ANÁLISIS DEL RENDIMIENTO DE UN MODELO DE REDES NEURONALES PARA LA DETECCIÓN DE PERSONAS EN ZONAS SEGURAS DE MEDIOS ACUÁTICOS

NATHALI VILLALBA SÁNCHEZ

INSTITUCIÓN UNIVERSITARIA PASCUAL BRAVO

FACULTAD DE INGENIERÍA

TECNOLOGÍA DESARROLLO DE SOFTWARE

MATEO RICO GARCÍA

JAIME SOTO URDANETA

13 DE MAYO DE 2025

Resumen

Este proyecto se basa en el análisis del rendimiento de un modelo de detección mediante visión artificial e inteligencia artificial, destinado a la función de prevenir accidentes en medios acuáticos. Para lo anterior se implementó el modelo Faster R-CNN, el cual aplica una red neuronal convolucional (R-CNN) utilizada en imágenes con el fin de simular condiciones del mundo real. También se implementó el método de aprendizaje por transferencia y se determinó realizar una evaluación de métricas de rendimiento de precisión y recall. Los resultados que se obtuvieron fueron una precisión máxima del 89%, un recall del 73% y la métrica F1-score, que combina precisión y recall, fue de 0.80 lo que es una clara evidencia del diseño robusto del modelo que también incluye escenarios previos. La implementación es efectiva en la identificación automática de personas en entornos acuáticos y es adecuada para su uso como sistema preventivo en contextos de contar con supervisión humana. Este trabajo tiene como finalidad servir de base para creaciones de softwares robustos y eficientes en el mundo real, con el objetivo de proporcionar sistemas que disminuyan el riesgo de pérdidas humanas en medios acuáticos.

Abstract

This project is based on the performance analysis of an object detection model using computer vision and artificial intelligence, aimed at preventing accidents in aquatic environments. For this purpose, the Faster R-CNN model was implemented, which applies a convolutional neural network (R-CNN) used on images to simulate real-world conditions. The transfer learning method was also implemented, and an evaluation of precision and recall performance metrics was carried out.

The results obtained showed a maximum precision of 89%, a recall of 73%, and an F1-score—which combines precision and recall—of 0.80, which clearly demonstrates the robust design of the model, including previous scenario training. The implementation is effective in the automatic identification of people in aquatic environments and is suitable for use as a preventive system in contexts that involve human supervision.

The aim of this work is to serve as a foundation for the development of robust and efficient software for real-world applications, with the goal of providing systems that help reduce the risk of human loss in aquatic environments.

Tabla de contenido

Lista de tablas

Lista de figuras

Planteamiento del problema

La detección de objetos es uno de los pilares fundamentales en el campo de la visión artificial. Se trata de la capacidad de un sistema para identificar y localizar objetos específicos dentro de una imagen o video. Esta tecnología ha evolucionado enormemente gracias a los avances en el aprendizaje profundo, que permiten identificar múltiples objetos en tiempo real con gran precisión.

A pesar de los avances, la detección de objetos sigue enfrentando desafíos importantes como lo son condiciones ambientales adversas, ocultamiento parcial, variabilidad en los datos, consumos de recursos, entre otros.

A pesar de estos retos, la evolución de esta tecnología ha sido clave para múltiples industrias, desde la seguridad y la medicina hasta la automoción y la agricultura.

Uno de los desafíos más críticos hoy en día es la detección precisa de personas en entornos acuáticos, ya sea en piscinas, lagos o el océano. Este problema es complejo debido a varios factores como por ejemplo reflejos y distorsión, movimiento del agua, variabilidad de posiciones, condiciones de iluminación.

Por esta razón, resulta fundamental investigar y desarrollar soluciones tecnológicas que permitan una detección eficaz de personas en ambientes acuáticos, con el fin de mejorar los sistemas de seguridad, prevenir accidentes y salvar vidas en contextos recreativos, deportivos o laborales. Esta necesidad urgente plantea la importancia de avanzar en modelos más robustos y precisos que puedan operar en condiciones adversas y en tiempo real.

Formulación

¿Puede un modelo de algoritmo basado en redes neuronales detectar de manera eficiente a personas en zonas seguras en medios acuáticos, y cómo podría este análisis contribuir a determinar su efectividad en términos de precisión y exactitud, para apoyar el desarrollo de soluciones tecnológicas orientadas a la prevención de accidentes?

Justificación

Abordar el problema de la detección de personas en el agua no es solo una cuestión de innovación tecnológica, sino una necesidad humanitaria y de seguridad pública. Según la Organización Mundial de la Salud (OMS), más de 236,000 personas mueren por ahogamiento cada año, situación que la posiciona en una de las principales causas de muerte accidental a nivel global. Estas cifras reflejan la urgencia de contar con sistemas de prevención y rescate más eficaces.

Actualmente, la seguridad acuática depende de salvavidas humanos, quienes pueden fatigarse, distraerse o no notar a alguien en peligro a tiempo, también de cámaras de seguridad convencionales, que solo registran imágenes sin capacidad de detección automática. y por otro lado sistemas de boyas o alarmas, que generalmente reaccionan después de que una persona ya está en una situación crítica. En resumen, el análisis del rendimiento de un modelo basado en redes neuronales para detectar automáticamente personas en medios acuáticos, es una iniciativa que responde a una necesidad urgente en el ámbito de la seguridad acuática. Teniendo en cuenta que no solo mejorará la protección de las personas, sino que también promoverá la innovación tecnológica y fortalecerá las capacidades de gestión y respuesta de las autoridades competentes. Por lo tanto, este proyecto se justifica como una medida crucial para salvaguardar vidas humanas y mejorar la experiencia de recreación en entornos acuáticos.

Objetivos

Objetivo general

Analizar el rendimiento de un modelo de algoritmo basado en redes neuronales para la detección de personas en zonas seguras en medios acuáticos, con el fin de determinar su efectividad en términos de precisión y exactitud.

Objetivos específicos

- Seleccionar un modelo de redes neuronales para detección de objetos que permita identificar personas en medios acuáticos en tiempo real, considerando criterios de eficiencia, robustez y precisión.
- Implementar un conjunto de datos compuesto por imágenes de personas en medios acuáticos que sirvan para el entrenamiento del modelo seleccionado.
- Optimizar el modelo seleccionado para la detección de personas con las imágenes recolectadas
- Evaluar los resultados del rendimiento obtenido por el modelo de detección de personas en medios acuáticos.

Marco Referencial

Antecedentes

A continuación, se presentan tres antecedentes recientes relacionados con la aplicación de redes neuronales para la detección y reconocimiento de objetos en entornos acuáticos, los cuales aportan una base de referencia para el presente estudio.

Calatayud Ferre (2022) realizó un estudio titulado *Detección de personas para rescate marítimo*, cuyo objetivo fue desarrollar un sistema de detección automática de personas en situaciones de peligro en el mar mediante la implementación de la red neuronal YOLOv3. Este sistema fue diseñado para operar en tiempo real, mejorando significativamente la eficiencia en operaciones de búsqueda y rescate marítimo. Para lograr este objetivo, el autor generó un conjunto de datos propio, compuesto por imágenes y videos específicos de personas en ambientes marinos, alcanzando resultados prometedores en términos de precisión y tiempo de respuesta.

Lisani Roca y Catalán Alemany (2020), en su publicación *Así enseñamos a los ordenadores* a identificar especies de peces, utilizaron técnicas de inteligencia artificial aplicadas al reconocimiento visual automático de especies marinas. El estudio destacó la importancia de entrenar redes neuronales convolucionales (CNN) para automatizar la identificación precisa de diferentes especies en imágenes submarinas. Los resultados indicaron que la aplicación de estas tecnologías reduce significativamente la intervención humana en la monitorización de los ecosistemas marinos, facilitando la obtención de datos útiles para la conservación marina.

López Rever (2021) presentó el trabajo *Detección de residuos mediante redes neuronales en entornos de playa*, centrado en la implementación de modelos de

inteligencia artificial para detectar y clasificar residuos sólidos en zonas costeras. Su estudio propuso el uso de redes neuronales para facilitar la limpieza y monitoreo continuo de playas, mostrando que estos modelos pueden adaptarse satisfactoriamente a ambientes acuáticos complejos y ayudar significativamente en las tareas de gestión ambiental.

Marco Teórico

El presente capítulo desarrolla el marco teórico que sustenta esta investigación, abordando los principales conceptos, enfoques y antecedentes relacionados con la visión artificial y su aplicación en la detección de personas en entornos acuáticos. Se parte del análisis de trabajos previos que han explorado el uso de algoritmos basados en redes neuronales para tareas de reconocimiento y localización de objetos, destacando sus logros, limitaciones y evolución reciente. Posteriormente, se profundiza en los fundamentos técnicos de la visión artificial, como disciplina que combina procesamiento de imágenes, inteligencia artificial y aprendizaje profundo, con el propósito de dotar a los sistemas computacionales de la capacidad para interpretar y analizar su entorno visual. Este capítulo proporciona el sustento conceptual necesario para comprender el desarrollo e implementación del modelo propuesto en este estudio.

Visión artificial

La visión artificial o también conocida como visión por computadora es una tecnología que fue desarrollada alrededor de los años 60, con el origen de un prototipo automatizado basado en cámaras y sistemas. estas procesaban las imágenes a través de ordenador y software, esta tecnología incluye métodos para recopilar, procesar, analizar y comprender las imágenes que vienen del mundo real. transformando todos estos datos en información digital la cual puede ser tratada por un ordenador.

Una forma simple de entender este proceso es pensar en las cámaras como el equivalente a nuestros ojos como humanos y en el ordenador como nuestro cerebro el cual puede analizar y comprender la imagen de lo que está observando, así los ordenadores pueden percibir geometría, espacios, colores, defectos e incluso con el sensor adecuado pueden percibir temperatura y radiación sin olvidar las ventajas que nos brinda el avance en el desarrollo de estos equipos como la velocidad de procesamiento lentes ópticos de mejor calidad, sistemas de control, etc.

La visión artificial requiere una gran cantidad de datos para analizar y reconocer imágenes. Dos tecnologías esenciales son el aprendizaje automático (APR) y las redes neuronales convolucionales (CNN). El APR utiliza una computadora para aprender a sí misma sobre el contexto de los datos visuales, mientras que las CNN ayudan a un modelo de APR a "mirar" imágenes compilándolas en píxeles y asignándoles etiquetas. Las CNN primero distinguen bordes definidos y formas simples, y luego completan la información a medida que ejecuta iteraciones de sus predicciones. Se utiliza para el reconocimiento de imágenes individuales y en aplicaciones de vídeo, ayudando a las computadoras a comprender las relaciones entre imágenes en una serie de cuadros. En este contexto la Figura 1 ejemplifica la aplicación práctica de la visión artificial en dispositivos móviles, al permitir la detección y traducción automática de texto en tiempo real. (IBM, s.f.).

Figura 1.

Uso de Google Lens para traducir texto mediante visión artificial.

Fuente. Xataka Android. (2020). Google Lens invade la traducción con cámara del Traductor de Google.

Redes neuronales artificiales

Desde sus inicios, la Inteligencia Artificial (IA) ha intentado replicar las habilidades propias del ser humano, esta comparación como se refleja en la Tabla 1 permite comprender un poco algunas de estas habilidades que busca replicar. Dentro de este amplio campo, las redes neuronales artificiales (RNA) han surgido con el mismo propósito, aunque con una diferencia fundamental. Mientras que la IA tradicional se desarrolló sobre la base de la arquitectura convencional de los ordenadores, utilizando algoritmos programados en lenguajes comprensibles para la máquina y optimizados para gestionar memoria y procesadores de manera eficiente, las redes neuronales artificiales adoptan un enfoque distinto.

En lugar de imitar el funcionamiento de una computadora clásica, las RNA buscan replicar el modelo de procesamiento del cerebro humano. A diferencia de los ordenadores, que operan con un número limitado de microprocesadores diseñados para ejecutar instrucciones de forma precisa y secuencial, el cerebro humano está compuesto por miles de millones de neuronas interconectadas, formando redes neuronales complejas. Cada neurona, individualmente, es un procesador simple y poco fiable si se compara con los microprocesadores de los dispositivos electrónicos; sin embargo, el poder del cerebro radica en su inmensa cantidad de neuronas (alrededor de 100 mil millones) y en su capacidad de operar en paralelo mediante conexiones sinápticas. Se estima que cada neurona puede estar conectada con unas 10.000 más, lo que da lugar a un sistema de procesamiento robusto y altamente eficiente.

Otro aspecto fundamental que distingue a las redes neuronales es su capacidad de aprendizaje. A diferencia de los programas convencionales, que requieren ser programados explícitamente para ejecutar tareas específicas, las neuronas no operan bajo un esquema predefinido, En su lugar, aprenden a partir de los datos que reciben, ajustando sus parámetros internos mediante un proceso de entrenamiento. Esta habilidad de adaptar su comportamiento en función de la información permite a las redes neuronales generalizar y resolver problemas complejos sin necesidad de instrucciones detalladas para cada situación.

En una red neuronal, no existe un único componente que controle el sistema, como ocurre con la CPU en una computadora, en su lugar, las neuronas se influyen entre sí mediante conexiones sinápticas, ya sean excitadoras o inhibidoras, lo que da lugar a una dinámica compleja de activaciones y desactivaciones. A través de este mecanismo, las redes neuronales se ajustan y evolucionan por sí solas, aprendiendo y adaptándose al entorno, permitiendo que surjan

capacidades avanzadas de procesamiento sin necesidad de una programación explícita. (Tecnobits, 2022).

Tabla 1.

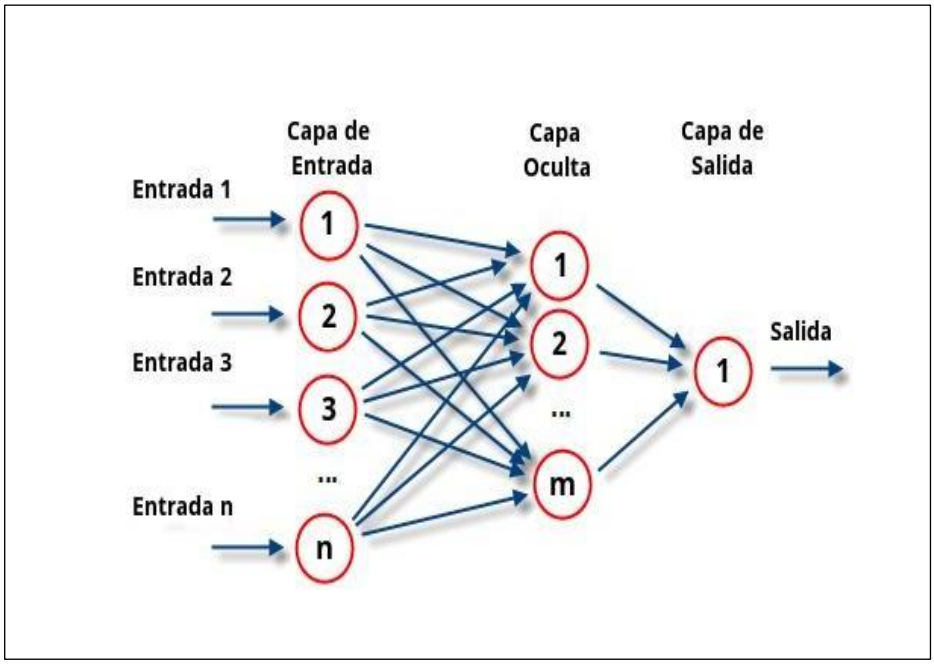
Comparación estructural y funcional entre el procesamiento biológico del cerebro humano y el procesamiento en redes neuronales artificiales

Característica	Cerebro	Redes Neuronales
Velocidad de procesamiento	Rápido en tareas cognitivas complejas, pero más lento en cálculos numéricos	Muy rápido en cálculos matemáticos y procesamiento de datos masivos
Estilo de procesamiento	Paralelo y distribuido en múltiples áreas del cerebro	Secuencial en algunas capas
Número de procesadores	Alrededor de 100 mil millones de neuronas	Miles o millones de nodos interconectados en una red artificial
Conexiones	Cada neurona puede conectarse con aproximadamente 10.000 más	Conexiones establecidas mediante pesos sinápticos entrenados
Almacenamiento del conocimiento	Basado en asociaciones y conexiones sinápticas que se fortalecen con la experiencia	Se almacenan en los pesos y parámetros ajustados durante el entrenamiento
Tolerancia a fallos	Alta, debido a la redundancia neuronal y la capacidad de reorganización (neuroplasticidad)	Baja, pequeñas modificaciones en los datos pueden afectar significativamente el rendimiento
Tipo de control del proceso	Autoorganizado y descentralizado	Basado en reglas predefinidas y supervisión durante el entrenamiento

Fuente. Tomado de Tecnobits (2022).

Figura 2.

Arquitectura de una red neuronal de tres capas.



Fuente. Tomado de ResearchGate. (2023)

Redes neuronales convolucionales (CNN)

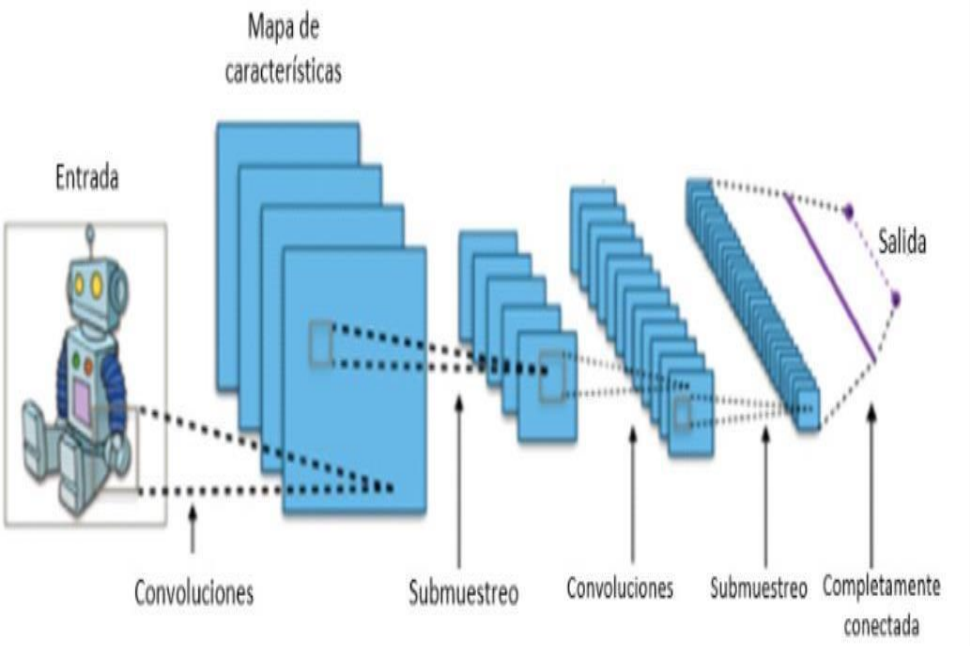
Las redes neuronales convolucionales poseen una estructura en múltiples capas, en las que cada una realiza varias convoluciones aplicando funciones no lineales como ReLU o tanh para producir resultados específicos. Estas redes están inspiradas en la estructura de la corteza visual animal, donde la respuesta de cada neurona puede expresarse mediante operaciones matemáticas conocidas como convoluciones, la Figura 2 muestra de manera esquemática la arquitectura de una red neuronal evidenciando como se organizan las neuronas artificiales para procesar datos en distintas fases.

El funcionamiento interno de este tipo de red puede detallarse en la Figura 3, la cual representa el flujo de procesamiento desde la entrada hasta la clasificación final, evidenciando la importancia de las capas de convolución, submuestreo y completamente conectadas.

Las neuronas visuales reaccionan individualmente a estímulos ubicados en áreas específicas denominadas campos receptivos, que se sobreponen entre sí formando una especie de malla visual. Una ventaja clave de estas redes es su capacidad de reconocer patrones sin importar su posición espacial exacta. En la corteza visual existen dos tipos principales de células: las simples, que responden especialmente a ciertos patrones dentro de su campo receptivo, y las complejas, que poseen campos receptivos mayores y pueden reconocer patrones de manera más flexible, sin depender estrictamente de su ubicación precisa. (Pacheco, 2017, como se citó en Moreira Ramos, 2021).

Figura 3.

Proceso de una red neuronal convolucional (CNN).



Fuente. Tomado de Atria Innovation. (s.f.).

Modelos de detección de objetos

Faster R-CNN:

Faster R-CNN, abreviatura de *"Faster Region-Convolutional Neural Network"*, es una arquitectura avanzada de detección de objetos que pertenece a la familia de las redes R-CNN. Fue presentada en 2015 por Shaoqing Ren, Kaiming He, Ross B. Girshick y Jian Sun. Su objetivo principal es ofrecer una solución unificada capaz no solo de identificar objetos en una imagen, sino también de ubicarlos con precisión. Para lograrlo, integra el poder de las redes neuronales convolucionales (CNN) y las redes de propuestas de regiones (RPN) en una arquitectura conjunta, lo que mejora tanto la velocidad como la precisión del modelo.

Esta arquitectura puede observarse esquemáticamente en la Figura 4, donde se representa la secuencia de componentes principales desde las capas convolucionales hasta las capas totalmente conectadas.

Esta arquitectura está compuesta por dos módulos principales:

1. Red de Propuestas de Regiones: Utiliza una Red de Propuestas de Regiones (RPN) para identificar áreas en la imagen que probablemente contengan objetos.
2. Detector Fast R-CNN (Ren, He, Girshick & Sun, 2015)

Arquitectura de Faster R-CNN

Modulo 1: Red de Propuestas de Regiones (RPN):

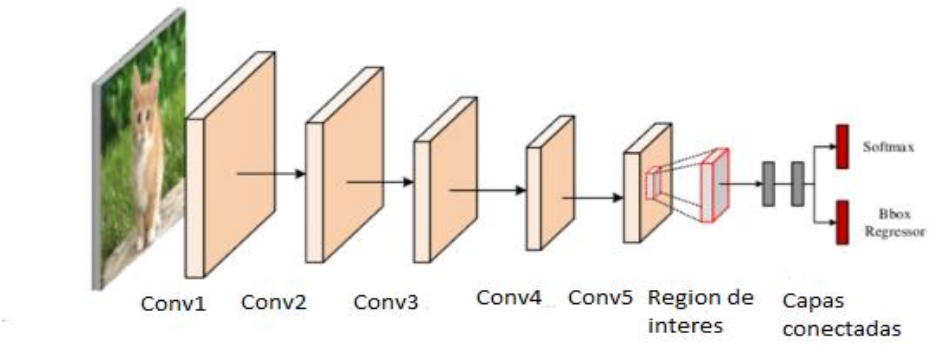
La RPN utiliza una ventana deslizante sobre los mapas de características y, en cada posición, genera múltiples "anclas" con diferentes tamaños y proporciones. Sus componentes claves son:

- **Cajas de anclaje (*Anchor Boxes*)**: Son fundamentales en el proceso de generación de propuestas de región dentro de Faster R-CNN. Estas son una serie de cuadros predefinidos con diferentes tamaños y formas que se colocan estratégicamente en distintas zonas del mapa de características extraído por la red convolucional (CNN). El objetivo es cubrir la mayor variedad posible de objetos en cuanto a tamaño y forma
- **Enfoque de Ventana Deslizante (*Sliding Window Approach*)**: La red de propuestas de regiones (RPN) aplica una estrategia de ventana deslizante sobre el mapa de características para analizar diferentes zonas de la imagen. En cada posición, una pequeña red convolucional.
- **Puntuación de Objetividad (Objectness Score):** Esta puntuación expresa qué tan probable es que una caja de anclaje contenga un objeto relevante en lugar de solo mostrar parte del fondo. En otras palabras, evalúa si vale la pena considerar esa región. Durante el entrenamiento, esta puntuación se utiliza para etiquetar las cajas como positivas (si tienen alta probabilidad de contener un objeto) o negativas (si muestran fondo), ayudando a la red a aprender a distinguir entre ambas.
- **IoU (Intersección sobre Unión – Intersection over Union):** La métrica IoU sirve para medir qué tanto se superponen dos cajas delimitadoras. Se calcula dividiendo el área de la zona en la que ambas cajas coinciden por el área total que cubren ambas combinadas. El valor de IoU va de 0 a 1, y cuanto más cercano esté a 1, mayor será la coincidencia entre las cajas. Esta métrica es esencial para decidir si una propuesta de región es lo suficientemente buena o no.
- **Supresión No Máxima (Non-Maximum Suppression - NMS):** Cuando varias cajas proponen el mismo objeto con puntuaciones similares, se genera redundancia. Para evitar

esto, se aplica la supresión no máxima. Este proceso consiste en conservar únicamente la caja que tiene la puntuación de objetividad más alta y descartar las demás que se superpongan demasiado con ella (según el valor de IoU). Así se garantiza que el modelo no detecte el mismo objeto varias veces.

Figura 4.

Arquitectura Faster R-CNN



Fuente. Tomado de Zhao et al., 2019. “traducida al español”

Modulo 2: Detector Fast R-CNN

El detector F**ast R-CNN** es una parte clave dentro de la arquitectura **Faster R-CNN**, ya que se encarga de identificar los objetos que aparecen en las regiones sugeridas por la red de propuestas de regiones (RPN).

- **RoI Pooling (Agrupación por Regiones de Interés)**: El proceso comienza con las regiones sugeridas por la RPN, las cuales pueden tener tamaños distintos. Para que estas puedan ser procesadas por las capas siguientes de la red, se utiliza **RoI pooling**, una

técnica que estandariza el tamaño de las regiones. Este método divide cada propuesta en una cuadrícula de tamaño fijo y aplica una operación de **max pooling** en cada celda, generando así un mapa de características con dimensiones constantes que facilita su análisis posterior.

- **Extracción de Características**: Después del pooling, los mapas de características se introducen en la red convolucional base (la misma usada por la RPN). Esta red extrae información representativa de cada región, capturando tanto detalles visuales como patrones estructurales importantes, lo cual permite identificar con mayor precisión qué hay en cada región.
- **Capas Totalmente Conectadas:** Una vez obtenidas las características, estas se pasan por varias capas completamente conectadas. Estas capas se encargan de realizar dos tareas simultáneamente: clasificar el contenido de cada región y ajustar las coordenadas de las cajas delimitadoras para que coincidan mejor con el objeto real.
- **Clasificación de Objetos:** La red calcula la probabilidad de que cada región pertenezca a una clase específica. Esta predicción se basa en las características extraídas y en los pesos aprendidos por la red durante el entrenamiento. También considera una clase adicional para el fondo (es decir, cuando no hay objeto en la región).
- **Regresión de las Cajas Delimitadoras:** Además de clasificar, la red también predice una serie de ajustes para mejorar la precisión de las cajas que rodean los objetos. Esta etapa busca que la caja generada se ajuste de forma más exacta al objeto real, refinando tanto su posición como su tamaño.
- **Función de Pérdida Multitarea (*Multi-task Loss Function*):** El entrenamiento del modelo se realiza optimizando una función de pérdida que combina dos tipos de errores;

el error de clasificación (para saber si la región contiene un objeto y de qué clase) y el error de regresión (para ajustar correctamente la caja que lo rodea).

- **Posprocesamiento (Post-Processing):** Una vez que se generan las predicciones (clases y cajas), se realiza un refinamiento final de los resultados. Aquí se aplica nuevamente la técnica de **supresión no máxima (NMS)**, que elimina duplicados innecesarios al conservar solo las cajas con mayores puntuaciones que no se sobrepongan demasiado. Esto asegura que cada objeto sea detectado una sola vez, de forma precisa.

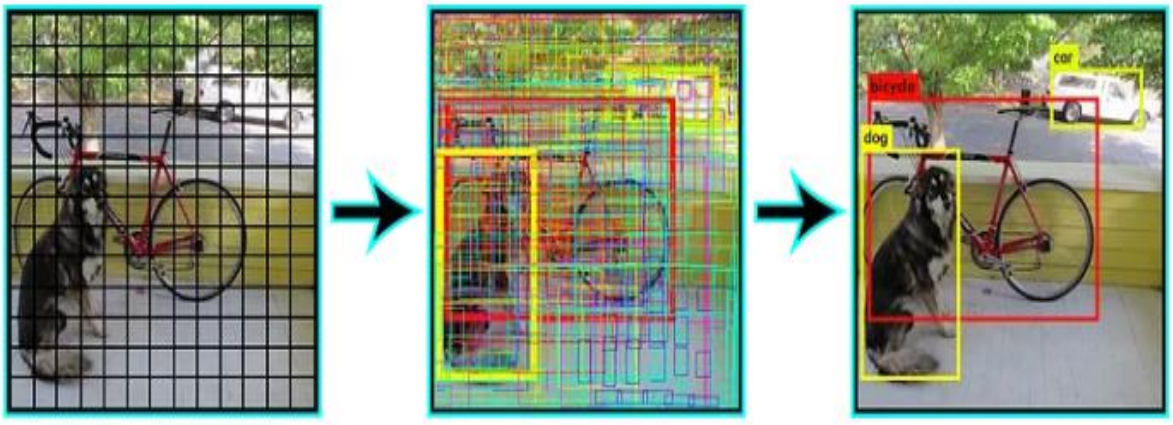
Ventajas clave

Una de las mayores ventajas de Faster R-CNN es su equilibrio entre precisión y eficiencia. Al integrar la RPN dentro del mismo modelo, se logra un sistema de detección mucho más rápido comparado con sus predecesores. Además, al compartir la mayoría de los cálculos entre la RPN y la red de detección, se reduce el tiempo de procesamiento, lo cual es fundamental en aplicaciones del mundo real como vigilancia, vehículos autónomos, o análisis de video en tiempo real.

Lo que hace realmente poderosa esta arquitectura es su enfoque completamente integrado. A diferencia de las versiones anteriores donde la propuesta de regiones era un proceso separado y costoso computacionalmente, Faster R-CNN lo unifica todo en una sola red entrenable de extremo a extremo. Esto permite mejorar tanto la precisión como el tiempo de inferencia, además, su diseño modular permite que se adapte fácilmente a nuevas clases o que se integre con nuevas tecnologías, como detección en vídeo o análisis multicámara. (Robu, 2021)

Figura 5.

Funcionamiento Yolo



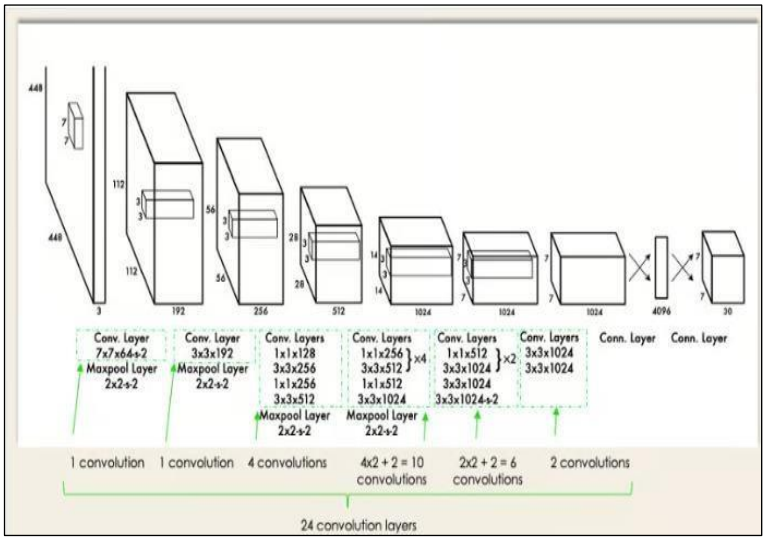
Fuente. Tomado de Moreira Ramos (2021)

Características Destacadas de YOLO

- Velocidad: YOLO es notablemente rápido, capaz de procesar imágenes a 45 cuadros por segundo (FPS), lo que lo hace adecuado para aplicaciones en tiempo real.
- Precisión: Ofrece una alta precisión en la detección, minimizando errores de fondo y mejorando la identificación correcta de objetos.
- Generalización: YOLO muestra una sólida capacidad de generalización para nuevos dominios, facilitando su aplicación en diversos contextos y escenarios.

Figura 6.

Arquitectura de la red Yolo



Fuente. Tomado de DataCamp. (2024).

Funcionamiento de YOLO

El proceso de detección de YOLO se basa en cuatro componentes principales:

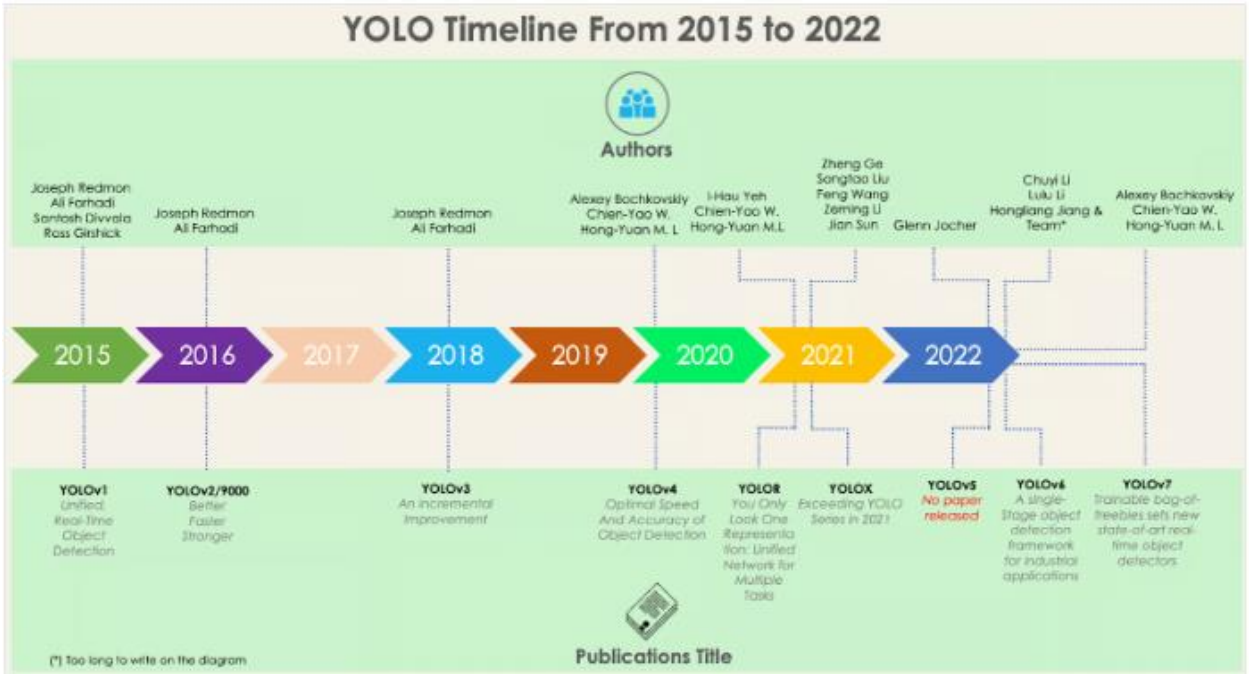
1. Bloques Residuales: La imagen de entrada se divide en una cuadrícula de celdas, y cada celda es responsable de predecir las cajas delimitadoras y las probabilidades de clase para los objetos cuyo centro cae dentro de la celda. En la Figura 5 donde se muestra como el modelo detecta múltiples objetos en tiempo real, utilizando un solo paso de inferencia sobre la imagen completa. La arquitectura general de esta red se representa en la Figura 6, que ilustra cómo se conectan los distintos bloques del modelo, desde la entrada de la imagen hasta la salida de predicciones.
2. Regresión de Cajas Delimitadoras: YOLO predice directamente las coordenadas de las cajas delimitadoras, junto con las probabilidades de clase, utilizando una única red de regresión.

3. Intersección sobre Unión (IoU): Esta métrica evalúa la precisión de las cajas delimitadoras predichas comparándolas con las reales, ayudando a filtrar predicciones inexactas.
4. Supresión de No Máximos: Se aplica para eliminar cajas redundantes o superpuestas, conservando únicamente las predicciones más precisas y relevantes.

La evolución de YOLO a través de sus distintas versiones, desde 2015 hasta 2022, se presenta en la **Figura 7**, donde se visualizan las mejoras progresivas en términos de velocidad, precisión y capacidades funcionales del algoritmo.

Figura 7.

Cronología de las versiones del algoritmo Yolo de 2015 a 2022



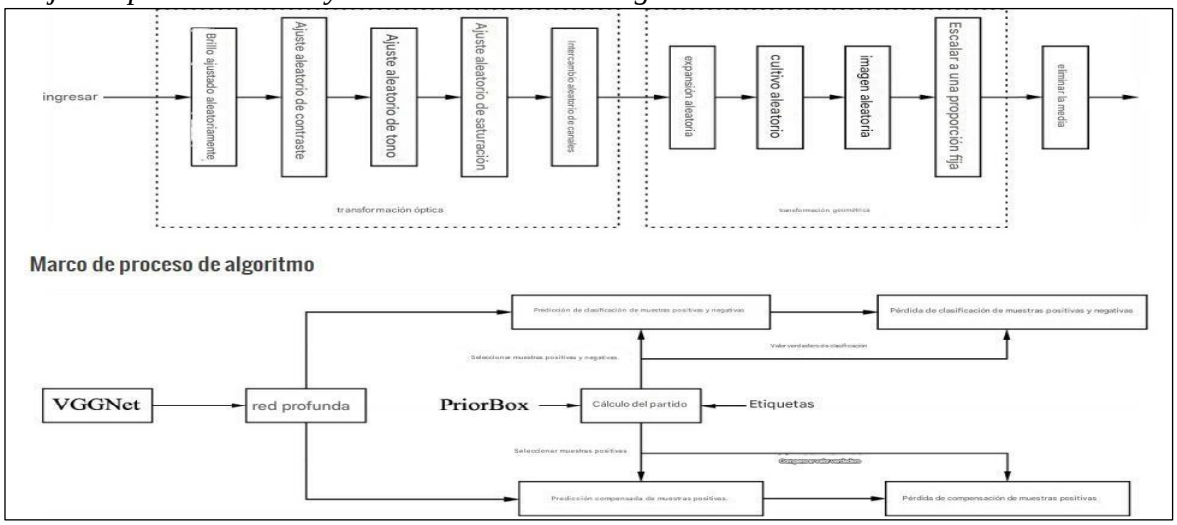
Fuente. Tomado de DataCamp. (2024).

SSD (Single Shot MultiBox Detector)

El Single Shot MultiBox Detector (SSD) es un modelo de detección de objetos de una sola etapa que realiza predicciones en múltiples escalas y desde diferentes niveles de características. A diferencia de modelos anteriores como Faster R-CNN, que emplean una arquitectura de dos etapas, SSD integra la detección y clasificación de objetos en un solo paso, lo que mejora significativamente la velocidad de procesamiento sin comprometer la precisión. Esta eficiencia lo hace especialmente adecuado para aplicaciones en tiempo real.

Figura 8.

Flujo del procesamiento y entrenamiento en el algoritmo SSD.



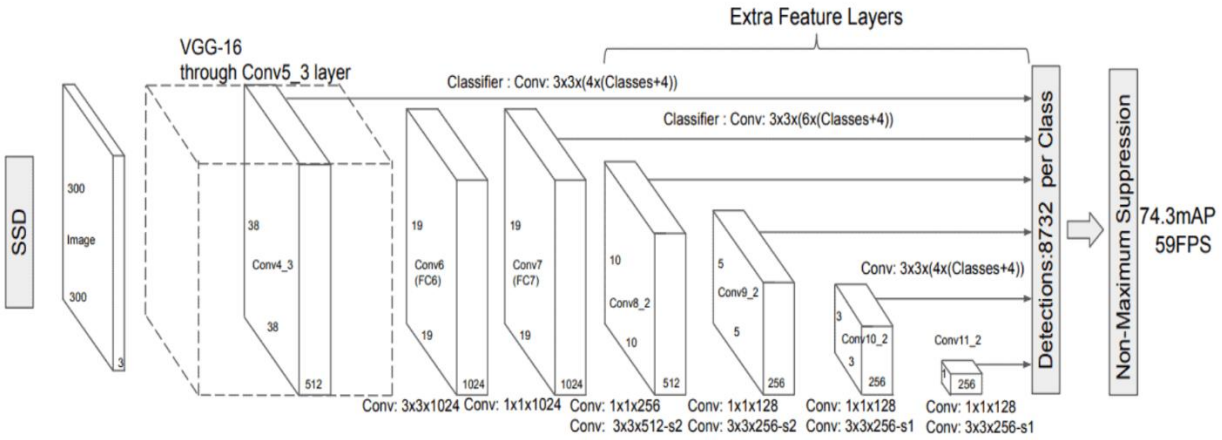
Fuente. Marco del proceso del algoritmo- Programmer Click. (s.f.).

La Figura 8 ilustra el marco general de procesamiento del algoritmo SSD, incluyendo las transformaciones aplicadas a las imágenes y los componentes clave que permiten la predicción y clasificación de objetos.

Por su parte, la Figura 9 presenta la arquitectura detallada del modelo SSD, basada en VGG16, con sus capas convolucionales y las capas adicionales para detección en distintas escalas.

Figura 9.

Arquitectura SSD



Fuente. Tomado de *"Object Detection With SSD and YOLO"* por E. Chikovani, Baeldung, s.f., bajo licencia CC BY-SA 4.0.

Arquitectura del SSD

El modelo SSD utiliza una versión modificada de la arquitectura VGG16 como su red base. Las capas completamente conectadas de VGG16 se reemplazan por capas convolucionales, permitiendo la extracción de mapas de características en diferentes resoluciones. Además, se añaden capas convolucionales adicionales después de la red base para detectar objetos a diferentes escalas. Cada una de estas capas se encarga de predecir la presencia de objetos y ajustar las cajas delimitadoras correspondientes.

Ventajas del SSD

- Velocidad: Al integrar la detección y clasificación en un solo paso, SSD es capaz de operar en tiempo real, procesando múltiples cuadros por segundo.
- Precisión en Múltiples Escalas: La capacidad de realizar predicciones en diferentes niveles de características permite al SSD detectar objetos de diversos tamaños con alta precisión.
- Simplicidad: La arquitectura unificada de SSD simplifica el proceso de entrenamiento y despliegue en comparación con modelos de dos etapas. (Programmer Click, s.f.)

Transfer Learning (Aprendizaje por Transferencia)

El aprendizaje por transferencia (Transfer Learning) se ha convertido en una técnica clave dentro del aprendizaje automático. Su enfoque radica en aprovechar el conocimiento adquirido en una tarea para aplicarlo a una tarea diferente pero relacionada, lo cual permite reducir la necesidad de grandes volúmenes de datos y recursos computacionales. A diferencia del aprendizaje tradicional, donde cada tarea debe resolverse desde cero, esta técnica permite una mayor generalización, acelerando el entrenamiento y reduciendo la complejidad de los modelos sin sacrificar precisión. Además, permite alcanzar buenos resultados con menos iteraciones y datos, lo que resulta beneficioso cuando se cuenta con hardware limitado o datasets pequeños (De Luca, Irigoitia, Pérez & Pons, 2021).

Aplicación del Aprendizaje por Transferencia (Transfer Learning)

Factores que inciden en el rendimiento del entrenamiento

- Procesamiento de imágenes: Cada imagen está representada como una matriz tridimensional, y su análisis mediante convoluciones está determinado por el tamaño

de la imagen y la configuración del kernel. Esto tiene un impacto directo en el tiempo de cómputo.

- Duración del entrenamiento: El tiempo requerido para entrenar la red depende en gran medida de la capacidad del hardware disponible.
- Tipo de software: La eficiencia del entrenamiento también está condicionada por la herramienta de software utilizada y su compatibilidad con el hardware.
- Número de clases: A mayor cantidad de clases (categorías a predecir), mayor será la complejidad del modelo.
- Tamaño del dataset: Un conjunto de datos extenso incrementa la calidad del modelo, pero también exige más tiempo de entrenamiento.
- Aumento de datos (Data Augmentation): Aplicar transformaciones como rotación o inversión a las imágenes permite generar nuevos ejemplos, mejorando la generalización del modelo.
- Hiperparámetros: Variables como la tasa de aprendizaje o el tamaño del lote (batch size) impactan directamente en el proceso de optimización.
- Calidad del etiquetado: La precisión de las etiquetas influye directamente en la capacidad del modelo para aprender correctamente.

Estrategias para el uso de modelos preentrenados

El uso de modelos preentrenados es una práctica común en Transfer Learning. Para adaptarlos a una nueva tarea, se reemplaza la capa clasificadora final por una personalizada, y luego se aplica una de las siguientes estrategias de entrenamiento representadas gráficamente en la Figura 10:

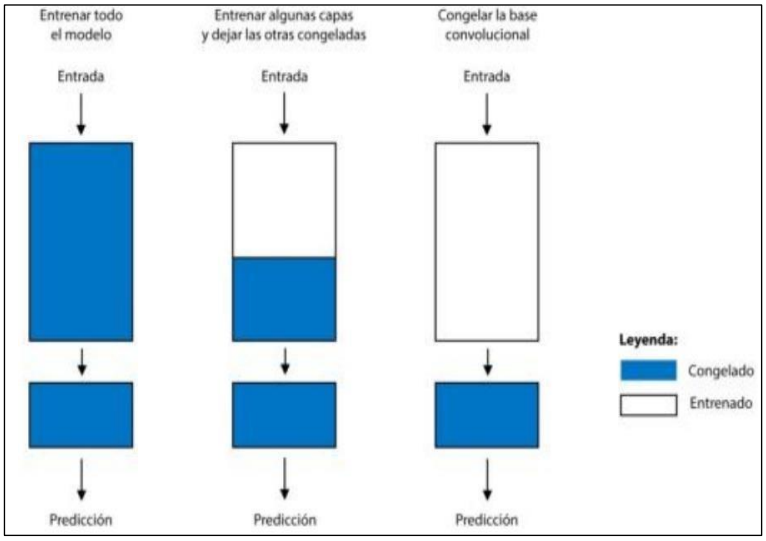
Entrenamiento completo del modelo: Consiste en reutilizar la arquitectura completa del modelo preentrenado, pero reentrenarlo completamente con el nuevo dataset. Esta estrategia requiere una gran cantidad de datos y recursos computacionales.

Entrenamiento parcial (congelar capas): Algunas capas, especialmente las iniciales, se mantienen fijas (congeladas), mientras que otras se ajustan. Esta estrategia es útil cuando se cuenta con un dataset pequeño, ya que las primeras capas extraen características genéricas que no es necesario modificar.

Uso como extractor de características: Se congela toda la red convolucional preentrenada y se utiliza como un extractor de características, alimentando un nuevo clasificador. Esta estrategia es ideal cuando los recursos computacionales son limitados o se dispone de pocos datos. (De Luca, Irigoitia, Pérez & Pons, 2021).

Figura 10.

Estrategias de entrenamiento Transfer Learning



Fuente. De Luca, Irigoitia, Pérez y Pons (2021).

Marco metodológico

Enfoque

La presente investigación adopta un enfoque cuantitativo, de tipo aplicado y con diseño experimental, con el propósito de analizar la viabilidad de utilizar redes neuronales, para la detección automática de personas en medios acuáticos. El enfoque cuantitativo permite evaluar objetivamente el desempeño del modelo mediante métricas como precisión y recall, facilitando la validación de hipótesis a partir de datos medibles.

El carácter aplicado de la investigación se justifica en tanto que busca resolver un problema concreto relacionado con la seguridad en entornos acuáticos mediante herramientas de inteligencia artificial. A diferencia de una investigación puramente teórica, este estudio implica la implementación de modelos reales, su entrenamiento con datos relevantes y la validación en escenarios controlados.

El diseño experimental se manifiesta en la manipulación intencionada de variables como el tipo de arquitectura de red neuronal, el conjunto de datos utilizado y los parámetros de entrenamiento, para observar su efecto en la capacidad del sistema de detección. Se utilizó un dataset construido manualmente con imágenes reales tomadas en piscinas, las cuales fueron anotadas para entrenar y evaluar el modelo. El proceso experimental implicó la división de los datos en conjuntos de entrenamiento y prueba, la configuración del modelo, su entrenamiento mediante técnicas de transferencia de aprendizaje (transfer learning), y la posterior evaluación de su rendimiento.

En síntesis, la metodología empleada permite determinar de forma rigurosa la efectividad del modelo propuesto para detectar personas en cuerpos de agua, con el fin de sentar las bases para futuras aplicaciones prácticas en contextos de seguridad acuática.

Prueba, Entrenamiento y Resultados

Fase 1. Elección del Modelo

La identificación precisa y oportuna de personas en entornos acuáticos representa un desafío técnico relevante para la implementación de sistemas de alerta temprana. En este contexto, la elección de un modelo de detección de objetos adecuado resulta fundamental para garantizar la eficacia del prototipo propuesto. Actualmente, existen diversos algoritmos de aprendizaje automático enfocados en la detección de objetos, entre los cuales destacan YOLO, Faster R-CNN y SSD, ampliamente reconocidos por su aplicación en tareas de visión por computador. Esta fase del proyecto tiene como propósito analizar y comparar las características de dichos modelos, con el fin de seleccionar aquel que ofrezca el mejor equilibrio entre precisión, velocidad de detección y adaptabilidad al contexto marino, particularmente orientado a la identificación de personas dentro o cerca del agua.

En el proceso de selección del modelo más adecuado para detectar personas en ambientes acuáticos, es fundamental conocer las características de los algoritmos más utilizados en la detección de objetos. De acuerdo con Rodríguez, Muñoz y Martínez (2023), YOLO (You Only Look Once) destaca por su velocidad, ya que divide la imagen en una cuadrícula y realiza predicciones simultáneas de clases y ubicaciones, lo que lo hace ideal para tareas en tiempo real.

Por otro lado, Faster R-CNN ofrece una mayor precisión al usar una red de propuestas de regiones (RPN) que identifica zonas de interés y las clasifica posteriormente, aunque con mayor

carga computacional. Finalmente, el algoritmo SSD (Single Shot MultiBox Detector) busca un equilibrio entre precisión y velocidad, al realizar detecciones desde diferentes escalas dentro de los mapas de características sin etapas intermedias. Estas características permiten comparar su aplicabilidad según los requerimientos del entorno acuático, donde la rapidez y exactitud en la detección pueden ser cruciales. Esta comparación se puede detallar en la Figura 11, donde se visualizan los resultados obtenidos por cada modelo al detectar personas y objetos en una escena acuática junto con si tiempo de ejecución. (Rodríguez, Muñoz y Martínez, 2023).

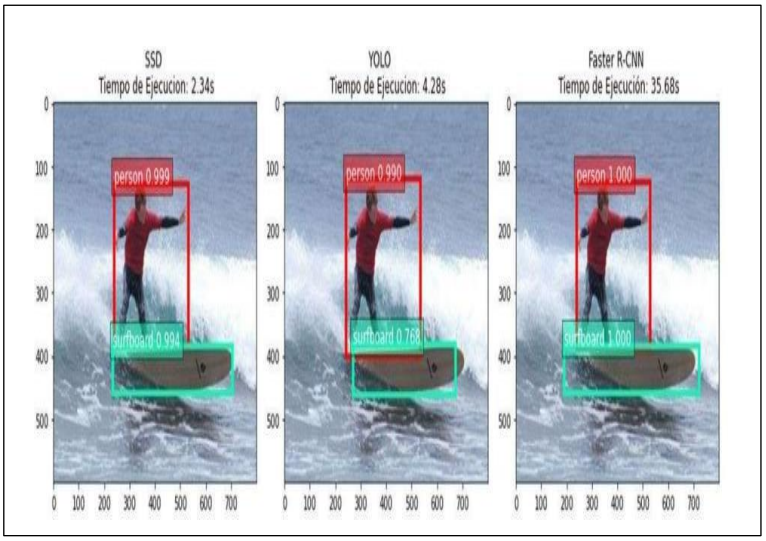
Respecto a los criterios de selección del modelo para definir el modelo más adecuado en la detección de personas en ambientes acuáticos, se consideraron tres criterios esenciales: precisión, robustez en condiciones visuales complejas y capacidad de generalización. La precisión es prioritaria, ya que una falsa negativa podría significar no detectar a una persona en riesgo. La robustez implica mantener el rendimiento frente a factores como oclusiones parciales, movimiento del agua o reflejos, frecuentes en entornos marinos. Finalmente, la generalización se refiere a la capacidad del modelo para adaptarse a distintas situaciones sin perder efectividad, considerando variaciones como clima, ubicación o tipo de cámara.

En relación con las ventajas comparativas del modelo elegido, de acuerdo con Rodríguez, Muñoz y Martínez (2023), el modelo Faster R-CNN ofrece una alta precisión al detectar objetos, gracias a su arquitectura basada en una red de propuestas de regiones (RPN) que mejora la localización y clasificación. Aunque este modelo puede requerir mayor capacidad computacional y tiempo de procesamiento frente a alternativas como YOLO o SSD, su exactitud lo hace idóneo para contextos donde la prioridad es evitar errores críticos de detección, como el monitoreo de bañistas en el agua. Además, Faster R-CNN permite un mejor ajuste a diferentes escalas y es menos propenso a errores en

imágenes con elementos visuales complejos, lo cual es una ventaja importante en escenarios acuáticos.

Figura 11.

Comparación visual entre los algoritmos SSD, YOLO y Faster R-CNN.



Fuente. Tomado de Academia Journals Monterrey – UACJ (2023).

En función de los criterios establecidos y del análisis comparativo, se determinó que Faster R-CNN es el modelo más adecuado para este proyecto. Su capacidad para detectar con precisión personas en imágenes con variaciones visuales significativas lo convierte en una herramienta confiable para sistemas de alerta en ambientes acuáticos. Aunque su velocidad es menor en comparación con otros modelos como YOLO, la prioridad del proyecto está centrada en la fiabilidad del reconocimiento, más que en la velocidad de procesamiento. Por esta razón, se ha seleccionado Faster R-CNN como base para el desarrollo del prototipo, utilizando técnicas de transfer learning para optimizar su desempeño en el contexto específico del proyecto

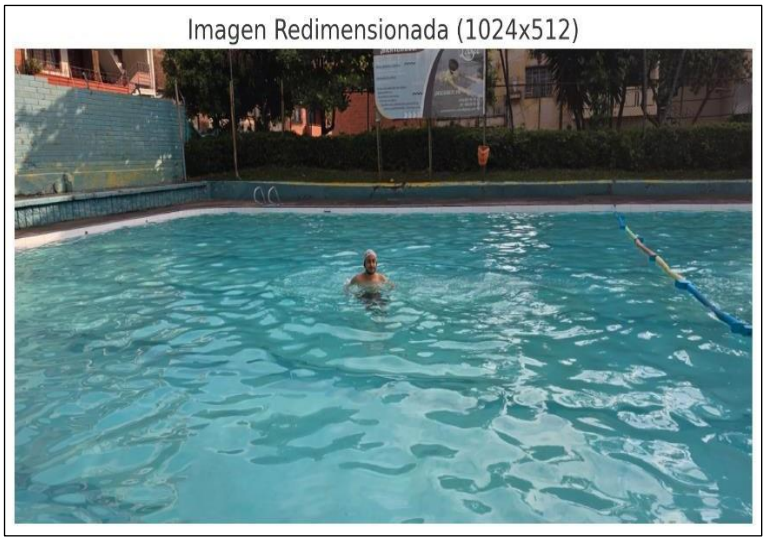
Fase 2. Construcción del Conjunto de Datos

El desempeño de los modelos de detección de objetos depende en gran medida de la calidad, variedad y precisión del conjunto de datos utilizado para su entrenamiento (Ian

Goodfellow, Yoshua Bengio, Aaron Courville, 2016). En esta fase, se procedió a la recolección, anotación y validación de un conjunto de imágenes enfocadas en la detección de personas en entornos acuáticos. Antes de pasar las imágenes por la herramienta que se utilizó en el proceso de anotaciones, estas se redimensionaron y normalizaron a un tamaño estándar (1024x512) para mejorar la eficiencia del entrenamiento como se muestra en la Figura 12.

Figura 12.

Imagen redimensionada utilizada para las pruebas (1024x512)

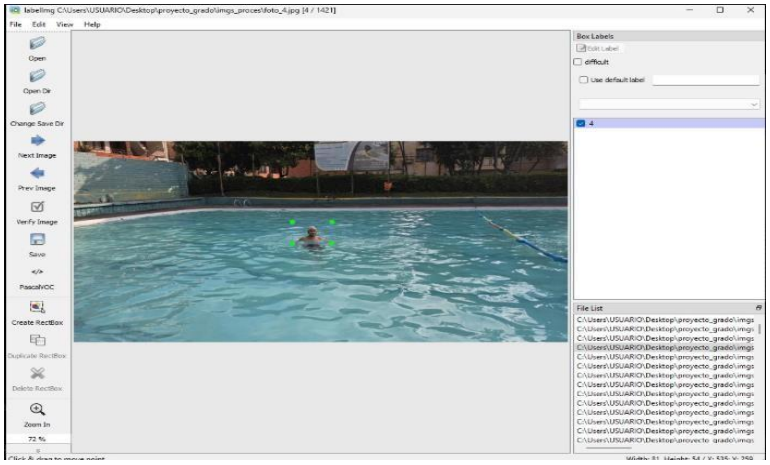


Fuente. Elaboración propia

El Dataset fue construido manualmente a partir de fotografías reales capturadas en piscinas, las cuales representan contextos de personas total o parcialmente visibles dentro del agua. En total, se recopilaron aproximadamente 800 imágenes, con distintas condiciones de luz, ángulos de toma y nivel de inmersión, lo que aporta variedad al entrenamiento del modelo y mejora su capacidad de generalización.

Figura 13.

Anotación manual de bounding box en LabelImg.

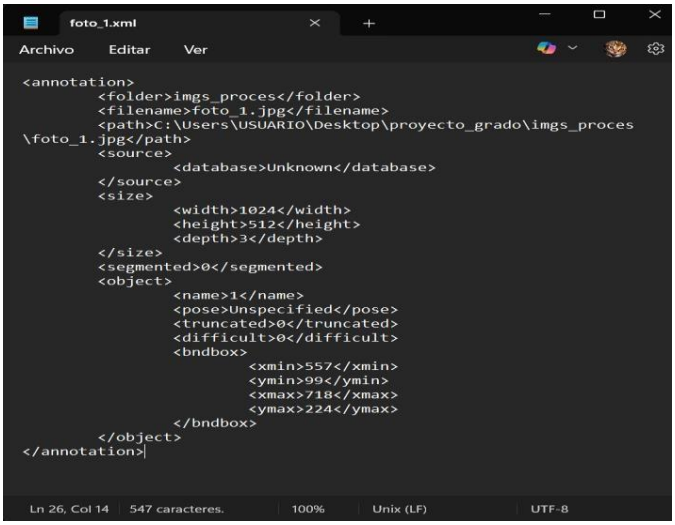


Fuente. Elaboración propia

Durante la construcción inicial del dataset, se utilizó la herramienta LabelImg, la cual permite generar anotaciones en formato XML bajo el estándar Pascal VOC. Con estas anotaciones se entrenó un primer modelo de detección. A pesar que ya hay herramientas automáticas que podrían haber hecho este proceso menos demorado se eligió realizar el proceso de manera manual por la ventaja que hay en la precisión en la anotación. La Figura 13 muestra la interfaz de LabelImg, donde se definieron manualmente las *bounding boxes* alrededor de las personas en cada imagen.

Figura 14.

Ejemplo de archivo XML de anotación generado para una imagen.



Fuente. Elaboración propia

Cada imagen fue anotada manualmente para definir las cajas delimitadoras (bounding boxes) y asignar la etiqueta correspondiente a cada objeto (en este caso, “persona”). Las anotaciones fueron validadas visualmente para asegurar consistencia y precisión, y aquellas imágenes con errores de iluminación, enfoque o sin presencia clara de personas fueron descartadas para evitar ruido en el entrenamiento. La Figura 14 presenta un ejemplo del archivo XML generado por *LabelImg*, mientras que la Figura 15 ilustra la conversión de ese mismo XML al formato JSON, requerido para el código personalizado de entrenamiento utilizado en este proyecto. Aunque inicialmente se utilizaron anotaciones en formato XML (generadas por LabelImg), este formato estaba estructurado según el estándar Pascal VOC, el cual no se ajustaba completamente a las necesidades del código personalizado de entrenamiento utilizado en el proyecto.

Figura 15.

Anotaciones convertidas a formato Json


```
{
    "nombre_archivo": "foto_1.jpg",
    "tama\u00f1o": {
        "ancho": 1024,
        "alto": 512,
        "profundidad": 3
    },
    "objetos": [
        {
            "nombre": "1",
            "xmin": 557,
            "ymin": 99,
            "xmax": 718,
            "ymax": 224
        }
    ]
}
```


Como las anotaciones fueron posteriormente convertidas en formato JSON para el primer entrenamiento podemos ver más claramente:

- Nombre del archivo
- Tamaño (alto, ancho, profundidad)
- Bounding boxes (xmin, ymin, xmax, ymax)
- Formato listo para entrenar con Pytorch y Faster R-CNN.

Continuando con el proceso se desarrolló un script en Python que estructuró el dataset a partir de las imágenes procesadas y sus respectivas anotaciones en formato JSON. Dicho script fue implementado a través de una clase personalizada que hereda de “torch.utils.data.Dataset”

Figura 16.

Importación de librerías en Pycharm

```
import torch
from torch.utils.data import Dataset
from PIL import Image
import json
import os
from torchvision.transforms import ToTensor
```

Se utilizaron librerías fundamentales como PIL para el manejo de imágenes, JSON para leer las anotaciones generadas y torch para estructurar los datos como tensores. La Figura 16 muestra el fragmento del código de las librerías necesarias y se define La clase DatasetPiscina como se detalla en la Figura 17, que hereda de Dataset de PyTorch, lo que permite que las imágenes y sus anotaciones puedan ser cargadas por un DataLoader.

Figura 17,

Definición inicial de la clase DatasetPiscina, utilizada para estructurar el dataset

```
class DatasetPiscina(Dataset):  4 usages
    def __init__(self, ruta_imagenes, ruta_anotaciones):
        self.ruta_imagenes = ruta_imagenes
        self.ruta_anotaciones = ruta_anotaciones
        self.imagenes = sorted([img for img in os.listdir(ruta_imagenes) if img.endswith(('.jpg', '.png'))])
```

El método __getitem__ implementado en la clase DatasetPiscina permite acceder a una imagen específica mediante su índice, obteniendo tanto la imagen como su correspondiente archivo de anotación en formato JSON. Para ello, se construye la ruta de acceso a la imagen, se convierte al formato RGB para su procesamiento y, posteriormente, se accede a la anotación renombrando la extensión de la imagen a. JSON, como se muestra en la Figura 18.

Figura 18.

Método getitem encargado de cargar una imagen y su respectiva anotación en formato JSON.

```
def __getitem__(self, indice):
    nombre_imagen = self.imagenes[indice]

    ruta_imagen = os.path.join(self.ruta_imagenes, nombre_imagen)
    imagen = Image.open(ruta_imagen).convert("RGB")

    nombre_json = nombre_imagen.replace( _old: '.jpg', _new: '.json')
    ruta_json = os.path.join(self.ruta_anotaciones, nombre_json)

    with open(ruta_json, 'r') as archivo:
        anotacion = json.load(archivo)
```

Nota. Fragmento de código encargado de cargar una imagen y su respectiva anotación en formato JSON desde el índice correspondiente.

Posteriormente como se observa en la Figura 19 Se recorren los objetos anotados en el archivo JSON para extraer las coordenadas de las personas en la imagen. Estas coordenadas se almacenan en formato de tensores, que son la entrada esperada por los modelos de PyTorch.

Figura 19.

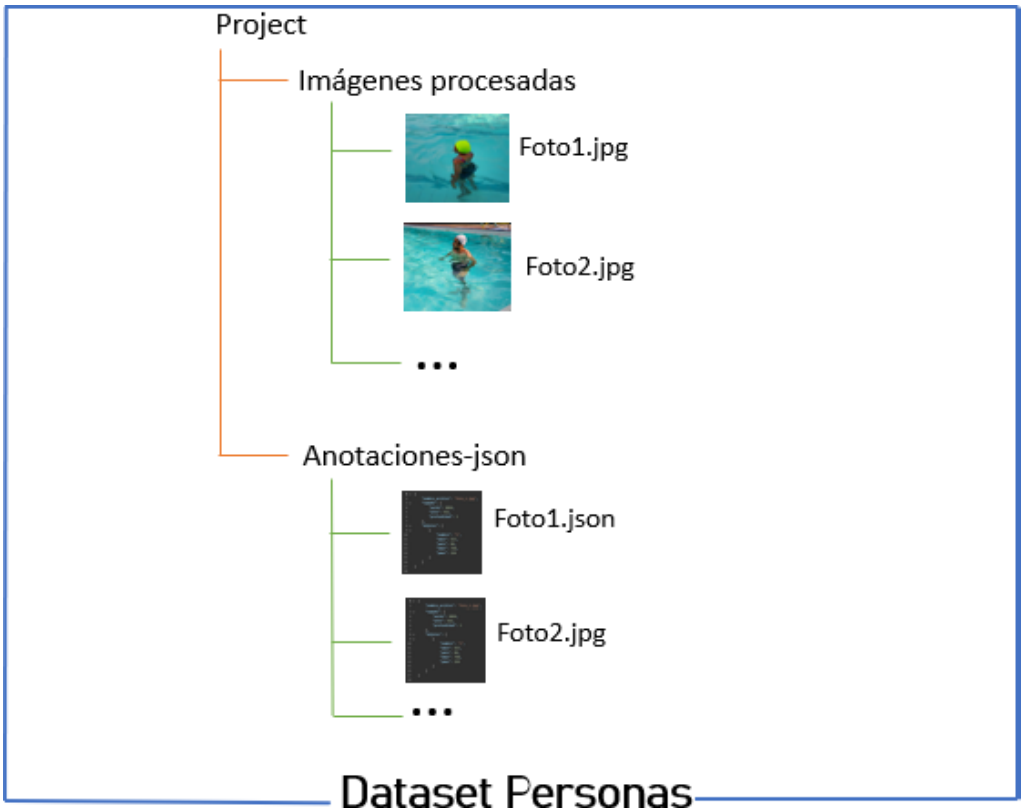
Extracción y conversión de las coordenadas y etiquetas a tensores para uso del entrenamiento.

```
30
31          cajas = []
32          etiquetas = []
33
34          for obj in anotacion['objetos']:
35              xmin = obj['xmin']
36              ymin = obj['ymin']
37              xmax = obj['xmax']
38              ymax = obj['ymax']
39              cajas.append([xmin, ymin, xmax, ymax])
40              etiquetas.append(1)  # '1' representa la clase "persona"
41
42          # Convertir a tensores
43          cajas = torch.as_tensor(cajas, dtype=torch.float32)
44          etiquetas = torch.as_tensor(etiquetas, dtype=torch.int64)
45
46          objetivo = {}
47          objetivo["boxes"] = cajas
48          objetivo["labels"] = etiquetas
49
50          # Aplicar transformación a la imagen
51          imagen = ToTensor()(imagen)
52
53          return imagen, objetivo
```

La imagen se transforma en tensor y se retorna junto con su información de etiquetas, lista para ser utilizada por el modelo durante el entrenamiento. Finalmente, para asegurar una correcta lectura del dataset durante el entrenamiento, se estructuraron las imágenes y anotaciones en carpetas independientes. La Figura 20 muestra la organización del dataset, el cual incluye las imágenes y sus respectivas anotaciones organizadas bajo una estructura de proyecto clara y coherente.

Figura 20.

Estructura del dataset de imágenes procesadas y anotaciones.



Fuente. Elaboración propia

Fase 3. Entrenamiento

Una vez construido y validado el dataset con sus respectivas anotaciones en formato. json, se procedió con la fase de entrenamiento del modelo de detección de objetos. El modelo seleccionado fue **Faster R-CNN**, el cual se implementó utilizando la biblioteca **PyTorch**, haciendo uso de su arquitectura preentrenada sobre el conjunto de datos COCO, mediante la técnica de **Transfer Learning**.

Importación de Librerías: En el bloque de importaciones (ver Figura 21), se incorporan las librerías fundamentales para el entrenamiento y uso del modelo. Se hace uso de torch y

torchvision para acceder a las funcionalidades del modelo preentrenado Faster R-CNN (fasterrcnn_resnet50_fpn) y se modifica su cabezal de predicción utilizando FastRCNNPredictor. Asimismo, se importa el conjunto de datos personalizado DatasetPiscina, previamente definido, para su integración en el DataLoader.

Figura 21.

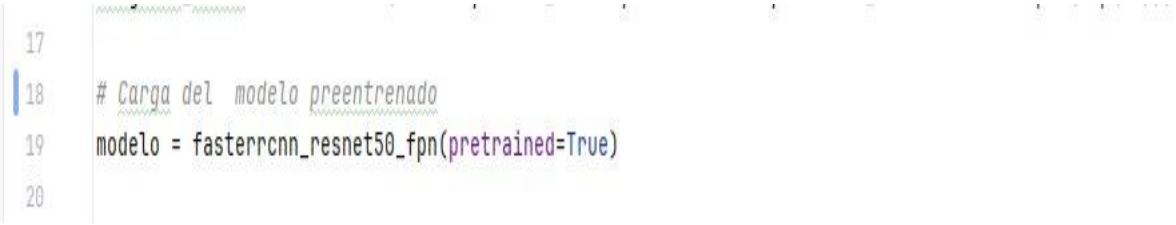
Módulos para el modelo Faster R-CNN

```
1  import torch
2  from torch.utils.data import DataLoader
3  from torchvision.models.detection import fasterrcnn_resnet50_fpn
4  from torchvision.models.detection.faster_rcnn import FastRCNNPredictor
5  from dataset_personas_piscina import DatasetPiscina
6
```

Carga del Modelo: La Figura 22 muestra cómo se modifica la capa final del modelo preentrenado Faster R-CNN (box_predictor) para ajustarse al número de clases específicas del proyecto: persona y fondo. El modelo originalmente fue entrenado con el conjunto de datos COCO, que contiene 91 clases. Sin embargo, en el presente trabajo solo se requiere detectar una clase de interés: “persona”. Por tanto, es necesario reemplazar la capa de predicción con una nueva instancia de FastRCNNPredictor, indicando explícitamente el número de clases (num_clases = 2), como se muestra en la Figura 23. Este cambio garantiza que el modelo no intente clasificar entre categorías innecesarias y se enfoque exclusivamente en la detección relevante para el contexto del proyecto.

Figura 22.

Visual de la carga del modelo



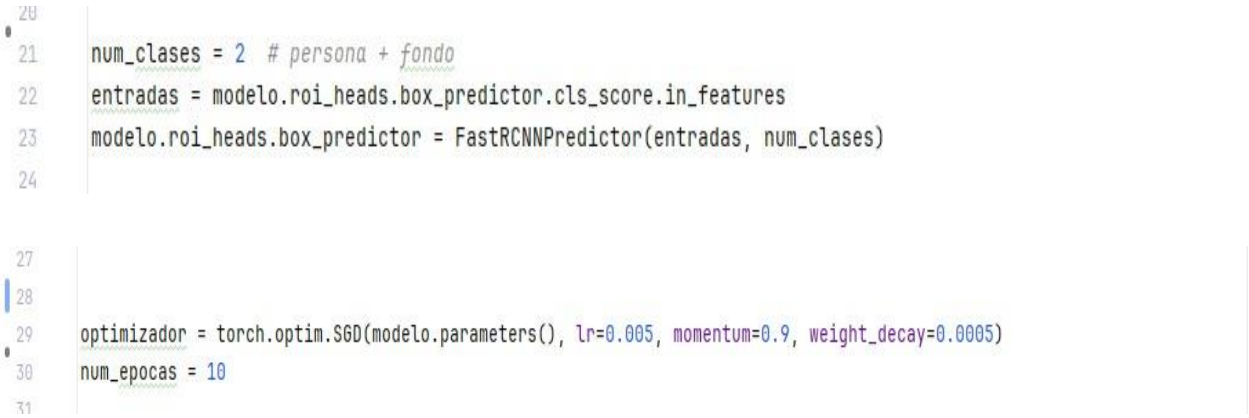
```
17
18  # Carga del  modelo preentrenado
19  modelo = fasterrcnn_resnet50_fpn(pretrained=True)
20
```

Configuración del entrenamiento

- Número de clases: 2 (persona y fondo)
- Clase 0: Fondo
- Clase 1: Persona en el agua

Figura 23.

Ajuste de la capa de predicción



```
20
21  num_clases = 2  # persona + fondo
22  entradas = modelo.roi_heads.box_predictor.cls_score.in_features
23  modelo.roi_heads.box_predictor = FastRCNNPredictor(entradas, num_clases)
24
```



```
27
28
29  optimizador = torch.optim.SGD(modelo.parameters(), lr=0.005, momentum=0.9, weight_decay=0.0005)
30  num_epocas = 10
31
```

- Épocas: 10
- Optimizador: SGD con tasa de aprendizaje 0.005, momentum=0.9 y weight_decay=0.0005

SGD (Stochastic Gradient Descent) es un algoritmo que ajusta los pesos del modelo para reducir la pérdida.

Cada época representa un ciclo completo en el que el modelo ve todo el conjunto de entrenamiento, el número de épocas es crucial para que el modelo aprenda lo suficiente sin sobreajustarse.

Figura 24.

Batch size: 4

```
15
16  cargador_datos = DataLoader(dataset, batch_size=4, shuffle=True, collate_fn=lambda x: tuple(zip(*x)))
17
```

Indica que, durante el entrenamiento, el modelo procesará **4 imágenes a la vez** en cada iteración (ver Figura 24) antes de actualizar los pesos. Esto ayuda a equilibrar el uso de memoria y la velocidad de entrenamiento.

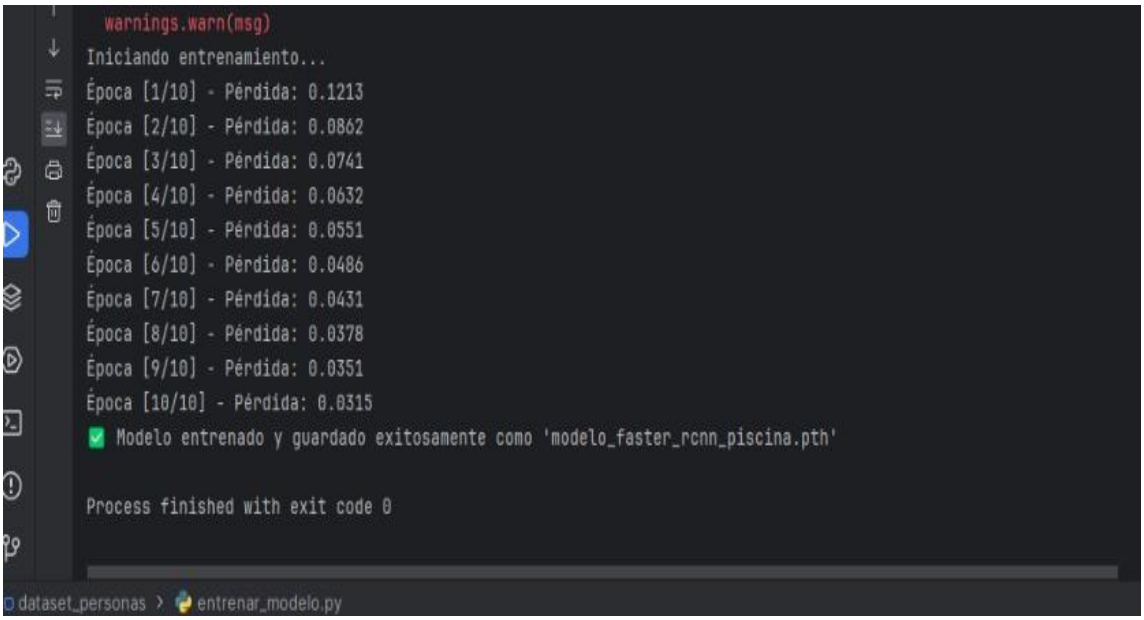
Visualización del Progreso y Resultado del Entrenamiento

La Figura 25 ilustra el proceso de entrenamiento del modelo Faster R-CNN utilizando las imágenes anotadas. Se observa cómo la pérdida (loss) disminuye progresivamente a lo largo de las 10 épocas, partiendo de un valor inicial de 0.1213 hasta alcanzar 0.0315 en la última iteración. Esta reducción constante es un indicativo de que el modelo fue capaz de aprender las características relevantes del conjunto de datos. Una vez finalizado el proceso, el modelo fue

guardado correctamente en el archivo modelo_faster_rcnn_piscina.pth para su posterior uso en tareas de inferencia.

Figura 25.

Entrenamiento 10 épocas



Fase 4. Prueba 1 y Análisis

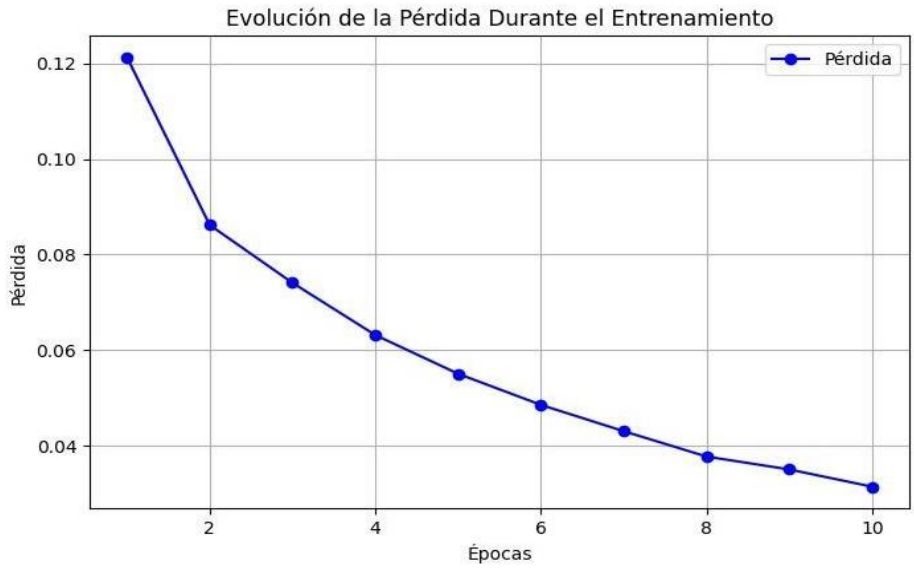
Ya teniendo el modelo entrenado se procede hacer predicciones en imágenes individuales, con el objetivo de medir qué tan bien funciona. Para lo anterior se creó un script de evaluación donde se carga el modelo entrenado junto con el Dataset de validación. Posterior se calcula la precisión y Recall para medir el rendimiento, se usa el cálculo de IoU para determinar si las predicciones son correctas y por último imprime los resultados.

Hasta esta fase se aclara que se usó CPU por lo que el modelo tuvo ciertas demoras en las inferencias y un umbral de confianza de 0.5. Cabe destacar que antes de probar en imágenes nuevas fue necesario mirar el comportamiento de la perdida de manera grafica.

En la figura 26 la pérdida disminuye con cada época de entrenamiento, esto significa que el modelo fue aprendiendo y mejorando su detección, adicionalmente la curva descendente indica que el modelo está convergiendo correctamente. Sin embargo, también se puede concluir por medio de la gráfica que la curva no llega aplanarse del todo. Esto sugiere que el modelo seguía aprendiendo al final de la décima época, por lo que sería conveniente aumentar el número de épocas para intentar mejorar la predicción del modelo.

Figura 26.

Evolución de la pérdida

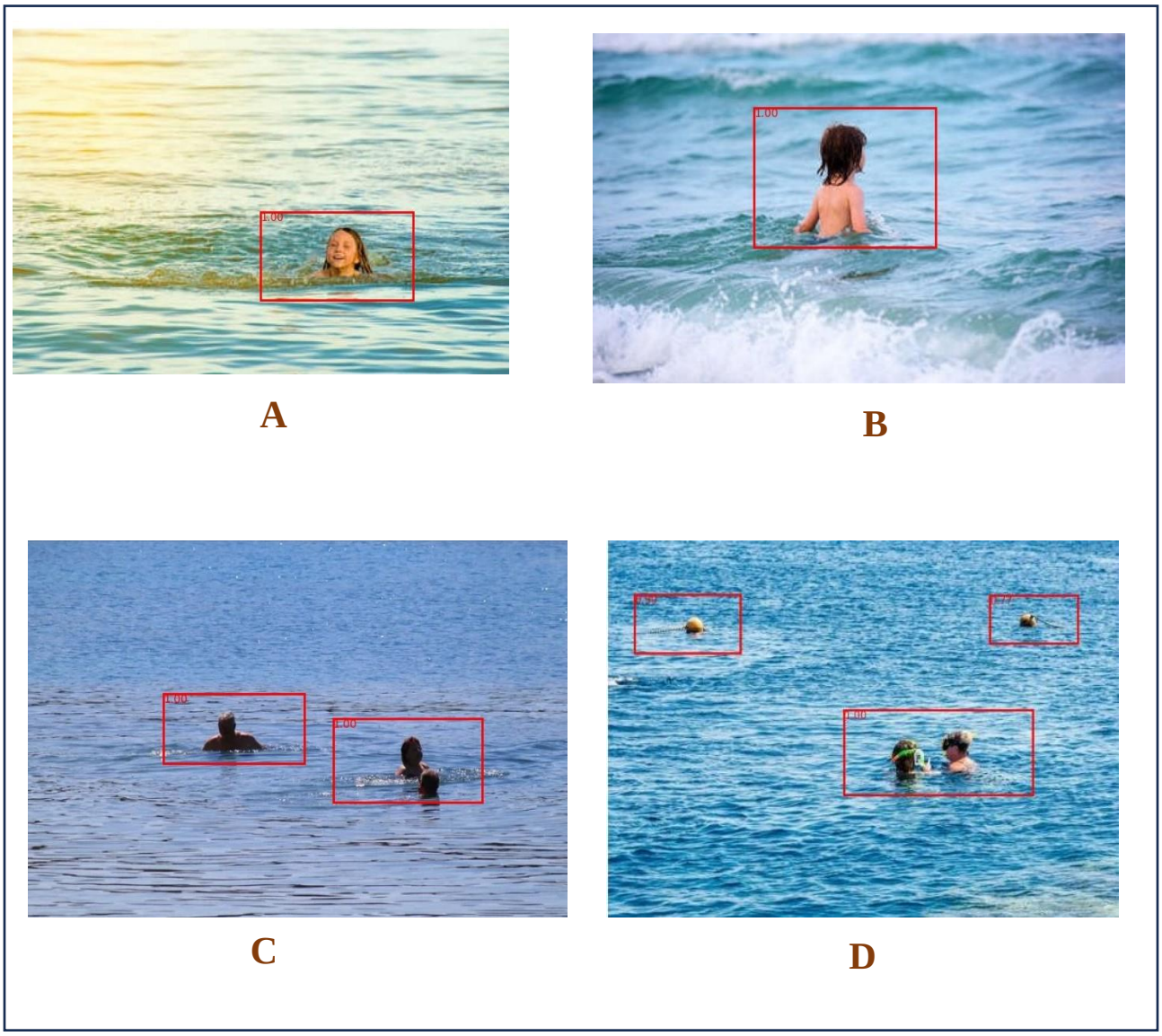


Primer resultado

Continuamos con la prueba con imágenes nuevas, para ello se creó un script que toma la nueva imagen y se le aplica el modelo entrenado con el objetivo de verificar que el modelo detecta correctamente personas en imágenes nuevas, comprobar que la detección es precisa y no tiene errores de clasificación y por último visualizar el resultado con las cajas de detección sobre la imagen.

Figura 27.

Resultado 1



En las primeras imágenes se puede observar que el modelo entrenado es capaz de detectar personas en el agua con un grado alto de confianza, lo que indicaba que el entrenamiento funcionaba correctamente, sin embargo, se buscó darle un grado de dificultad al modelo a ver como seria el resultado y se muestra que el modelo ha detectado correctamente a las personas en el agua, pero también ha identificado incorrectamente como se observa en la Figura 27 en la imagen ”D” se notan objetos que no son personas, como boyas o elementos flotantes. Este es un caso de falsos positivos, lo que indica que el modelo a veces confunde ciertos objetos con personas.

Lo anterior hizo que se reevaluara el modelo con algunas pequeñas modificaciones como las siguientes:

- Revisión del Dataset
- Aumento en la cantidad de datos
- Uso de GPU en reemplazo de CPU
- Ajuste del umbral de confianza
- Aumento de épocas de entrenamiento

Fase 5. Prueba 2 y Análisis

Revisión del Dataset y Aumento en la Cantidad de Datos

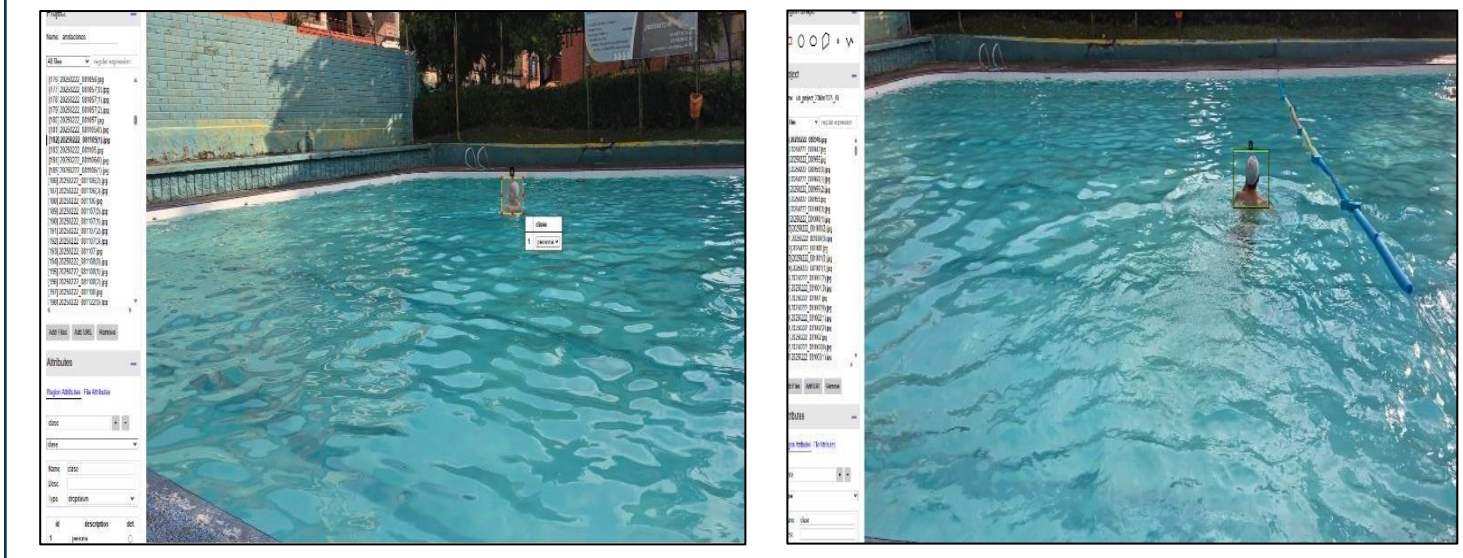
En la revisión se agregaron más imágenes para, es decir se pasó de 700 imágenes a 1200 imágenes para posteriormente realizar nuevamente la anotación de estas.

Modificación del anotador *labelimg* por *VGG 16*

Para las nuevas anotaciones se realizó el proceso con el anotador *VGG 16* como se muestra en la Figura 28, por facilidad en el formato generado ya que no requeriría conversión de XML a JSON. Adicional la interfaz que facilita la anotación visual y colaborativa y por último que soporta anotaciones más detalladas como regiones específicas, lo que podría aumentar la precisión del modelo. Para lo anterior solo fue necesario acceder a su página oficial, no fue necesario instalar ningún programa.

Figura 28.

Anotacion en VGG16 annotator



Uso de GPU en Reemplazo de CPU

En este proyecto, se utilizó inicialmente una CPU para las primeras pruebas, pero se reconoce que el uso de GPU constituye una mejor práctica cuando se cuenta con acceso a recursos computacionales avanzados, como los disponibles en entornos universitarios o plataformas en la nube. Esto no solo optimiza los tiempos de desarrollo, sino que también permite explorar configuraciones más profundas del modelo sin restricciones de hardware. Acelera el entrenamiento: un modelo que tarda 6 horas en CPU puede tardar solo 40 minutos en GPU.

- Permite usar batch sizes más grandes lo que mejora la estabilidad del aprendizaje.
- Facilita el ajuste fino de modelos preentrenados (transfer learning) más rápido.
- Reduce errores por tiempo prolongado de entrenamiento, como calentamiento del sistema o interrupciones.

Ajuste del umbral de confianza

Inicialmente se determinó usar un umbral de 0.5, sin embargo, en el posterior análisis y evaluación se aumentó el umbral a 0.8 con el objetivo de filtrar únicamente las detecciones más seguras y reducir las agrupaciones de personas en una misma caja.

- Aumento de épocas de entrenamiento

Originalmente, el modelo estaba configurado para entrenar durante 10 épocas. Sin embargo, al analizar la gráfica de evolución de la pérdida (loss) que se obtuvo durante el entrenamiento, se notó que la línea descendente seguía bajando con una tendencia clara, sin estabilizarse completamente.

Esta observación llevó a aumentar el número de épocas a 20 como se muestra en la Figura 29 y Figura 30, con el objetivo de aprovechar el aprendizaje progresivo del modelo y permitirle seguir ajustando sus parámetros para mejorar la detección.

Figura 289.

Entrenamiento 20 épocas

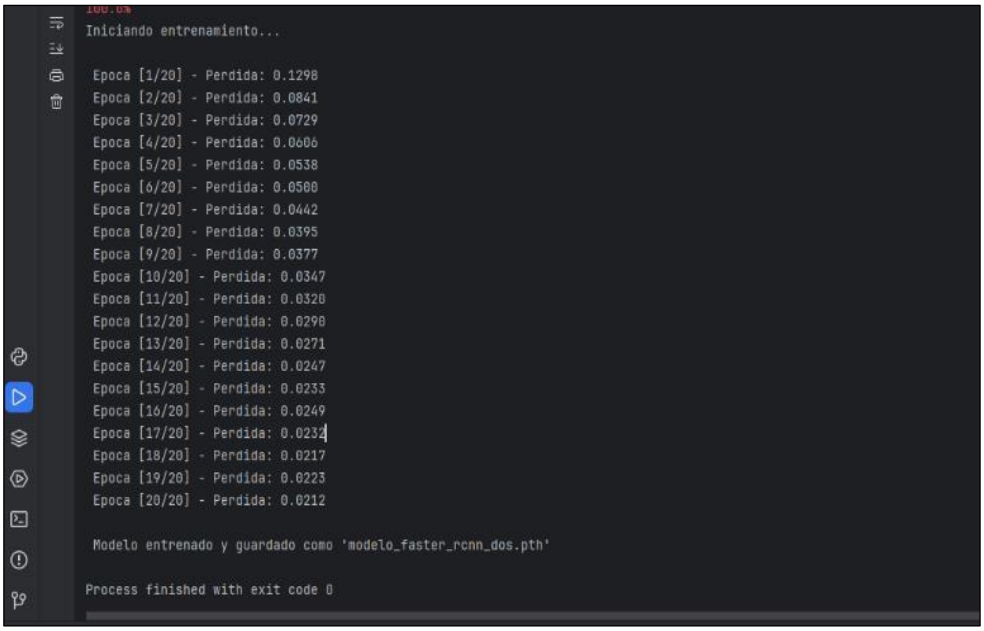
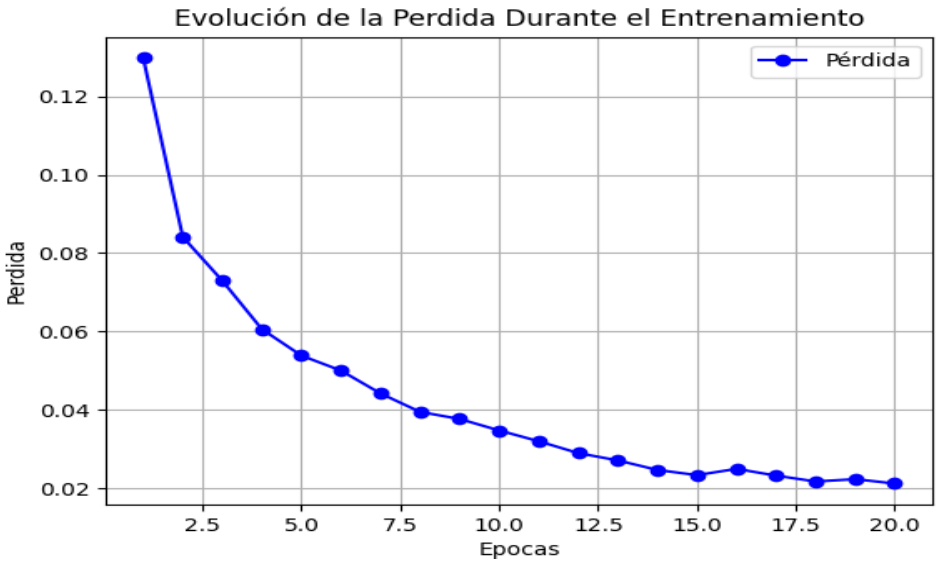


Figura 290.

Evolución perdida 20 épocas

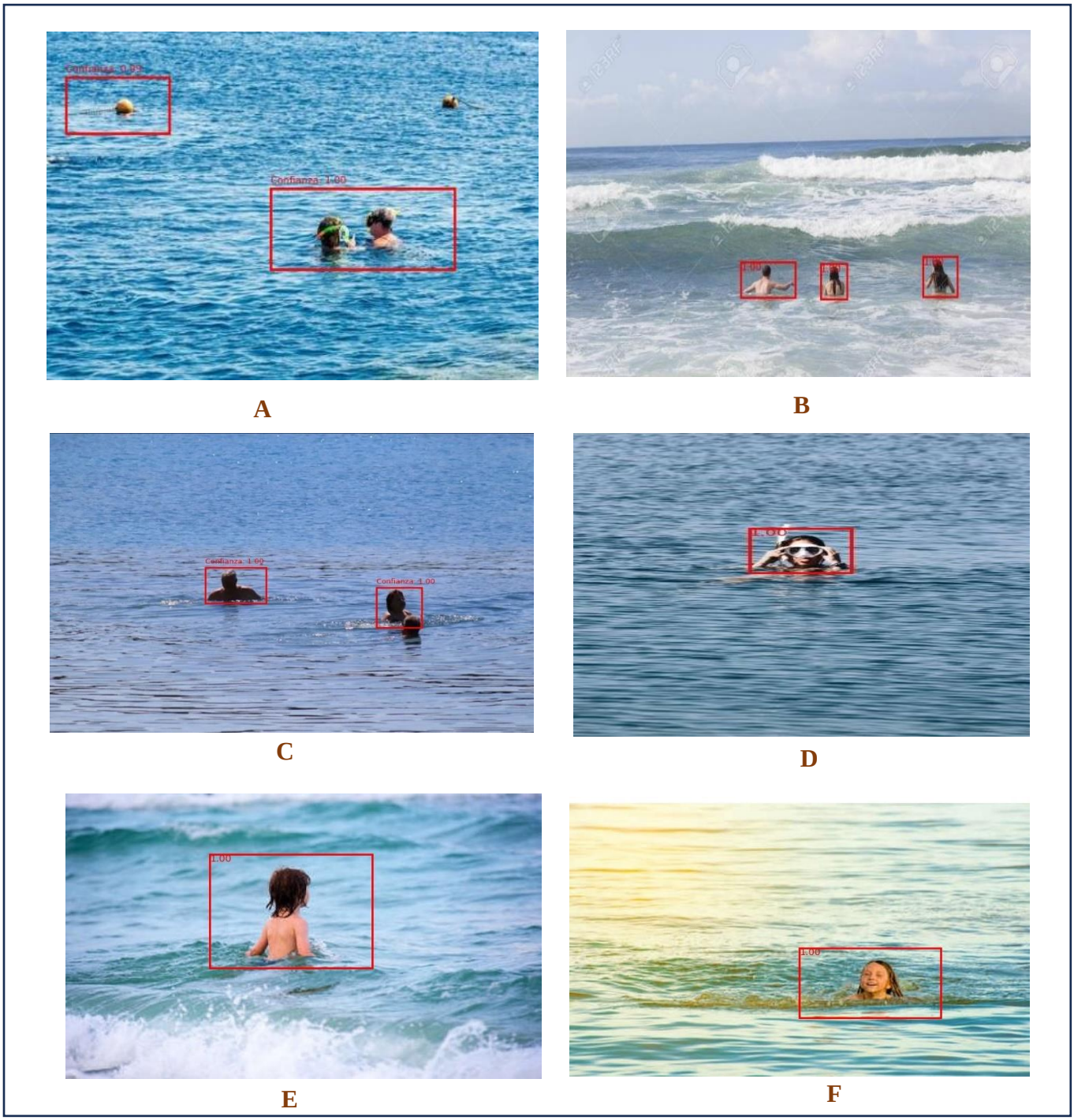


Resultados finales evaluación del modelo

De acuerdo con el falso positivo que se logró evidenciar al probar modelo con diferentes imágenes inicialmente, se realizaron varias modificaciones para tratar de mejorar el rendimiento y eficacia del modelo. En la Figura 31 se logra evidenciar que la mejora de rendimiento no fue de un 100%, específicamente en la imagen “A”, podemos ver que sigue teniendo errores en la detección. Se determino evaluar algunas de las mismas imágenes del entrenamiento anterior como las “C”, “E” y “F” y se visualiza que sigue realizando detecciones correctamente. Así mismo se probó el modelo en imágenes nuevas donde talvez el grado de dificultad de detección en estas no es tan alto sin embargo hubo una detección de estas imágenes 100% exitosa.

Figura 301.

Resultado 2 del segundo entrenamiento



Intersección sobre la Unión (IoU)

De acuerdo con los resultados anteriores se decide medir qué tan precisas son las predicciones del modelo en comparación con las cajas reales. Para lo anterior se seleccionaron 12 imágenes para medir cuánto se superponen las dos cajas delimitadoras (boundin boxes): caja predicha por el modelo y la caja real.

Posteriormente se realizó anotación manual de estas y se creó un script para compararlas contra las predicciones del modelo. Al tener las anotaciones tanto manuales como las del modelo en formato JSON se procedió a realizar el cálculo de precisión y recall junto con la definición de falsos positivos, falsos negativos y verdaderos positivos y verdaderos negativos.

La Tabla 2 muestra el cálculo porcentual de cada tipo de resultado, según la formula clásica de evaluación y detección de objetos, mientras que la Tabla 3 muestra un buen desempeño del modelo entrenado para la detección de personas en medios acuáticos. Se identificaron correctamente 41 personas (verdaderos positivos), con 2 verdaderos negativos, 5 falsos positivos y 15 falsos negativos. La precisión alcanzada fue del 89%, lo que indica que el modelo rara vez detecta incorrectamente. El recall fue del 73%, evidenciando que el modelo detectó la mayoría de las personas presentes, aunque aún tiene margen de mejora. La métrica F1-score, que combina precisión y recall, fue de 0.80, mostrando un equilibrio favorable en el rendimiento del sistema.

Tabla 2.

Cálculo porcentual basado en la fórmula clásica de evaluación de detección de objetos

Tipo de Resultado	Fórmula	Porcentaje
Verdaderos Positivos	TP / (TP + FN + FP) × 100	65.08%
Verdaderos Negativos	TN / (TP + FN + FP) × 100	3.17%
Falsos Positivos	FP / (TP + FN + FP) × 100	7.94%
Falsos Negativos	FN / (TP + FN + FP) × 100	23.81%

Fuente. Elaboración propia

Tabla 3.

Métricas de evaluación

Métrica	Valor
Verdaderos Positivos (TP)	**41**
Verdaderos Positivos (TP)	**2**
Falsos Positivos (FP)	**5**
Falsos Negativos (FN)	**15**
Precisión	**0.89**
Recall	**0.73**
F1-Score	**0.80**

Fuente. Elaboración propia

La siguiente matriz de confusión (ver Tabla 4) presenta la distribución de los resultados obtenidos por el modelo evaluado, con base en los porcentajes observados en la gráfica. Esta herramienta permite visualizar el desempeño del modelo al clasificar correctamente o incorrectamente los casos positivos y negativos. En este caso particular, se evidencian principalmente verdaderos positivos, junto con una proporción menor de falsos negativos y falsos positivos, lo cual ofrece una visión clara sobre los aciertos y errores cometidos por el modelo durante su proceso de predicción.

Tabla 4.

Matriz de confusión

	Predicción: Positivo	**Predicción: Negativo**
Real: Positivo	65.1 (**Verdaderos Positivos**)	23.8 (**Falsos Negativos**)
Real: Negativo	7.9 (**Falsos Positivos**)	3.2 (**Verdaderos Negativos**)

Conclusiones y recomendaciones

El propósito de este estudio consistió en construir un prototipo que integrase tecnología de visión artificial para detectar personas en el agua, y, de acuerdo a los resultados obtenidos, el objetivo se ha cumplido sin ningún tipo de excepción.

En primer lugar, se realizó una investigación comparativa de diferentes modelos de detección de objetos, según su eficiencia y precisión y al analizar sus ventajas y desventajas de seleccionó el módelo Faster R-CNN, ya que mostró un buen equilibrio en exactitud y robustez, adicionalmente que se adaptó bien al tipo de imágenes con las que se trabajaron.

Posteriormente se elaboró e implementó un conjunto de datos con imágenes tomadas en entornos acuáticos, destacando que se tomaron donde la parte del cuerpo estaba parcialmente cubierta por el agua. Este paso fue fundamental para que el modelo pudiera aprender de ejemplos realistas.

Después se entrenó el modelo seleccionado usando el conjunto de datos (Dataset) construido, y se realizaron varios ajustes para que el rendimiento fuera cada vez mejor, gracias a eso, se logró que el modelo respondiera bien no en un 100% pero los resultados del entrenamiento fueron muy aceptables.

Por último, se evaluó el desempeño general del sistema. Los resultados mostraron que el modelo fue capaz. El modelo Faster R-CNN obtuvo una precisión del 89 %, un recall del 73 % y un F1 Score de 0,80, lo que pone en evidencia el potencial del modelo para detectar personas en situaciones simuladas, incluso en condiciones complicadas como oclusiones parciales, escalas de imagen diferentes, etc. Los resultados obtenidos justifican el uso de redes neuronales para la

detección automatizada de personas en entornos de riesgo (playas o piscinas), y posibilitan futuras implementaciones en el mundo real donde la detección de bañistas sea la prioridad.

A partir de los resultados obtenidos y de las limitaciones encontradas durante el transcurso del proyecto se presentan una serie de recomendaciones enfocadas a mejorar el sistema, pero también líneas futuras de trabajo. En primer lugar, se recomienda incrementar el número de épocas de entrenamiento, dado que las curvas de pérdida no se estabilizaban al finalizar las 20 épocas, lo que indicaba que el modelo seguía aprendiendo.

En segundo lugar, incrementar la diversidad del conjunto de imágenes del problema a tratar es un aspecto muy relevante. Aunque el conjunto de datos fue aumentado hasta 1200 imágenes, se siguieron detectando situaciones con errores en casos de iluminación complicada, oclusiones parciales o presencia de objetos distractores. también ayudan a reducir la tasa de falsos positivos.

Como tercera línea de mejora, se deberían aplicar técnicas más sofisticadas de aumentación de los datos que fortalezcan la capacidad de generalización del modelo a diferentes contextos. Por ejemplo, realizar transformaciones que incluyen rotaciones, zooms, cambios de brillo y contraste, desenfoques o recortes parciales podrían llevarse a cabo a través de librerías especializadas como Albumentations o a través de transformaciones de torchvision.

Para concluir, una línea de trabajo futuro clave sería la validación del modelo aplicado a la realidad. Hasta el momento el sistema sólo ha sido validado usando imágenes estáticas, pero hay que probarlo también en secuencias de vídeo, idealmente en vídeos de playas o piscinas, lo que permite saber su rendimiento en tiempo real, como el número de cuadros por segundo (FPS) o la latencia del sistema, que son criterios fundamentales para evaluar su idoneidad para luego aplicarlo en la vigilancia del agua.

Referencias

Academia Journal Monterrey – UACJ. (2023). *Comparación de los algoritmos en aprendizaje automático YOLO v3, SSD y Faster R-CNN* (PDF). Universidad Autónoma de Ciudad Juárez. https://catdi.uacj.mx/bitstream/handle/20.500.11961/26137/2023%20%20AcademiaJournal.pdf

Artiam Innovation. (s.f.). *¿Qué son las redes neuronales y sus funciones?* https://artiaminnovation.com/blog/que-son-las-redes-neuronales-y-sus-funciones/

Chikovani, E. (s.f.). *Object detection with SSD and YOLO*. Baeldung. https://www.baeldung.com/cs/object-detection-ssd-yolo

Robu, R. (2021, 21 de agosto). *Understanding and implementing Faster R-CNN*. Medium. https://medium.com/@RobuRishabh/understanding-and-implementing-faster-r-cnn-248f7b25ff96

DataCamp. (2024, enero 29). *Explicación de la detección de objetos YOLO*. https://www.datacamp.com/es/blog/yolo-object-detection-explained

De Luca, A., Irigoitia, M., Pérez, G., & Pons, C. (2021*). Uso de la técnica de Transfer Learning en Machine Learning para la clasificación de productos en el Banco Alimentario de La Plata* (Tesis de grado, Universidad Nacional de La Plata). https://sedici.unlp.edu.ar/bitstream/handle/10915/130474/Documento_completo.pdf

IBM. (s.f.). *¿Qué es la visión por computadora?* IBM. https://www.ibm.com/mx-es/topics/computer-vision

Medium. (2024, octubre 14). *Understanding and implementing Faster R-CNN*. https://medium.com/@RohitRisabac/understanding-and-implementing-faster-r-cnn-248f7b25ef96

Moreira Ramos, D. L. (2021). *Aplicación de un modelo de reconocimiento de objetos utilizando YOLO (You Only Look Once)* (Tesis de pregrado, Universidad Estatal Península de Santa Elena). Repositorio UPSE. https://repositorio.upse.edu.ec/bitstream/46000/5755/1/UPSE-TTIT-2021-0008.pdf

Programmer Click. (s.f.). *Arquitectura del modelo SSD con VGG16*. https://programmerclick.com/article/30304504320

Programmer Click. (s.f.). *Esquema del proceso de algoritmo SSDJ*. https://programmerclick.com/article/30304504320

Programmer Click. (s.f.). *Explicación del modelo SSD (Single Shot MultiBox Detector)*. https://programmerclick.com/article/30304504320

Ren, S., He, K., Girshick, R., & Sun, J. (2015). *Faster R-CNN: Towards real-time object detection with region proposal networks. arXiv*. https://arxiv.org/pdf/1506.01497v1

ResearchGate. (2022). *Esquema de red neuronal de tres capas.* https://www.researchgate.net/figure/Esquema-de-red-neuronal-de-tres-capas_fig2_366368920

UPSE – Universidad Estatal Península de Santa Elena. (s.f.). *Repositorio institucional UPSE*. https://repositorio.upse.edu.ec

Tecnobits. (2022, abril 27). *¿Qué son las redes neuronales artificiales?* https://tecnobits.com/que-son-las-redes-neuronales-artificiales/

Xataka Android. (2020, febrero 6). *Google Lens invade la traducción con cámara del Traductor de Google.* https://www.xatakandroid.com/aplicaciones-android/google-lens- invade-traduccion-camara-traductor-google