking* identifica según la posición del recuadro del objeto en la imagen y por su disposición de píxeles, guardando esta información en memoria durante un periodo de tiempo del orden de segundos. Existe un riesgo de que en el video se deje de identificar una vaca y luego se la identifique en una posición muy distinta. Este riesgo se considera muy bajo, ya que se mitiga notoriamente debido a los parámetros de precisión y *recall* que tiene el modelo de detección utilizado. Para mitigar aún más este riesgo, es importante realizar solo videos de ida, ya que hacer ida y vuelta, es probable que lleve a un conteo duplicado al ver nuevamente a los mismos animales.

Por lo tanto, utilizaremos solamente videos de ida, comenzando a grabar en el momento que se llega a la parcela y finalizando el video al realizar toda la recorrida por la parcela, asegurando asi que no se cuenten duplicados y optimizando el tiempo de procesamiento y el almacenamiento que ocupa el video.

En conclusión, se va a tomar como pre-condicion hacer un vuelo en una sola dirección, manteniendo la altura y evitando que las vacas se puedan escapar del frame, para que no vuelvan a entrar y haya una duplicación.

6.5.3 Respuesta del modelo predictivo ante variación de altura

En concordancia a lo anterior, ahora se realiza la prueba para el modelo predictivo, donde se filma un grupo de animales y se va elevando la altura del dron lentamente, de esta manera, se puede observar los factores de confianza que muestra para las diferentes alturas, lo que permite saber hasta qué altura es lo suficientemente confiable como para garantizar un buen funcionamiento. Dicha grabación, contempla alturas desde 5 hasta 55 metros. Esta es una comparación justa ya que se entrenó el modelo previamente con imágenes en todas estas alturas (ver imágenes 58, 59, 60, 61, 62, 63, 64 y 65).

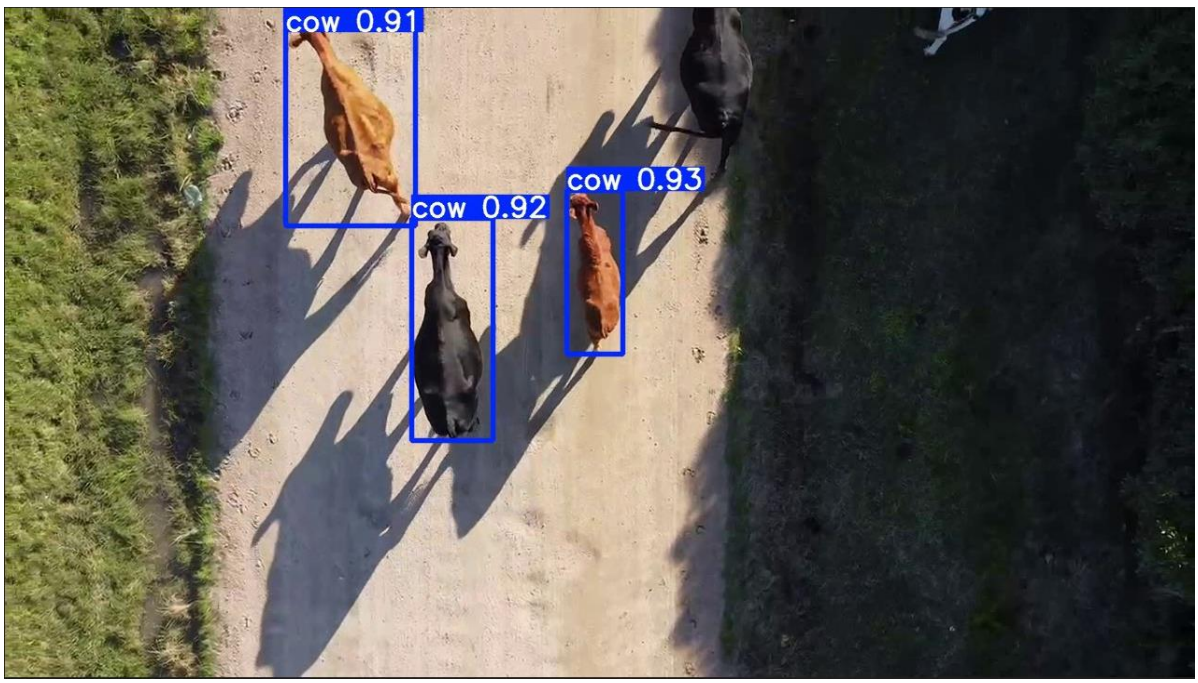


Imagen 58 - Modelo sobre imagen a 10 metros.

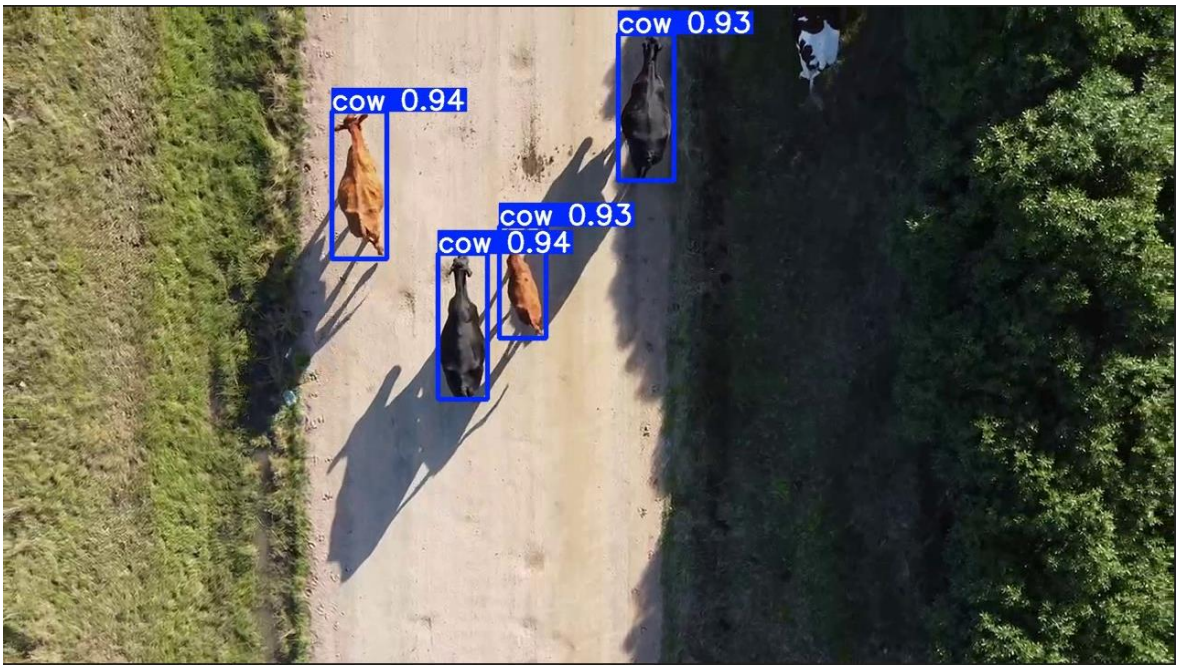


Imagen 59 - Modelo sobre imagen a 15 metros.

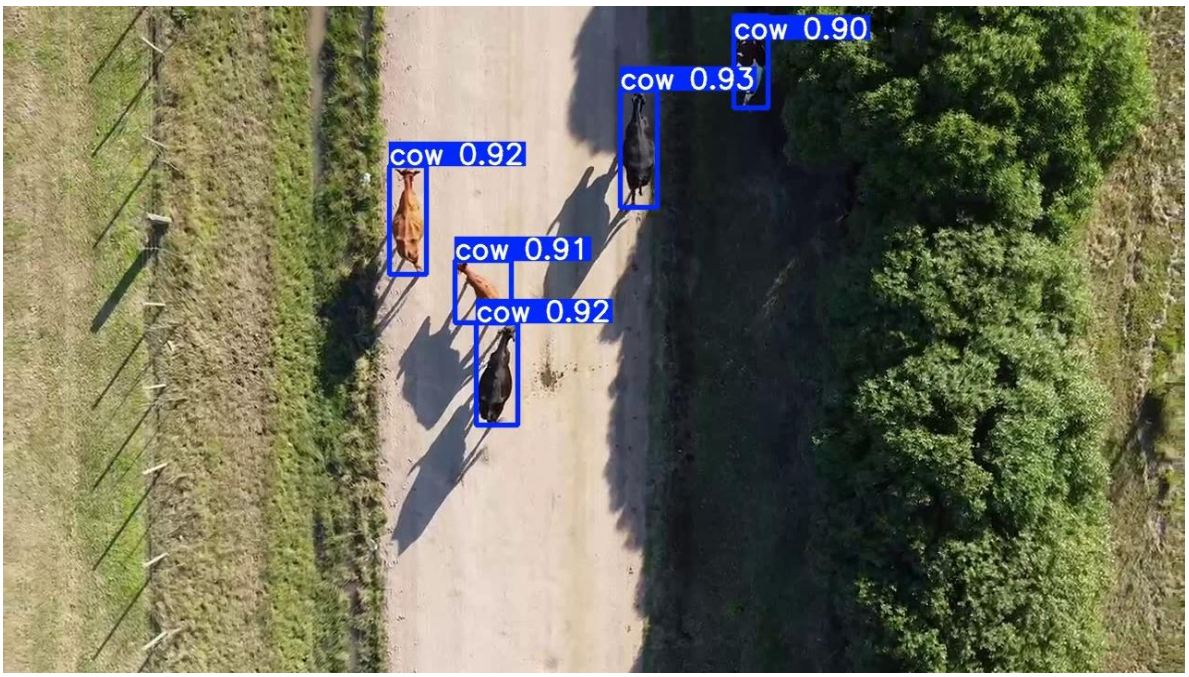


Imagen 60 - Modelo sobre imagen a 20 metros.

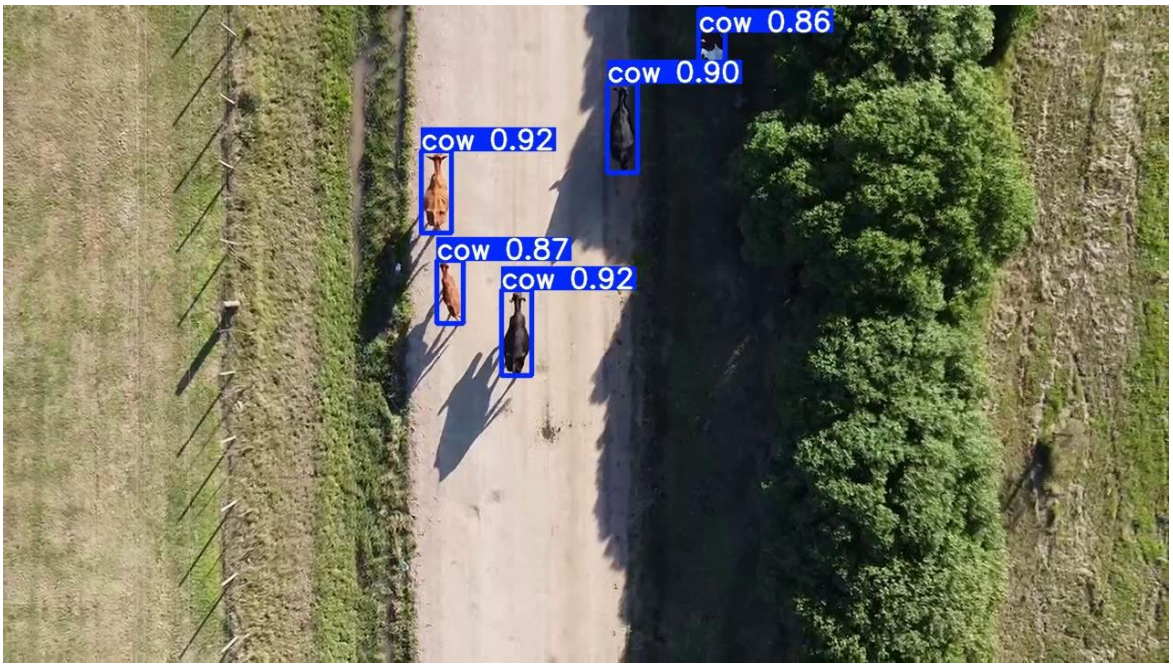


Imagen 61 - Modelo sobre imagen a 25 metros.

Imagen 62 - Modelo sobre imagen a 30 metros.

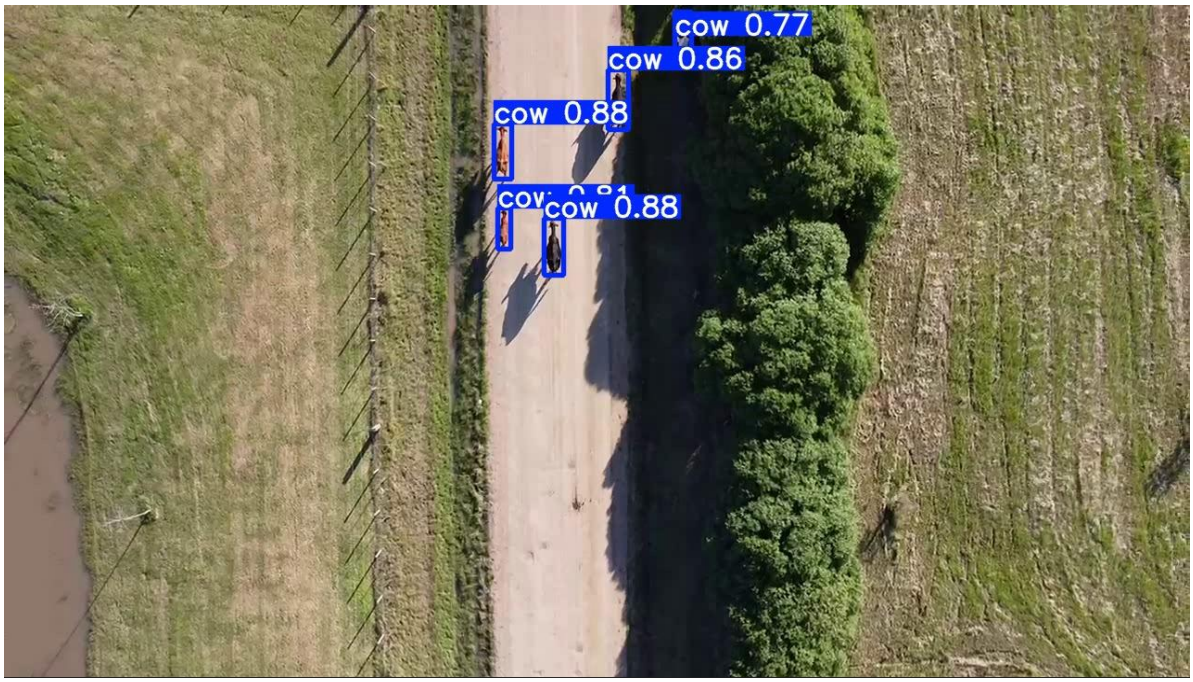


Imagen 63 - modelo sobre imagen a 35 metros.

Imagen 64 - Modelo sobre imagen a 40 metros.

Imagen 65 - Modelo sobre imagen a 50 metros.

Tras estas pruebas y el estudio anterior, se concluye que lo ideal, es recorrer a una altura de aproximadamente 35 metros, para dicho punto se logra un equilibrio entre precisión y tiempo total de recorrido. A partir de dicha altura, comienzan a decaer los factores de confianza que devuelve el modelo de detección.

Aun así, se pueden realizar recorridos a gran altura para ciertas excepciones, para estas imágenes, se considera un umbral de confianza de 0.7, que surge en concordancia a la evaluación del modelo realizado. Sin embargo, se podría bajar dicho umbral a un valor más bajo, realizando un conteo a gran altura, pero con posibilidades de algún falso positivo detectado.

6.6 Problemas y soluciones

Tras entrenar varios modelos, el equipo nota ciertos detalles en los diferentes modelos, los cuales fueron problemas que se tuvieron que investigar para solucionar.

El primero de ellos fue el sobreajuste (*overfitting*). Este problema consiste en que el modelo entrenado, funcione de manera excelente para el set de validación, pero no logre un correcto funcionamiento para imágenes que no conoce, en otras palabras, el modelo tras entrenar no generaliza, sino que adapta sus parámetros para las imágenes que se le dieron en el entrenamiento. Este problema fue muy común cuando se contaba con una cantidad de imágenes etiquetadas pequeña, ya que no existe variedad y el modelo no logra generalizar. Luego se tomaron ciertas medidas, como es el evitar el *leakage*.

El *data leakage*, se da cuando el modelo en su entrenamiento accede a información a la que no debería. Esto puede ser causado, por ejemplo, cuando existe una imagen en el set de entrenamiento muy similar a una en la de validación. Al estar entrenando con una imagen similar a la que se evalúa, las métricas predictivas quedan sobre estimadas, y arrojan resultados erróneos, ya que en realidad el modelo se estaría ajustando a ese caso en particular y no a generalizar.

Para evitar esto, al tomar *frames* de un video, se elige una diferencia temporal suficiente que garantice cambios, del orden de segundos, y considerando que el dron está en movimiento a una velocidad importante, los *frames* son bastante diferentes.

Con estos cuidados, se logra superar el problema de sobre ajuste, logrando un modelo que generalice. Para comprobar que efectivamente se supera el sobre ajuste y el modelo funciona correctamente, se arma un set de pruebas, con imágenes diferentes a las del set de entrenamiento y validación, en conjunto a el uso de imágenes de sets públicos. Esto se describe de mejor manera en el siguiente capítulo.

6.7 Evaluación del modelo

En este capítulo se estudia el grado de cumplimiento de los requerimientos planteados en el capítulo 3, donde se especifica que debe cumplir el modelo de *machine learning*, se desarrollan las pruebas realizadas y los buenos resultados obtenidos.

Además de evaluar métricas tradicionales como precisión y recall, también se ha considerado la capacidad del modelo para operar eficazmente en una variedad de condiciones de campo, tales como diferentes tipos de ganado, variaciones climáticas y cambios en la altura o el ángulo de vuelo del dron. Este enfoque permite asegurar que el modelo pueda mantener su rendimiento en distintos escenarios. Cabe destacar que el modelo fue optimizado para detectar razas de ganado cárnico en Uruguay, entre ellas Angus, Braford, Brangus, Aberdeenangus, Hereford, entre otras. No se asegura el correcto funcionamiento de las detecciones para las razas lecheras o de origen Árabe como las que predominan en Brasil.

Se discute sobre la evaluación del modelo de detección final, mostrando los resultados de precisión, *recall* y de mAP para diferentes rangos de IoU. También se demuestran ciertos indicadores de por qué el modelo converge en su entrenamiento y logra generalizar, sin indicios de *overfitting*.

6.7.1 Evaluación de la precisión y *recall*

Para comenzar, se estudian los parámetros más relevantes al evaluar un modelo, tanto sobre el set de datos de entrenamiento, como el set de datos de validación.

Primero, analizamos los valores de mAP (promedios de precisión media) para los distintos rangos de IoU (Intersección sobre unión). Parámetros que fueron previamente explicados, en este caso los resultados son los siguientes:

$$mAP50-95\ Train\colon 0.933\ mAP50-95\ Val\colon 0.838$$

$$mAP50\ Train\colon 0.995\ mAP50\ Val\colon 0.9905$$

$$mAP75\ Train\colon 0.994\ mAP75\ Val\colon 0.9511$$

Rango de 50-95: El hecho de que el mAP en entrenamiento sea mayor que en validación es normal, ya que el modelo ha sido ajustado directamente en los datos de entrenamiento, por lo que suele rendir mejor en ellos.

La diferencia entre 0.838 (validación) y 0.933 (entrenamiento) es razonable, lo que indica que el modelo está generalizando bien y no parece haber sobreajuste (*overfitting*) de manera significativa.

Cuando IoU es 0.5, ambos valores (validación y entrenamiento) son muy altos, lo que indica que el modelo detecta bien los objetos en general. La diferencia aquí es mínima, lo que sugiere que el modelo está generalizando bien también en validación para un umbral de IoU más bajo.

Con un IoU más estricto (0.75), el modelo sigue funcionando muy bien tanto en entrenamiento como en validación, aunque de nuevo, el rendimiento es ligeramente mejor en el conjunto de entrenamiento, como es esperable.

El modelo está funcionando bien, con un buen equilibrio entre los resultados de entrenamiento y validación. La diferencia en los valores mAP entre entrenamiento y validación es aceptable y no indica problemas evidentes de sobreajuste.

Ahora, analicemos la precisión y el *recall* para ambos conjuntos de datos. Estos parámetros, están directamente relacionados con la eficacia del modelo.

Los resultados para la precisión son los siguientes:

$$Precision\ Train: 0.997\ Precision\ Val: 0.984$$

Recordemos que la precisión es idealmente uno, y este valor se refiere a cuando no existen falsos positivos y se detectan efectivamente todos los verdaderos positivos.

En nuestro caso, se tienen valores muy cercanos al ideal. La precisión en el conjunto de entrenamiento es muy alta (0.997), lo que indica que el modelo hace pocas predicciones incorrectas en este conjunto. La precisión en el conjunto de validación es también alta (0.984), pero ligeramente menor. Esta pequeña diferencia sugiere que el modelo generaliza bien.

Los resultados de *recall* reflejan los siguientes valores:

$$Recall\ Train: 0.994\ Recall\ Val: 0.978$$

El *recall* es similar a la precisión, es idealmente 1 cuando detecta todos los verdaderos positivos, pero la diferencia es que, la razón que hace disminuir este parámetro, es cuando existen falsos negativos (ver imagen 66).

```
Speed: 0.2ms preprocess, 28.2ms inference, 0.0ms loss, 0.8ms postprocess per image
Results saved to [1mruns\detect\val70[0m
mAP50-95 Train: 0.9327826371116442 mAP50-95 Val: 0.8380388568051519
mAP50 Train: 0.9946811774024316 mAP50 Val: 0.9904629654663935
mAP75 Train: 0.993727156856867 mAP75 Val: 0.9514406780611851
Precision Train: [    0.99653] Precision Val: [    0.98416]
Recall Train: [    0.99436] Recall Val: [    0.97758]
```

Imagen 66 - Evaluación para set de entrenamiento y validación.

De los resultados vemos que en el conjunto de entrenamiento (*Train*) hay un alto *recall* de 0.994, lo que significa que el modelo detecta la mayoría de los casos positivos en el entrenamiento, con muy pocos falsos negativos. En el conjunto de validación es algo menor,

de 0.978, lo cual es esperado, pero la diferencia es lo suficientemente pequeña como para afirmar que el modelo también generaliza bien a nuevos datos.

Con estos parámetros tenemos señales de que el modelo está generalizando bien y mostrando resultados bastante buenos. Sin embargo, esto lo podemos estudiar de mejor manera si analizamos los datos que se arrojan al finalizar el entrenamiento del modelo.

6.7.2 Gráficos del entrenamiento

Al finalizar un entrenamiento, los modelos de Yolo automáticamente generan unos gráficos con datos interesantes acerca del entrenamiento (ver imagen 67).

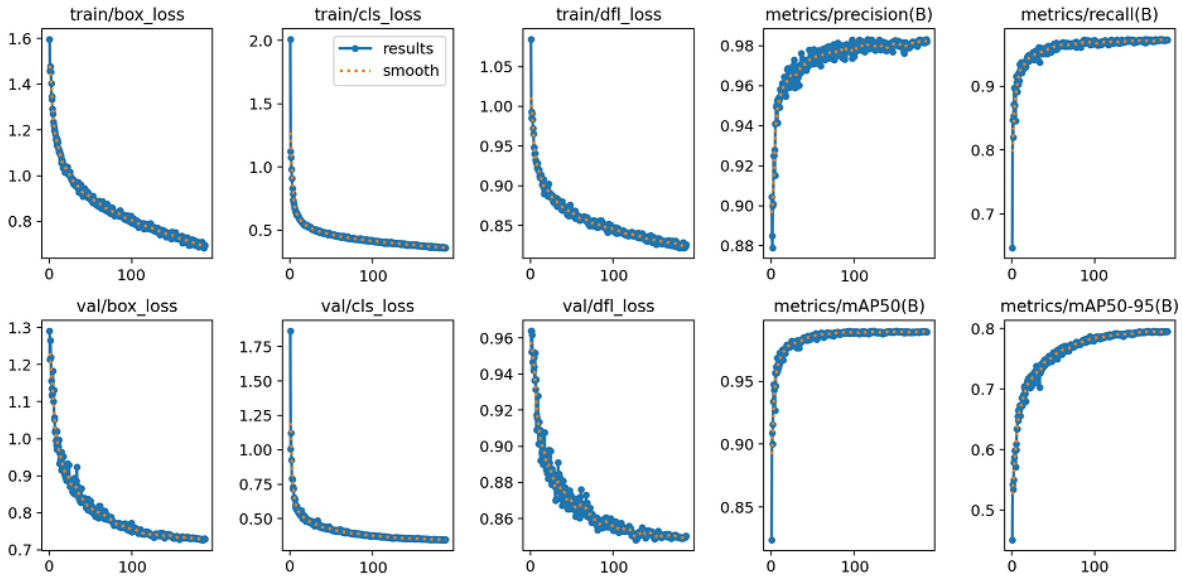


Imagen 67 - Resultados del entrenamiento.

Estos gráficos representan la evolución del modelo a medida que pasaron las iteraciones completas a través de todo el set de datos de entrenamiento y validación.

Cada circulo amarillo representa una *epoch*, y para cada *epoch* se genera un punto en cada gráfico. Los cuatro gráficos de la primera fila corresponden al set de datos de entrenamiento, mientras que la fila de abajo es sobre el set de imágenes de validación.

Resumiendo lo explicado en el marco teórico, cuando hay sobreajuste (*overfitting*), los gráficos de *box_loss*, *cls_loss* y *dfl_loss* decaen bruscamente para el set de entrenamiento y aumentan en el de validación. En este caso no ocurre esto, y como las curvas de validación y entrenamiento muestran formas similares, en conjunto a lo analizado anteriormente, se puede asegurar que nuestro modelo generaliza y no sobre ajusta (ver imagen 68).

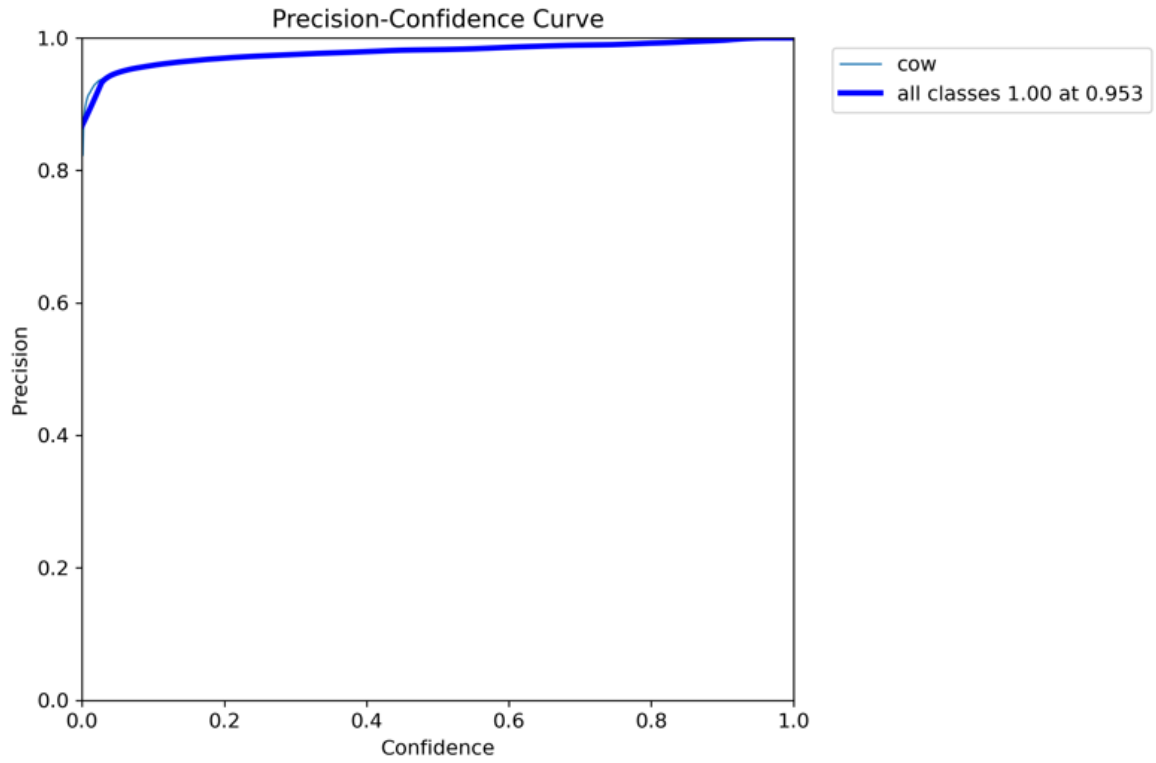


Imagen 68 - Curva de precisión respecto a la confianza.

Esta curva representa la precisión, que como ya se mencionó, mide el total de verdaderos positivos sobre el total de detecciones, respecto al nivel de confianza, que consiste en la probabilidad que estima el modelo de que una detección sea efectivamente una vaca.

Para niveles de confianza bajos (cerca de 0 en el eje X), el modelo hace predicciones menos seguras, lo que puede incluir más falsos positivos, bajando la precisión. Sin embargo, ya con una confianza de 0, la precisión es del orden del 90%.

A medida que aumenta la confianza, el modelo se vuelve más preciso, lo que significa que hace predicciones más seguras.

La leyenda que aparece a la derecha sugiere que, a partir de 0.953 de confianza, el modelo alcanza una precisión de 1, lo que significa que, en ese punto, todas las predicciones que hace son correctas (no hay falsos positivos). Es un buen indicio de que el modelo es muy preciso cuando está muy seguro de sus predicciones.

Ahora se realiza un gráfico de la precisión respecto al *recall* (ver imagen 69).

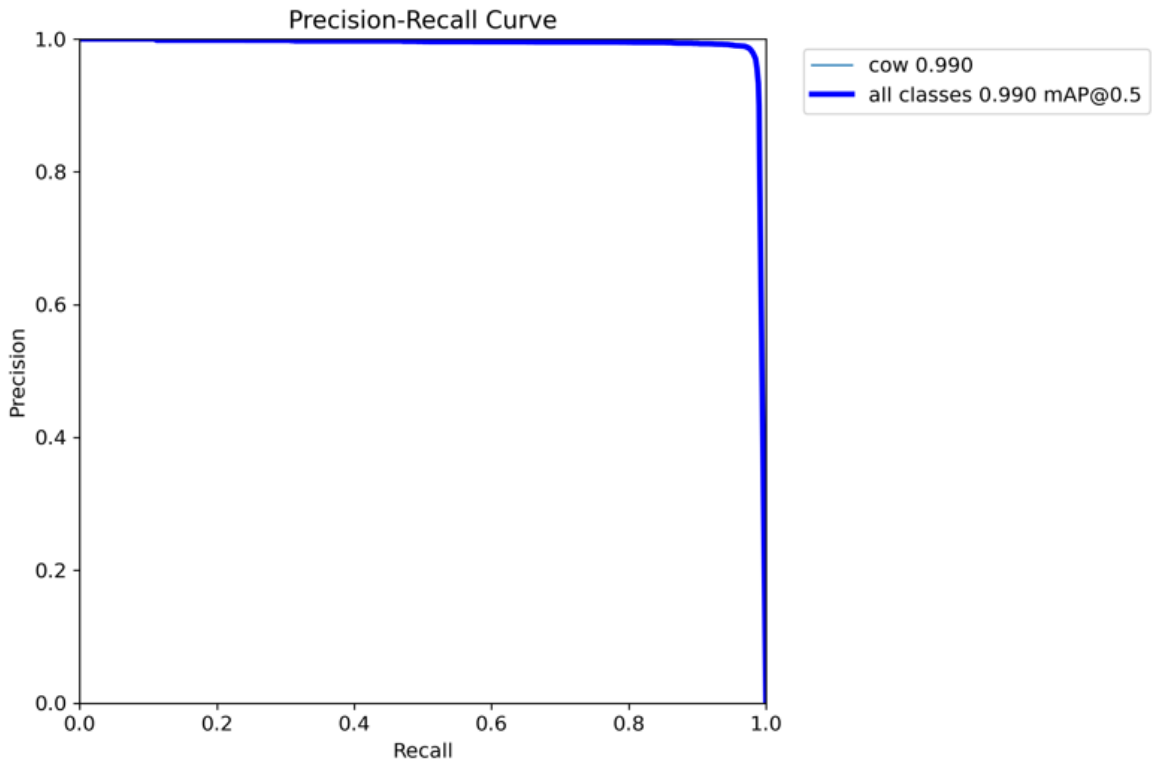


Imagen 69 - Gráfico de precisión respecto al recall.

Esta curva gráfica los parámetros a los que tanta importancia se les ha dado en el documento, ya que son muy representativos de la efectividad del modelo.

La curva se ajusta a la esquina superior derecha, lo que es ideal y sugiere que el modelo está operando con un rendimiento muy alto. Esta forma de curva es indicativa de un modelo que tiene:

- Precisión cercana a 1: Esto significa que, cuando el modelo identifica algo como una vaca, es muy probable que realmente sea una vaca. Los falsos positivos son mínimos.

- *Recall* cercano a 1: El modelo está siendo capaz de encontrar casi todas las vacas reales en las imágenes. Los falsos negativos (es decir, vacas reales que no fueron detectadas) son también mínimos.

Normalmente, la precisión y el *recall* suelen compensarse. En muchos casos, al aumentar la precisión para minimizar los falsos positivos, el *recall* tiende a bajar y se pierden algunos verdaderos positivos. Sin embargo, en este caso, ambas métricas están cerca de su valor máximo simultáneamente.

Resta solamente graficar el *recall* respecto a la confianza (ver imagen 70).

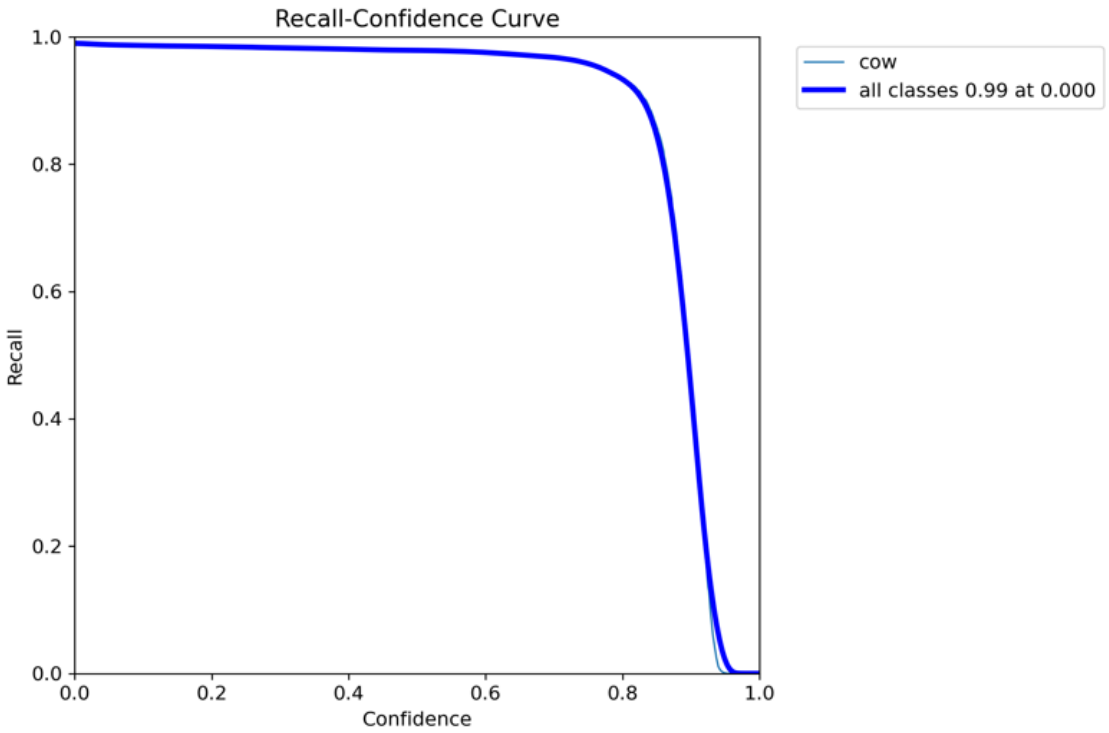


Imagen 70 - Curva del recall respecto a la confianza.

Este gráfico muestra cómo varía el *recall* según el umbral de confianza del modelo:

En el eje x, se representa la confianza con la que el modelo hace predicciones. Valores más altos indican que el modelo está más seguro de su predicción. En el eje y, se muestra cuántos de los verdaderos positivos son detectados por el modelo.

Se puede ver que el *recall* es bastante alto (cerca de 1) en la mayor parte del rango de confianza (hasta alrededor de 0.8). Esto significa que el modelo es capaz de detectar casi todas las vacas incluso con niveles de confianza bajos.

Sin embargo, cuando la confianza se aproxima a valores cercanos a 1, el *recall* disminuye rápidamente. Esto es esperado, ya que un umbral de confianza más alto tiende a reducir el número de predicciones hechas. Este gráfico indica que lo ideal es poner como valor umbral de confianza, un número cercano a 0.8, para así mantener un alto *recall*. En todos estos gráficos y valores, se denota un gran rendimiento del modelo.

7. Conclusiones

El equipo concluye que el proyecto de tesis "Sistema de conteo de ganado" refleja el impacto significativo que el uso de drones y tecnologías de *machine learning* puede tener en la gestión de los recursos ganaderos. Este trabajo ha demostrado que es posible optimizar procesos tradicionalmente manuales, como el conteo de ganado, mediante una solución tecnológica innovadora y adaptable tanto a pequeños productores como a grandes establecimientos rurales.

A lo largo del desarrollo de este proyecto, el equipo adquirió una valiosa experiencia en varias áreas tecnológicas. En primer lugar, el trabajo con *machine learning* permitió profundizar en técnicas como el *fine-tuning* de modelos pre entrenados, la creación de conjuntos de datos específicos y la evaluación de modelos de detección. Se aprendió a entrenar y ajustar modelos de reconocimiento de imágenes.

También se aprendió sobre el desarrollo de una aplicación web siendo una oportunidad para trabajar con tecnologías utilizadas en la industria como Django, PostgreSQL y Celery, permitiendo construir una plataforma robusta y escalable que centraliza la gestión de los datos generados. Además, se realizó un despliegue en la nube utilizando lo que también brindó conocimientos sobre infraestructura y servicios *cloud*, fundamentales para garantizar la escalabilidad y accesibilidad global del sistema.

El proyecto ha cumplido con los objetivos propuestos, implementando un sistema funcional integrado por un dron con cámara y un modelo de reconocimiento de imágenes que permite el conteo automático de vacas. El sistema es fácil de usar para productores sin conocimientos previos en uso de tecnología, lo que garantiza su adopción en diversos entornos rurales. Las pruebas realizadas han demostrado la precisión y eficacia del sistema, alcanzando resultados satisfactorios en cuanto a la detección de animales.

Uno de los aspectos más destacados del proyecto es la reducción significativa del tiempo y el esfuerzo necesarios para realizar los conteos. Los métodos tradicionales, como el rodeo manual o el uso de bastones *RFID*, demandan muchas horas de trabajo y pueden ser propensos a errores. El uso de drones no sólo acelera este proceso, sino que también minimiza el contacto físico con los animales, lo que reduce el estrés del ganado y mejora su bienestar general. Esto tiene implicaciones directas en la productividad y en la calidad de vida de los trabajadores rurales.

Además, la integración de la plataforma web permite a los usuarios acceder a datos históricos, visualizar vídeos procesados y obtener estadísticas que facilitan la toma de decisiones informadas sobre la gestión de su ganado. La escalabilidad del sistema, tanto a nivel nacional como internacional, es otra de las fortalezas del proyecto, permitiendo su adaptación a diferentes contextos geográficos y productivos.

En cuanto a los requerimientos específicos del modelo de *machine learning*, se logró que el sistema sea capaz de procesar videos de cualquier resolución capturados por distintos drones, identificando y contando correctamente el número de vacas en cada cuadro. Esto se hace utilizando un modelo pre entrenado de Yolo V8 ajustado mediante técnicas de *fine-tuning* sobre un gran conjunto de datos, con aproximadamente 20.000 animales etiquetados en imágenes tomadas por nosotros. Las pruebas de validación y el proceso de entrenamiento permitieron mejorar la precisión del modelo, que alcanzó los estándares exigidos, con un

recall y una precisión cercanos al 99%, cumpliendo así los requerimientos técnicos del sistema.

La evaluación del modelo de *machine learning* abarca pruebas en diferentes condiciones ambientales, de iluminación y desde diversos ángulos de captura, incluyendo pruebas con animales similares como perros, caballos y ovejas. Durante estas pruebas se verificó que el modelo funcionara adecuadamente en condiciones reales, asegurando su capacidad para detectar y contar el ganado incluso en situaciones de alta aglomeración.

En conclusión, el "Sistema de Conteo de Ganado" es una solución a una problemática real, y en su desarrollo se aprendió sobre tecnologías no investigadas anteriormente. Se considera que la solución desarrollada es totalmente funcional y puede ser de gran valor para los productores ganaderos. Esto se basa en las evaluaciones finales realizadas, donde los resultados colmaron nuestras expectativas.

8. Referencias bibliográficas

Todas las imágenes tomadas desde dron, incluyendo las 3000 para el armado del *dataset*, fueron tomadas por nosotros y pertenecen a videos.

[1] Vola tus Drones, "DJI Mavic 2 Enterprise Dual Thermal Drone", sep. 2024, [Online]. Available: https://volatusdrones.com/blogs/posts-without-blog/dji-mavic-2-enterprise-dual-thermal-drone

[2] Heliguy, "DJI Matrice 100", sep. 2024, [Online]. Available: https://www.heliguy.com/cdn/shop/articles/dji-matrice-100_4cc5cb48-537c-4dd1-93fe-2c69cbe00042_1512x.jpg

[3] Hstore, "Imagen de producto", sep. 2024, [Online]. Available: https://www.hstore.cl/shop/pub/media/catalog/product/cache/e4d64343b1bc593f1c5348fe05efa4a6/f/o/foto_313_1592.jpg

[4] DJI, "Imagen de producto", sep. 2024, [Online]. Available: https://se-cdn.djiits.com/tpc/uploads/carousel/image/5809969aa387a06a0522450e1749f2ea@ultra.jpg

[5] DJI Store Uruguay, "DJI Mini 3 Pro", sep. 2024, [Online]. Available: https://djistore.com.uy/wp-content/uploads/2022/05/mini-3-pro-04-100x100.jpeg

[6] ArduPilot, "ArduPilot UK", sep. 2024, [Online]. Available: https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.ardupilot.co.uk%2F&psig=AOvVaw3X7boS9yJW-yqe1QJpVE7d&ust=1726101968491000&source=images&cd=vfe&opi=89978449&ved=0CBQQjRxqFwoTCJiErIfVuYgDFQAAAAAdAAAAABAE

[7] Kinsta, "Flask vs Django", sep. 2024, [Online]. Available: https://kinsta.com/es/blog/flask-vs-django/

[8] Strapi, "Relational Databases: PostgreSQL vs MariaDB vs MySQL vs SQLite", sep. 2024, [Online]. Available: https://dev.to/strapi/relational-databases-postgresql-vs-mariadb-vs-mysql-vs-sqlite-5dn7

[9] PostGIS, "PostGIS: Spatial and Geographic Objects for PostgreSQL", sep. 2024, [Online]. Available: https://postgis.net/

[10] Rock Content, "Bootstrap: Guía para principiantes de qué es, por qué y cómo usarlo", sep. 2024, [Online]. Available: https://rockcontent.com/es/blog/bootstrap/

[11] Hostinger, "¿Qué es Nginx?", sep. 2024, [Online]. Available: https://www.hostinger.es/tutoriales/que-es-nginx

[12] Flask Project, "Gunicorn", sep. 2024, [Online]. Available: https://flask.palletsprojects.com/es/main/deploying/gunicorn/

[13] OpenWebinars, "¿Qué es Celery? Introducción y primeros pasos", sep. 2024, [Online]. Available: https://openwebinars.net/blog/que-es-celery-introduccion-y-primeros-pasos/

[14] Amazon Web Services, "¿Qué es Redis?", sep. 2024, [Online]. Available: https://aws.amazon.com/es/elasticache/what-is-redis/

[15] Amazon Web Services, "Docker", sep. 2024, [Online]. Available: https://aws.amazon.com/es/docker/

[16] Amazon Web Services, "Uploading and copying objects using multipart upload", sep. 2024, [Online]. Available: https://docs.aws.amazon.com/AmazonS3/latest/userguide/mpuoverview.html

[17] Amazon Web Services, "Working with presigned URLs", sep. 2024, [Online]. Available: https://docs.aws.amazon.com/AmazonS3/latest/userguide/using-presigned-url.html

[18] Amazon Web Services, "User guide: S3 Storage", sep. 2024, [Online]. Available: https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html

[19] Youtube, "Python Django Tutorial: Full-Featured Web App Part 1 - Getting Started", sep. 2024, [Online[. Available: https://www.youtube.com/watch?v=UmljXZIypDc

[20] I. Goodfellow, Y. Bengio, y A. Courville, "Deep Learning. Cambridge", sep. 2024, [Online]. Available: https://www.deeplearningbook.org/

[21] Ultralytics, "GitHub issue #898: YOLOv3", sep. 2024, [Online]. Available: https://github.com/ultralytics/yolov3/issues/898

[22] Built In, "Data Leakage in Machine Learning: Detect and Minimize Risk", sep. 2024, [Online]. Available: https://builtin.com/machine-learning/data-leakage

[23] Amazon Web Services, "¿Qué es el sobre ajuste?", sep. 2024, [Online]. Available: https://aws.amazon.com/what-is/overfitting/

[24] Universidad ORT Uruguay - Fundamentos de Ingeniería de Software - Aulas, "1.1. Git", sep. 2023, [Online]

[25] Athos Capital, "La importancia de Hugging Face", sep. 2024, [Online]. Available: https://www.athos-cap.com/post-12-la-importancia-de-hugging-face/

[26] HumanSignal, "LabelImg GitHub repository", sep. 2024, [Online]. Available: https://github.com/HumanSignal/labelImg

[27] Hugging Face, "Object Detection Leaderboard", sep. 2024, [Online]. Available: https://huggingface.co/spaces/hf-vision/object_detection_leaderboard.

[28] Ultralytics, "Tips for Best Training Results - YOLOv5", sep. 2024, [Online]. Available: https://docs.ultralytics.com/yolov5/tutorials/tips_for_best_training_results/.

[29] Ultralytics, "Ultralytics GitHub repository", sep. 2024, [Online]. Available: https://github.com/ultralytics

[30] Digital Commons, "Comparison of Tracking-By-Detection Algorithms for Real-Time Satellite Component Tracking", n.d., [Online]. Available: https://digitalcommons.usu.edu/cgi/viewcontent.cgi?article=5751&context=smallsat.

[31] YouTube, "Playlist de videos del proyecto", sep. 2024, [Online]. Available: https://www.youtube.com/playlist?list=PL0TI0oNPkR9qPixm25RedazeUhbpLaPT9

Las siguientes referencias, no se mencionan estrictamente en alguna parte del texto, pero se investigaron durante el desarrollo del proyecto:

Maquinac, "Los drones ya se usan hasta para contar vacas", n.d., [Online]. Available: https://maquinac.com/2015/09/los-drones-ya-se-usan-hasta-para-contar-vacas/

ABC News, "Cow muzzles unlocking potential for facial recognition", dic. 2021, [Online]. Available: https://www.abc.net.au/news/rural/2021-12-22/une-researcher-developes-facial-recognition-for-cattle/100713090

123RF, "Drone cuenta las vacas. Pastor de robots. Internet de las cosas en la agricultura. Ilustración vectorial", n.d., [Online]. Available: https://es.123rf.com/photo_89969240_drone-cuenta-las-vacas-pastor-de-robots-internet-de-las-cosas-en-la-agricultura-ilustraci%C3%B3n.html

Ruralnet, "Conteo de ganado vacuno u ovino con drones", jul. 2021, [Online]. Available: https://ruralnet.com.ar/2021/07/05/conteo-de-ganado-vacuno-u-ovino-con-drones/

Core Electronics, "Object and Animal Recognition With Raspberry Pi and OpenCV - Tutorial Australia", ago. 2021, [Online]. Available: https://core-electronics.com.au/guides/object-identify-raspberry-pi/

9. ANEXOS

ANEXO 1: Visitas al campo

Se planifica implementar un ciclo de mejora continua en las visitas al campo, donde los nuevos datos obtenidos se utilizarán para ajustar y optimizar el modelo, asegurando que el sistema mantenga su rendimiento y se adapte a cambios en el entorno a lo largo del tiempo.

Las visitas al campo se realizan con un objetivo planeado anteriormente, luego se actúa sobre dicho objetivo, evaluando posteriormente para actuar en función a si se cumple o no el objetivo. Para cada campo se planifica, se lleva a cabo, se verifica y se actúa, con el fin de mejorar continuamente.

Anteriormente se menciona la visita a dos campos, que fueron los primeros visitados donde se armó un set de datos básico para realizar una comparativa primaria entre modelos pre-entrenados. En este capítulo, se describen las visitas que se hicieron posteriormente.

Además de esas dos, se visitaron 4 establecimientos rurales, siendo que algunos de ellos se fue múltiples veces.

Se visitan los siguientes campos:

1) Establecimiento “El Talar”
2) Campo Manchega
3) Estancia “Santa Isabel” - 3 veces
4) Estancia “Santa Maria”
5) Laboratorios Microsules
6) Barracas - 4 veces

Visita a Laboratorio Microsules

Se realiza una visita a un establecimiento llamado “Laboratorio Microsules”, se va a este campo con la intención de recopilar nuevas imágenes para entrenar y de generar material donde se pueda evaluar el funcionamiento del modelo a diferentes alturas.

Se obtienen aproximadamente 12 minutos de video útiles, donde había a disposición 200 vacas las cuales se recorrieron dos veces. Estas vacas se encontraban mayormente aglomeradas en corrales (ver imagen 71).

Imagen 71 - Vacas aglomeradas en un corral.

En este campo se encontraban vacas de ganadería extensiva y también vacas en corrales. Siendo esta la primera vez que se obtienen imágenes de vacas en corrales, lo cual hace que este material sea muy útil para entrenar bajo estas circunstancias, donde las vacas están muy juntas.

Se realiza una prueba de altura, realizando un video desde el dron a cierto grupo de vacas, partiendo desde una altura de 8 metros hasta 50 metros.

Verificación

Primero, se entrena el modelo con nuevas imágenes de este campo, dividiendo un set de entrenamiento y uno de validación. Como es la primera vez que enfrentamos vacas en corrales, los resultados no son lo suficientemente buenos, debido al escaso entrenamiento en estas situaciones. El modelo presenta una gran precisión en vacas individuales.

La prueba de altura es satisfactoria, se observa que el rango óptimo de vuelo es de hasta 45 metros aproximadamente, luego la definición de la cámara comienza a decaer en calidad.

Conclusión

Se debe continuar entrenando el modelo, principalmente en vacas muy apretadas. También se concluye que los vuelos deben ser del rango entre 20 y 45 metros. Siendo que, a mayor altura, más rápido se recorre la superficie total de un campo, sacrificando calidad, pero no de manera significativa.

Visita a establecimiento “Barracas”

El equipo visita un campo en Canelones. Este campo se llama “Barracas”. Donde hay a disposición más de 500 animales.

Planificación

Se tienen una serie de objetivos a evaluar en este campo:

- Evaluar *tracking buffer*: Visualizar una vaca, sacarla de foco y ver si al retornar, el *hashing* de identificación del modelo de Yolo es suficiente para identificarla con el mismo ID anterior.

- Realizar tomas panorámicas y ver el funcionamiento del modelo.

- Tomar imágenes de autos, barriles, y diferentes objetos que se puedan encontrar en un campo y ver si el modelo lo toma como falsos positivos.

- Generar un nuevo set de datos de entrenamiento y validación.

Verificación

Tras llevar a cabo los objetivos planteados, se observan los siguientes resultados.

El modelo de *hashing* para la identificación del *tracking* no es suficiente como para identificar una vaca que se va de la toma de imagen. Se probó variando el *buffer* en memoria que se tiene, pero a pesar de persistir el *hash* en memoria, no lo reconoce como si fuera el mismo. Se investigó el funcionamiento de esta identificación, donde la manera de identificar es según la disposición de los píxeles dentro de la caja de detección de un objeto, junto a la posición en la imagen donde se encuentra la caja. Por lo que es sumamente difícil que, al retornar, la vaca se vea desde el mismo punto de vista y el recuadro esté en una posición muy similar a la anterior.

Las tomas panorámicas no son buenas, ya que se realiza una deformación de la imagen en ciertos lugares al avanzar. Realizándose en estático, se observa solamente una pequeña parte del campo.

Se observa un buen funcionamiento del modelo ante falsos positivos, no detecta otro tipo de objetos como vacas, particularmente se realizaron pruebas con tanques de agua, un auto y balanzas.

Se pudo generar gran cantidad de videos para realizar conteos (ver imagen 72).

Imagen 72 - Vacas en ganadería extensiva.

En este caso, se realizaba ganadería extensiva, por lo que el funcionamiento de detección arroja resultados positivos de buena precisión. Se realiza un conteo utilizando línea, donde se cuentan efectivamente todas las instancias de vaca.

Conclusión

Ante estas verificaciones, el equipo concluye que debe elaborar rutas que aseguren no repetir la misma vaca, debido a que el modelo de *hashing* no asegura identificarla de nuevo, por lo que pasarla por encima nuevamente significa contarla dos veces.

También se concluye que los videos deben grabarse de la manera estándar, en resoluciones de aspecto de imagen 16:9 o 4:3, ya que imágenes panorámicas o de otro tipo deforman la imagen.

La manera en la que se cuenta utilizando una línea, puede no ser la más efectiva, se plantea ver de cambiar esto a futuro.

Visita al campo “Santa Isabel”

Esta vez se concurre a Mercedes, a visitar un campo llamado “Santa Isabel”. Se va a visitar posteriormente más días, por lo que la primera visita es de reconocimiento del campo y recopilación de imágenes.

Planificación

Como volveremos a este campo, se va a realizar una toma de datos simplemente para entrenar, así las siguientes veces, utilizamos todas las imágenes para probar el modelo.

Se busca continuar mejorando el modelo y probar el funcionamiento del conteo.

Verificación

En esta visita se encontraron ciertas deficiencias en el modelo de detección. Previo a entrenar, se probó las imágenes con el modelo de detección que se tenía hasta el momento.

En este campo hay un perro negro y blanco, que en tomas aéreas se ve muy similar a una vaca pequeña, lo cual arrojo un falso positivo con un alto factor de confianza (ver imagen 73).

Imagen 73 - Perro negro y blanco.

En este campo se encontró una situación ideal, ya que realizan un pastoreo en superficies poco anchas y muy largas. Planteando superficies de ancho de aproximadamente 30 metros y un largo de kilómetros. Por lo que las vacas se pueden contar todas en una sola recorrida, y evitar contar una más de una vez.

También se realizaron grabaciones a la máxima velocidad que permite el dron, de lo que se notó que el estabilizador de imagen, no funciona muy bien a esta velocidad. Si bien aún se logra detectar el ganado, no se obtienen imágenes de calidad.

Conclusiones

Ante la problemática de falsas detecciones de perros, el equipo plantea dos posibles soluciones. La primera sería entrenar un modelo de detección de perros. Por lo que se tendrían dos clases a detectar y se filtra para contar solo las que sean vacas. Sin embargo, esto no es sencillo, ya que existen muchas razas de perros y no es fácil recopilar una gran cantidad de imágenes que aseguren detectar perros. Entonces, la solución que se plantea es incluir en el *dataset* fotos de vacas con perros al lado, donde solo se etiquetan las vacas como instancias de detección. De esta manera el perro interpreta las imágenes de perros sueltos como "*Background Images*" y entiende que, los perros no son vacas.

El equipo también concluye en este campo que se debe cambiar la manera de contar. Utilizar *tracking* junto a una línea por la cual pasan los objetos que se cuentan, no es eficiente. Por lo que se busca realizar un *script* el cual cuente todas las instancias de identificación únicas que se encuentren dentro del recuadro de la imagen.

Visita al campo “Santa María”

La siguiente visita se realizó en un campo de Flores, llamado “Santa María”. En este campo se dedican a la ganadería de engorde, y las vacas se encuentran dispuestas mayoritariamente en corrales, por lo que están apretadas.

Planificación

En este campo, se busca comprobar el funcionamiento del modelo de conteo, teniendo vacas apretadas, se puede verificar la detección de vacas cuando la intersección entre las cajas de detección es muy grande.

Verificación

Se realizan grabaciones de tiempo considerable, sobrevolando el ganado en diferentes situaciones. Ya sea dentro de los corrales, como en el traslado hacia dentro y fuera de los mismos. También se realiza una grabación donde los animales corren por el campo de manera extensiva, con una separación considerable (ver imagen 74).

Imagen 74 - Pasillo de vacas junto a un caballo.

Este establecimiento también contaba con ovejas y caballos, esto nos permitió tomar capturas en la cuales podemos ver el ganado desfilando junto a los caballos al lado de los corrales de las ovejas y así poder verificar que no haya falsa detección de vacas con una oveja ni con un caballo (ver imagen 75).

Imagen 75 - Foto aérea donde se ven ovejas.

Conclusiones

El balance tras la evaluación de los videos es que el modelo funciona de manera excelente para la disposición del ganado de manera extensiva, donde no se notan fallas en las detecciones. Sin embargo, cuando las vacas se encuentran muy juntas y el parámetro IoU (*Intersection Over Union*) es muy grande, el modelo falla y puede detectar dos vacas como si fuera una sola. Esto se considera una falla solucionable con más entrenamiento sobre ganado en este contexto.

Segunda visita al campo "Santa Isabel"

Nuevamente se visita el campo de Mercedes, "Santa Isabel". Esta vez el equipo busca verificar que se hayan solucionado los problemas de la primera visita, para continuar con la mejora continua del modelo.

Planificación

Anteriormente se tuvo fallas en la detección de vacas, donde el modelo consideraba que un perro era una vaca con un alto factor de confianza. Desde la primera visita a esta, se entrena el modelo con fotos de estos perros vistos anteriormente, sin clasificarlos como vacas.

Verificación

Para verificar, el equipo genera videos de manera similar a como lo hizo la primera vez, videos donde se visualiza ganado, pero además en ciertos instantes aparecen perros. También, se generan videos similares a los de la primera visita, donde se sobrevuelan vacas con una separación considerable (ver imagen 76).

Imagen 76 - Conteo sobre vacas separadas.

Conclusiones

Tras procesar los videos, se ve que el modelo ya no presenta la falsa detección de perros como vacas. A su vez, se realiza el conteo de ganado sobre los videos donde las vacas están de manera extensiva por el campo, en la primera visita se había realizado el conteo con una modalidad vieja. Por lo que en este campo no solo se verifica que se solucionaron las falsas detecciones, también se verifica el funcionamiento del nuevo modelo de conteo.

Visitas al campo para mejora continua

Tras las visitas descritas anteriormente, el equipo considera satisfactorio el trabajo realizado, por lo que simplemente va a realizar series de visitas sin un fin en particular. Donde se realizan grabaciones del ganado para realizar los conteos, buscando algún escenario donde se pueda fallar, para seguir entrenando el modelo, o simplemente para generar contenido del trabajo realizado.

En total, se realizaron 4 visitas de este tipo en total, donde se visita nuevamente el campo "Barracas" en tres oportunidades, y se visita por segunda vez el campo "Santa María".

Fallas detectadas

En estos campos se realizan los videos de rutina sobrevolando el ganado en distintos contextos. En una instancia, se realiza un conteo el cual consiste en recorrer 500 metros ida y vuelta, para poder cubrir el ancho total. En este video, el modelo de conteo mostró un funcionamiento ideal, donde detecta todas las vacas en el video y no presenta falsos positivos. Pero existe un problema y es que ante la imposibilidad de identificar ganado, se contó dos veces la misma vaca. Ya que una vaca aparece tanto en la ida como en la vuelta, debido a que se mueve horizontalmente. Lamentablemente, solucionar esta falla es muy compleja, ya que realizar un modelo de identificación a 30 metros de altura es sumamente difícil.

Soluciones

Debido a que no se puede desarrollar un modelo de identificación de ganado, la única manera de minimizar riesgos de conteo doble de una misma instancia cuando se realizan videos de ida y vuelta, es optimizando la grabación y la ruta del dron. Sabemos que, a mayor altura, el ángulo de vista de la cámara es mayor, por lo que se cubre una superficie más grande. Por lo que, para mitigar riesgos, los videos se deben realizar a una altura considerable, pudiendo evitar movimientos horizontales de las vacas. Normalmente las vacas se encuentran muy estáticas o se mueven muy lento, por lo que el desplazamiento horizontal suele ser muy pequeño. Sobrevolando a una altura de 40 o 50 metros debería ser suficiente para evitar este tipo de problemas.

En contraparte, realizar las grabaciones desde tan alto, genera que las imágenes de las vacas se vean en peor definición, ya que cada vaca ocupa una menor cantidad de píxeles. Por lo que se debe buscar un punto de equilibrio en la altura del dron, donde la definición sea suficiente para generar factores de confianza altos, y se minimice la posibilidad de conteo doble.

ANEXO 2: Fragmentos de conteos en los diferentes establecimientos

Se agregan capturas de pantalla de videos de conteos realizados en el proyecto.

(ver imágenes 77, 78, 79, 80 y 81).

Imagen 77 - Fragmento de conteo 1.

Imagen 78 - Fragmento de conteo 2.

Imagen 79 - Fragmento de conteo 3.

Imagen 80 - Fragmento de conteo 4.

Imagen 81 - Fragmento de conteo 5.

Estas imágenes se incluyen en el documento a modo de referencia, todas pertenecen a un video de conteo. Estos videos son accesibles desde la aplicación web, y hay algunos disponibles en Youtube [31], donde hay videos desde los primeros prototipos de conteo hasta el modelo final.

ANEXO 3: Caso de uso

En este anexo se describe paso a paso el flujo que se debe seguir para garantizar los mejores resultados de este producto.

Para comenzar, se debe tener un dron. Este dron no debe ser necesariamente el mismo que se menciona en este proyecto como elegido, sino que puede ser otro que cumpla con los requisitos planteados.

Teniendo este dron, se deben generar las grabaciones. Idealmente, se debe apuntar la cámara totalmente hacia abajo, buscando que quede paralela al suelo. Esto se debe a que, para entrenar, se usó principalmente este punto de vista, donde el modelo reconoce mejor si se ve el lomo de la vaca como un plano.

Otra precaución al momento de grabar es que como no se identifica ganado, no se puede recorrer dos veces sobre la misma vaca. Entonces, se debe tener el dron siempre en movimiento e idealmente, en el mismo sentido que el ganado.

Teniendo la grabación pronta, se debe extraer el video de la tarjeta de memoria del dron, o transferir el video directamente desde el celular en drones más avanzados. Luego, se ingresa en la aplicación web y se *loguea* al usuario correspondiente (ver imagen 82).

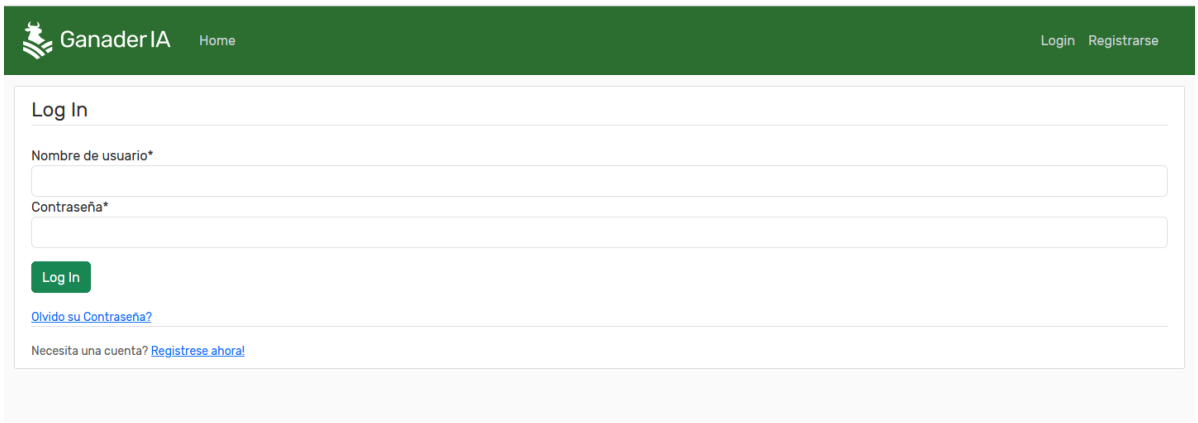


Imagen 82 - Home de la aplicación web.

Tras iniciar sesión, arriba aparece una opción denominada ‘Mis Establecimientos’. En caso de que no se tenga ningún establecimiento creado, se puede hacer dando click en la esquina superior derecha de la pantalla, donde aparece la foto de perfil (ver imagen 83).

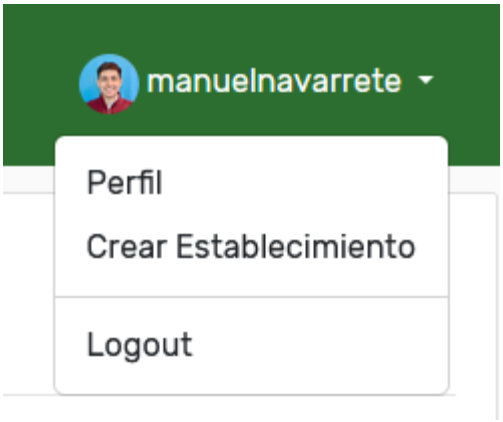


Imagen 83 - Esquina superior derecha de la aplicación web.

Tras tener un establecimiento creado, se accede al mismo desde la pestaña 'Mis Establecimientos'. Al seleccionar el mismo, se despliega información y los miembros que pertenecen a dicho establecimiento (ver imagen 84).

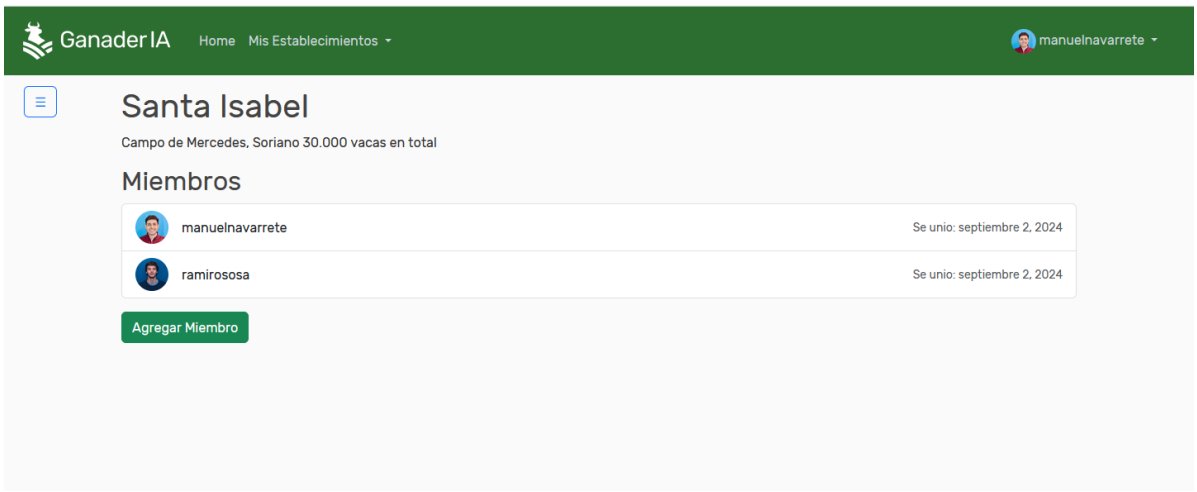


Imagen 84 - Pantalla de un establecimiento.

En la izquierda, bajo el logo de Ganader IA, hay un menú desplegable, donde se accede a las pasturas y lotes del campo. Cuando no se tiene nada creado, se deben crear tanto un lote como una pastura, en orden indistinto. Para hacer esto, se dirige en el menú desplegable a la opción 'Mis Lotes', y al seleccionar la opción, aparece un botón que indica 'Agregar Lote' (ver imagen 85).

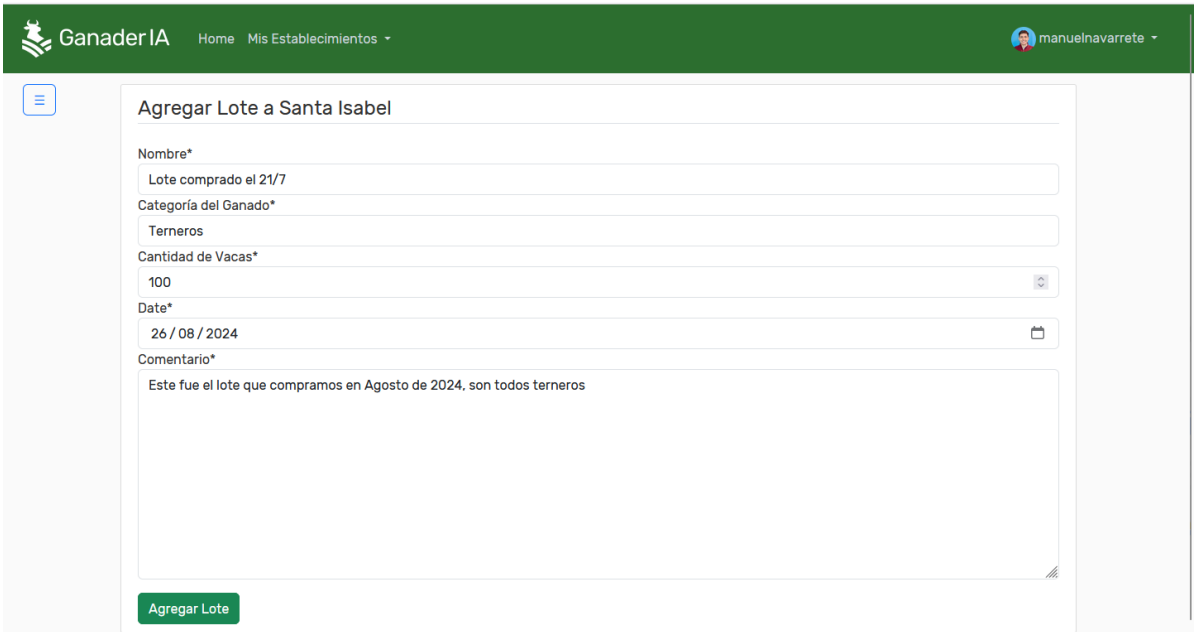


Imagen 85 - Menú de creación de lote.

Se completan todas las opciones y se presiona el botón de "Agregar Lote". Luego se dirige a la pestaña de pasturas y realiza un procedimiento similar, creando una pastura. Al crear la pastura, le aparece la opción de asignarle un lote, en este caso, se le asigna el único que existe creado (ver imagen 86).

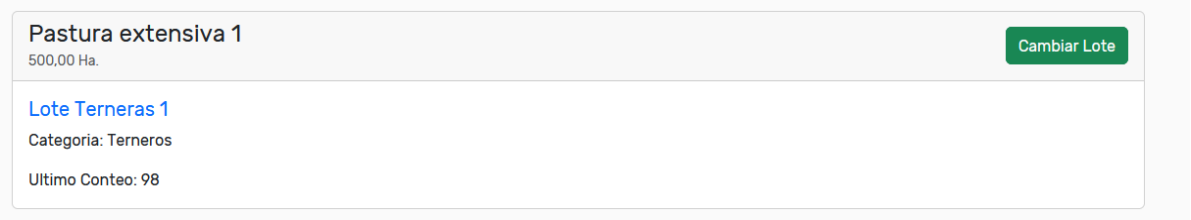


Imagen 86 - Vista de pasturas.

Ahora podemos realizar un conteo. Nos dirigimos al lote, tanto dándole click desde la pastura, como desde el menú desplegable (ver imagen 87).

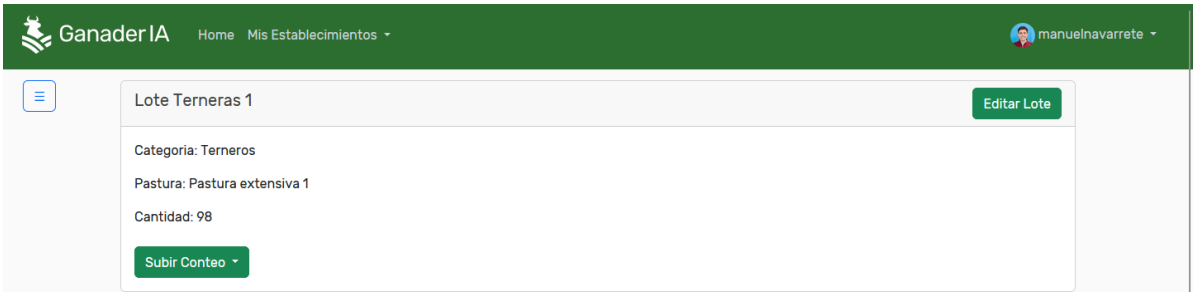


Imagen 87 - Vista de un lote.

En esta pestaña, se debe seleccionar el botón de "Subir Conteo", donde brinda la opción de subir un video o de subir un conteo que fue realizado manualmente.

Al seleccionar conteo por video, se abre un menú en el cual se debe subir el archivo, indicar fecha y una descripción (ver imagen 88).

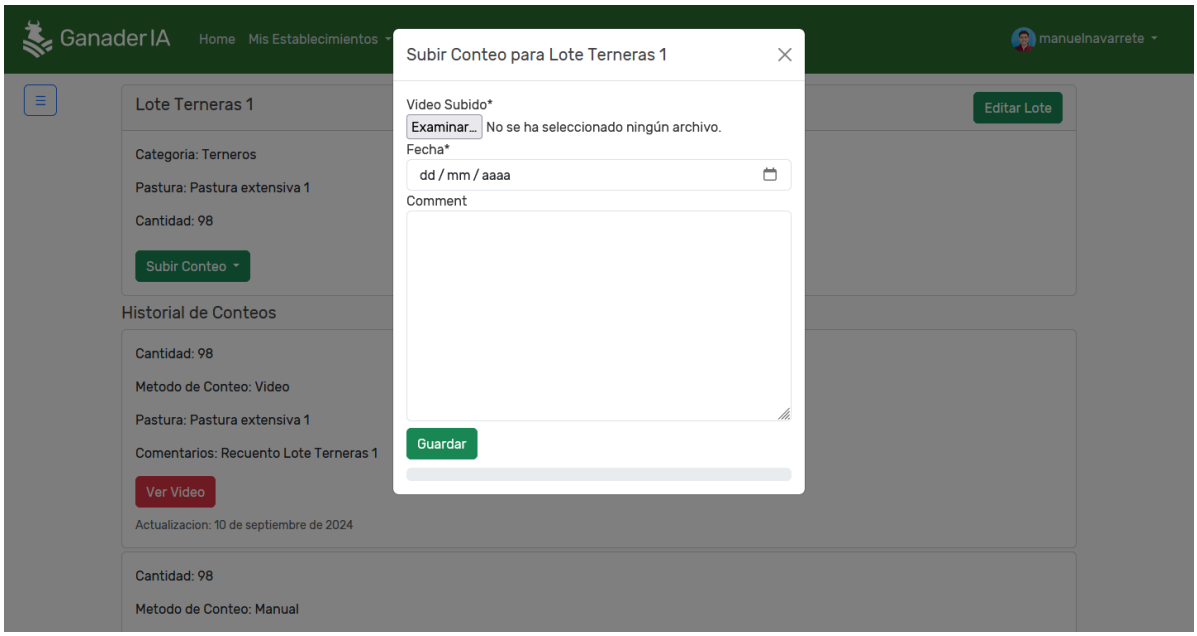


Imagen 88 - Subida de video de conteo.

Al guardarlo, el video comienza a subirse y se muestra una barra que carga a medida que se da la subida del archivo. Posteriormente, aparece una alerta que indica que el video ya ha sido subido, y dentro del lote se verá una barra de carga que muestra cómo va el proceso del video (ver imagen 89).

Imagen 89 - Barra de proceso de subida.

Al finalizar esto, se actualiza la información de la cantidad de vacas arriba, y aparece abajo, en el historial de conteos, este último procesamiento, con la opción de ver el video procesado si interesa. En caso de seleccionar la opción de ver vídeo, se abre una pestaña como la siguiente (ver imagen 90).

Imagen 90 - Vista de video procesado.

De esta manera, se culmina un caso de uso típico para el conteo de ganado mediante el sistema desarrollado. Siguiendo estos pasos, los usuarios pueden aprovechar de forma eficiente las capacidades de reconocimiento y conteo de ganado ofrecidas por la plataforma, con resultados precisos y centralizados en una aplicación.

ANEXO 4: Entrevista al Ing. Ignacio Lopez

El 24 de Octubre 2023 nos reunimos con Ignacio Lopez, el trabaja en Dronefies Labs programando drones DJI.

La reunión fue muy interesante ya que nos dio otra perspectiva y entendimiento sobre el mundo de los drones. Las necesidades de un *SDK on-board* o *mobile* y los requerimientos a nivel software de cada dron, las compatibilidades y demás.

Nos comentó que utilizan DJI ya que es de lo mejor en el mercado, es lo más adaptable y versátil. La diferencia principal entre un *on-board SDK* y otro tipo de *SDK* es que el *on-board* es básicamente una computadora, donde el dron puede hacer procesamiento en vuelo, esto puede llegar a ser muy bueno dependiendo de las necesidades que se tengan y de la arquitectura del producto. Una de nuestras propuestas era hacer el procesamiento de las imágenes en vuelo y simplemente pasar a una base de datos los datos necesarios para que se desplieguen en una web. Sin embargo, concluimos que no tenemos urgencia en procesar los datos sobre las vacas en vuelo, sino que simplemente se puede utilizar el video tomado por el dron luego. En ese caso una *mobile SDK* nos permitiría tener una aplicación que haga de interfaz entre el operador del dron y el mismo ya que podríamos programar un vuelo realizar ciertas maniobras de movimiento y luego subir los videos a través de la aplicación cuando se tenga conectividad para luego sí realizar el procesamiento de las imágenes.

Ignacio también nos hizo varias aclaraciones acerca de la utilización del *SDK* de Mavic, puede que alguna versión no soporte lo que queremos hacer o que se discontinúe y la próxima versión de dron no sea soportada por la misma *API*. Nos comentó que no son tan amigables en ese aspecto.

Luego de la reunión descartamos la utilización de un Matrice ya que el *SDK on-board* tiene mucho peso en el *budget* de *power* del dron y se acaba rápidamente la batería, por lo cual un *SDK on-board* no es factible para nuestro caso de uso.

De todas formas, nos menciona que DJI es lo mejor que había en el mercado, y nos recomendó para nuestro caso de uso Mavic 3 o Mini 3 pro, por sus capacidades y cámara.