UNIVERSIDAD NACIONAL DE SAN ANTONIO ABAD DEL CUSCO

FACULTAD DE INGENIERÍA ELÉCTRICA, ELECTRÓNICA, INFORMÁTICA Y MECÁNICA

ESCUELA PROFESIONAL DE INGENIERÍA INFORMÁTICA Y DE SISTEMAS

TESIS

IMPLEMENTACIÓN DE UN PROTOTIPO DE SISTEMA CON VISIÓN COMPUTACIONAL PARA DETECTAR MOTOCICLISTAS SIN CASCO DE SEGURIDAD

PRESENTADO POR:

Br. NELSON GONZALES HUISA

PARA OPTAR AL TÍTULO PROFESIONAL DE:

INGENIERO INFORMÁTICO Y DE SISTEMAS

ASESOR:

Dr. RONY VILLAFUERTE SERNA

CO-ASESOR:

Ph.D. CARLOS FERNANDO MONTOYA CUBAS

CUSCO - PERÚ

2025

INFORME DE ORIGINALIDAD

(Aprobado por Resolución Nro.CU-303-2020-UNSAAC)

El que suscribe, **Asesor** del trabajo deinvestigación/tesis titulada: IMPLEMENTACIÓN DE UN PROTOTIPO DE SISTEMA CON VISIÓN COMPUTACIONAL PARA DETECTAR MOTOCICLISTAS SIN CASCO DE SEGURIDAD

presentado por: NELSON GONZALES HUISA con DNI Nro 74638174 presentado por: ..... con DNI Nro.: ..... para optar el título profesional/grado académico de INGENIERO INFORMÁTICO Y DE SISTEMAS

Informo que el trabajo de investigación ha sido sometido a revisión por 03 veces, mediante el Software Antiplagio, conforme al Art. 6° del ***Reglamento para Uso de Sistema Antiplagio de la UNSAAC*** y de la evaluación de originalidad se tiene un porcentaje de 4 **%.**

Evaluación y acciones del reporte de coincidencia para trabajos de investigación conducentes a grado académico o título profesional, tesis

Porcentaje	Evaluación y Acciones	Marque con una (X)
Del 1 al 10%	No se considera plagio.	X
Del 11 al 30 %	Devolver al usuario para las correcciones.	
Mayor a 31%	El responsable de la revisión del documento emite un informe al inmediato jerárquico, quien a su vez eleva el informe a la autoridad académica para que tome las acciones correspondientes. Sin perjuicio de las sanciones administrativas que correspondan de acuerdo a Ley.	

Por tanto, en mi condición de asesor, firmo el presente informe en señal de conformidad y **adjunto** la primera página del reporte del Sistema Antiplagio.

Cusco, 18 de Agosto de 20 25

.....

Firma

Post firma RONY VILLAFUERTE SERNA

Nro. de DNI 23957778

ORCID del Asesor 0000-0003-4607-522X

Se adjunta:

1. Reporte generado por el Sistema Antiplagio.
2. Enlace del Reporte Generado por el Sistema Antiplagio: oid: 27259:484294565

Nelson Gonzales Huisa

motociclistasSinCascoV3

Universidad Nacional San Antonio Abad del Cusco

Detalles del documento

Identificador de la entrega
trn:oid:::27259:484294565

Fecha de entrega
18 ago 2025, 8:38 a.m. GMT-5

Fecha de descarga
18 ago 2025, 11:08 a.m. GMT-5

Nombre de archivo
Tesis (5).pdf

Tamaño de archivo
9.7 MB

103 Páginas

18.811 Palabras

107.677 Caracteres

4% Similitud general

El total combinado de todas las coincidencias, incluidas las fuentes superpuestas, para ca...

Filtrado desde el informe

- Bibliografía
- Texto citado
- Coincidencias menores (menos de 20 palabras)

Exclusiones

- N.º de coincidencias excluidas

Fuentes principales

4%	Fuentes de Internet
0%	Publicaciones
2%	Trabajos entregados (trabajos del estudiante)

Marcas de integridad

N.º de alerta de integridad para revisión

Caracteres reemplazados
88 caracteres sospechosos en N.º de páginas
Las letras son intercambiadas por caracteres similares de otro alfabeto.

Los algoritmos de nuestro sistema analizan un documento en profundidad para buscar inconsistencias que permitirían distinguirlo de una entrega normal. Si advertimos algo extraño, lo marcamos como una alerta para que pueda revisarlo.

Una marca de alerta no es necesariamente un indicador de problemas. Sin embargo, recomendamos que preste atención y la revise.

Dedicatoria

Este trabajo está dedicado a mi familia, especialmente a mis padres, quienes día a día luchan incansablemente para que sus hijos sean los mejores. Han sido los pilares fundamentales para que esto sea posible. A mi hermano mayor, por su apoyo constante en los momentos más difíciles, siempre siendo un ejemplo y una fuente de fortaleza.

Agradecimientos

Agradezco a mi familia, amigos y profesores por su apoyo incondicional durante la realización de esta tesis. Sin su motivación, orientación y paciencia, este logro no habría sido posible.

Resumen

El aumento en el uso de motocicletas representa una problemática creciente para la seguridad vial, debido al incumplimiento de normas de tránsito por parte de los motociclistas. En la actualidad, monitorear y hacer cumplir estas normativas es un desafío para el personal policial, debido a la alta densidad de tránsito en zonas urbanas y la limitada presencia policial en áreas rurales. Ante esta situación, esta investigación propone automatizar esta tarea mediante el desarrollo de un prototipo basado en visión computacional. El desarrollo del prototipo comenzó con la creación de un dataset propio, compuesto por imágenes etiquetadas específicamente para identificar motociclistas que no usan casco de seguridad. Luego, se evaluaron los modelos de detección de objetos Faster R-CNN y YOLO, comparando sus ventajas en términos de precisión y rendimiento en escenarios de tiempo real. Tras un análisis detallado, se seleccionó YOLO como el modelo más adecuado debido a su capacidad para procesar de manera eficiente en tiempo real. Se evaluaron dos variantes de YOLO analizando precisión, recall y F1-score, lo que permitió seleccionar el modelo más adecuado para el prototipo. También se integró el algoritmo DeepSORT, el cual permitió monitorear a motociclistas en movimiento, estimar su velocidad y detectar infracciones por no usar casco y por exceso de velocidad, abordando dos factores críticos asociados a la seguridad vial.

Palabras clave: detección de objetos, YOLO, Faster R-CNN, DeepSORT.

Abstract

The increase in motorcycle usage represents a growing concern for road safety, due to motorcyclists' non-compliance with traffic regulations. Currently, monitoring and enforcing these norms is a challenge for law enforcement personnel, given the high traffic density in urban areas and the limited police presence in rural regions. In response to this situation, this research proposes automating the task through the development of a computer vision-based prototype. The development began with the creation of a custom dataset consisting of images specifically labeled to identify motorcyclists who do not wear safety helmets. Subsequently, the object detection models Faster R-CNN and YOLO were evaluated, comparing their advantages in terms of accuracy and performance in real-time scenarios. After a detailed analysis, YOLO was selected as the most suitable model due to its efficient real-time processing capabilities. Two variants of YOLO were assessed by analyzing precision, recall, and F1-score, which facilitated the selection of the most appropriate model for the prototype. The DeepSORT algorithm was also integrated, enabling the system to monitor moving motorcyclists, estimate their speed, and detect infractions for not wearing helmets and speeding—addressing two critical factors associated with road safety.

Keywords: object detection, YOLO, Faster R-CNN, DeepSORT.

Introducción

En la actualidad, el uso de motocicletas va en aumento, lo cual está generando problemas de seguridad vial, especialmente en áreas urbanas y rurales, donde las motocicletas son un medio de transporte muy común. Este incremento en el uso de motocicletas también conlleva un aumento en los accidentes debido a la falta de uso del casco de seguridad. A pesar de la existencia de normativas que exigen su uso, el control policial no es suficiente para garantizar su cumplimiento. Muchos motociclistas circulan a velocidades no permitidas en zonas urbanas, lo que incrementa aún más el riesgo de accidentes, por más que lleven casco. Este proyecto propone automatizar la detección de motociclistas sin casco de seguridad y con exceso de velocidad, con el objetivo de mejorar la seguridad vial y proteger la integridad de los ciudadanos.

El segundo capítulo de esta tesis presenta el marco teórico que respalda el desarrollo del sistema propuesto. Se revisan los antecedentes nacionales e internacionales, seguidos de las bases teóricas sobre seguridad vial, visión computacional y los modelos de detección de objetos. Se abordan los algoritmos de machine learning y deep learning, con especial énfasis en el modelo YOLO (You Only Look Once) para detección en tiempo real y Deep Sort para el seguimiento de objetos, proporcionando los fundamentos necesarios para entender la implementación del sistema.

El tercer capítulo describe la metodología de implementación del prototipo, comenzando con la construcción del dataset a través de la recolección y etiquetado de imágenes de motociclis-

tas con y sin casco. Luego, se realiza el preprocesamiento de las imágenes, normalizándolas y ajustando su tamaño para optimizar el entrenamiento. Se evalúa diferentes modelos de detección de objetos, discutiendo las ventajas de Faster R-CNN y YOLO. Posteriormente, se detalla el proceso de entrenamiento del modelo seleccionado. Una vez implementado Deep-Sort para el seguimiento de objetos, se calcula la velocidad de las motocicletas. Finalmente, se construye el prototipo, integrando todas las partes para su funcionamiento en tiempo real. Este capítulo proporciona un paso a paso del proceso de desarrollo e implementación del prototipo.

En el capítulo final, se discute los resultados obtenidos en relación con los objetivos planteados al inicio del proyecto. Se analiza los hallazgos en comparación con los antecedentes para entender mejor el impacto de las soluciones propuestas. Se evalúa tanto los logros alcanzados como las limitaciones que surgieron durante el desarrollo del trabajo. A partir de esto, se ofrece una conclusión sobre los objetivos cumplidos y la aportación del proyecto. Finalmente, se da recomendaciones para futuras investigaciones.

Listado de Abreviaturas

- **YOLO:** You Only Look Once
- **Faster R-CNN:** Faster Region Convolutional Neural Network
- **DeepSORT:** Deep Learning-based SORT (Simple Online and Realtime Tracking)
- **FPS:**Frames Per Second (Cuadros Por Segundo)
- **mAP:** Mean Average Precision
- **IoU:** Intersection over Union (Intersección sobre Unión)
- **TP:**True Positive (Verdadero Positivo)
- **FP:** False Positive (Falso Positivo)
- **FN:** False Negative (Falso Negativo)
- **Recall:** Métrica que mide la recuperación de las detecciones correctas.
- **Precision:** Precisión en las detecciones correctas.
- **F1-Score:** Medida combinada de la precisión y el recall.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Aspectos Generales

1.1. Planteamiento del Problema

1.1.1. Descripción del problema

En la actualidad, las motocicletas son uno de los medios de transporte más populares a nivel mundial debido a su eficiencia en el consumo de combustible y su capacidad para agilizar los desplazamientos, incluso en rutas congestionadas como el trayecto al trabajo. Sin embargo, existe la problemática de la seguridad vial a nivel mundial. La Organización Mundial de la Salud publicó en el año 2023 un informe titulado "Global status report on road safety 2023", donde mencionan que casi el 21 % de todas las muertes en accidentes de tránsito reportadas en la encuesta se incluyen vehículos motorizados de dos y tres ruedas, como motocicletas (OMS, 2023). Además, el 33 % de los motociclistas involucrados en choques fatales estaban excediendo los límites de velocidad, una proporción mayor en comparación con conductores de otros tipos de vehículos (NHTSA, 2023). Por eso es importante mencionar que los motociclistas que no utilizan cascos de seguridad ni respetan los límites de velocidad incrementan la probabilidad de sufrir lesiones graves.

En caso de un accidente, la cabeza es una de las partes del cuerpo más expuestas y

delicadas, y un daño grave en esta área puede resultar en lesiones graves e incluso la muerte instantánea del conductor, considerando que el cerebro es un órgano vital. Por ende, el casco actúa como una envoltura protectora que cubre toda la cabeza, proporcionando una barrera frente a impactos. Los cascos de calidad reducen el riesgo de muerte en más de seis veces y reducen el riesgo de lesión cerebral hasta en un 74 % (OMS, 2023).

En el caso de Perú, la seguridad vial ya es un serio problema frente al aumento del uso de motocicletas. Según INEI, La tasa de crecimiento promedio de las motocicletas fue del 3.2 %, mientras que la de los autos fue del 2.9 % (Asociación Movemos, 2023). Este incremento representa un desafío crítico para las fuerzas policiales, quienes son encargadas de la fiscalización vial, ya que las estadísticas indican que, durante el primer semestre del 2023, en Perú se registraron 17,153 vehículos menores involucrados en siniestros de tránsito, siendo las motocicletas los vehículos con mayor incidencia (61.8 %) (Observatorio Nacional de Seguridad Vial, 2023). Por este motivo, existe la necesidad de implementar tecnologías modernas como la visión computacional para los fiscalizadores de tránsito que operan en diferentes partes del Perú, como la ciudad de Cusco. Esto permitiría identificar a los motociclistas que infringen las normas de seguridad vial, como el no uso del casco de seguridad. Esta negligencia por parte de los motociclistas es preocupante, ya que aumenta el riesgo de sufrir lesiones graves, peor aún si exceden los límites de velocidad.

1.1.2. Identificación del problema

En la ciudad de Cusco, al igual que en otras ciudades del Perú, existe la necesidad de tecnologías modernas, como la visión computacional, que ayuden a los fiscalizadores de tránsito a optimizar el control de ciertas normas incumplidas por los motociclistas. Según las estadísticas, los motociclistas son vulnerables y la causa principal de su muerte es la gravedad de las lesiones en la cabeza. En Perú ya existen normativas que regulan esta

problemática, pero no hay un control óptimo de su cumplimiento. Estas normativas son establecidas por organizaciones como la SUTRAN, donde en el artículo 105 se establece la normativa que exige el uso de cascos protectores autorizados por parte de conductores y acompañantes de motocicletas (SUTRAN, 2014). Asimismo, el Ministerio de Transportes y Comunicaciones (MTC) establece que deben respetar los límites máximos de velocidad establecidos para vehículos automotores: 50 km/h en avenidas y 30 km/h en calles y jirones (El Peruano, 2024). Es necesario abordar el cumplimiento de estas normas de seguridad vial en los motociclistas para salvaguardar su integridad.

1.2. Formulación del Problema

1.2.1. Problema General

¿Cómo implementar un prototipo de sistema para identificar a motociclistas que infringen las normas de tránsito al circular sin casco de seguridad?

1.2.2. Problemas Específicos

- ¿Existe un dataset de imágenes de motociclistas con y sin casco de seguridad, según a la realidad de tránsito en el Perú?
- ¿Cuál es el mejor modelo de visión computacional para el monitoreo eficiente en tiempo real de motociclistas?

1.3. Objetivos

1.3.1. Objetivo General

Implementar un prototipo de sistema para identificar a motociclistas que infringen las normas de tránsito al circular sin casco de seguridad.

1.3.2. Objetivos Específicos

- Construir un dataset de imágenes de motociclistas con y sin casco de seguridad, según a la realidad de tránsito en el Perú.
- Evaluar modelos de visión computacional para el monitoreo eficiente en tiempo real de motociclistas.

1.4. Justificación

1.4.1. Conveniencia

En la actualidad, las motocicletas se han convertido en un medio de transporte popular para las personas debido a su facilidad para desplazarse de un lugar a otro. Pero, esta creciente popularidad ha llevado a que muchos motociclistas circulen sin utilizar el casco de seguridad, lo que incrementa el riesgo de lesiones graves en caso de accidentes. Las autoridades encargadas de la fiscalización no cuentan con los recursos suficientes para controlar de manera efectiva esta infracción, lo que agrava la situación. Esto se refleja en las estadísticas: según el Observatorio Nacional de Seguridad Vial, en 2023 se registraron 455 siniestros relacionados con motocicletas en Perú. De estos, 405 terminaron en muertes, evidenciando que la gran mayoría de estos accidentes tuvieron consecuencias fatales (Panamericana, 2024).

Ante la deficiencia en la fiscalización y el control de la seguridad vial en motociclistas, actualmente no existe ninguna herramienta automatizada que permita identificar de manera eficiente a los motociclistas con y sin casco de seguridad. Por esta razón, la presente investigación tiene como objetivo automatizar este proceso utilizando tecnologías de inteligencia artificial.

En otros países, ya se han implementado soluciones automatizadas para mejorar la fiscalización vial. Por ejemplo, Estados Unidos utiliza cámaras de velocidad automatizadas para controlar el exceso de velocidad, mostrando resultados significativos en la reducción de accidentes.

Tabla 1.1
Comparación de accidentes de tránsito con y sin cámaras de velocidad

Sin cámaras de velocidad	**Con cámaras de velocidad**
El exceso de velocidad causó el 29 % de las muertes por accidentes de tránsito en EE. UU. (12,330 en 2021).	Reducción del 19 % en accidentes totales y del 21 % en accidentes graves, especialmente en zonas escolares y de construcción.

Nota. Adaptado de Forbes Advisor y NHTSA (2021).

Esta evidencia respalda el desarrollo de herramientas similares basadas en inteligencia artificial para abordar las infracciones cometidas por motociclistas, como el no uso de casco y el exceso de velocidad.

1.4.2. Relevancia

Esta investigación busca mejorar la seguridad vial en motociclistas, enfocándose en la identificación de aquellos que circulan sin casco. La implementación de un sistema automatizado de detección mediante visión computacional tiene el potencial de reducir el número de accidentes, ya que permite identificar las condiciones de los motociclistas en las vías. Este sistema puede facilitar la implementación de políticas de seguridad vial más efectivas para

los fiscalizadores y de esta forma también promover un entorno más seguro en las vías.

1.4.3. Implicancias Prácticas

Las implicancias prácticas de esta investigación podría contribuir a la modernización de la seguridad vial al permitir a los fiscalizadores optimizar recursos mediante la automatización de la detección de infracciones de motociclistas, reduciendo la necesidad de supervisión de una persona. Su implementación en un futuro próximo fomentaría el uso del casco de seguridad, disminuyendo accidentes y promoviendo una cultura de prevención y responsabilidad vial. Con el tiempo, este sistema podrá integrarse en plataformas más amplias de monitoreo vial, mejorando la gestión del tráfico y garantizando mayor seguridad en las vías.

1.4.4. Valor Teórico

Este proyecto de investigación representa una contribución al avance tecnológico en el ámbito de la seguridad vial, mediante la implementación innovadora de técnicas de visión computacional y seguimiento de objetos actualmente disponibles. Su enfoque permite maximizar el aprovechamiento de las cámaras de vigilancia utilizadas por los fiscalizadores de tránsito, fortaleciendo la capacidad para identificar y monitorear a los motociclistas infractores de manera eficiente y automatizada.

1.4.5. Utilidad Metodológica

La metodología desarrollada en este proyecto puede ser reutilizada y adaptada a otros contextos similares. Las técnicas de visión computacional y seguimiento de objetos que se aplican en este proyecto de investigación no solo sirven para la detección de motociclistas sin casco, sino que también podrían ser usadas en otros sistemas de monitoreo, como la gestión del tráfico vehicular, el control de accesos o incluso en la vigilancia de espacios

públicos. Esto hace que la propuesta sea flexible y útil para diferentes propósitos, ya que puede ajustarse fácilmente con nuevos datos o configuraciones específicas, permitiendo que otros investigadores o desarrolladores la aprovechen en sus propios proyectos.

1.5. Delimitación de estudio

1.5.1. Delimitación Espacial

La investigación se enmarca en el ámbito de la seguridad vial y está diseñada para su uso potencial por entidades fiscalizadoras de tránsito. La recopilación de datos se realizó en la ciudad de Cusco, accediendo a las cámaras de videovigilancia de seguridad ciudadana en áreas urbanas con alta densidad vehicular, y en el distrito de Yanaoca, provincia de Canas, donde se obtuvieron imágenes capturadas con una cámara en un entorno rural. Los videos obtenidos de ambas ubicaciones se utilizaron para las pruebas del sistema, garantizando su aplicabilidad tanto en contextos urbanos como rurales, fortaleciendo su relevancia para mejorar la fiscalización en diversos escenarios viales de la región.

1.5.2. Delimitación Temporal

Esta proyecto de investigación fue desarrollada durante el año 2024.

1.6. Método

1.6.1. Alcance

Este proyecto de investigación propone el desarrollo de un prototipo de sistema que detecta motociclistas con y sin casco de seguridad. El prototipo también tiene la capacidad de medir la velocidad a la que circulan. Si se implementara en el ámbito de la seguridad vial,

el sistema podría contribuir a reducir los accidentes de tránsito y automatizar la supervisión de las normas viales en motociclistas.

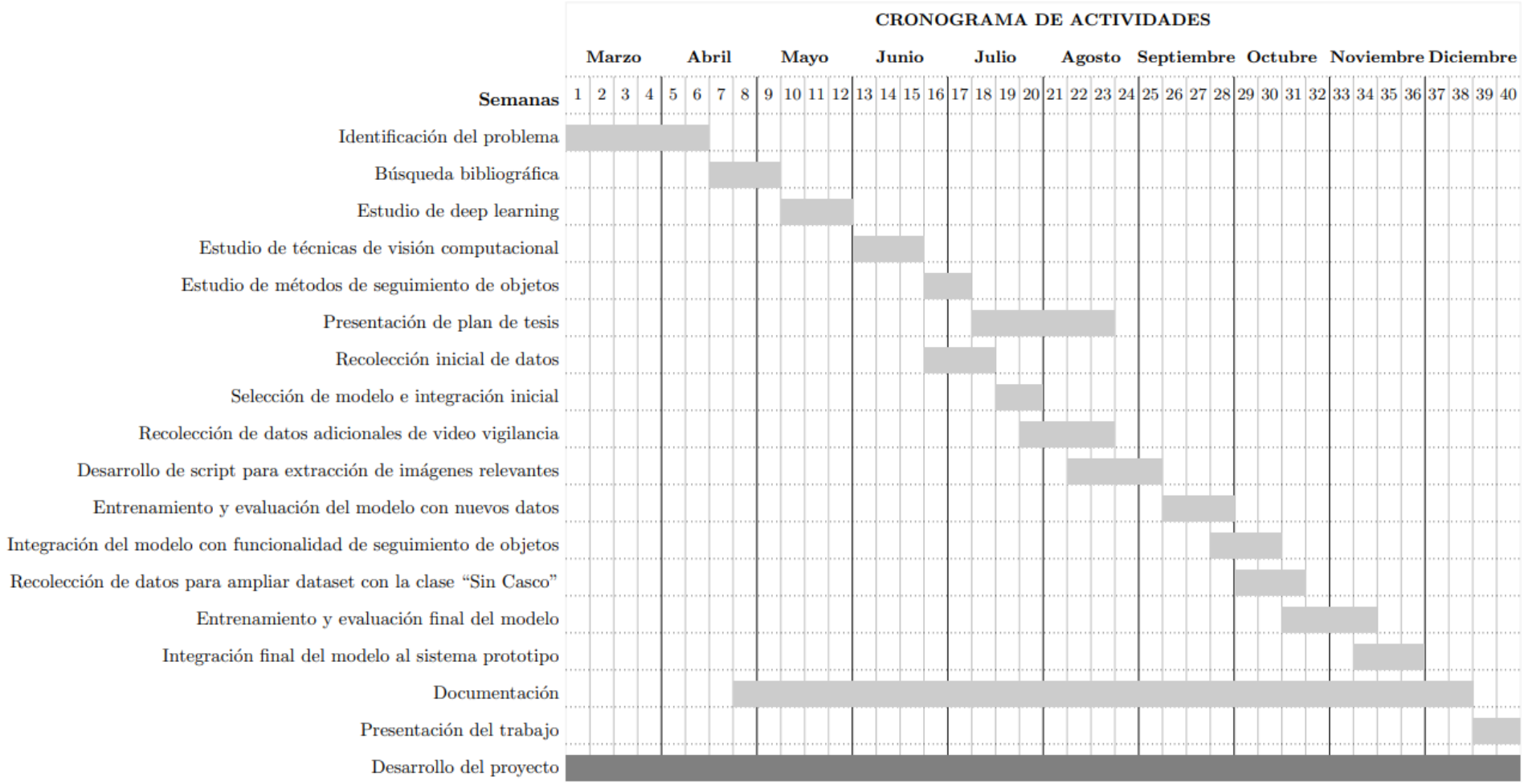
1.6.2. Diseño

Esta investigación, debido al uso de algoritmos de visión computacional para la construcción del prototipo, se enmarca en una investigación aplicada, en la que se emplean conocimientos previos para resolver un problema relacionado con los motociclistas. Según Ander-Egg (2011), la investigación aplicada mantiene una estrecha relación con la investigación básica, pues depende de sus descubrimientos y avances, enfocándose en la resolución de problemas mediante la aplicación y utilización del conocimiento.

1.6.3. Para el desarrollo de la parte informática

Se adoptó la metodología ágil SCRUM por su enfoque flexible e iterativo, ideal para este proyecto de investigación. Según Scrum.org (2022), SCRUM se define como "una forma de hacer el trabajo en equipo en pequeños pedazos a la vez, con experimentación continua y circuitos de retroalimentación en el camino para aprender y mejorar a medida que avanza". El desarrollo se organizó en sprints cortos, lo que permitió ajustar y mejorar continuamente cada fase del proyecto. En una primera etapa, se construyó un prototipo inicial del sistema, al cual se le añadieron funcionalidades mejoradas en cada iteración.

1.6.4. Cronograma de Actividades



Capítulo 2

Marco Teórico

2.1. Antecedentes

2.1.1. Antecedentes Internacionales

(Wei et al., 2021),*Real-time automatic helmet detection of motorcyclists in urban traffic using improved YOLOv5 detector*, Hefei University of Technology, China.

Conclusiones:

- En este artículo se presenta un método de detección de cascos de motociclistas de extremo a extremo en tiempo real basado en el algoritmo YOLOv5. El método desarrollado automatiza la detección de motocicletas en vídeos o imágenes y evalúa si los conductores llevan casco. El enfoque se divide en dos etapas: detección de motocicletas y detección de cascos, cada una entrenada con modelos específicos, YOLOv5-MD y YOLOv5-HD, respectivamente. Estos modelos aseguran alta precisión manteniendo un tamaño compacto: 20.6 MB para el modelo de primera etapa y 14.8 MB para el de segunda etapa. Para validar la efectividad del método, se propone el conjunto de datos HFUT-MH, obtenido de diversas escenas de tráfico en China con condiciones climáti-

cas variadas, oclusión y iluminación variable. En este conjunto de datos, la detección de cascos alcanzó un 97.7 % de precisión, con una puntuación F1 del 92.7 y una velocidad de detección de 63 FPS, asegurando tanto precisión como velocidad en tiempo real. Además, el método puede determinar la sobrecarga de motocicletas calculando el número de cascos y la ausencia de estos.

Comentario: Este proyecto muestra metodologías para la detección de motociclistas sin casco, lo cual sirve para emplear esas metodologías y adaptarlas al uso de nuevas versiones de YOLO. Al contar con un método bien fundamentado y validado experimentalmente, podré diseñar un marco teórico robusto para mi investigación. Además, la utilización de las versiones recientes de YOLO en lugar de YOLOv5, puede ofrecer potenciales mejoras en términos de precisión y eficiencia en la detección de objetos.

(Duong et al., 2023),*Robust Automatic Motorcycle Helmet Violation Detection for an Intelligent Transportation System*, Universidad Sungkyunkwan, Corea del Sur.

Conclusiones:

- La detección automatizada del uso del casco de motocicleta basada en videovigilancia tiene el potencial de mejorar la eficacia de las iniciativas de educación y aplicación de la ley, lo que podría resultar en carreteras más seguras. No obstante, los métodos actuales de detección enfrentan desafíos, como la dificultad para identificar motocicletas individuales y distinguir entre conductores que usan casco y pasajeros. Este artículo presenta un marco para detectar e identificar motocicletas de manera independiente, mientras se monitorea específicamente el uso del casco por parte del conductor. El enfoque de clasificación del uso del casco demuestra una mayor eficiencia en comparación con investigaciones previas. Destacando la alta precisión del aprendizaje profundo, la

técnica logró una puntuación de 0.7754 en la tabla de clasificación del desafío AI City 2023 Challenge Track 5. Para futuras mejoras, se planea integrar un rastreador en el marco principal para obtener información consolidada de múltiples marcos y mejorar la clasificación individual de cada ciclista.

Comentario: Este proyecto determina la automatización en la detección de motociclistas sin casco, lo cual no solo mejora la educación y la aplicación de la ley en materia de seguridad vial, sino que también proporciona una base para justificar y respaldar nuestro propio proyecto. La metodología empleada en este estudio ofrece una guía valiosa para implementar un enfoque que garantice resultados más precisos y efectivos en una investigación.

(Aman et al., 2022), *Detecting, Tracking and Counting Motorcycle Rider Traffic Violations on Unconstrained Roads.*, International Institute of Information Technology, Hyderabad, India.

Conclusiones:

- Este artículo propuso una arquitectura innovadora diseñada para detectar, rastrear y contar infracciones relacionadas con el uso del casco y la conducción triple en calles concurridas de Asia, utilizando una cámara montada en un vehículo. El enfoque desarrollado aborda simultáneamente ambas violaciones. El marco propuesto mejora los enfoques existentes al manejar eficazmente oclusiones de conductores, minimizando falsos negativos y falsos positivos de otras motocicletas en entornos con alta densidad de tráfico. Se demostró la eficacia del uso de aprendizaje curricular para la detección de infracciones de motocicletas y cuadros delimitadores de trapecio en lugar de los tradicionales cuadros rectangulares. Este trabajo sienta las bases para la implementación de tales sistemas en la mejora de la seguridad vial, permitiendo la vigilancia efectiva

en diversos puntos de la ciudad sin el costo significativo asociado a una red de cámaras estáticas. Como futuro desarrollo, se plantea explorar la implementación del sistema en una configuración de GPU distribuida para la vigilancia integral de toda la ciudad.

Comentario: Este proyecto presenta una arquitectura innovadora en la detección de motociclistas sin casco, utilizando un trapecio como cuadro delimitador en lugar del tradicional cuadro rectangular. Esta metodología resulta útil para una investigación, especialmente en el marco teórico, al abordar la importancia de los cuadros delimitadores de objetos utilizados en proyectos de visión computacional.

2.1.2. Antecedentes Nacionales

(Gonzales del Valle Romero and Neyra Espinoza, 2022),*Implementación de una red neuronal artificial convolucional para la detección y conteo de vehículos en una sección del estacionamiento de la Universidad Ricardo Palma*, Universidad Ricardo Palma, Perú.

Conclusiones:

- Se implementaron las redes neuronales artificiales convolucionales YOLO v5 y Faster R-CNN, aplicando transfer learning a ambas mediante los métodos de congelamiento de capas y fine tuning respectivamente. Se utilizó un dataset compuesto por 460 imágenes, evaluando los resultados en términos de precisión, FPS, porcentaje de error promedio y falsos positivos bajo umbrales de detección del 35 % y 55 %. Basado en estos resultados (detallados en la Tabla 6), se seleccionó YOLO v5 como la red principal para la detección de automóviles, demostrando un promedio de error de detección del 5.33 %.

- Se aplicaron métodos de limitación para el conteo de vehículos, utilizando líneas de interés y regiones de interés. Estos sistemas fueron evaluados en tiempo real, con-

siderando la morfología específica de los estacionamientos y la tasa de error en las detecciones de espacios (como se observa en las Figuras 24 y 27). Se optó por el método de selección por región de interés, delimitando secciones específicas para el conteo mediante procesamiento de imágenes.

- Se desarrolló una interfaz gráfica de usuario para ambas redes neuronales artificiales convolucionales. Esta interfaz permite inicializar la cámara y la red neuronal, mostrando los resultados del conteo de espacios de estacionamiento en tiempo real. Además, incluye funcionalidades para iniciar y detener el sistema de detección y conteo de vehículos.

Comentario: Este proyecto ofrece una detallada explicación sobre la implementación de un sistema de detección y conteo de vehículos, utilizando métodos como YOLO v5 y Faster RCNN, los cuales forman parte de la visión computacional. este enfoque es de gran utilidad, ya que la metodología empleada para la identificación y conteo de vehículos también contribuye al proyecto de investigación.

(Perez Silva, 2022),*Reconocimiento de placas vehiculares mediante visión computacional para mejorar el acceso a un parqueadero*, Universidad Señor de Sipan, Perú.

Conclusiones:

- Para el diseño del sistema de reconocimiento de placas vehiculares, se definieron como requisitos funcionales el proceso de captura de imágenes de los vehículos, su preprocesamiento, la localización de la placa, así como la segmentación y el reconocimiento de los caracteres que la componen, con el objetivo de verificar el acceso autorizado al parqueadero a través de una consulta a la base de datos de vehículos. Los requisitos no funcionales se agruparon en torno a atributos como rendimiento, usabilidad y disponibilidad.

- En base a los requisitos identificados, se diseñó una arquitectura de software compuesta por dos componentes principales. El primero incluye la capa de interfaz de usuario, la capa lógica de la aplicación que abarca las funcionalidades de preprocesamiento, segmentación y reconocimiento de placas, y la capa de acceso a datos que facilita la comunicación con el servidor de base de datos. El segundo componente es el servidor de base de datos, encargado de almacenar los registros de vehículos autorizados.

- El preprocesamiento de la imagen vehicular se realizó mediante la conversión de la imagen de formato RGB a HSV, la maximización del contraste para resaltar los tonos y contornos, el suavizado con desenfoque gaussiano y, finalmente, la conversión a escala de grises para optimizar los cálculos posteriores. Durante la segmentación de la imagen, se implementó una funcionalidad para identificar la placa y segmentar cada uno de los caracteres que la componen. En la etapa de reconocimiento, los caracteres segmentados fueron comparados con una matriz de caracteres, utilizando el algoritmo KNN para su clasificación.

- En las pruebas realizadas en un entorno supervisado, el sistema alcanzó una precisión del 89.9 % para el reconocimiento de caracteres numéricos y del 80.5 % para los caracteres alfabéticos, lo que indica que los algoritmos empleados en el sistema de reconocimiento son efectivos. Además, el tiempo medio de procesamiento para reconocer un vehículo autorizado, basado en su placa, fue de 0.034 segundos, lo que permite un rendimiento de hasta 29 imágenes procesadas por segundo. El sistema fue desarrollado en Python, utilizando la librería OpenCV para visión por computadora, lo que garantiza su compatibilidad con los sistemas operativos Windows, Linux y Mac OS.

Comentario: Este proyecto aborda la problemática de la falta de uso de un software con visión computacional para el control de parqueaderos de vehículos. El objetivo principal fue implementar un sistema de reconocimiento de placas utilizando técnicas de visión compu-

tacional. Donde, se emplearon algoritmos de tratamiento de imágenes, como la obtención de contornos, umbralización y segmentación, como parte del preprocesamiento de datos antes del entrenamiento del modelo. Por lo que es necesario considerar estos pasos para entrenar un modelo de visión computacional en una investigación.

(Wong Leon, 2022), *Una revisión sistemática de literatura sobre implementación de sistemas de control de trafico*, Universidad Católica Sedes Sapientiae, Perú.

Conclusiones:

- A través de esta revisión de la literatura, se concluye que el uso de modelos o algoritmos de inteligencia artificial (IA) representa la opción más adecuada para abordar los problemas de congestión vehicular y el control de semáforos. Esto se debe a su capacidad para optimizar el control de señales de tránsito, ya que permiten predecir la densidad del tráfico de manera eficiente.

- En cuanto al procesamiento de imágenes, se determina que el modelo YOLO es el más adecuado para el reconocimiento de vehículos, basado en los resultados de los estudios revisados. Esto se debe a que YOLO puede detectar múltiples objetos simultáneamente, realiza decisiones rápidamente basadas en los resultados obtenidos y tiene un bajo consumo de recursos computacionales, lo que lo hace ideal para su implementación en tiempo real en sistemas de control de semáforos.

- El análisis de los artículos revisados también sugiere que, para entrenar el modelo de reconocimiento de vehículos, es importante priorizar secuencias de imágenes que muestren claramente los vehículos y sus matrículas al menos una vez por cuadro. Esto mejora la precisión del modelo al diferenciar vehículos de otros objetos en movimiento.

- Finalmente, se concluye que el algoritmo más preciso para construir un sistema de

control de semáforos es el filtro bayesiano aplicado a las redes neuronales convolucionales (CNN). Este algoritmo permite modelar la incertidumbre en los datos de tráfico, facilitando el aprendizaje y la predicción de la densidad vehicular, lo que contribuye a mejorar la precisión de las decisiones sobre el cambio de luces en los semáforos.

Comentario: Este proyecto presenta una revisión literaria exaustiva de los métodos y algoritmos eficientes para el desarrollo de sistemas de control de tráfico, donde menciona el uso del filtro bayesiano, redes neuronales convolucionales y el modelo YOLO. La implementación de estos sistemas es clave para mejorar la fluidez del tránsito y reducir la congestión vehicular, lo que hace viable emplear el modelo YOLO en mi investigación.

2.2. Bases Teóricas

2.2.1. La Seguridad Vial

La seguridad vial implica la prevención y mitigación de accidentes de transito, priorizando la protección de la vida y la salud de las personas involucradas. Incluye tanto el uso de tecnología en vehículos terrestres (como automóviles, camiones, motocicletas y bicicletas) como la aplicación de normativas de tráfico y la responsabilidad de los usuarios de la vía. La falta de organización estatal y la ausencia de regulación sobre el comportamiento humano, tanto individual como colectivo, obstaculizan el logro de resultados óptimos en materia de seguridad vial (Ahorra Seguros, s.f.). (Ahorra Seguros, 2020).

Importancia de la seguridad vial. La importancia de la seguridad vial radica en su impacto en la sociedad en términos de vidas humanas, salud, calidad de vida y costos económicos. Respecto a este último, organismos internacionales estiman que cada año los accidentes de tránsito demandan entre el 1 5 % del PIB de cada país. Los accidentes de

tráfico son una de las principales causas de muerte y lesiones en todo el mundo, y afectan desproporcionadamente a los grupos más vulnerables, como peatones, ciclistas y motociclistas. La seguridad vial es fundamental para garantizar la movilidad segura y sostenible de las personas, así como para promover el desarrollo socio-económico y la equidad en el acceso al transporte (Goldman, 2020).

Seguridad vial para motociclistas. El aumento de la popularidad de las motocicletas entre segmentos de la población previamente excluidos es evidente. Sin embargo, este crecimiento acelerado en la presencia de motocicletas en las vías ha generado una serie de repercusiones negativas, incluyendo un incremento en los accidentes, lesiones y fatalidades. De hecho, las estadísticas revelan que los motociclistas enfrentan un riesgo de mortalidad hasta 26 veces mayor en comparación con los ocupantes de automóviles en caso de accidentes viales. Esta vulnerabilidad intrínseca a este medio de transporte ha dado lugar a un aumento notable tanto en la cantidad como en la gravedad de las lesiones sufridas en estos sucesos(PAHO, 2020). Por tanto, resulta necesario abordar de manera prioritaria el tema de la seguridad vial para motociclistas.

Obligaciones y recomendaciones:

- Por el hecho de que las motocicletas son vehículos inestables y vulnerables es necesario que cumplan ciertas normas establecidas por entidades encargadas de fiscalizar la seguridad vial, en el caso de Perú las normas exigen que, el conductor y el acompañante tienen que usar, obligatoriamente, un casco protector autorizado, para evitar sufrir lesiones graves o incluso la muerte. El incumplimiento de esta normativa es una infracción grave, que se sanciona con una multa de S/ 412 y la acumulación de 40 puntos en el récord del conductor (El Peruano, 2024).

- Al elegir un casco, asegúrate de que cumpla con los estándares de seguridad establecidos

y que se ajuste correctamente a tu cabeza. Un casco mal ajustado puede ser menos efectivo en caso de impacto. Además, es recomendable reemplazar el casco después de cualquier impacto significativo a esto es recomendable acompañar un guante, ya que su capacidad para proteger puede verse comprometida(Sura, 2018).

2.2.2. Visión computacional

La visión computacional les permite a las maquinas ver y entender. Para poder hacer funciones similares al ojo humano es necesario que estas máquinas sean entrenadas con cámaras, datos y algoritmos (IBM, 2019).

Análisis de Imagen mediante visión computacional. Para el análisis de imágenes es necesario emplear sensores, IA, aprendizaje automático y Utilizar algoritmos y entrenarlos con grandes conjuntos de datos para luego reconocer patrones y comprender el contenido de las imágenes de esa forma imitar el ojo humano (Microsoft, 2021):

- Un sensor captura una imagen. A menudo, este sensor es simplemente una cámara, pero puede ser una cámara de vídeo, un dispositivo médico de diagnóstico por imagen o cualquier otro tipo de dispositivo que capture una imagen para analizarla.

- A continuación, se envía la imagen a un dispositivo de interpretación. El dispositivo de interpretación usa el reconocimiento de patrones para descomponer la imagen, comparar los patrones que contiene con su biblioteca de patrones conocidos y determinar si hay contenido en la imagen que coincida. El patrón puede ser algo general, como la apariencia de un determinado tipo de objeto, o podría basarse en identificadores únicos, como los rasgos faciales.

- Un usuario solicita información específica sobre una imagen y el dispositivo de interpretación proporciona la información solicitada en función del análisis que ha realizado de la imagen.

Procesamiento y aumento de imágenes. Según (Purestorage, 2024), el preprocesamiento de datos implica convertir la información cruda en un formato más útil y comprensible, facilitando así su análisis y el entrenamiento de modelos. Esta etapa es clave para mejorar tanto la precisión como el rendimiento de los algoritmos de aprendizaje automático, ya que permite corregir problemas como datos incompletos, ruido, errores y valores anómalos. El aumento de imagen es un paso en el que se aplican aumentos a las imágenes existentes en su conjunto de datos. Este proceso puede ayudar a mejorar la capacidad de su modelo para generalizar y, por lo tanto, desempeñarse de manera más efectiva en imágenes invisibles (Roboflow, 2023).

Herramientas. En la actualidad, existen herramientas para el preprocesamiento y aumento de imágenes, siendo una de ellas *Roboflow*. Esta plataforma proporciona un conjunto de funciones que optimizan y preparan las imágenes para su uso en tareas de visión por computadora (Roboflow, 2023). En cuanto al **preprocesamiento**, Roboflow ofrece las siguientes funciones:

- **Redimensionamiento de imágenes**: Permite ajustar todas las imágenes a una misma resolución, lo cual es esencial para que los modelos trabajen con entradas consistentes.
- **Conversión de color**: Transforma las imágenes a escala de grises, RGB u otros formatos según sea necesario.
- **Recorte automático (Auto-orientación)**: Ajusta la orientación de las imágenes y elimina bordes innecesarios.
- **Eliminación de ruido**: Aplica filtros para reducir interferencias visuales que podrían afectar el rendimiento del modelo.
- **Mejora de contraste y brillo**: Ajusta estos parámetros para destacar las carac-

terísticas importantes de la imagen.

- **Normalización**: Escala los valores de los píxeles a un rango adecuado (por ejemplo, de 0 a 1).

En cuanto al **aumento de imágenes** (*data augmentation*), Roboflow también proporciona las siguientes funciones:

- **Rotaciones aleatorias**: Gira las imágenes en distintos ángulos para simular diferentes perspectivas.
- **Traslaciones y escalado**: Desplaza o cambia el tamaño de los objetos en la imagen para aumentar la variedad espacial.
- **Reflejo horizontal o vertical**: Invierte las imágenes para ampliar la diversidad de ejemplos.
- **Agregado de ruido aleatorio**: Introduce ligeras variaciones para mejorar la tolerancia del modelo a imperfecciones.
- **Brillo y contraste aleatorio**: Aplica cambios variables para que el modelo se adapte a diferentes condiciones de iluminación.
- **Recortes aleatorios (Random crop)**: Extrae secciones aleatorias de la imagen original para evitar el sobreajuste.

Procesamiento de imágenes. Según (IBM, 2024), La segmentación de imágenes procesa datos visuales a nivel de píxel, utilizando diversas técnicas para anotar píxeles individuales como pertenecientes a una clase o instancia específica. Las técnicas tradicionales de segmentación de imágenes determinan las anotaciones analizando las cualidades inherentes de cada píxel (conocidas como "heurística"), como el color y la intensidad, mientras

que los modelos de deep learning utilizan redes neuronales complejas para un sofisticado reconocimiento de patrones. Los resultados de este proceso de anotación son máscaras de segmentación, que representan el límite y la forma específicos, píxel a píxel, de cada clase, que generalmente corresponde a diferentes objetos, características o regiones dentro de la imagen.

Clases semánticas: "cosas contables" y "cosas incontables". Se refieren a categorías que ayudan a clasificar los píxeles según sus características visuales. La diferencia entre los métodos de segmentación de imágenes radica en cómo manejan estas clases, lo que influye en la precisión de la segmentación.

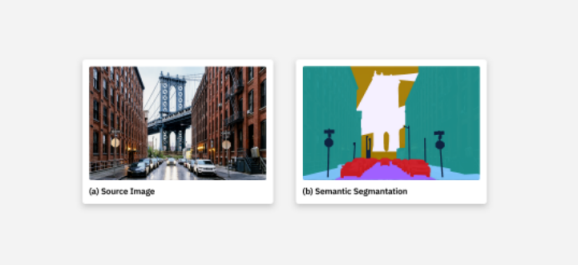
- Las cosas contables son clases de objetos con formas características, como "coche", "árbol" o "persona". Normalmente, las cosas contables tienen instancias claramente definidas que son contables. Tienen relativamente poca variación de tamaño de un caso a otro, así como partes constituyentes distintas de la cosa en sí: por ejemplo, todos los coches tienen ruedas, pero una rueda no es un coche.

- Cosas incontables hace referencia a clases semánticas de forma amorfa y tamaño muy variable, como "cielo", "agua" o "hierba". Normalmente, las cosas no tienen instancias individuales claramente definidas y contables. A diferencia de las cosas contables, las cosas incontables no tienen partes diferenciadas: una brizna de hierba y un campo de césped son ambos igualmente "hierba".

Segmentación semántica. La segmentación semántica considera todos los píxeles como pertenecientes a entidades indivisibles; no hace una distinción entre objetos individuales y fondos. Por ejemplo, si se entrena un modelo de segmentación semántica para identificar clases específicas en una imagen de una calle, generará máscaras de segmentación que delinean los contornos de cada clase relevante, tanto de objetos individuales (como autos o postes de luz) como de elementos del entorno (como carreteras y aceras). Sin embargo, no

diferencia ni cuenta el número de instancias de la misma clase. Por ejemplo, varios autos estacionados uno detrás del otro podría ser interpretados simplemente como un segmento largo de “autos”.

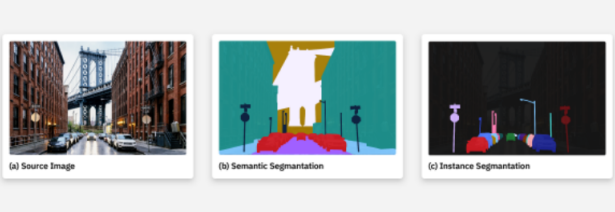
Figura 2.1
Segmentación semántica



Nota. Tomado de (IBM, 2024)

Segmentación de instancias. La segmentación de instancias va más allá al separar objetos contables de aquellos que no lo son, superando así la limitación de la detección de objetos. En lugar de solo proporcionar un cuadro delimitador, crea máscaras de segmentación precisas. Comparado con la segmentación semántica, es una tarea más desafiante. A diferencia de la segmentación semántica, que puede agrupar objetos contables de la misma clase, los modelos de segmentación de instancias deben discernir y delinear cada uno individualmente, incluso cuando se superponen. Por ejemplo, examinemos cómo ambos modelos abordan la identificación de coches estacionados en una calle de la ciudad.

Figura 2.2
Segmentación de instancias



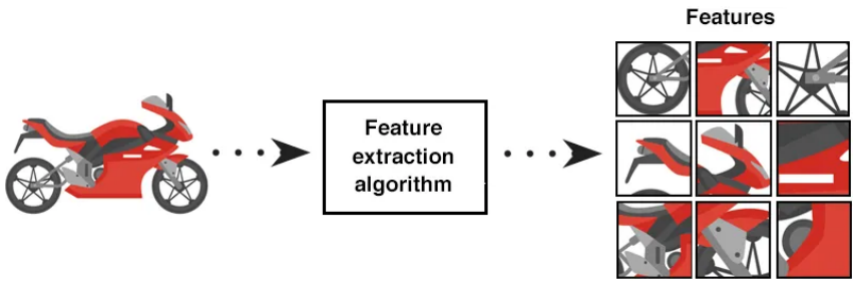
Nota. Tomado de (IBM, 2024)

Los algoritmos de segmentación de instancias pueden dividirse en enfoques de una o dos etapas. Los modelos de dos etapas, como R-CNN, primero detectan objetos para generar cuadros delimitadores y luego refinan la segmentación y clasificación dentro de cada uno. En

contraste, los modelos de una sola etapa, como YOLO, ejecutan la detección, clasificación y segmentación simultáneamente, lo que les permite lograr segmentación de instancias en tiempo real. Si bien los enfoques de una sola etapa priorizan la velocidad, podrían sacrificar precisión, mientras que los de dos etapas tienden a ofrecer mayor precisión, a expensas de la velocidad.

Extracción de características. Según (Manning Publications, 2020). Para comprender la extracción de características en visión computacional es necesario comprender que una característica es un dato mensurable en una imagen único para un objeto específico. Puede ser un color distinto o una forma específica como línea, borde o segmento. Una buena característica distingue objetos, como una rueda que claramente diferencia motocicletas de perros. Sin embargo, una sola característica puede no ser suficiente para distinguir entre objetos similares, como bicicletas y motocicletas, en cuyo caso se necesitan características adicionales como espejos o matrículas. En proyectos de aprendizaje automático, transformamos los datos crudos (imagen) en un vector de características para instruir al algoritmo de aprendizaje sobre las características del objeto, lo que facilita su reconocimiento y clasificación.

Figura 2.3
Extracción de características

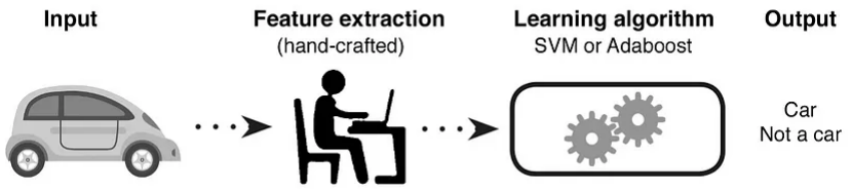


Nota. Tomado de (Manning Publications, 2020)

El aprendizaje automático tradicional utiliza características elaboradas a mano. En problemas de aprendizaje automático tradicionales, dedicamos tiempo a la selección manual de características. Confiando en nuestro conocimiento del dominio, creamos

características para mejorar los algoritmos. Luego, utilizamos clasificadores como SVM o Adaboost. Algunas características elaboradas a mano incluyen: HOG, Haar Cascades, SIFT y SURF.

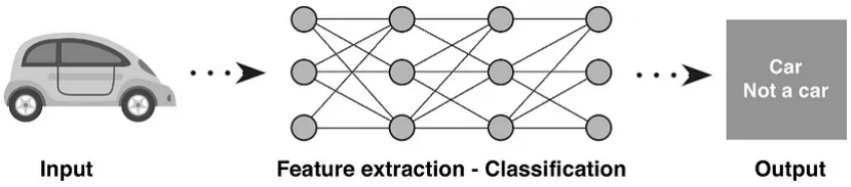
Figura 2.4
Extracción de características tradicional



Nota. Tomado de (Manning Publications, 2020)

Deep learning extrae automáticamente características. En el aprendizaje profundo, las características se extraen automáticamente. La red neuronal asigna pesos a sus conexiones para aprender la importancia de estas características en la salida. Al pasar la imagen por las capas de la red, esta identifica patrones para crear características. Las redes neuronales funcionan como extractores y clasificadores de características, siendo entrenables de principio a fin, a diferencia de los modelos de ML tradicionales que utilizan características elaboradas manualmente.

Figura 2.5
Extracción de características automático



Nota. Tomado de (Manning Publications, 2020)

2.2.3. Machine Learning

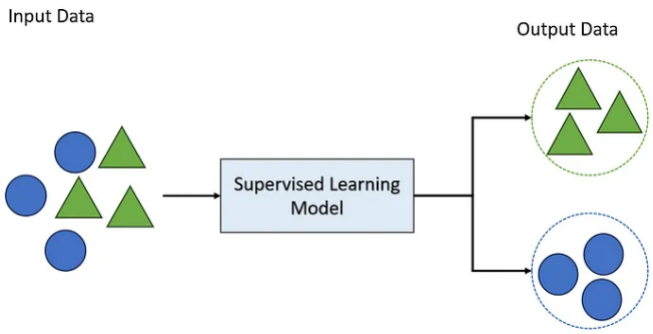
Machine learning es una rama de la inteligencia artificial (IA) y la informática que se centra en el uso de datos y algoritmos para imitar la forma en la que aprenden los seres

humanos, con una mejora gradual de su precisión (IBM, 2021).

Métodos de Machine Learning (IBM, 2021). Los modelos de machine learning se dividen en tres categorías principales.

- Machine learning supervisado: El aprendizaje supervisado, también conocido como machine learning supervisado, se define por su uso de los conjuntos de datos etiquetados para entrenar los algoritmos para clasificar datos o predecir resultados con precisión. A medida que se introducen datos de entrada en el modelo, este adapta sus pesos hasta que se haya ajustado correctamente. Esto ocurre como parte del proceso de validación cruzada para asegurarse de que el modelo evite el sobreajuste o el subajuste. El aprendizaje supervisado permite a las organizaciones resolver una amplia variedad de problemas del mundo real a escala como, por ejemplo, la clasificación de spam en una carpeta distinta de la bandeja de entrada. Algunos métodos utilizados en el aprendizaje supervisado son las redes neuronales, Naïve Bayes, la regresión lineal, la regresión logística, el bosque aleatorio y la máquina de vectores de soporte (SVM).

Figura 2.6
Machine Learning supervisado

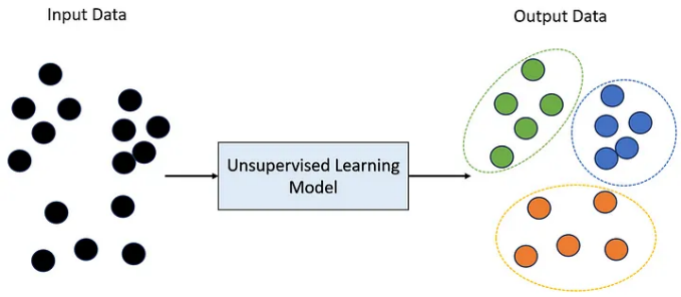


Nota. Tomado de (IBM, 2021)

- Machine learning no supervisado: El aprendizaje no supervisado, también conocido como machine learning no supervisado, utiliza algoritmos de machine learning para analizar y agrupar en clústeres conjuntos de datos sin etiquetar. Estos algoritmos des-

cubren agrupaciones de datos o patrones ocultos sin necesidad de ninguna intervención humana. La capacidad de este método para descubrir similitudes y diferencias en la información lo convierten en ideal para el análisis de datos exploratorios, las estrategias de venta cruzada, la segmentación de clientes y el reconocimiento de imágenes y patrones. También se utiliza para reducir el número de características de un modelo mediante el proceso de reducción de dimensionalidad. El análisis de componentes principales (PCA) y la descomposición en valores singulares (SVD) son dos de los enfoques más habituales para realizar este proceso. Otros algoritmos utilizados en el aprendizaje no supervisado son las redes neuronales, la agrupación en clúster de medias K y los métodos de agrupación probabilística.

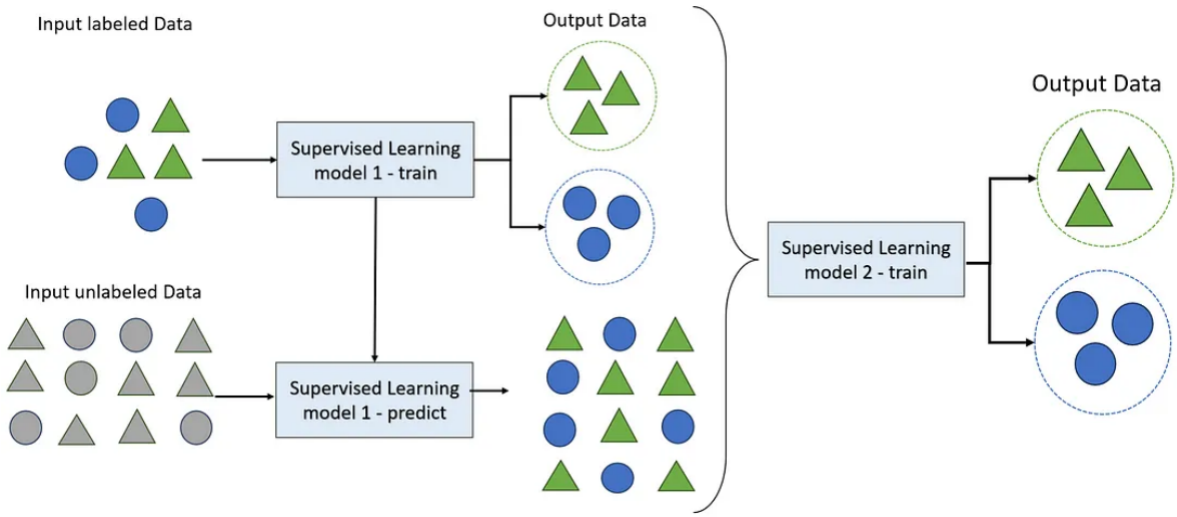
Figura 2.7
Machine Learning no supervisado



Nota. Tomado de (IBM, 2021)

- El aprendizaje semisupervisado ofrece un punto intermedio entre el aprendizaje supervisado y no supervisado. Durante el entrenamiento, utiliza un conjunto de datos etiquetados más pequeño para guiar la clasificación y la extracción de características de un conjunto de datos sin etiquetar de mayor tamaño. El aprendizaje semisupervisado puede resolver el problema de no tener suficientes datos etiquetados para un algoritmo de aprendizaje supervisado. También es útil si el coste de etiquetar datos suficientes es demasiado elevado.

Figura 2.8
Machine Learning semisupervisado



Nota. Tomado de (IBM, 2021)

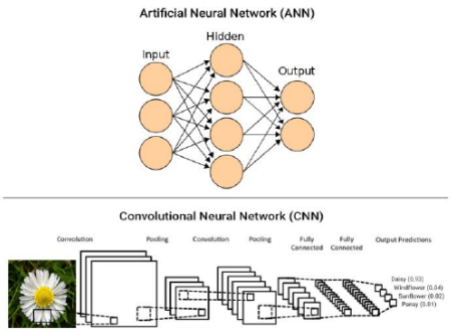
2.2.4. Deep Learning

Deep learning se refiere a técnicas de inteligencia artificial que emplean redes neuronales para crear e interpretar imágenes complejas, mejorando sistemas de reconocimiento de imágenes y otros algoritmos (Chojecki, 2019).

Red Neuronal Convolucional (CNN). Según Gelvez (2019), citado en (Gonzales del Valle Romero and Neyra Espinoza, 2022), las redes neuronales convolucionales se definen como "un tipo de redes neuronales artificiales utilizadas para el análisis y reconocimiento de imágenes, donde las capas convolucionales son fundamentales para generar mapas de características a partir de filtros aplicados a partes de una entrada" (p. 2).

Figura 2.9

Red Neuronal Convencional y una Red Neuronal Convolucional (CNN)



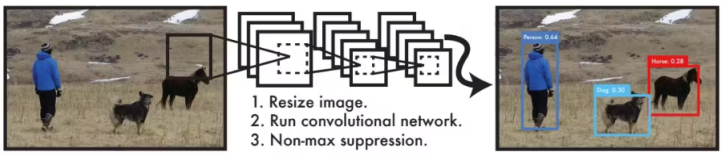
Nota. Tomado de (Gogul and Sathiesh Kumar, 2017)

En la investigación de Deep Learning, las CNN se aplican específicamente para aplicaciones de visión por computadora que involucran clasificación de imágenes y reconocimiento de objetos (Gogul and Sathiesh Kumar, 2017), como se muestra en la figura anterior. **Transferencia de Aprendizaje** El Transfer Learning ha tenido un gran éxito con el crecimiento del Deep Learning. Frecuentemente, los modelos utilizados en este campo necesitan grandes tiempos de cálculo y muchos recursos. Sin embargo, utilizando como punto de partida modelos pre-entrenados, el Transfer Learning permite desarrollar rápidamente modelos eficaces y resolver problemas complejos de Computer Vision o Natural Language Processing (DataScientest, 2022).

2.2.5. YOLO (You Only Look Once)

YOLO, desarrollado por Joseph Redmon y otros, trata la detección de objetos como regresión para cajas delimitadoras y probabilidades de clase. Analiza la imagen completa en tiempo de prueba, siendo rápido y adecuado para aplicaciones en tiempo real (Acharya, 2024).

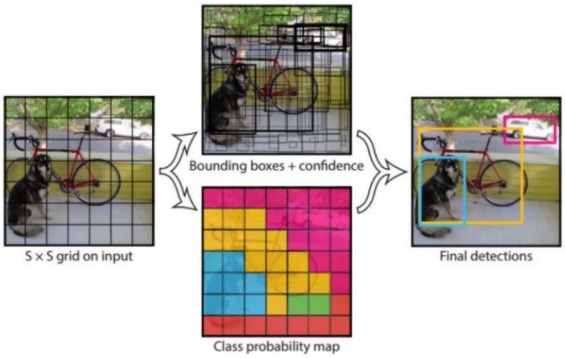
Figura 2.10
YOLO (You Only Look Once)



Nota. Tomado de (Encord, 2024)

Según Joiya (2022), YOLO analiza la imagen completa en su totalidad para predecir los cuadros delimitadores y luego calcula la probabilidad de clase para etiquetarlos. Este enfoque limita la cantidad de cuadros generados para cumplir con sus objetivos de manera eficiente. Con una capacidad de clasificación de hasta 155 FPS (fotogramas por segundo) en tiempo real, YOLO duplica la precisión media promedio (mAP) en comparación con otros clasificadores de objetos. Es una red convolucional única que predice simultáneamente múltiples cuadros delimitadores para varios objetos y asigna probabilidades de clase a cada uno.

Figura 2.11
YOLO Bounding Box, detección y localización de objetos



Nota. Tomado de (Joiya, 2022)

El método divide la imagen en una cuadrícula de M regiones, donde cada una tiene dimensiones iguales de P x P. Cada región de la cuadrícula es responsable de detectar y localizar los objetos que se encuentren en su área.

- Estas regiones predicen las coordenadas de los cuadros delimitadores en relación con

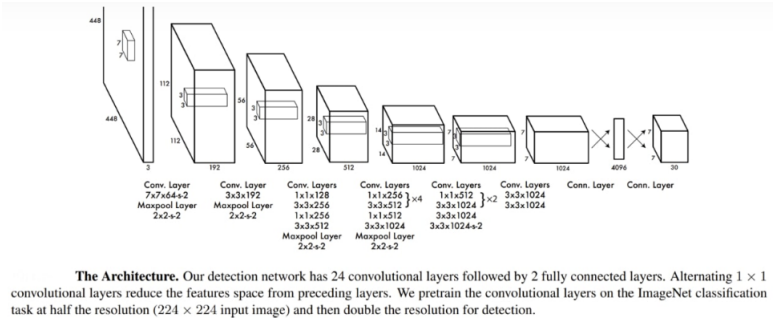
las celdas, junto con la etiqueta del objeto y la probabilidad de su presencia.

- Este enfoque reduce significativamente la carga computacional, ya que cada celda se encarga tanto de la detección como del reconocimiento.

- Además, se utiliza la supresión no máxima (Non-Max Suppression) para filtrar las predicciones, eliminando cuadros superpuestos y duplicados.

Arquitectura de YOLO. El algoritmo YOLO utiliza una única Red Neuronal Convolucional (CNN) que divide la imagen en una cuadrícula. Cada celda de la cuadrícula predice una cierta cantidad de cuadros delimitadores (bounding boxes). Además, cada celda predice una probabilidad de clase, la cual indica la probabilidad de que un objeto específico esté presente en dicho cuadro (Encord, 2024).

Figura 2.12
Arquitectura de YOLO



The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

Nota. Tomado de (Encord, 2024)

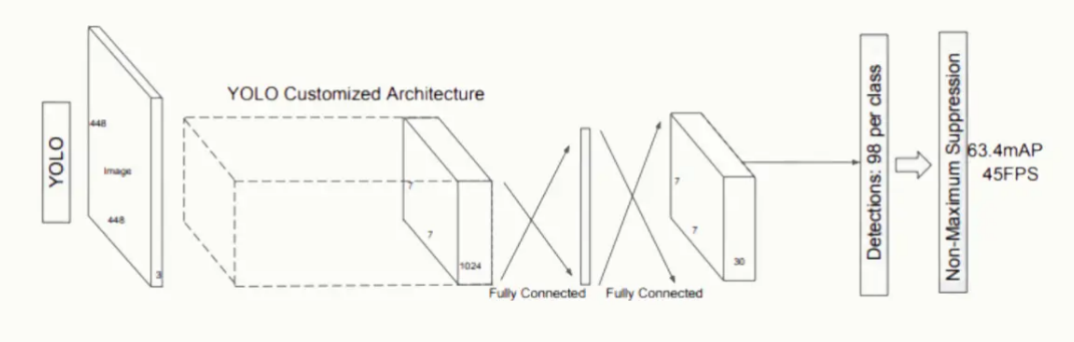
Proceso de reconocimiento del cuadro delimitador. Según, (Acharya, 2024) En el proceso de reconocimiento de cuadros delimitadores en YOLO, se siguen los siguientes pasos:

- Creación de la Cuadrícula: La imagen se divide en una cuadrícula de tamaño SxS. Cada celda de la cuadrícula es responsable de predecir un objeto si el centro del objeto cae dentro de ella.

- Predicción de Cuadros Delimitadores: Cada celda de la cuadrícula predice B cuadros delimitadores y puntajes de confianza para esos cuadros. El puntaje de confianza refleja cuán seguro está el modelo de que un cuadro contiene un objeto y qué tan preciso cree que es el cuadro.

- Predicción de Probabilidad de Clase: Cada celda de la cuadrícula también predice C probabilidades de clase condicionales (una por clase para los objetos potenciales). Estas probabilidades están condicionadas a la existencia de un objeto en el cuadro.

Figura 2.13
Cuadro delimitador en YOLO



Nota. Tomado de (Encord, 2024)

YOLO ofrece diversas versiones desarrolladas por Ultralytics, por lo que fue necesario determinar la versión adecuada. Según Sharma and Kumar (2024), en su investigación titulada *"Comparative performance of YOLOv8, YOLOv9, YOLOv10, YOLOv11 and Faster R-CNN models for detection of multiple weed species"*, concluyen que:

> "El menor tiempo de inferencia de los modelos YOLO, especialmente YOLOv11 y YOLOv8, los hace muy adecuados para aplicaciones como pulverizadores de precisión, drones y robots desmalezadores. Además, la capacidad de estos modelos para generalizar bien a partir de datos limitados los convierte en herramientas valiosas para aplicaciones agrícolas, donde no siempre se dispone de conjuntos de datos anotados de manera exhaustiva."

En la siguiente tabla se puede apreciar las comparaciones que hizo el autor, y se indica que YOLOv8 y YOLOv11 ofrecen un mejor equilibrio entre precisión y velocidad, lo que los hace ideales para aplicaciones en tiempo real (Sharma and Kumar, 2024).

Tabla 2.1
Análisis del rendimiento de los modelos

Modelo	**mAP@0.5 (Train)**	**mAP@0.5 (Test)**	**mAP@0.5–0.95 (Test)**	**Inference Time (Test-ms)**	**Precision (Test)**	**Recall (Test)**
YOLOv11	0.914	0.921	0.621	13.5	0.894	0.854
YOLOv10	0.899	0.872	0.591	19.3	0.889	0.804
YOLOv9	0.926	0.935	0.650	54.2	0.894	0.893
YOLOv8	0.921	0.925	0.650	23.0	0.860	0.858
Faster RCNN	0.889	0.865	0.615	63.8	0.901	0.825

Nota. Tomado de (Sharma and Kumar, 2024)

En las siguientes tablas, se presentan las variantes de YOLOv8 y YOLOv11, detallando sus configuraciones específicas, características principales y los contextos en los que destacan. Esto ayudará a comprender las diferencias entre cada variante.

Tabla 2.2
Variantes de YOLOv8.

Modelo	**tamaño (píxeles)**	**mAPval 50-95**	**Velocidad CPU ONNX (ms)**	**Velocidad A100 TensorRT (ms)**	**parámetros (M)**	**FLOPs (B)**
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

Nota. Tomado de ultralytics

Tabla 2.3
Variantes de YOLOv11.

Modelo	**tamaño (píxeles)**	**mAPval 50-95**	**Velocidad CPU ONNX (ms)**	**Velocidad T4 TensorRT10 (ms)**	**parámetros (M)**	**FLOPs (B)**
YOLO11n	640	39.5	56.1	1.5	2.6	6.5
YOLO11s	640	47.0	90.0	2.5	9.4	21.5
YOLO11m	640	51.5	183.2	4.7	20.1	68.0
YOLO11l	640	53.4	238.6	6.2	25.3	86.9
YOLO11x	640	54.7	462.8	11.3	56.9	194.9

Nota. Tomado de ultralytics

Tras analizar las variantes de YOLOv8 y YOLOv11, se eligieron YOLOv8s y YOLO11s por su compatibilidad con el hardware disponible (GPU NVIDIA GeForce GTX 1650 de 4GB). Estas versiones ofrecen alta velocidad de detección sin requerir hardware avanzado, a diferencia de otras variantes que exigen mayores capacidades computacionales. Se seleccionaron estos modelos para evaluar su precisión y recall, con el fin de determinar cuál integrar al prototipo, garantizando un rendimiento eficiente.

2.2.6. Deep Sort

Segun Muhammad (2022), DeepSORT(Simple Online and Realtime Tracking with a Deep Association Metric) es una mejora del algoritmo SORT(Simple Real-time Tracker) que se utiliza para el seguimiento de objetos en visión por computadora. Este algoritmo asigna un identificador único a cada objeto rastreado e incorpora aprendizaje profundo mediante descriptores de apariencia. Esto permite reducir los cambios de identidad y mejora la eficiencia del seguimiento.

SORT

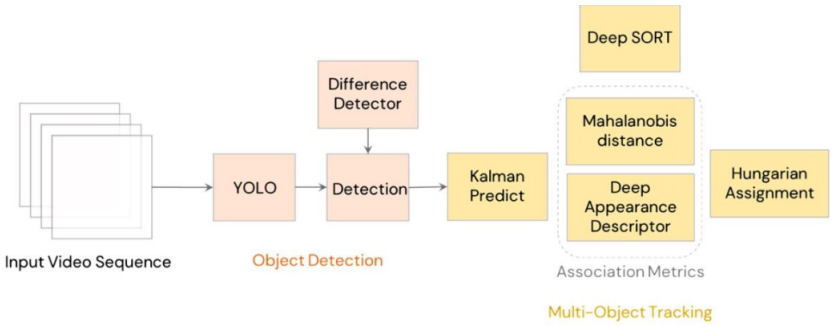
Segun Bewley et al. (2016) en su investigacion titulado “Simple Online and Realtime Tracking” indican que es un enfoque práctico para el seguimiento de múltiples objetos,

centrado en asociar objetos de manera eficiente para aplicaciones en tiempo real. Donde, para la detección y seguimiento utilizaron el filtro de Kalman y el algoritmo húngaro.

Arquitectura de Deep SORT

El algoritmo realiza un enfoque de dos etapas: primero generando detecciones de objetos y luego asociando esas detecciones a pistas existentes (Kouidri, 2023). En la siguiente imagen se muestra la arquitectura de este algoritmo.

Figura 2.14
Arquitectura de Deep SORT



Nota. Tomado de (Kouidri, 2023)

¿Por qué DeepSORT?

DeepSORT es un algoritmo de seguimiento de objetos ampliamente utilizado en el campo de la visión computacional debido a sus características destacadas. Según Kouidri (2023), se considera un modelo adecuado para la construcción del prototipo de sistema por las siguientes razones:

a) **Ventajas de DeepSORT:** DeepSORT combina información de movimiento con características de apariencia, lo que le confiere varias ventajas, entre las cuales destacan:

- **Resiliencia frente a oclusiones:** Gracias a los descriptores de apariencia, este algoritmo puede identificar objetos incluso después de haber sido momentáneamente ocultados.

- **Diferenciación de objetos similares:** Los descriptores visuales permiten distinguir entre elementos que comparten características similares.
- **Capacidad de procesamiento en tiempo real:** A pesar de su complejidad, DeepSORT logra un desempeño casi en tiempo real, aprovechando técnicas optimizadas para la extracción y asociación de características.
- **Alta compatibilidad:** Es adaptable a cualquier detector moderno de objetos, lo que aumenta su versatilidad.

b) **Aplicaciones reales de DeepSORT:** Este algoritmo tiene una amplia gama de aplicaciones, entre las cuales se destacan:

- **Vigilancia:** Mejora la seguridad al seguir a personas u objetos en entornos concurridos.
- **Deportes:** Permite el seguimiento de jugadores para evaluar estrategias y el rendimiento en diferentes disciplinas.
- **Vehículos autónomos:** Ayuda a seguir el movimiento de peatones y vehículos, incrementando la seguridad y eficiencia de la navegación.
- **Retail:** Facilita el análisis del comportamiento del cliente mediante el seguimiento de sus movimientos dentro de una tienda.

Capítulo 3

Desarrollo del proyecto

3.1. Metodología de desarrollo

Para llevar a cabo la construcción del prototipo, se empleó la metodología SCRUM, que se basa en la división del proyecto en sprints, siguiendo los principios de las metodologías ágiles.. Cada sprint representa una etapa definida de trabajo que abarca objetivos específicos. Esta estructura permitió organizar las actividades de manera incremental, favoreciendo la iteración y la mejora continua a lo largo del desarrollo del proyecto.

Historia de Usuario 1 - Sprint 1: Recolección inicial de datos

- **Rol:** Investigador.
- **Objetivo:** Capturar un conjunto inicial de datos para el entrenamiento del modelo.
- **Backlog:**
  - Capturar imágenes de motociclistas en diferentes ubicaciones de Cusco.

    - Seleccionar y organizar las imágenes relevantes.

- **Duración:** 3 semanas.

- **Beneficio:** Contar con un primer conjunto de datos para iniciar el entrenamiento del modelo.

Historia de Usuario 2 - Sprint 2: Selección de modelo e integración inicial

- **Rol:** Desarrollador de visión computacional.

- **Objetivo:** Seleccionar un modelo base e integrarlo en el primer prototipo funcional.

- **Backlog:**

    - Evaluar y seleccionar el modelo más adecuado para el prototipo.

    - Entrenar el modelo con el conjunto inicial de datos.

    - Construir el primer prototipo del sistema e integrar el modelo.

- **Duración:** 2 semanas.

- **Beneficio:** Tener un prototipo básico que detecte motociclistas en videos.

Historia de Usuario 3 - Sprint 3: Recolección de datos adicionales de videovigilancia

- **Rol:** Investigador.

- **Objetivo:** Ampliar el dataset utilizando datos de las cámaras de videovigilancia.

- **Backlog:**
  - Solicitar acceso a cámaras de videovigilancia.
  - Seleccionar grabaciones de video con la mejor resolución y mejor escenario.
- **Duración:** 4 semanas.
- **Beneficio:** Ampliar la diversidad y calidad del dataset inicial.

Historia de Usuario 4 - Sprint 4: Desarrollo de script para extracción de imágenes relevantes

- **Rol:** Desarrollador.
- **Objetivo:** Automatizar la extracción de imágenes relevantes de videos de cámaras de videovigilancia.
- **Backlog:**
  - Diseñar el script para procesar videos y extraer imágenes.
  - Eliminar imágenes extraídas menos relevantes.
- **Duración:** 4 semanas.
- **Beneficio:** Acelerar la generación de conjuntos de datos a partir de videos.

Historia de Usuario 5 - Sprint 5: Entrenamiento y evaluación del modelo con nuevos datos

- **Rol:** Especialista en visión computacional.

- **Objetivo:** Mejorar el modelo utilizando los datos recolectados de videovigilancia.
- **Backlog:**
  - Etiquetar las imágenes del nuevo conjunto de datos.
  - Entrenar el modelo con el conjunto ampliado.
  - Evaluar el desempeño del modelo actualizado.
- **Duración:** 3 semanas.
- **Beneficio:** Aumentar la precisión y robustez del modelo.

Historia de Usuario 6 - Sprint 6: Integración del modelo con funcionalidad de seguimiento de objetos

- **Rol:** Desarrollador.
- **Objetivo:** Incorporar una funcionalidad de seguimiento al prototipo.
- **Backlog:**
  - Implementar un algoritmo de seguimiento de objetos (DeepSort).
  - Integrar la funcionalidad al prototipo.
  - Validar el rendimiento del prototipo con videos de prueba.
- **Duración:** 3 semanas.
- **Beneficio:** Prototipo con detección y seguimiento de motociclistas.

Historia de Usuario 7 - Sprint 7: Recolección de datos para ampliar dataset con la clase "Sin_Casco"

- **Rol:** Investigador.
- **Objetivo:** Recolectar datos locales específicos utilizando una cámara personal.
- **Backlog:**
  - Planificar rutas y ubicaciones en el distrito de Yanaoca, provincia de Canas.
  - Capturar imágenes y videos de motociclistas.
  - Emplear el script para procesar videos.
  - Seleccionar imágenes relevantes para el dataset.
- **Duración:** 4 semanas.
- **Beneficio:** Incorporar datos contextualizados para mejorar la detección en un entorno específico.

Historia de Usuario 8 - Sprint 8: Entrenamiento y evaluación final del modelo

- **Rol:** Especialista en visión computacional.
- **Objetivo:** Refinar el modelo con los datos recolectados en Yanaoca.
- **Backlog:**
  - Etiquetar el nuevo conjunto de datos.
  - Entrenar y evaluar el modelo con todos los datos recolectados.

- **Duración:** 4 semanas.
- **Beneficio:** Modelo optimizado para detección en entornos variados.

Historia de Usuario 9 - Sprint 9: Integración final del modelo al sistema prototipo

- **Rol:** Desarrollador.
- **Objetivo:** Completar la integración del modelo y validar el sistema prototipo.
- **Backlog:**
  - Integrar el modelo final al prototipo.
  - Realizar pruebas con videos en distintas resoluciones.
  - Optimizar el rendimiento del prototipo para garantizar su funcionalidad.
- **Duración:** 3 semanas.
- **Beneficio:** Sistema funcional listo para evaluación y presentación final.

3.2. Construcción del dataset

Antes de trabajar con cualquier tecnología de inteligencia artificial, es fundamental contar con un conjunto de datos adecuado para la tecnología a emplear. En el presente proyecto, se utilizó visión computacional, lo que requiere un dataset de imágenes para entrenar los modelos de detección.

Equipos empleados

- **Teléfono móvil Samsung Galaxy A32**
  - Tipo: Cámara trasera de 64 megapíxeles.
  - Resolución: 4624 × 3468 píxeles.
  - Uso: Captura de imágenes estáticas en zonas urbanas y rurales.
  - Condiciones: Luz natural, capturas diurnas.
- **Cámara de videovigilancia fija**
  - **Tipo:** Cámara de seguridad instalada en semáforos de los distritos de Cusco y Santiago.
  - **Resolución:** 1280 × 720 píxeles, 1920 × 1080 píxeles y 3840 × 2160 píxeles.
  - **Ubicación:** Av. Cultura – Magisterio (Cusco), Av. Sol – Qoricancha (Cusco) y Mercado Huancaro (Santiago).
  - **Uso:** Obtención de grabaciones históricas de motociclistas en tránsito.
  - **Condiciones:** Mañana (7:30 a.m. – 8:30 a.m.), mediodía (12:40 p.m. – 1:30 p.m.) y noche (7:30 p.m. – 8:30 p.m.).

Fases de construcción

- **Primera fase:** En esta etapa inicial, se recopiló un dataset compuesto por 160 imágenes obtenidas mediante capturas directas en diferentes puntos de la ciudad de Cusco. Este conjunto de datos permitió desarrollar un prototipo básico del sistema.
- **Segunda fase:** Durante esta fase, el dataset fue ampliado a 720 imágenes gracias al acceso a las cámaras de videovigilancia de Seguridad Ciudadana de Cusco. Las

imágenes adicionales se generaron a partir de los videos registrados por estas cámaras, lo que incrementó significativamente la diversidad y el volumen de datos.

- **Tercera fase:** Finalmente, el dataset se amplió a un total de 1560 imágenes. Estas imágenes fueron recopiladas fuera de la ciudad de Cusco, específicamente en el distrito de Yanaoca, provincia de Canas. En esta etapa, se emplearon tanto capturas directas como el procesamiento de videos para garantizar una representación más amplia de escenarios y condiciones.

3.2.1. Imágenes capturadas mediante fotografías

Según la Autoridad de Transporte Urbano (ATU, 2024), las cámaras de videovigilancia están ubicadas principalmente en avenidas de alto tránsito vehicular. En esta investigación se consideraron como sitios estratégicos dichas avenidas principales, debido a su constante flujo de vehículos. Las fotografías fueron capturadas con un dispositivo móvil en estos puntos, Este enfoque permitió garantizar la diversidad en las condiciones ambientales, como iluminación, ángulos y escenarios, en el siguiente gráfico se muestran alguna de estas imágenes capturadas.

Figura 3.1
Captura directa de imágenes

La recolección de datos mediante este método garantiza imágenes de alta calidad, lo que es ideal para entrenar modelos de visión computacional. Sin embargo, debido a que el entrenamiento requiere un volumen significativo de datos, el proceso de captura manual resulta ser considerablemente lento, lo que lo hace menos eficiente en proyectos que demandan grandes conjuntos de datos.

3.2.2. Imágenes obtenidas a partir de videos

En la segunda etapa como se menciono anteriormente, se solicitaron grabaciones de video a las municipalidades de Cusco y del distrito de Santiago, las cuales provenían de sus cámaras de videovigilancia y estaban en formato de video. Para convertir estos videos en imágenes individuales utilizables en el dataset, se desarrolló un script personalizado como parte del proyecto.

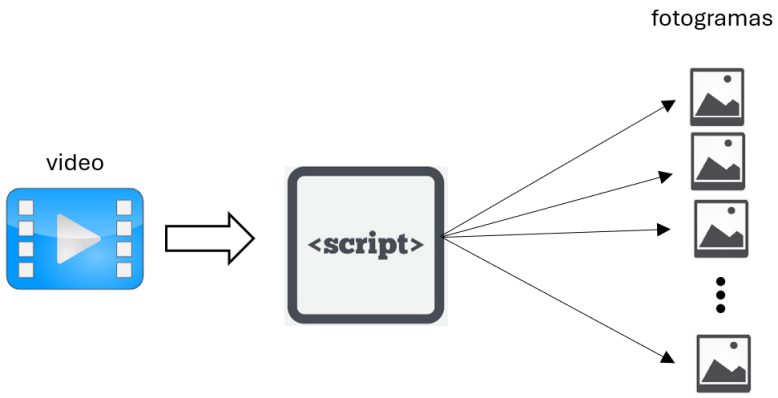
Desarrollo del Script. El script fue implementado en Python y utilizó el modelo YOLO preentrenado con el dataset COCO como base para la detección de motocicletas. Su diseño permitió analizar cada cuadro del video, identificar motocicletas presentes y capturar automáticamente los fotogramas relevantes. Estas imágenes fueron almacenadas en una carpeta predefinida, incrementando la eficiencia en la generación de datos y garantizando su relevancia para el dataset.

Algorithm 1 Pseudocódigo del script para detección y extracción de motocicletas

Cargar el modelo YOLO preentrenado con el dataset COCO
Seleccionar video para procesamiento
Abrir el archivo de video
while el video está abierto **do**
    Leer el siguiente cuadro del video
    Convertir el cuadro a tensor (formato compatible con el modelo YOLO)
    Realizar detección de objetos en el cuadro usando el modelo YOLO
    for cada detección **do**
        if la clase detectada es motocicleta **then**
            Obtener las coordenadas del cuadro delimitador (bounding box)
            Recortar el fotograma usando las coordenadas del cuadro delimitador
            Guardar el fotograma recortado como imagen en la carpeta predefinida
            Registrar información de la detección (como la probabilidad y las coordenadas)
        end if
    end for
end while
Cerrar el archivo de video

A continuación, se presenta un gráfico que ilustra el comportamiento del script en acción. Este gráfico permite visualizar de manera detallada la obtención de imágenes a partir de video.

Figura 3.2
Obtención de imágenes a partir de video



Post procesamiento. Las imágenes de calidad replicarán la iluminación, los ángulos y las distancias de la cámara que se encontrarían en la ubicación de destino. Un conjunto de datos de alta calidad contiene distintos ejemplos del tema deseado. En términos generales,

si no puede reconocer a su sujeto objetivo a partir de una imagen, tampoco puede hacerlo un algoritmo (TagX, 2023).

Las imágenes relevantes seleccionadas para el dataset fueron aquellas que mostraban motociclistas claramente visibles y en situaciones útiles para el entrenamiento del modelo. Este filtrado aseguró que las imágenes cumplieran con los estándares necesarios para mejorar la precisión y eficacia del sistema de detección.

Figura 3.3
Imágenes Relevantes

Por otro lado, se eliminaron imágenes irrelevantes, como aquellas que estaban duplicadas, borrosas, donde los motociclistas no se visualizaban con nitidez, motocicletas sin conductor y otras que no aportaban valor al dataset. Este proceso permitió depurar el conjunto de datos y garantizar su calidad.

Figura 3.4
Imágenes irrelevantes

Se recolectó un total de 1560 imágenes que contienen motociclistas en diferentes situaciones. Estas imágenes fueron clasificadas en tres categorías aquellas en las que todos los motociclistas portan casco de seguridad, aquellas en las que ninguno lo porta, y aquellas donde existe motociclistas con y sin casco en una misma escena.

Tabla 3.1
Distribución de imágenes según uso del casco

Categoría	Cantidad de imágenes
Solo motociclistas con casco	677
Solo motociclistas sin casco	716
Motociclistas con y sin casco	167
Total	**1560**

3.3. Selección de modelos

3.3.1. Modelo de detección de objetos

La elección de un modelo adecuado en esta investigación es fundamental, ya que de ello depende el desarrollo de un prototipo de sistema eficiente para la detección de motociclistas con y sin casco de seguridad. Dado que este prototipo esta pensado para que funcione en tiempo real, procesando videos en vivo para alertar a las autoridades o registrar infracciones de manera inmediata, es esencial optar por modelos que ofrezcan un equilibrio óptimo entre precisión y velocidad. También es importante mencionar que en muchos casos el hardware disponible cuenta con recursos limitados, resulta crucial emplear un modelo que maximice el aprovechamiento de los recursos computacionales.

Por lo tanto, la selección del modelo de detección no solo debe garantizar resultados precisos, sino también ser capaz de operar en entornos prácticos con eficiencia. En este contexto, se han evaluado dos modelos mas comunes en el campo de la visión computacional: Faster R-CNN y YOLO.

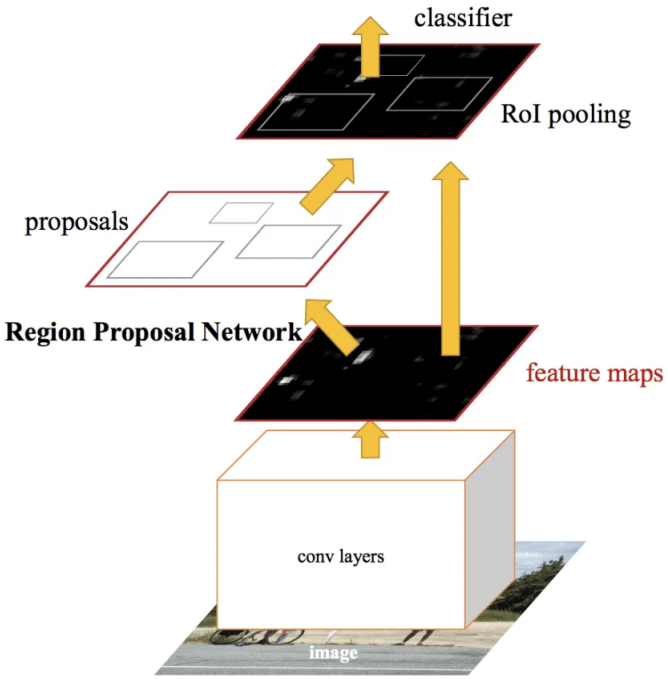
Faster R-CNN. Según Sojasingarayar (2022), Faster R-CNN es una red neuronal convolucional profunda diseñada para la detección de objetos, que se presenta como un sistema unificado de extremo a extremo. Esta red permite predecir con rapidez y precisión las ubicaciones de diferentes objetos. Para comprender en profundidad Faster R-CNN, es fundamental conocer las redes de las que evolucionó, como R-CNN y Fast R-CNN. En esencia, Faster R-CNN es una mejora de Fast R-CNN, y su principal ventaja radica en la integración

de la red de propuestas de regiones (RPN), lo que le otorga mayor velocidad. La arquitectura de Faster R-CNN está compuesta por dos módulos principales:

- RPN (Red de propuestas de regiones): Una red neuronal convolucional que sugiere áreas de interés y el tipo de objeto presente en esas regiones.
- Fast R-CNN: Otra red convolucional que analiza las áreas propuestas, extrae sus características y produce tanto los cuadros delimitadores como las etiquetas de clase.

Ambos módulos trabajan sobre la misma salida de una red CNN profunda. En este sistema, la red de propuestas de regiones actúa como un mecanismo de atención, guiando a Fast R-CNN hacia las áreas que deben ser examinadas con mayor detalle.

Figura 3.5
Arquitectura de Faster R-CNN



Nota. Tomado de (Sojasingarayar, 2022)

Análisis y conclusión sobre el modelo Faster R-CNN frente a YOLO. Al comparar las características de Faster R-CNN con las características ya mencionadas en el capitulo 2 de YOLO, queda claro que ambos algoritmos tienen ventajas particulares que los hacen útiles en diferentes escenarios. La selección del algoritmo adecuado para nuestro

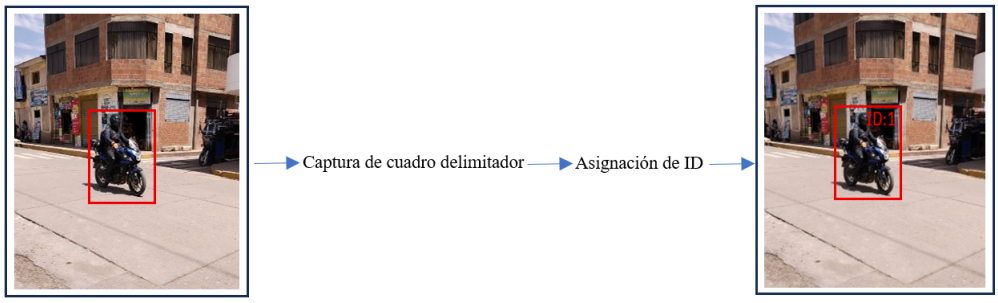
prototipo de sistema debe considerar un equilibrio entre precisión y velocidad.

El propósito principal no radica en determinar el mejor detector, sino en identificar aquel que ofrezca el balance óptimo entre velocidad y precisión según los requisitos de una tarea. Aunque Faster R-CNN y YOLO son precisos, YOLO destaca por su simplicidad, eficiencia y capacidad de entrenamiento de extremo a extremo. Su enfoque de detección en un solo paso lo convierte en una opción preferida para aplicaciones en tiempo real, ya sea en imágenes estáticas o en video. Además, su representación más generalizada de los objetos, junto con su robustez y rapidez, lo posiciona como un algoritmo altamente recomendable para tareas de detección de objetos (Joiya, 2022).

3.3.2. Modelo de seguimiento de objetos

El seguimiento de objetos en este proyecto de investigación es fundamental, ya que, como se expone en el Capítulo 1, la gravedad de las lesiones en la cabeza de un motociclista está estrechamente relacionada con la velocidad a la que transita. Para abordar este aspecto, se ha optado por utilizar DeepSORT en donde se menciona en el capítulo 3 sus ventajas.

Figura 3.6
Flujo del modelo de seguimiento de objetos



En el flujo descrito, se observa cómo el modelo de seguimiento de objetos realiza la captura del cuadro delimitador del motociclista, lo que permite identificar su posición y dimensiones dentro del fotograma. A partir de esta detección, se le asigna un ID único que se mantiene a lo largo de los siguientes cuadros, facilitando el seguimiento continuo del mismo

objeto. Esto permite distinguir y rastrear múltiples objetos simultáneamente, incluso cuando se cruzan o se ocultan parcialmente.

3.4. Construcción del prototipo

3.4.1. Etiquetado del dataset y entrenamiento del modelo YOLO

Definición de clases. El primer paso en el proceso de etiquetado consiste en definir de manera clara las clases que el modelo YOLO debe identificar. Estas clases representan los objetos de interés dentro del dataset y son esenciales para estructurar correctamente la información que el modelo utilizará durante el entrenamiento.En este proyecto, las clases definidas son las siguientes.

Tabla 3.2
Clases Definidas para el Modelo

Clase	Descripción
Con_Casco	Indica que el motociclista lleva casco de seguridad.
Sin_Casco	Señala que el motociclista no lleva casco de seguridad.

En la figura siguiente se representa gráficamente cómo se definen las clases mencionadas en la tabla. Estas áreas serán las regiones de interés de cada imagen de nuestro dataset.

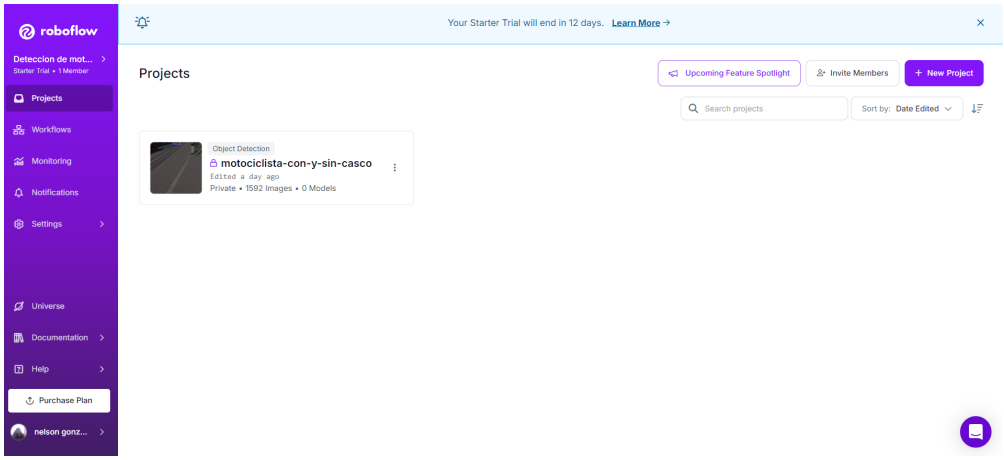
Figura 3.7
Definición de clases

Herramientas. Para el proceso de etiquetado de las imágenes del dataset, se utilizó

Roboflow, una herramienta que ofrece diversas funciones, tal como se detallan en el capítulo 2, **Roboflow** es una aplicación de web ampliamente reconocida en el ámbito de la visión por computadora. Esta herramienta destaca por su interfaz sencilla e intuitiva, lo que la hace ideal para usuarios con diferentes niveles de experiencia en el etiquetado de datasets.

Una vez ingresado al dashboard de Roboflow, se cargó el dataset con todas las imágenes listas para el etiquetado. La plataforma permite configurar la ubicación donde se almacenarán las anotaciones generadas. Su interfaz facilita el trazado manual de cuadros delimitadores alrededor de los objetos de interés y la asignación de etiquetas predefinidas, permitiendo un etiquetado rápido y eficiente.

Figura 3.8
Carga de dataset en Roboflow

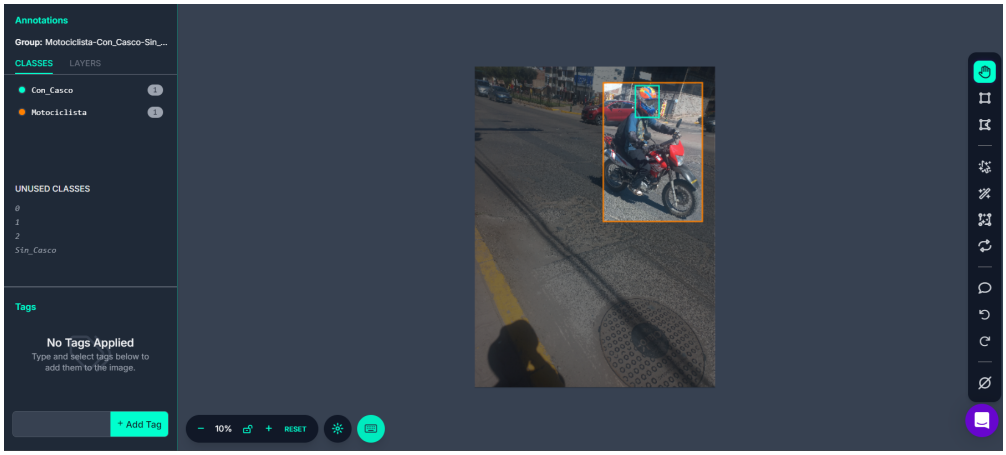


Nota. Tomado de Roboflow

En esta imagen se muestra el proceso de etiquetado de un motociclista con casco de seguridad utilizando la herramienta **Roboflow**. Se traza un cuadro delimitador alrededor del motociclista y se asigna la etiqueta correspondiente, lo que es esencial para entrenar el modelo de detección.

Figura 3.9

Etiquetado de motociclista con casco de seguridad

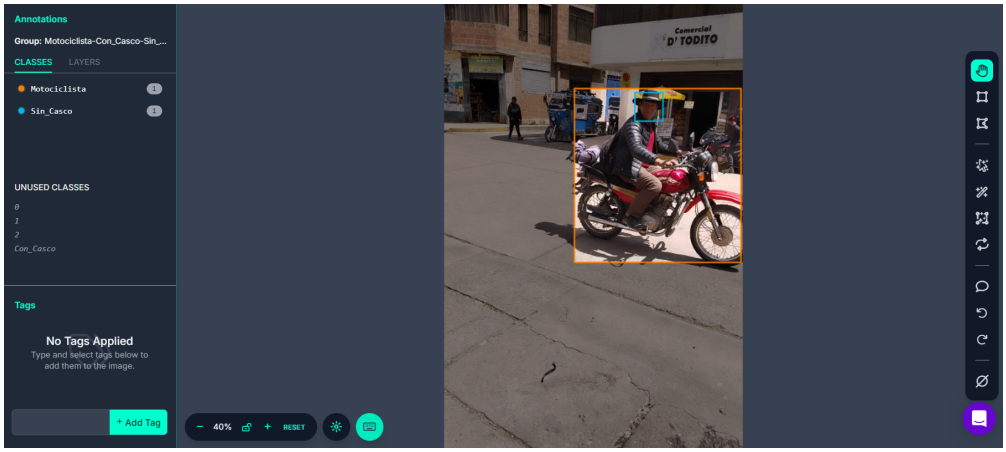


Nota. Tomado de Roboflow

De manera similar, en esta imagen se muestra el etiquetado de un motociclista sin casco de seguridad. El proceso es igual, pero con la etiqueta correspondiente a la falta de casco, lo que permitirá al modelo aprender a diferenciar ambos casos.

Figura 3.10

Etiquetado de motociclista sin casco de seguridad



Nota. Tomado de Roboflow

Roboflow genera automáticamente archivos de anotaciones en formato .txt compatibles con el modelo YOLO. Estos archivos incluyen las coordenadas de los cuadros delimitadores y las etiquetas asociadas a cada objeto, proporcionando la información clave necesaria para el entrenamiento del modelo.

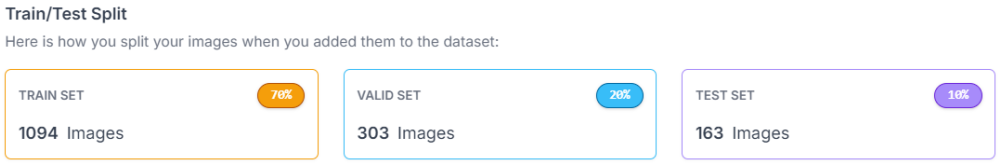
Distribución de datos. Para entrenar el modelo YOLO, se dispone de un dataset

compuesto por 1560 imágenes completamente etiquetadas, seleccionadas cuidadosamente para garantizar una representación diversa de los escenarios y condiciones posibles. Estas imágenes incluyen distintas resoluciones, ángulos de cámara, niveles de iluminación y contextos, lo que asegura que el modelo pueda aprender a generalizar mejor en situaciones reales. El dataset ha sido dividido estratégicamente en tres subconjuntos principales, cada uno diseñado con un propósito específico dentro del proceso de desarrollo del modelo y para optimizar su rendimiento:

- **Train:** Este subconjunto, que constituye el mayor porcentaje del dataset, se utiliza para ajustar los parámetros del modelo durante el entrenamiento. El modelo aprende a identificar patrones relevantes a partir de los datos contenidos en este conjunto.
- **Validation (Val):** Este conjunto se emplea para evaluar el desempeño del modelo mientras se entrena. Sirve como un indicador clave para detectar problemas como el sobreajuste (overfitting), asegurando que el modelo no memorice los datos de entrenamiento y sea capaz de generalizar correctamente.
- **Test:** Este conjunto está reservado para realizar una evaluación objetiva del modelo una vez completado el entrenamiento. Permite medir su rendimiento en datos que no ha visto previamente, garantizando una valoración imparcial de su capacidad predictiva.

Roboflow nos permite distribuir el dataset en porcentajes, lo que facilita organizar los datos según las necesidades del proyecto. En nuestro caso, priorizaremos el entrenamiento, distribuyendo los datos de la siguiente manera: 70 % para entrenamiento, 20 % para validación y 10 % para pruebas. Estos porcentajes son ajustables según el tamaño del dataset y los requerimientos específicos del proyecto. La siguiente imagen muestra cómo se han dividido los datos en este proyecto.

Figura 3.11
Distribución de datos



Nota. Tomado de Roboflow

Preprocesamiento de datos. El preprocesamiento de datos es una etapa fundamental en la preparación de un dataset, ya que permite ajustar las imágenes para garantizar su compatibilidad y optimizar el rendimiento del modelo. En nuestro caso, utilizamos las herramientas de Roboflow para aplicar dos operaciones principales:

- **Auto-Orient:** Corrige automáticamente la orientación de las imágenes para asegurar que todas tengan el mismo eje de referencia, evitando inconsistencias durante el entrenamiento.
- **Resize (Stretch to 640x640):** Cambia el tamaño de las imágenes estirándolas a una resolución uniforme de 640x640 píxeles, lo cual es necesario para que sean compatibles con el modelo YOLO y para reducir la carga computacional sin perder información relevante.

Estas transformaciones garantizan que los datos estén preparados y normalizados para el proceso de entrenamiento.

Aumento de datos. El **aumento de datos** es una técnica utilizada para incrementar de manera artificial el tamaño de un dataset, aplicando transformaciones a las imágenes existentes. Esto ayuda a mejorar la capacidad del modelo para generalizar y adaptarse a diferentes escenarios, especialmente cuando el dataset original es limitado.

En este proyecto, se utilizó la herramienta de *Roboflow* para realizar los siguientes aumentos:

- **Rotación:** Se aplicaron rotaciones aleatorias entre $-10°$ y $+10°$.
- **Desenfoque:** Se añadió un desenfoque de hasta 0.1 píxeles.
- **Ruido:** Se incorporó ruido en hasta el 0.1 % de los píxeles.

En la siguiente imagen podemos ver una imagen con las tres transformaciones realizadas para el aumente de datos.

Figura 3.12
Transformaciones realizadas

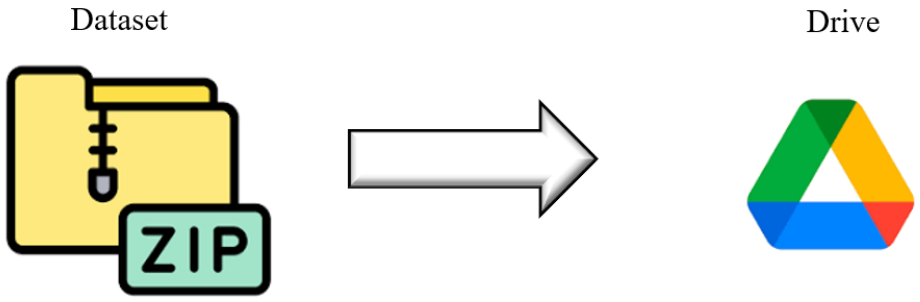
Estas transformaciones enriquecen el dataset al introducir variaciones que simulan condiciones reales, lo que contribuye a mejorar la precisión y robustez del modelo durante el entrenamiento. Asimismo, se utilizó un Maximum Version Size de 5936 (x5), multiplicando por cinco el tamaño del dataset original.

Exportación de dataset. Una vez realizado el preprocesamiento y el aumento de datos, Roboflow permite exportar el dataset en formato .zip. Este archivo contiene el dataset dividido en las carpetas train, val y test, con las imágenes y sus correspondientes archivos de anotación (.txt). Dentro del archivo .zip se encuentra también el archivo data.yaml, que contiene la configuración del dataset, esencial para entrenar el modelo. A continuación se muestra el archivo `data.yaml` que genero roboflow la cual contiene la configuración del dataset:

```
train: /ruta/a/train/images

val: /ruta/a/val/images

test: /ruta/a/test/images

nc: 2

names: ['Con_Casco','Sin_Casco']
```

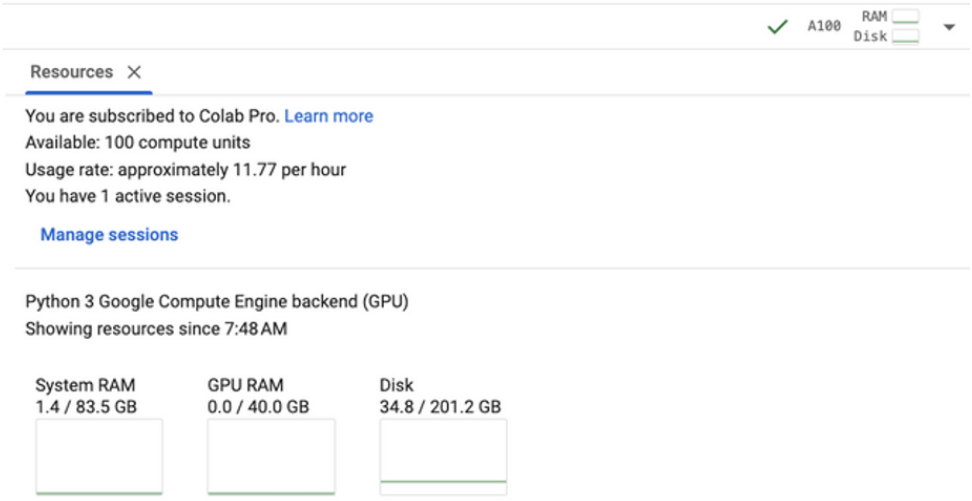
Carga de dataset para entrenamiento. El proceso de entrenamiento del modelo YOLO (You Only Look Once) se llevó a cabo utilizando Google Colab Pro, una herramienta que ofrece acceso a GPUs potentes, ideal para tareas de entrenamiento intensivo en visión computacional. Para iniciar, el conjunto de datos (dataset) previamente etiquetado se subió a Google Drive en un archivo comprimido, lo que permitió una integración fluida con el entorno de Colab. Esto garantizó un acceso rápido y seguro a los archivos durante todas las fases del entrenamiento.

Figura 3.13
Carga de dataset a drive



Una vez que el conjunto de datos ha sido cargado correctamente en Google Drive, se procede a configurar el entorno de ejecución para el entrenamiento del modelo. En esta etapa, es esencial seleccionar la opción de utilizar una GPU como recurso de cómputo, ya que esto acelera significativamente el procesamiento y mejora el rendimiento durante el entrenamiento. Google Colab Pro ofrece acceso a GPUs potentes, como la A100, que son ideales para tareas de entrenamiento intensivas y permiten reducir considerablemente los tiempos de entrenamiento. Para entrenar el modelo se eligio la GPU A100, que está especialmente diseñada para manejar cargas de trabajo pesadas de manera eficiente.

Figura 3.14
Entorno de ejecución google colab



Una vez seleccionado el entorno de ejecución adecuado, el siguiente paso consiste

en acceder a los datos almacenados en Google Drive y prepararlos para el entrenamiento del modelo. Este proceso es crucial, ya que se debe garantizar que el conjunto de datos esté disponible en el entorno de ejecución de Google Colab y se encuentre en un formato adecuado para ser utilizado durante el proceso de entrenamiento. Para lograr esto, primero se monta Google Drive en el entorno de Colab utilizando la librería drive de google.colab, lo que permite acceder a los archivos almacenados en la nube de Google directamente desde el entorno de ejecución. Luego, es necesario descomprimir el archivo ZIP que contiene el conjunto de datos. Este archivo debe extraerse en una carpeta local dentro de Colab para facilitar su acceso y manipulación durante el entrenamiento. A continuación, se muestran los comandos necesarios para montar Google Drive y descomprimir el archivo del conjunto de datos:

Figura 3.15
Acceso y extracción de dataset

```
from google.colab import drive
drive.mount('/content/drive')
#descomprimir dataset comprimido
!unzip /content/drive/MyDrive/Tesis/DataSet/Imagenes-sin-recortar.zip -d /content/dataset
```

Tal como se abordó en la sección 3.3.1 sobre las capacidades de los modelos YOLOv8s y YOLO11s, en este punto se procedió a entrenar ambos modelos utilizando el dataset previamente preparado. Para el entrenamiento de los modelos, se empleó los siguientes fragmentos de código en Python, utilizando la biblioteca Ultralytics:

Entrenamiento. Una vez cargado el dataset, se instaló la biblioteca Ultralytics, necesaria para trabajar con los modelos YOLO. Esta biblioteca facilita el entrenamiento, evaluación e implementación de los modelos al incluir todas las herramientas y dependencias requeridas, como PyTorch.

```
!pip install ultralytics
```

Tal como se abordó en la sección 3.1 sobre las capacidades de los modelos YOLOv8s y

YOLO11s, en este punto se procedió a entrenar ambos modelos utilizando el dataset previamente preparado. Para el entrenamiento de los modelos, se empleó los siguientes fragmentos de código en Python, utilizando la biblioteca Ultralytics:

Figura 3.16
Entrenamiento YOLOv8s

```
from ultralytics import YOLO
model = YOLO('yolov8s.pt')
results = model.train(
    data='/content/dataset/data.yaml', # uso de data.yaml de roboflow
    epochs=100,          # Nro de epocas
    imgsz=640,           # Tamaño de las imágenes segun a las preprocesadas
    batch=16,            # un batch razonable que no comprometa estabilidad
    lr0=0.001,           # Tasa de aprendizaje inicial
    workers=8,            # Hilos de preprocesamiento
    device=0             # Utilizar GPU
)
```

Figura 3.17
Entrenamiento YOLO11s

```
from ultralytics import YOLO
model = YOLO('yolo11s.pt')
results = model.train(
    data='/content/dataset/data.yaml', # uso de data.yaml de roboflow
    epochs=100,          # Nro de epocas
    imgsz=640,           # Tamaño de las imágenes segun a las preprocesadas
    batch=32,            # Un batch razonable que no comprometa estabilidad
    lr0=0.001,           # Tasa de aprendizaje inicial
    workers=8,           # Hilos de preprocesamiento
    device=0             # Utilizar GPU
)
```

En estos entrenamientos de los modelos YOLOv8s y YOLO11s, los hiperparámetros se seleccionaron cuidadosamente para garantizar un equilibrio entre rendimiento y estabilidad. Se utilizó un tamaño de imagen de 640 píxeles para adaptarse al dataset preprocesado, 100 épocas para asegurar la convergencia del modelo, y un tamaño de lote de 32 para optimizar el uso de la memoria de la GPU sin comprometer la estabilidad. La tasa de aprendizaje inicial de 0.001 se estableció para prevenir divergencias en las etapas iniciales, mientras que 8 hilos de preprocesamiento y el uso de la GPU facilitaron el manejo eficiente del dataset y redujeron significativamente los tiempos de entrenamiento.

Evaluación de Resultados de entrenamiento. En la evaluación de los resultados de los modelos de detección de objetos, es fundamental considerar métricas de precisión y el recall. Según GeeksforGeeks (2024), estas métricas son esenciales para evaluar modelos de clasificación multiclase. Los conceptos clave para su cálculo son:

- **TP (Verdaderos Positivos):** Cantidad de instancias predichas correctamente como pertenecientes a la clase.
- **FP (Falsos Positivos):** Cantidad de instancias predichas incorrectamente como pertenecientes a la clase.
- **FN (Falsos Negativos):** Cantidad de instancias que pertenecen a la clase pero fueron predichas como otra clase.

La Precisión mide la proporción de predicciones correctas entre todas las instancias predichas como positivas para la clase y recall mide la proporción de instancias correctamente predichas como positivas entre todas las instancias reales de la clase. Estas métricas se calculan de la siguiente manera.

$$\text{Precisión} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

En esta investigación el objetivo principal es la detección de motociclistas sin casco por lo que se prioriza la detección de la clase "Sin_Casco":

- **Verdaderos positivos (TP):** Motociclistas sin casco correctamente detectados.
- **Falsos positivos (FP):** Motociclistas clasificados como "sin casco" pero que en realidad usaban casco.

- **Falsos negativos (FN):** Motociclistas sin casco que no fueron detectados.

Las métricas de precisión y recall permiten evaluar el desempeño del modelo en la detección de motociclistas sin casco: una alta precisión refleja un bajo número de falsos positivos, es decir, pocos errores al clasificar incorrectamente a motociclistas con casco como 'sin casco', mientras que un alto recall indica que el modelo detecta la mayoría de los motociclistas sin casco, minimizando los falsos negativos. El F1-score es una métrica utilizada en aprendizaje automático que fusiona la precisión y el recall en un único valor para medir el desempeño de un modelo de clasificación. Resulta especialmente útil cuando se requiere balancear ambos aspectos. Este valor se obtiene a través de la media armónica entre precisión y recall, lo que penaliza considerablemente a los modelos con un desequilibrio entre estas dos métricas (Foqum, 2023).

Una vez finalizado el proceso de entrenamiento, se procedió a evaluar las métricas de los modelos YOLOv8s y YOLO11s utilizando sus respectivos datos de prueba. Esta evaluación permitió analizar el desempeño de los modelos en términos de precisión, recall, mAP (mean Average Precision) y otras métricas clave para medir su efectividad en la detección de motociclistas con y sin casco. Los datos de prueba fueron fundamentales para garantizar una evaluación objetiva, ya que no formaron parte del conjunto de entrenamiento, lo que permite observar cómo los modelos generalizan frente a nuevos datos.

Con el objetivo de analizar de manera completa las métricas relevantes, como la precisión y el recall, se incluirá también el F1 score, que representa el promedio armónico entre estas dos métricas. Los resultados se presentarán en las siguientes tablas.

Tabla 3.3
Métricas de YOLOv8s

Clase	**Precisión**	**Recall**	**F1-score**
Con_Casco	0.947	0.841	0.891
Sin_Casco	0.976	0.984	0.980
Total	0.961	0.912	0.936

Tabla 3.4
Métricas de YOLO11s

Clase	Precisión	Recall	F1-score
Con_Casco	0.943	0.942	0.942
Sin_Casco	0.990	0.928	0.958
Total	0.966	0.935	0.950

Tras analizar los resultados obtenidos de ambos modelos, se concluyó que la evaluación del desempeño se centró principalmente en el recall, ya que el objetivo principal del prototipo era priorizar la detección de motociclistas sin casco y reducir los falsos negativos. En este aspecto, el modelo YOLO11s mostró un mejor rendimiento, alcanzando un recall de 0.935, frente al 0.912 obtenido por YOLOv8s. El F1-score promedio de las dos clases para YOLO11s fue de 0.950, superando el 0.936 registrado por YOLOv8s, lo que confirma su superioridad en este contexto.

3.4.2. Implementación del prototipo del sistema

Hardware empleado. Para la implementación del prototipo del sistema, se utilizó una máquina con las siguientes especificaciones técnicas:

- **Procesador:** Intel Core i7 de 10ª generación.
- **Memoria RAM:** 8 GB.
- **Tarjeta Gráfica:** NVIDIA GeForce GTX 1650 Ti con 4 GB de VRAM.

Es necesario considerar las características del hardware para ejecutar algoritmos de visión computacional de manera eficiente, especialmente aprovechando la capacidad de procesamiento paralelo de la GPU en aplicaciones que lo requieren, como en este caso.

Configuración de entorno. Para evitar conflictos con las librerías instaladas en el sistema principal y garantizar un entorno controlado, se creó un **entorno virtual** utilizando la herramienta `venv`, incluida en Python. Este entorno fue configurado específicamente para

el desarrollo del prototipo, asegurando que todas las dependencias necesarias sean gestionadas de manera independiente.

Primero, se utilizó el siguiente comando para crear el entorno virtual en la carpeta de trabajo:

```
python -m venv yolovenv
```

Una vez creado, se activó con el comando correspondiente en el sistema operativo windows:

```
yolovenv\Scripts\activate
```

Con el entorno activado, se procedió a instalar las librerías necesarias según las especificaciones de la tarjeta gráfica NVIDIA GeForce GTX 1650 Ti, que requiere CUDA 11.8. Los comandos utilizados fueron los siguientes:

- Para instalar PyTorch con soporte para CUDA:

```
pip install torch torchvision torchaudio --index-url

https://download.pytorch.org/whl/cu118
```

- Para instalar YOLO11s:

```
pip install ultralytics
```

Este enfoque permitió evitar conflictos con las librerías del sistema operativo, facilitó la replicabilidad y optimizó el uso del hardware al aprovechar el soporte de CUDA en la GPU.

Implementación de prototipo e integración del modelo. Para el desarrollo del prototipo, se utilizó Python 3.12.5 como lenguaje de programación, aprovechando las

bibliotecas necesarias como OpenCV y YOLO11s. El primer paso fue cargar el modelo YOLO entrenado previamente con nuestro dataset, adaptado específicamente para la detección de motociclistas con y sin casco. Este modelo se cargó utilizando la función load_model, que carga el archivo .pt de YOLO11s optimizado para el uso en detección de objetos en videos.

A continuación, se procedió a seleccionar el archivo de video para su procesamiento. Al abrir el archivo de video con OpenCV, se inició un ciclo donde se leyeron fotogramas del video uno a uno. Para cada fotograma, se aplicó la detección de objetos utilizando el modelo YOLO. El pseudocódigo para esta parte del proceso es el siguiente:

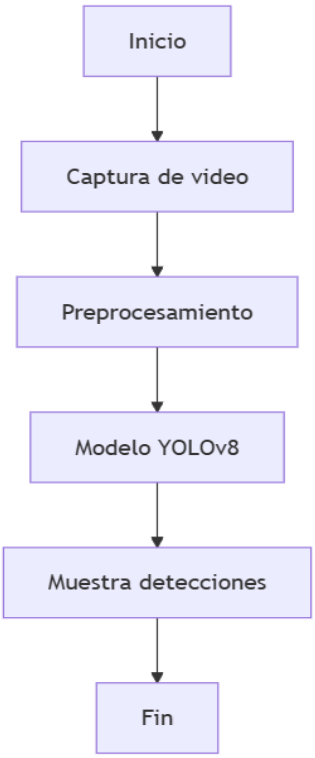
Algorithm 2 Proceso de detección de motociclistas con YOLO11s

```
Cargar el modelo YOLO entrenado "Mi_Modelo_Yolo11s.pt"
Seleccionar archivo de video para procesamiento
Abrir el archivo de video
while el video esté abierto do
    Leer el siguiente fotograma del video
    Convertir el fotograma a formato compatible con el modelo YOLO
    Realizar detección de objetos en el fotograma usando el modelo YOLO
    for cada detección do
        Obtener la clase detectada, las coordenadas del cuadro delimitador y la confianza
        if la confianza es mayor que 0.5 then
            if la clase detectada es Motociclista then
                Dibujar el cuadro delimitador alrededor del motociclista
                Registrar la información de la detección (coordenadas, clase, confianza)
            else if la clase detectada es Con_Casco then
                Dibujar el cuadro delimitador alrededor del casco
                Registrar la información de la detección (coordenadas, clase, confianza)
            else if la clase detectada es Sin_Casco then
                Dibujar el cuadro delimitador alrededor de la cabeza del motociclista
                Registrar la información de la detección (coordenadas, clase, confianza)
            end if
        end if
    end for
    Mostrar el fotograma con los cuadros delimitadores y la información en pantalla
    if se presiona la tecla 'q' then
        Salir del bucle de procesamiento
    end if
end while
Cerrar el archivo de video
Liberar los recursos de la captura de video
```

El pseudocódigo describe la primera parte de un prototipo para la detección de moto-

ciclistas en videos utilizando YOLO11s, enfocado en identificar si llevan casco o no. Detalla desde la carga del modelo hasta la visualización de los resultados en tiempo real. En el siguiente gráfico se presenta el diagrama de flujo.

Figura 3.18
Diagrama de flujo del prototipo con integración del modelo



En esta sección, se presentarán los resultados obtenidos con la primera etapa del prototipo desarrollado para la detección de motociclistas. A través de este prototipo, hemos utilizado el modelo YOLO11s para identificar motociclistas en videos, evaluando su uso de casco de seguridad. En la siguiente imagen, se puede observar claramente la detección de un motociclista que cumple con las normas de seguridad, es decir, que lleva puesto el casco correspondiente.

Figura 3.19

Resultado de detección de motociclista con casco

Por otro lado, en la siguiente imagen se muestra el resultado de la detección de un motociclista que no lleva casco de seguridad, lo cual representa una infracción a las normativas de seguridad vial. Esta fase del prototipo es igualmente importante, ya que permite evaluar la capacidad del modelo para identificar situaciones de riesgo, lo que podría tener un impacto directo en la mejora de la seguridad vial y en la prevención de accidentes. Ambos resultados demuestran el progreso logrado hasta el momento y la efectividad del sistema para clasificar correctamente a los motociclistas en función de su cumplimiento con las normativas de seguridad.

Figura 3.20
Resultado de detección de motociclista sin casco

3.4.3. Implementación de la funcionalidad de detección de velocidad

Instalación de requerimientos. Para incorporar la funcionalidad de detección de velocidad en el sistema, se integró el módulo `DeepSort` en el entorno virtual, tal como se había detallado previamente en el punto 3.3.2. Esta integración permitió el seguimiento y la asignación de una identidad única a cada motociclista detectado en el video, lo que facilita la medición de su velocidad en tiempo real.

El proceso de instalación implicó la configuración de las dependencias necesarias para que el módulo `DeepSort` pudiera operar correctamente en el entorno de trabajo. Esto incluyó la ejecución del siguiente comando para instalar el módulo en el entorno virtual:

```
pip install deep-sort-realtime
```

Integración al prototipo. La integración de la funcionalidad de detección de velocidad en el prototipo permite que el prototipo sea mas completo, permitiendo registrar la velocidad de motociclistas ya que la velocidad es uno de los factores que determina la gravedad de lesiones en la cabeza. Esta funcionalidad utilizará el algoritmo DeepSort en combinación con el modelo entrenado con YOLO11s, permitiendo calcular la velocidad de las motocicletas detectadas en tiempo real. La metodología implementará la detección inicial con YOLO11s, que proporcionará cuadros delimitadores para los objetos de interés, mientras que DeepSort se encargará del seguimiento de las trayectorias de las motocicletas, asignándoles identificadores únicos para rastrear sus movimientos a lo largo de la escena, a continuación se mostrara el seudocódigo.

Algorithm 3 Seguimiento de objetos y cálculo de velocidad con YOLO11s y DeepSort

function CALCULAR_VELOCIDAD(*bbox, id_rastreo, fps*)
    conversion_pixeles_a_metros ← 0,06 ▷ Factor para convertir píxeles a metros.
    Calcular centro del cuadro delimitador (cx, cy).
    if *id_rastreo* en posiciones anteriores **then**
        Obtener posición previa (cx_prev, cy_prev) del *id_rastreo*.
        $distancia\_pixeles \leftarrow \sqrt{(cx - cx\_prev)^2 + (cy - cy\_prev)^2}$
        $distancia\_metros \leftarrow distancia\_pixeles \times conversion\_pixeles\_a\_metros$
        $velocidad \leftarrow distancia\_metros \times fps$ ▷ Velocidad en metros/segundo.
    else
        $velocidad \leftarrow 0$ ▷ Velocidad inicial cuando no hay posición previa.
    end if
    Guardar posición actual (cx, cy) como posición previa para el *id_rastreo*.
    Suavizar velocidad:
    Agregar *velocidad* a una lista de historial (*historial_velocidades*).
    if *historial_velocidades* tiene más de 5 elementos **then**
        Eliminar el elemento más antiguo de *historial_velocidades*.
    end if
    velocidad_suavizada ← Promedio(*historial_velocidades*)
    return *velocidad_suavizada*
end function
function RASTREAR_Y_DETECTAR(*detecciones, imagen, rastreador, fps*)
    Actualizar rastreador con *detecciones*.
    for cada objeto rastreado en rastreador **do**
        if el rastreo está confirmado **then**
            Obtener *id_rastreo*, cuadro delimitador y clase.
            if clase == "Motociclista" **then**
                Calcular velocidad con CALCULAR_VELOCIDAD.
                Dibujar cuadro y mostrar velocidad.
            else if clase == Çon_Casco" **then**
                Dibujar cuadro y mostrar confianza.
            else if clase == "Sin_Casco" **then**
                Dibujar cuadro y mostrar confianza.
            end if
        end if
    end for
    return Imagen procesada.
end function

En el seudocódigo, se destacan dos funciones clave que desempeñaron un papel esencial en el seguimiento de objetos y en el cálculo de la velocidad. Estas funciones fueron fundamentales para el desarrollo del sistema, ya que permiten tanto rastrear los objetos a lo largo del video como calcular su velocidad en tiempo real. Un aspecto crucial dentro de la función de cálculo de velocidad es la conversión de píxeles a metros, un factor de conversión que establece la relación entre el tamaño físico real de un objeto, como una motocicleta, y el número de píxeles que ocupa en el video. Este factor es esencial para transformar las medidas en el espacio digital a unidades físicas. La fórmula para calcular este *conversion_pixeles_a_metros* es la siguiente:

$$conversion\_pixeles\_a\_metros = \frac{longitud\_real\_en\_metros}{longitud\_en\_pixeles}$$

Este valor de conversión se utiliza para transformar las distancias medidas en píxeles a distancias en metros, permitiendo obtener resultados más precisos y acordes a las medidas reales.

Para calcular la velocidad de la motocicleta, se hace necesario determinar el centro de los cuadros delimitadores que representan a cada objeto (motocicleta) en el video. Los centros de estos cuadros se utilizan para calcular la distancia recorrida entre dos cuadros consecutivos. La fórmula para calcular la distancia entre dos puntos en el espacio bidimensional (en píxeles) es la siguiente:

$$distancia\_en\_pixeles = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Donde (x_1, y_1) y (x_2, y_2) representan las coordenadas de los centros de los cuadros delimitadores en dos fotogramas consecutivos. Esta distancia en píxeles se convierte a metros utilizando el *pixel_to_meter_ratio*:

$$distancia\_en\_metros = distancia\_en\_pixeles \times conversion\_pixeles\_a\_metros$$

Luego, para obtener la velocidad en metros por segundo (m/s), se toma en cuenta la tasa de fotogramas por segundo (fps) del video, que indica cuántos cuadros se procesan por segundo. La fórmula para calcular la velocidad es:

$$velocidad = distancia\_en\_metros \times fps$$

La velocidad final promedio se calcula utilizando el promedio aritmético de las últimas 5 velocidades registradas, como se muestra a continuación:

$$v_{\text{prom}} = \frac{v_1 + v_2 + v_3 + v_4 + v_5}{5}$$

Donde v_1, v_2, v_3, v_4, v_5 son las velocidades de los últimos 5 fotogramas procesados. Este cálculo ayuda a suavizar las fluctuaciones en la velocidad, proporcionando una estimación más estable y precisa del movimiento del objeto.

Una vez implementada esta funcionalidad, se realizaron pruebas para evaluar la detección y el cálculo de la velocidad. A continuación, se presentan algunos de los resultados obtenidos en dichas pruebas.

Figura 3.21
Resultados de detección y velocidad

Con casco

Sin casco

En las imágenes presentadas, se pueden observar resultados que reflejan con alta precisión la realidad. La primera captura fue tomada cuando el motociclista se encontraba detenido en un semáforo, mostrando una velocidad de 0.01 m/s. Por otro lado, la segunda captura corresponde a un motociclista en movimiento, con una velocidad de 36.16 m/s. Es importante señalar que estas pruebas se realizaron utilizando videos con frecuencias de 30 y 60 cuadros por segundo (fps).

Capítulo 4

Análisis y discusión de resultados

4.1. Análisis de resultados respecto a los objetivos

- En la construcción del dataset, las imágenes fueron obtenidas de diferentes cámaras, lo que resultó en tamaños variables. Durante la primera fase, el dataset inicial estaba compuesto por 112 imágenes, lo que generó una sobrecarga de recursos en el entrenamiento debido a la diversidad en los tamaños y orientaciones de las imágenes. Para resolver este problema, en la siguiente fase se incrementó el dataset a 1560 imágenes y se aplicó un preprocesamiento que incluyó la corrección de la orientación de las imágenes y su redimensionamiento a 640 $\times$ 640 píxeles. Posteriormente, para mejorar aún más las condiciones del modelo, se llevó a cabo un aumento de datos mediante técnicas como rotación, desenfoque y ruido, alcanzando un total de 5936 imágenes. Este proceso permitió optimizar el dataset para un entrenamiento más eficiente y robusto.

- Para la detección de objetos, se evaluaron Faster R-CNN y YOLO. Se decidió emplear YOLO debido a su capacidad para realizar detecciones en tiempo real, mientras que Faster R-CNN se destaca por su mayor precisión. En la construcción del prototipo, es fundamental considerar tanto la eficiencia computacional como la capacidad

de procesamiento en tiempo real, aspectos en los que YOLO resulta más adecuado. YOLO cuenta con varias variantes mejoradas desarrolladas por Ultralytics. Para esta investigación, se evaluaron las variantes YOLOv8s y YOLO11s, seleccionando la versión "s"(small) debido a su costo computacional moderado en comparación con las versiones "m"(medium), "l"(large) y "x"(extra-large). En cuanto a la evaluación del rendimiento, se busca minimizar tanto los falsos positivos como los falsos negativos, manteniendo un equilibrio adecuado entre ambos, lo cual se mide a través del F1-score. En esta investigación, la clase *Sin_Casco* es particularmente relevante, ya que reducir los falsos negativos en esta categoría es crucial para prevenir accidentes. Según la tabla 3.3, el modelo YOLOv8s alcanzó un F1-score de 0.936. Por otro lado, el modelo YOLO11s logró un F1-score de 0.950, como se muestra en la tabla 3.4, evidenciando una ventaja del modelo YOLO11s frente a YOLOv8s.

- En cuanto al seguimiento de objetos, se utilizó DeepSORT, lo que contribuyó a mantener el seguimiento de los motociclistas. Un factor clave fue la conversión de píxeles a metros, que relaciona la distancia real con la ocupada por el objeto en píxeles en la imagen. La distancia entre fotogramas se calcula a partir de los centros de los cuadros delimitadores, y se considera la frecuencia de cuadros (fps) para calcular la velocidad, como se muestra en el pseudocódigo de la sección 3.4.3. Las pruebas realizadas, que se presentan en la figura 3.21, arrojaron resultados que son muy similares a la realidad, lo que demuestra un buen desempeño del sistema. En la primera captura, con el motociclista detenido en un semáforo, se registró una velocidad de 0.01 m/s, y en la segunda, con el motociclista en movimiento, una velocidad de 36.16 m/s.

- El prototipo fue desarrollado en una máquina convencional equipada con un procesador Intel i7 de décima generación y una tarjeta gráfica NVIDIA GeForce GTX 1650 con 4 GB de VRAM. Para la implementación, se configuró un entorno virtual utilizando venv

en el sistema operativo Windows 11, en el cual se instalaron todas las dependencias necesarias. Posteriormente, se utilizó el modelo previamente entrenado para realizar la detección en tiempo real de motociclistas con y sin casco de seguridad, así como el cálculo de su velocidad. De esta manera, se logró construir un prototipo funcional.

4.2. Discusión de resultados respecto a los antecedentes

- En el trabajo presentado por (Wei et al., 2021), han propuesto un conjunto de datos enfocados estrictamente en la detección de cascos, capturados de diversas escenas de trafico en china, condiciones climáticas variadas oclusión y iluminación variable, donde alcanzaron una puntuación F1 de 0.927. En la presente investigación se construyo un dataset de diferentes escenarios y para asegurar la diversidad de datos se hizo un aumento de datos con el que se llego a un f1 score de 0.950 con YOLO11s.

- En el trabajo presentado por Duong et al. (2023), se enfatiza la importancia de la automatización en la detección del uso del casco de motociclistas mediante videovigilancia, con el fin de mejorar la eficacia de las iniciativas educativas y el cumplimiento de la ley. Este sistema permite identificar de manera individual el uso del casco tanto del conductor como del pasajero. En la presente investigación, también se destaca la relevancia de automatizar la seguridad vial y la detección del uso del casco en ambos, conductor y pasajero. Además, se incluye la medición de la velocidad a la que circula el vehículo, ya que esto también forma parte de las normativas para motociclistas. Este enfoque adicional es uno de los aspectos que distingue y enriquece el prototipo propuesto.

- En el trabajo presentado por Aman et al. (2022), se abordan las violaciones de tránsito

relacionadas con el uso del casco, con un enfoque en mejorar la gestión de oclusiones, especialmente al tratar eficazmente las obstrucciones causadas en la detección. El objetivo principal de su investigación fue minimizar los falsos negativos y falsos positivos en entornos de alta densidad de tráfico. En la presente investigación, se abordó este desafío utilizando DeepSort, una técnica encargada de realizar el seguimiento de los motociclistas. Este método demostró ser eficaz no solo para gestionar las oclusiones, sino también para calcular la velocidad, lo cual fue aprovechado en el desarrollo del prototipo del sistema.

- En el trabajo presentado por Gonzales del Valle Romero y Neyra Espinoza (2022), se abordaron las redes neuronales artificiales convolucionales YOLOv5 y Faster R-CNN, empleando transfer learning. Se utilizó un dataset con 460 imágenes para el conteo de vehículos. En esta investigación, se seleccionó YOLOv5 como la red principal, demostrando un mejor desempeño frente a Faster R-CNN. Estos sistemas fueron evaluados en tiempo real. En la presente investigación, también se discutieron estos modelos de detección de objetos, YOLO y Faster R-CNN, concluyendo que YOLO presenta un mejor desempeño en sistemas de tiempo real.

Conclusiones

1. El prototipo implementado demuestra su efectividad al identificar en tiempo real a motociclistas que infringen las normas de tránsito al circular sin casco de seguridad. mediante técnicas de visión computacional basadas en el modelo YOLO.

2. Se construyó un dataset de 1,560 imágenes de motociclistas con y sin casco, obtenidas a partir de grabaciones de cámaras de videovigilancia y capturas con un teléfono móvil en diversos escenarios y ángulos. Las imágenes fueron preprocesadas para estandarizar su tamaño debido a sus resoluciones variables. Además, se aplicaron técnicas de aumento de datos, como rotación, adición de ruido y desenfoque, para fortalecer su diversidad y obtener un dataset robusto.

3. Se evaluaron los modelos de visión computacional Faster R-CNN y YOLO, eligiendo YOLO por su capacidad para realizar detecciones en tiempo real. Se probaron las versiones YOLOv8s y YOLOv11s, siendo YOLO11s el que mostró un mejor rendimiento en la detección de motociclistas sin casco, alcanzando un F1-score de 0.950. Para la medición de velocidad, se utilizó DeepSort para el seguimiento de los motociclistas. La velocidad se calculó mediante un factor de conversión de píxeles a metros, considerando también los fotogramas por segundo (FPS) del video procesado.

Recomendaciones

- El dataset utilizado en esta investigación presenta limitaciones en la calidad de las imágenes, ya que muchas son de baja resolución. A pesar de estas deficiencias, los resultados fueron satisfactorios, lo que resalta la capacidad del modelo para manejar datos imperfectos. Para futuros trabajos, se recomienda mejorar la calidad y diversidad del dataset.

- Se recomienda integrar al prototipo un sistema de detección y reconocimiento de placas vehiculares, lo que permitiría identificar de manera precisa a los motociclistas infractores. Esta funcionalidad fortalecería el sistema al vincular de forma directa las infracciones con los responsables, facilitando la generación de evidencias claras y automatizadas para la aplicación de sanciones. La inclusión de esta tecnología aumentaría la efectividad del sistema en la gestión de la seguridad vial, contribuyendo a la mejora en el control de las normativas de tránsito y la reducción de accidentes relacionados con motociclistas que no cumplen con las normas de seguridad.

- Realizar ajustes mediante la incorporación de más datos, algunos de los cuales ya han sido obtenidos de las cámaras de videovigilancia de tránsito de la ciudad de Cusco, e integrar el prototipo con plataformas de monitoreo utilizadas por las autoridades locales de tránsito, lo que permitiría la generación de alertas automáticas y reportes en tiempo real.

- En esta investigación, la detección de velocidad de motociclistas aún no ha alcanzado su óptimo desempeño, ya que depende en gran medida de factores como la ubicación de las cámaras y otras variables que pueden ser mejoradas. Sin embargo, a pesar de estas limitaciones, se lograron resultados satisfactorios utilizando DeepSort.

- El hardware utilizado en esta investigación presenta latencia al ejecutar el modelo de forma local. Se recomienda emplear un equipo con GPU superior a 4 GB de VRAM y una memoria RAM mayor a 8 GB.

Bibliografía

Acharya, A. (2024). Yolo object detection explained: Evolution, algorithm, and applications. url: https://encord.com/blog/yolo-object-detection-guide/.

Ahorra Seguros (2020). Seguridad vial en peru. url: https://ahorraseguros.pe/guias/seguridad-vial-en-peru/. [recuperado el 24-03-2024].

Aman, G., Dev, A., Anbumani, S., C.V., J., Ravi, S., and Rohit, S. (2022). Detecting, tracking and counting motorcycle rider traffic violations on unconstrained roads. `https://doi.org/10.48550/arXiv.2204.08364`.

Ander-Egg, E. (2011). *Aprender a Investigar*. Editorial Brujas, Argentina.

Asociación Movemos (2023). Las motos ¿un problema o una soluciÓn para la movilidad? url: hhttps://movemos.pe/uploads/Reporte-sobre-motocicletas-Asociacion-MOVEMOS.pptx.pdf.

ATU (2024). Atu ya cuenta con 130 cámaras de videovigilancia para realizar fiscalización electrónica al transporte informal en 10 distritos. url: https://www.gob.pe/institucion/atu/noticias/956051-atu-ya-cuenta-con-130-camaras-de-videovigilancia-para-realizar-fiscalizacion-electronica-al-transporte-informal-en-10-distritos.

Bewley, A., Ge, Z., Ott, L., Ramos, F., and Upcroft, B. (2016). Simple online and realtime tracking. https://ar5iv.labs.arxiv.org/html/1602.00763.

Chojecki, C. (2019). What is deep learning? url: https://medium.com/swlh/what-is-deep-learning-b2cd80911cbc.

DataScientest (2022). ¿qué es el transfer learning? url: https://datascientest.com/es/que-es-el-transfer-learning.

Duong, T., Long, P., Hyung-Joon, J., Huy-Hung, N., Hyung-Min, J., Tai, T., and Jae, J. (2023). Robust automatic motorcycle helmet violation detection for an intelligent transportation system. https://openaccess.thecvf.com/content/CVPR2023W/AICity/papers/Tran_Robust_Autom.

El Peruano (2024). ¿eres motociclista? conoce cuáles son las normas de tránsito y evita las multas. url: https://www.elperuano.pe/noticia/234855-eres-motociclista-conoce-cuales-son-las-normas-de-transito-y-evita-las-multas.

Encord (2024). Yolo object detection explained: Evolution, algorithm, and applications. https://encord.com/blog/yolo-object-detection-guide/. Accessed: 2024-12-10.

Foqum (2023). F1: Métrica en aprendizaje automático. Accedido el 12 de diciembre de 2024.

GeeksforGeeks (2024). Calculating precision and recall for multiclass classification using confusion matrix. url: https://www.geeksforgeeks.org/calculating-precision-and-recall-for-multiclass-classification-using-confusion-matrix/. Accessed: 2024-12-12.

Gelvez, A. (2019). Redes neuronales convolucionales y redes neuronales recurrentes en la transcripción automática. *ResearchGate, 2. doi: 10.13140/RG.2.2.10855.39843.*

Gogul, I. and Sathiesh Kumar, V. (2017). Flower species recognition system using convolution neural networks and transfer learning. https://www.researchgate.net/

publication/320746968_Flower_species_recognition_system_using_convolution_neural_networks_and_transfer_learning.

Goldman, A. M. (2020). Seguridad vial. impacto económico, sanitario y social. url: https://economiaygestiondelasalud.finance.blog/2020/02/23/seguridad-vial-impacto-economico-sanitario-y-social/.

Gonzales del Valle Romero, G. F. and Neyra Espinoza, W. J. (2022). Implementación de una red neuronal artificial convolucional para la detección y conteo de vehículos en una sección del estacionamiento de la universidad ricardo palma. url: https://hdl.handle.net/20.500.14138/5957.

IBM (2019). ¿qué es la visión por computadora? url: https://www.ibm.com/topics/computer-vision.

IBM (2021). ¿qué es machine learning? url: https://www.ibm.com/es-es/topics/machine-learning.

IBM (2024). ¿qué es la segmentación de imágenes? url: https://www.ibm.com/es-es/topics/image-segmentation: :text=La

Joiya, F. (2022). Object detection: Yolo vs faster r-cnn. https://www.irjmets.com/uploadedfiles/paper//issue_9_september_2022/30226/final/fin_irjmets1664212182.pdf.

Kouidri, A. (2023). Mastering deep sort: The future of object tracking explained. https://www.ikomia.ai/blog/deep-sort-object-tracking-guide#what-is-deep-sort.

Manning Publications (2020). The computer vision pipeline. url: https://manningbooks.medium.com/the-computer-vision-pipeline-part-4-feature-extraction-6343ef063588.

Microsoft (2021). ¿qué es la visión artificial? url: https://azure.microsoft.com/es-es/resources/cloud-computing-dictionary/what-is-computer-visionclasificaci

Muhammad, M. (2022). What is deepsort and how to implement yolov7 object tracking using deepsort. url: https://medium.com/@m.moinfaisal/what-is-object-tracking-and-why-deepsort-and-how-to-implement-yolov7-object-tracking-using-deepsort-f0c952f89b06.

NHTSA (2023). Casi un tercio de las muertes de tráfico son en choques relacionados con la velocidad. url: https://www.nhtsa.gov/es/press-releases/aumentan-muertes-por-exceso-de-velocidad.

Observatorio Nacional de Seguridad Vial (2023). Boletín estadístico de siniestralidad vial, primer semestre 2023. url: https://www.onsv.gob.pe/post/boletin-estadistico-de-siniestralidad-vial-primer-semestre-2023/.

OMS (2023). Global status report on road safety 2023. url: https://iris.who.int/bitstream/handle/10665/375016/9789240086517-eng.pdf?sequence=1isAllowed=y.

PAHO (2020). La motocicleta en el transito en las américas. url: https://www.paho.org/es/documentos/hoja-informativa-motocicleta-transito-americas.

Perez Silva, E. W. (2022). Reconocimiento de placas vehiculares mediante vision computacional para mejorar el acceso a un parqueadero. url: https://orcid.org/0000-0003-3345-8774.

Purestorage (2024). What is data preprocessing for machine learning? `https://www.purestorage.com/knowledge/what-is-data-preprocessing.html#:~:text=Data%20preprocessing%20for%20machine%20learning%20(ML)%20refers%20to%20the%20preparation,and%20accuracy%20of%20the%20models.` Accessed: 2024-11-05.

Roboflow (2023). Datasets. https://docs.roboflow.com/datasets/. Accessed: 2024-11-05.

Scrum.org (2022). What is scrum? https://www.scrum.org/resources/what-scrum-module. Accedido el 15 de Octubre de 2024.

Sharma, A. and Kumar, V. (2024). Comparative performance of yolov8, yolov9, yolov10, yolov11 and faster r-cnn models for detection of multiple weed species. *ScienceDirect*.

Sojasingarayar, A. (2022). Faster r-cnn vs yolo vs ssd — object detection algorithms. https://medium.com/ibm-data-ai/faster-r-cnn-vs-yolo-vs-ssd-object-detection-algorithms-18badb0e02dc.

Sura (2018). Recomendaciones para ser un motociclistas responsable. url: https://segurossura.com/co/blog/movilidad/recomendaciones-para-ser-un-motociclista-responsable/.

SUTRAN (2014). Aprueban texto Único ordenado del reglamento nacional de trÁnsito - cÓdigo de trÁnsito decreto supremo nº 016-2009-mtc. url: https://www.sutran.gob.pe/wp-content/uploads/2015/08/D_$NRO_0$16 − 2009 − $MTC_AL_0$5,05,14.*pdf*.

TagX (2023). What is good quality training dataset for machine learning. url: https://medium.com/@tagx20/what-is-good-quality-training-dataset-for-machine-learning-efa2473c1ede.

Wei, J., Shiquan, X., Zhen, L., Yang, Z., Hai, M., ShuJie, L., and Ye, Y. (2021). Real-time automatic helmet detection of motorcyclists in urban traffic using improved yolov5 detector. url: https://doi.org/10.1049/ipr2.12295.

Wong Leon, E. R. adn Coral Ygnacio, M. A. (2022). Una revisión sistemáti-

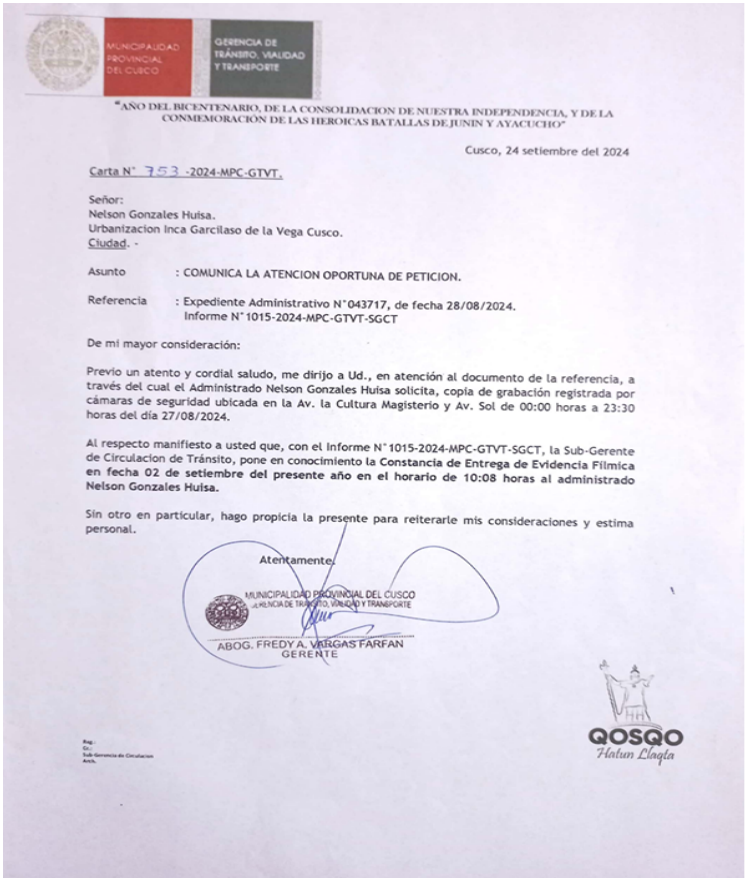
ca de literatura sobre implementación de sistemas de control de trafico. url: https://revistas.ulima.edu.pe/index.php/Interfases/article/view/6779/6803.

Capítulo 5

Anexos

5.1. Documentos empleados para el acceso a grabaciones de cámaras de video vigilancia

Figura 5.1
Municipalidad de Cusco



MUNICIPALIDAD PROVINCIAL DEL CUSCO
GERENCIA DE TRÁNSITO, VIALIDAD Y TRANSPORTE

"AÑO DEL BICENTENARIO, DE LA CONSOLIDACION DE NUESTRA INDEPENDENCIA, Y DE LA CONMEMORACIÓN DE LAS HEROICAS BATALLAS DE JUNIN Y AYACUCHO"

Cusco, 24 setiembre del 2024

Carta N° 753 -2024-MPC-GTVT.

Señor:
Nelson Gonzales Huisa.
Urbanizacion Inca Garcilaso de la Vega Cusco.
Ciudad. -

Asunto : COMUNICA LA ATENCION OPORTUNA DE PETICION.

Referencia : Expediente Administrativo N°043717, de fecha 28/08/2024.
Informe N°1015-2024-MPC-GTVT-SGCT

De mi mayor consideración:

Previo un atento y cordial saludo, me dirijo a Ud., en atención al documento de la referencia, a través del cual el Administrado Nelson Gonzales Huisa solicita, copia de grabación registrada por cámaras de seguridad ubicada en la Av. la Cultura Magisterio y Av. Sol de 00:00 horas a 23:30 horas del día 27/08/2024.

Al respecto manifiesto a usted que, con el informe N°1015-2024-MPC-GTVT-SGCT, la Sub-Gerente de Circulacion de Tránsito, pone en conocimiento la **Constancia de Entrega de Evidencia Filmica en fecha 02 de setiembre del presente año en el horario de 10:08 horas al administrado Nelson Gonzales Huisa.**

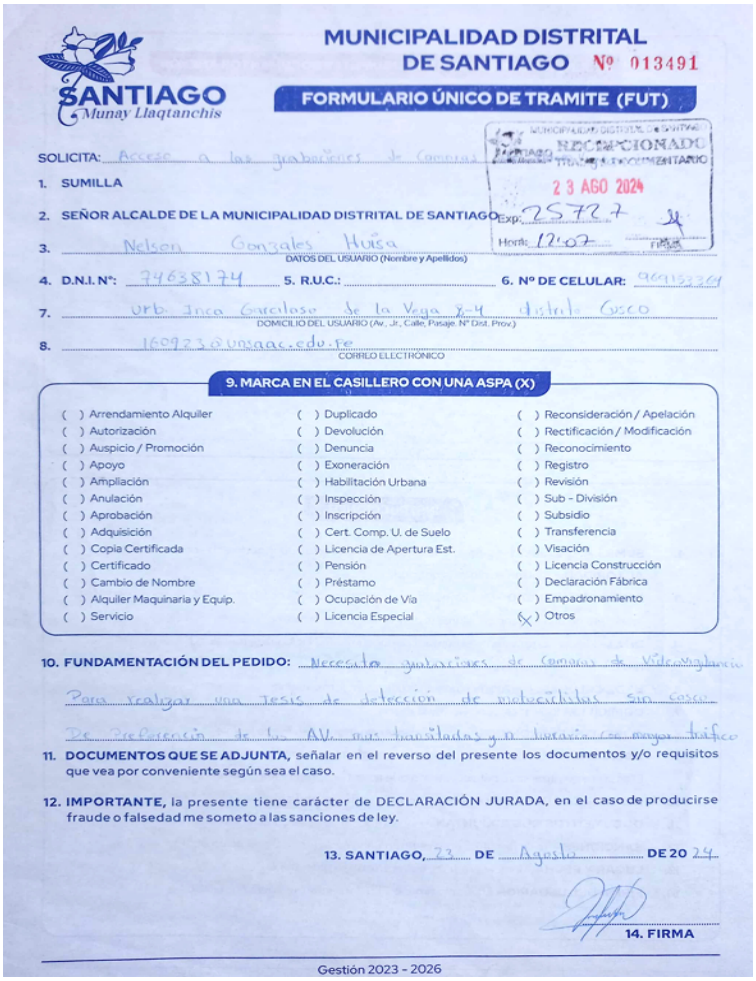
Sin otro en particular, hago propicia la presente para reiterarle mis consideraciones y estima personal.

Atentamente,

MUNICIPALIDAD PROVINCIAL DEL CUSCO
GERENCIA DE TRÁNSITO, VIALIDAD Y TRANSPORTE
ABOG. FREDY A. VARGAS FARFAN
GERENTE

QOSQO
Hatun Llaqta

Figura 5.2
Municipalidad de Santiago



SANTIAGO
Munay Llaqtanchis

MUNICIPALIDAD DISTRITAL DE SANTIAGO Nº 013491

FORMULARIO ÚNICO DE TRAMITE (FUT)

RECEPCIONADO
23 AGO 2024
Exp: 25727
Hora: 12:07

SOLICITA: Acceso a las grabaciones de Camaras

1. SUMILLA
2. SEÑOR ALCALDE DE LA MUNICIPALIDAD DISTRITAL DE SANTIAGO
3. Nelson Gonzales Huisa
   DATOS DEL USUARIO (Nombre y Apellidos)
4. D.N.I. N°: 74638174 5. R.U.C.: 6. Nº DE CELULAR: 960153364
7. Urb. Inca Garcilaso de la Vega K-4 distrito Cusco
   DOMICILIO DEL USUARIO (Av., Jr., Calle, Pasaje, N° Dist. Prov.)
8. 160923@unsaac.edu.pe
   CORREO ELECTRONICO

9. MARCA EN EL CASILLERO CON UNA ASPA (X)

( ) Arrendamiento Alquiler	( ) Duplicado	( ) Reconsideración / Apelación
( ) Autorización	( ) Devolución	( ) Rectificación / Modificación
( ) Auspicio / Promoción	( ) Denuncia	( ) Reconocimiento
( ) Apoyo	( ) Exoneración	( ) Registro
( ) Ampliación	( ) Habilitación Urbana	( ) Revisión
( ) Anulación	( ) Inspección	( ) Sub - División
( ) Aprobación	( ) Inscripción	( ) Subsidio
( ) Adquisición	( ) Cert. Comp. U. de Suelo	( ) Transferencia
( ) Copia Certificada	( ) Licencia de Apertura Est.	( ) Visación
( ) Certificado	( ) Pensión	( ) Licencia Construcción
( ) Cambio de Nombre	( ) Préstamo	( ) Declaración Fábrica
( ) Alquiler Maquinaria y Equip.	( ) Ocupación de Vía	( ) Empadronamiento
( ) Servicio	( ) Licencia Especial	(X) Otros

10. FUNDAMENTACIÓN DEL PEDIDO: Necesito grabaciones de camaras de videovigilancia
Para realizar una tesis de detección de vehículos sin casco
De preferencia de la AV. mas transitadas y a horarios con mayor trafico

11. **DOCUMENTOS QUE SE ADJUNTA,** señalar en el reverso del presente los documentos y/o requisitos que vea por conveniente según sea el caso.

12. **IMPORTANTE,** la presente tiene carácter de DECLARACIÓN JURADA, en el caso de producirse fraude o falsedad me someto a las sanciones de ley.

13. SANTIAGO, 23 DE Agosto DE 20 24

14. FIRMA

Gestión 2023 - 2026