UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Diseño e implementación de sistema de asistencia de manejo de un vehículo terrestre a través de visión por computadora

Trabajo de graduación en modalidad de Trabajo Profesional presentado por
Marco Antonio Jurado Velasquez
para optar al grado académico de Licenciado en Ingeniería en Ciencia de la Computación y Tecnologías de la Información

Guatemala
2024

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Diseño e implementación de sistema de asistencia de manejo de un vehículo terrestre a través de visión por computadora

Trabajo de graduación en modalidad de Trabajo Profesional presentado por
Marco Antonio Jurado Velasquez
para optar al grado académico de Licenciado en Ingeniería en Ciencia de la Computación y Tecnologías de la Información

Guatemala
2024

Vo.Bo.:

(f) _____

MSc. Alan Gerardo Reyes Figueroa

Tribunal Examinador:

(f) _____

MSc. Alan Gerardo Reyes Figueroa

(f) _____

Lic. Marcos Antonio Gutiérrez Suárez

(f) _____

Ing. Eddy Omar Castro Jauregui

Fecha de aprobación: Guatemala, 06 de diciembre de 2024.

Prefacio

La tecnología de asistencia de manejo es un campo en rápida evolución, con aplicaciones que pueden salvar vidas y mejorar la experiencia de conducción. Desde el inicio de mi carrera en Ciencias de la Computación y Tecnologías de la Información (TI), me ha fascinado la idea de cómo la visión por computadora puede integrarse en sistemas vehiculares para mejorar la seguridad y eficiencia en las carreteras. Este proyecto de graduación representa la culminación de años de estudio y la oportunidad de aplicar mis conocimientos en un área que considero vital para el futuro del transporte en Guatemala.

A medida que la tecnología continúa avanzando, espero que este proyecto pueda servir como una base para futuros desarrollos en sistemas de asistencia vehicular. Estoy convencido de que la visión por computadora tiene un papel crucial en la creación de vehículos más seguros y eficientes.

Agradecimientos

Este proyecto de graduación no habría sido posible primeramente sin el apoyo de Dios, quien me ha dado la bendición de poder tener la educación y me ha guiado todo el camino.

No puedo dejar de agradecer a mi familia, especialmente a mis padres, por su amor incondicional y apoyo constante a lo largo de mi vida. Sin su aliento y sacrificio, este logro no habría sido posible. A mis hermanos, quienes siempre han estado a mi lado brindándome su apoyo y consejo, y a la persona que ha sido mi mayor apoyo emocional, por su paciencia, su amor incondicional, comprensión y por ser una fuente constante de motivación.

También quiero agradecer a mis profesores del Departamento de Ciencias de la Computación y TI de la Universidad del Valle de Guatemala, quienes me brindaron los conocimientos y herramientas necesarios para llevar a cabo este trabajo. Su dedicación a la enseñanza ha sido una fuente constante de inspiración.

Índice

Lista de figuras

Lista de cuadros

Resumen

El sistema de asistencia de manejo de un vehículo terrestre desarrollado en este proyecto emplea visión por computadora y análisis de datos de sensores para detectar elementos críticos en carreteras, ofreciendo alertas en tiempo real. Basado en el modelo YOLOv8, el sistema identifica y clasifica señales de tránsito y otros objetos relevantes con alta precisión, logrando una tasa de precisión de hasta 91.32 %. Adicionalmente, se integraron datos simulados de sensores de velocidad y aceleración, lo que permite ajustar las recomendaciones de seguridad según la dinámica de conducción. Esta capacidad de ajuste contribuye a una experiencia de conducción más segura y eficiente, ya que las alertas se adaptan a las condiciones del entorno. El sistema fue evaluado mediante métricas de rendimiento, demostrando tiempos de respuesta promedio de 94.46 ms, lo cual asegura una reacción rápida y oportuna. Además, se configuró un conjunto de alertas que prioriza notificaciones críticas sin generar distracciones excesivas para el conductor. Esto refuerza la efectividad del sistema al ofrecer información relevante en el momento adecuado, mejorando la seguridad vial.

CAPÍTULO 1

Introducción

La visión por computadora es una disciplina dentro de la inteligencia artificial que brinda a una máquina la capacidad de interpretar y entender el mundo visual, procesando imágenes para extraer información relevante (Rezaei y Klette, 2017). Este campo es fundamental en el desarrollo de sistemas avanzados de asistencia al conductor, que utilizan esta tecnología para identificar elementos clave en el entorno vial, mejorando significativamente la seguridad y la experiencia de conducción (Horgan, Hughes, McDonald, y Yogamani, 2015). Al aplicar algoritmos de detección y clasificación de objetos, estos sistemas pueden reconocer y responder a situaciones complejas con condiciones de iluminación, clima y el entorno, asegurando una conducción más segura y eficiente (Velez y Otaegui, 2015).

El proyecto finalizado representará un sistema de asistencia al conductor, diseñado para mejorar la seguridad y la experiencia de conducción a través de tecnologías avanzadas de visión por computadora. Este sistema estará capacitado para detectar y clasificar elementos críticos en calles y carreteras, como señales de tránsito, utilizando un modelo de *machine learning* de vanguardia, específicamente adaptado y entrenado para esta tarea. Además, integrará variables con información de velocidad y aceleración, proporcionando una respuesta comprensiva y ajustada a las condiciones dinámicas de la conducción.

El sistema desarrollado en este proyecto integra modelos de aprendizaje automático y procesamiento de datos para detectar y analizar señales de tránsito, complementadas con datos simulados de sensores como la aceleración y la velocidad de un vehículo en un entorno simulado. Todo el procesamiento se lleva a cabo en el *backend*, sin una interfaz gráfica de usuario, y las pruebas se realizan exclusivamente en simulaciones. El sistema gestiona localmente tanto las imágenes como los datos simulados, eliminando la necesidad de una base de datos externa, con el objetivo de generar alertas personalizadas basadas en las señales detectadas y la información de los sensores simulados.

Mediante el uso de un *set* de datos manualmente generado y enriquecido con técnicas de data *augmentation*, junto con la integración del set de datos llamado Señales de tránsito *Dataset* que incluye imágenes de señalizaciones de tránsito específicas de Colombia, el modelo de detección de objetos se entrenará para reconocer y categorizar los distintos elementos del entorno vial. Este enfoque combinado permitirá al sistema no solo identificar dichos elementos con precisión sino también adaptarse a variaciones extremas en las condiciones de iluminación, clima, y contextos tanto urbanos como rurales, en el ámbito guatemalteco. La incorporación de data augmentation asegura que el modelo sea robusto frente a estas variaciones, lo cual es crucial para su desempeño en el mundo real (Shorten y Khoshgoftaar, 2019). Al mismo tiempo, el uso del set de datos Señales de tránsito *Dataset* proporcionado por el usuario Tesis en la plataforma de Roboflow añade una valiosa diversi-

dad al conjunto de entrenamiento, mejorando la capacidad del sistema para generalizar a través de normativas de tránsito (Tesis, 2022).

La implementación del sistema en un pipeline de machine learning permitirá procesar el flujo de video, junto con los datos concurrentes de sensores de velocidad. Este enfoque holístico asegura una toma de decisiones precisa y oportuna, ofreciendo advertencias y recomendaciones al conductor para prevenir incidentes y mejorar la navegación (Grigorescu, Trasnea, Cocias, y Macesanu, 2020). En este proyecto, se prioriza la creación y optimización de los modelos de detección y clasificación, sin el desarrollo de una interfaz visual ni mecanismos de alerta en tiempo real para la interacción con el usuario. Esta estrategia permite concentrar todos los esfuerzos en mejorar la capacidad técnica del sistema para procesar y analizar el entorno vial visualmente, sin la complejidad adicional de desarrollar y probar componentes de software interactivo para usuarios finales. El éxito del proyecto se medirá en su capacidad técnica para detectar y clasificar objetos con precisión, proporcionando una ayuda para el conductor.

CAPÍTULO 2

Objetivos

2.1. Objetivo general

Desarrollar e implementar un sistema de asistencia al conductor que integre tecnologías de visión por computadora y análisis de datos de sensores para detectar y clasificar elementos críticos en calles y carreteras, contribuyendo a la experiencia del conductor y su percepción de seguridad y eficiencia en la conducción.

2.2. Objetivos específicos

- Crear y afinar un modelo de machine learning, basado en algoritmos como YOLO (*You Only Look Once*), que sea capaz de identificar y clasificar con precisión y rapidez señales de tránsito y otros elementos a partir de flujos de video.
- Desarrollar un mecanismo para incorporar y analizar datos de sensores de velocidad y aceleración, permitiendo que el sistema ajuste las alertas y recomendaciones de seguridad de acuerdo con la dinámica de conducción actual y las condiciones externas reglamentadas por señales de tránsito en el entorno.
- Configurar un conjunto de alertas dentro del sistema de asistencia al conductor que informe de manera efectiva y oportuna sobre los potenciales peligros detectados, mejorando la capacidad del conductor para tomar decisiones informadas y reaccionar adecuadamente en situaciones críticas.

CAPÍTULO 3

Justificación

La problemática de abordar la seguridad vial nunca ha sido más importante, como lo demuestran las alarmantes estadísticas presentes en el Instituto Nacional de Estadística (INE) de accidentes de tránsito reportados por la Policía Nacional Civil (PNC) en el territorio guatemalteco en el año 2022. En el periodo de 2022 a primer trimestre del año 2024, se reportaron 19,698 personas lesionadas en accidentes de tránsito, una cifra que subraya la extensa magnitud del problema en una de las escalas no letales (INE, 2024). Estas lesiones no solo representan un costo humano significativo en términos de dolor y sufrimiento para las víctimas y sus familias, sino que también imponen una carga económica considerable en términos de atención médica y pérdida de productividad de los centros de asistencia médica. La implementación de un sistema de asistencia al conductor se presenta como una solución prometedora para reducir estos riesgos, ofreciendo la posibilidad de disminuir la frecuencia y severidad de estos incidentes mediante tecnología de detección y prevención de las señalizaciones de tránsito que existen para resguardar el bienestar de los ciudadanos que circulen en calles o carreteras.

Además, en el mismo periodo vio un total devastador de 5,065 fallecimientos debido a incidentes de tránsito (de Estadística, 2024). Cada uno de estos fallecimientos es una tragedia que deja un vacío en las familias y comunidades afectadas. Esta cifra resalta la necesidad crítica de sistemas de seguridad vehicular que puedan llegar a prevenir tales fatalidades. Un sistema de asistencia al conductor que utilice visión por computadora para detectar peligros potenciales al no respetar la señalización del entorno vial podría ser clave en la alerta temprana a los conductores sobre situaciones de riesgo, mejorando así la toma de decisiones en momentos críticos.

El número de accidentes de tránsito, que asciende a 17,895 incidentes reportados por la Policía Nacional Civil en el periodo de 2022 a primer trimestre del año 2024, subraya la urgencia de adoptar medidas para mejorar la seguridad vial en Guatemala (de Estadística, 2024). En este contexto, es crucial considerar la normativa vigente, como lo establece el Decreto Número 132-96, Ley de Tránsito, y su posterior reforma Decreto 84-2005, que marcan los lineamientos para el comportamiento en las vías públicas y la regulación de los vehículos. Asimismo, el Acuerdo Gubernativo No. 293-98, que reglamenta específicamente el tránsito, pone de manifiesto la necesidad de cumplir con una serie de disposiciones orientadas a garantizar la seguridad tanto de conductores como de peatones. Estas legislaciones abogan por un tránsito ordenado y seguro, promoviendo la responsabilidad y el respeto entre los usuarios de la vía.

En este entorno regulado por leyes y reglamentos, la introducción de sistemas avanzados de asistencia al conductor emerge como un complemento vital para fortalecer la seguridad vial. La aplicación de tecnologías basadas en algoritmos de *machine learning* para la detección y clasificación

de objetos no solo alinea con las disposiciones de la Ley de Tránsito y su reglamento, sino que también representa un avance significativo en la prevención de accidentes. Este enfoque tecnológico, que se suma a las estrategias de educación vial y a la aplicación rigurosa de las normativas, promete una mejora notable en la capacidad de respuesta de los pilotos frente a situaciones potencialmente peligrosas en la carretera, contribuyendo así a un entorno más seguro para todos.

El desarrollo de este proyecto responde directamente a estas estadísticas, buscando brindar una posible solución para reducir el número de accidentes, lesiones y fallecimientos sino también mejorar la experiencia general de conducción. Al integrar análisis avanzados de video y datos de sensores, el sistema propuesto facilitará una conducción más segura y consciente, adaptándose dinámicamente a las condiciones de la carretera y alertando a los conductores sobre peligros y resguardando la vida de muchos.

CAPÍTULO 4

Marco teórico

4.1. Introducción a la visión computacional

La visión computacional es una subdisciplina de la inteligencia artificial enfocada el desarrollo de técnicas para que un entorno digital de una computadora pueda interpretar y ser capaz de comprender el contenido de una fotografía digital. Aquí se combinan principios del área de la ciencia de la informática y de las matemáticas aplicadas para en conjunto crear algoritmos con la capacidad de procesar y analizar datos visuales como lo hace el mismo sistema visual de los humanos. El objetivo principal de la subdisciplina en cuestión es darle la capacidad visual de comprensión del entorno físico real, el reconocimiento de patrones y poder interpretar toda la información al traducirlo a un ámbito computacional (Szeliski, 2022).

Con el paso del tiempo la visión computacional ha mostrado una evolución significativa comparado con sus inicios en la década de 1960. Inicialmente los esfuerzos se centraban en la digitalización y el procesamiento básico de imágenes. Esto permitió que nuevas tecnologías tomaran lugar como lo fue el reconocimiento óptico de caracteres (OCR por sus siglas en inglés) en los años 70. Con esto se llegaron a avances como lo fue el permitir a las máquinas tener la capacidad de leer texto impreso. Con el paso de las decadas se ha impulsado el desarrollo de algoritmos avanzados y mejoras en la capacidad de procesamiento para tareas más complejas, como el reconocimiento facial y la conducción autónoma (Anand y Priya, 2019).

4.1.1. Definición de visión computacional

En el campo de la visión computacional, derivado de la inteligencia artificial, buscamos darle a las computadoras la capacidad de interpretar y entender el entorno visual del mundo tangible que nos rodea, de una forma similar a la forma que lo hacemos los seres humanos. Esto implica la creación de modelos y algoritmos que pueden tomar una entrada y procesarla, analizarla y extraer la información que compone la misma. Dichos modelos usan técnicas avanzadas de aprendizaje de máquina (conocido como *machine learning*), como lo son las redes neuronales convolucionales (CNN). Esto les permite identificar patrones y reconocer objetos dentro de las imágenes (E.R., 2018).

El principal objetivo del campo de visión computacional es realizar tareas complejas basadas en la interpretación visual, como la detección y reconocimiento de objetos. Esta tecnología tiene

una gran cantidad de aplicaciones críticas en diversos campos como lo es la automoción, donde la inclusión de la misma ayuda a tener una mejor precisión, eficiencia y toma de decisiones de las operaciones y funcionamiento (Tseng y Jan, 2018).

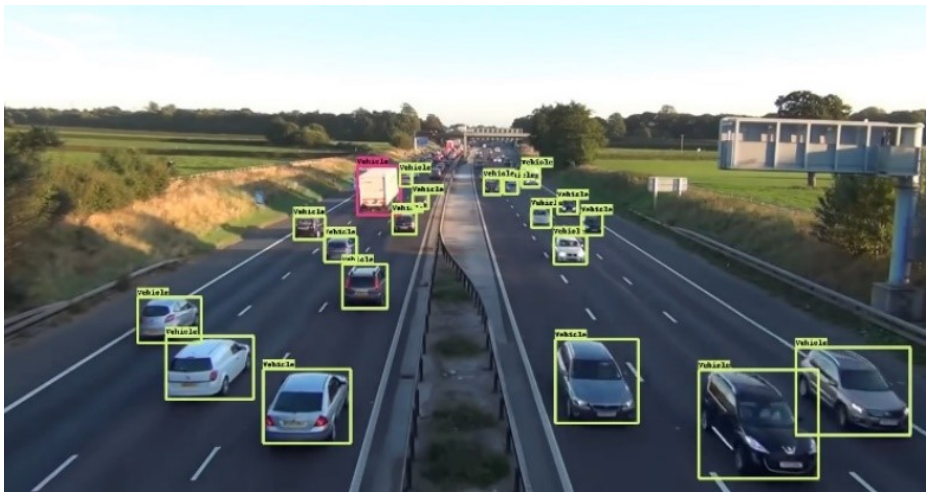

Figura 4.1: Reconocimiento de vehículos mediante algoritmo de detección de objetos con visión computacional.

(Cork, 2020)

4.1.2. Historia y evolución de la visión computacional

En la década de 1960, los proyectos iniciales de visión computacional buscaban lograr realizar tareas básicas de segmentación y extracción de objetos de una imagen. Luego, en los años 80 tenemos a de David Marr y su propuesta de un marco teórico que brinda la descripción de la visión como un proceso jerárquico. Al mismo tiempo, Kunihiko Fukushima presenta la red neuronal con capas convolucionales llamada Neocognitron que posee la capacidad de reconocer patrones (Alvarado, 2009).

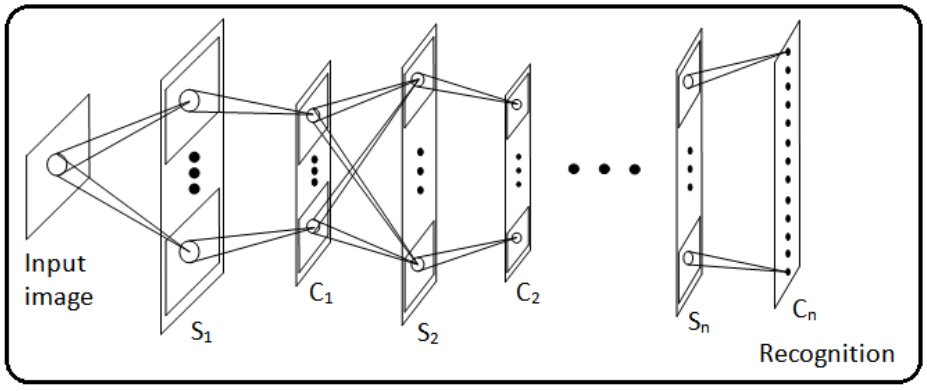

Figura 4.2: Diagrama de las multiples capas del diseño original de Neocognitron diseñado por Kunihiko Fukushima.

(Makone, 2020)

El enfoque toma un cambio en los años 90 hacia el reconocimiento de objetos basado en características locales con algoritmo como SIFT de David Lowe (Lowe, 1999). Poco tiempo después, en el año 2001, Paul Viola y Michael Jones desarrollaron el primer sistema de detección de rostros en tiempo real utilizando características simples y técnicas de *boosting* (Viola y Jones, 2001).

En el año 2012 es presentado el siguiente avance exponencial con el modelo ganador de la competencia ImageNet de red neuronal profunda, AlexNet. Con su funcionalidad de poder reducir significativamente los errores en el reconocimiento de imágenes, impulsa el uso de *deep learning* en visión computacional, dando lugar a avances en los siguiente años (Patel y Patel, 2020).

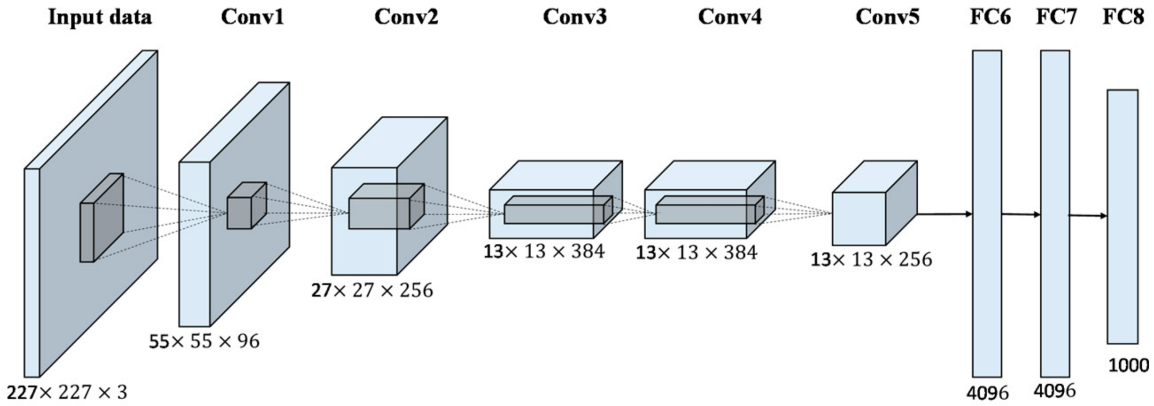

Figura 4.3: Arquitectura de Alexnet.

(Han, Zhong, Cao, y Zhang, 2017)

La visión por computadora ha seguido evolucionando constante con nuevos modelos capaces de generar y editar imágenes con precisión, ampliando las aplicaciones que tiene el campo de la inteligencia artificial y la visión por computadora en múltiples industrias como la automoción, la medicina y la seguridad. El desafío en la actualidad es meramente mejorar la precisión y eficiencia de los algoritmos y modelos para poder interpretar escenas complejas y la integración de múltiples fuentes de datos de entrada (Hu, 2023).

4.1.3. Importancia y aplicaciones en diversas industrias

La visión computacional es una herramienta tecnológica vital para desarrollar sistemas avanzados de asistencia al conductor (*Advanced Driver Assistance Systems (ADAS)* por sus siglas en inglés) y vehículos autónomos en la industria automotriz. Para estos se utilizan cámaras y algoritmos con la intención de lograr detectar y reconocer objetos a los alrededores del vehículo. Dentro de los objetos podemos destacar peatones, señales de tránsito, otros vehículos y obstáculos. Gracias a ello es posible lograr toma de decisiones para evitar colisiones, mantener ya sea al vehículo en su carril correcto o para optimizar la conducción en diversas condiciones de tránsito. Es importante destacar que se busca en todo momento proteger la vida de las personas tanto dentro del vehículo como fuera del mismo (Konstantinidis, Mouroutsos, y Gasteratos, 2021).

En cuanto a la industria de la agricultura la visión computacional es empleada como la herramienta que permite monitorear el estado de los cultivos. Mediante el uso de cualquier opción que permita realizar captura de imagenes aereas de sus cultivos, como por ejemplo drones equipados con cámaras, es posible realizar un análisis de la plantación. Esto ayuda a detectar rápidamente signos de plagas, malezas o alguna deficiencia de nutrientes en los cultivos que no podría haber sido observada sin la gran cantidad de horas invertidas en ello. Así mismo se puede analizar el crecimiento tanto de

la vegetación o cultivos de una zona en especial para determinar la protección de la misma zona o la eficiencia del cuidado de la misma (Dhanya y cols., 2022).

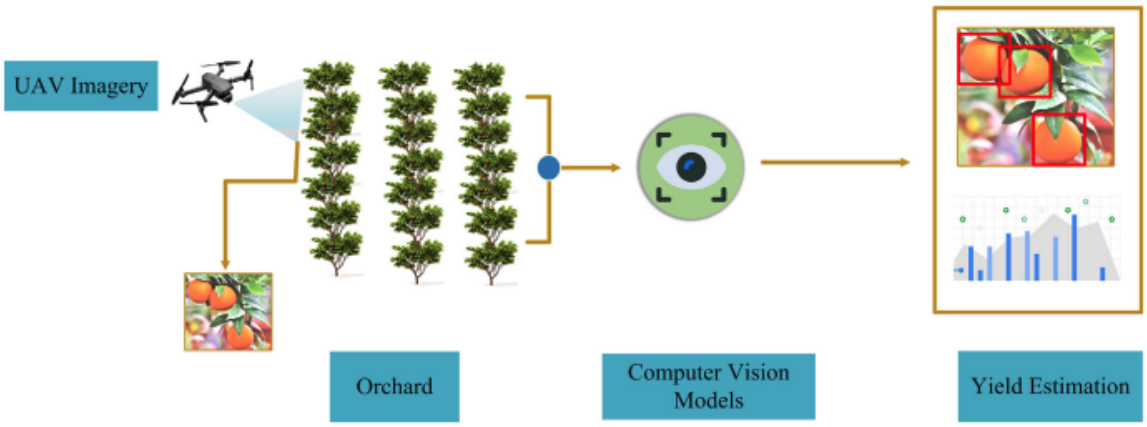

Figura 4.4: Estimación del rendimiento del huerto mediante visión por computadora.

(Dhanya y cols., 2022)

En el campo de la medicina, la visión por computadora es una herramienta fundamental para realizar análisis y ayudar a dar diagnósticos médicos con mayor precisión y rapidez. Un ejemplo de dicha aplicación es para poder analizar imágenes de resonancias magnéticas (MRI, por sus siglas en inglés), tomografías computarizadas (CT, por sus siglas en inglés) y radiografías. Con esto se puede hacer la detección de enfermedades que pueden ser difíciles de detectar hasta para el ojo entrenado por su patrón inusual y de baja magnitud ayudando así a los profesionales del sector de la salud en la toma de decisiones y diagnósticos (Esteva y cols., 2021).

Finalmente podemos hablar de la utilización de la herramienta de visión computacional en la seguridad y vigilancia. Estas permiten detección de brechas a sistemas e intrusos a espacios físicos y realizar el reconocimiento facial de los perpetradores. Esto nos brinda la facilidad de no solo mantener la seguridad y la integridad de un espacio en específico, pero también detectar las personas que ejecutan esta acción y poder brindar la mayor cantidad de información a las autoridades correspondientes (Fang y cols., 2020).

4.2. Conceptos fundamentales de la visión computacional

4.2.1. Captura y procesamiento de imágenes

Capturar y procesar imágenes es parte de un proceso importante en la visión por computadora donde traducimos las imágenes digitales en datos útiles para ser analizados. Esto se puede realizar por diversos métodos iniciando por la captación de las ondas de luz en señales eléctricas para ser digitalizadas, produciendo píxeles de la escena del mundo tangible en cuestión (Burger y Burge, 2022). Luego se empleado una gran serie de técnicas para mejorar la calidad de las imágenes y que estas puedan ser procesadas y analizadas. Una de estas, es la eliminación de ruido, ajustes de contrastes y sombras de la imagen donde buscamos resaltar y poder apreciar todas las características disponibles en la escena para luego poder hacer el reconocimiento de patrones y objetos (Cadena y cols., 2017).

Este proceso no está limitado a mejorar la calidad visual, sino que también incluye la transfor-

mación de las imágenes para que sean aptas a diferentes aplicaciones especificas a su campo. Por ejemplo, en el campo médico, las imágenes deben ser procesadas para resaltar detalles específicos que son críticos para diagnosticar un paciente. En la industria automotriz, las imágenes deben ser ajustadas para facilitar la detección de señales de tráfico y peatones en condiciones climáticas y lumínicas variadas encontradas en el ámbito especifico. Esto nos permite poder tener la facilidad e innovación de vehículos autónomos y sistemas ADAS (Konstantinidis y cols., 2021).

4.2.2. Segmentación y detección de bordes

Otra parte del proceso de interpretación y análisis de imágenes es la segmentación y detección de bordes. Se busca hacer la división de una imagen en partes más pequeñas, simplificando el análisis y dando el espacio para la identificación de objetos de interés en esos fragmentos de la imagen original. Una de las técnicas de segmentación más utilizada es la umbralización, que divide las regiones por la intensidad de los píxeles. También podemos mencionar las técnicas que buscan hacer la división de las zonas por pixeles con características similares entre ellos (Barik y Mondal, 2010).

En cuanto a la detección de bordes, tenemos el enfoque de identificar y delinear los límites de los objetos dentro de una imagen, buscando destacar las transiciones significativas en la intensidad de los píxeles. Un ejemplo de algoritmo que realizar está tarea es Sobel, que utiliza la convolución con matrices para encontrar gradientes en la imagen, mientras que algoritmos más avanzados, como Canny, emplean un proceso multietapa que incluye la detección de bordes en diferentes escalas y la supresión de no máximos para afinar los bordes detectados (S. Wang, Ge, y Liu, 2006).

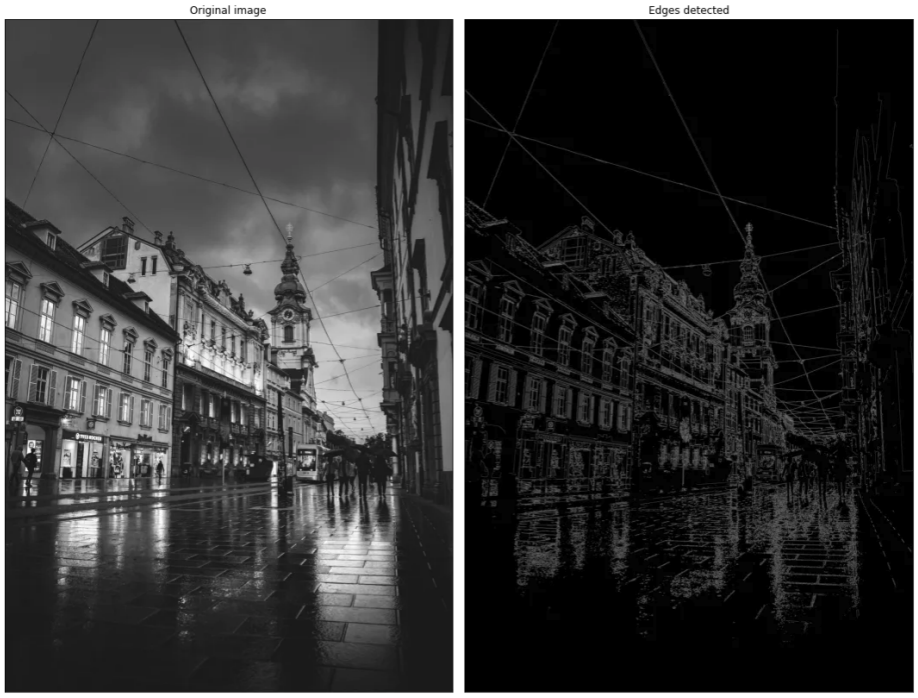

Figura 4.5: Imagen con detección de bordes mediante algoritmo de Canny Edge.

(Bakshee, 2021)

Gracias a los modelos basados en redes neuronales convolucionales se logrado evolucionar el proceso de detección de bordes. Con algoritmos como *DeepEdge* que con múltiples niveles de características mejoran su precisión y nitidez se logra trazar de una mejor forma los bordes en la

imagen. Combinando entonces la detección de bordes con la segmentación podemos tener una mejor interpretación de la imagen, detección de objetos y brindar respuestas más certeras (Bertasius, Shi, y Torresani, 2015).

4.2.3. Representación y descripción de objetos

La representación y descripción de objetos permiten que los sistemas sean capaces de interpretar mediante un proceso analítico los elementos presentes en una escena captada en una imagen. Por representación de objetos entendemos la manera en que un objeto se modela dentro de un sistema mientras que por descripción es el cómo se extraen y utilizan las características de dicho modelo para un posible proceso de análisis (Toshev, 2011). Para lograr hacer la representación hay diversas técnicas que pueden variar desde el uso de modelos sencillos en dos dimensiones hasta complejas estructuras de tres o más dimensiones de análisis.

Los enfoques en dos dimensiones normalmente utilizan descriptores como los histogramas de gradientes orientados (HOG), que capturan la forma y estructura de los objetos a partir de los gradientes de intensidad en una imagen (Poojari, 2023). Por otro lado, los modelos en tercera dimensión proporcionan una representación más detallada y robusta de los objetos, permitiendo su análisis desde múltiples ángulos y en diferentes condiciones de iluminación y oclusión (Hoiem y Savarese, 2011). En cuanto a los descriptores de objetos, estos son una parte importante para la identificación y clasificación precisa de los elementos en la escena analizada. Técnicas como SURF (*Speeded-Up Robust Features*) son utilizadas para extraer características distintivas de los objetos que son invariantes a transformaciones de escala y rotación. Estos descriptores permiten que los sistemas comparen y reconozcan objetos en diferentes imágenes eficientemente (Hladnik y Gabrijelčič, 2016). Ambos procesos los podemos encontrar en modelos como lo son las CNN (Montserrat, Lin, Allebach, y Delp, 2017).

4.3. *Machine Learning* en visión computacional

4.3.1. Introducción a *machine learning* y su relación con la visión computacional

Gracias a la relación entre *machine learning* (ML) y visión computacional (CV) se logra tener la capacidad de interpretar y extraer información importante de imágenes por medio de algoritmos de aprendizaje supervisado y no supervisado. Estas formas de aprendizaje con la ventaja del uso de redes neuronales permiten que reconocer patrones, característicos, y objetos en las escenas de una imagen en tiempo real de una forma eficiente y precisa (A. A. Khan, Laghari, y Awan, 2021).

Es importante hacer mención que cada tipo de aprendizaje tiene un enfoque meramente distinto para la misma finalidad de brindar la capacidad de detección y extracción de características de los objetos modelados. En el caso del supervisado, no supervisado y por refuerzo se aportan metodologías únicas para abordar los desafíos en el proceso de analizar e interpretar el mundo tangible y extraer dichos patrones de una imagen capturada (Sarker, 2021).

4.3.2. Tipos de aprendizaje: supervisado, no supervisado y por refuerzo

El aprendizaje supervisado, utilizado normalmente en tareas de clasificación y regresión, toma como entrada conjuntos de datos previamente etiquetados, donde el modelo aprende a mapear las entradas y salidas puntuales. En el otro lado tenemos el aprendizaje no supervisado, normalmente

utilizado en análisis de agrupamientos o la reducción de dimensionalidad, que trabaja sin etiquetas predefinidas enfocándose exclusivamente en identificar patrones y estructuras en los datos captados. Finalmente tenemos el aprendizaje por refuerzo toma una metodología basada en recompensas y castigos, donde la unidad de aprendizaje del modelo aprende a tomar decisiones en entornos dinámicos. Dicho modelo hace continuamente ajustes y cambios para maximizar la cantidad de recompensas obtenidas en sus tareas (Sarker, 2021). Estos enfoques de entrenamiento son habitualmente implementados en el desarrollo de redes neuronales y redes neuronales convolucionales con la intención de llegar a simular la complejidad y funcionamiento del cerebro humano.

4.3.3. Redes neuronales y redes neuronales convolucionales (CNN)

Ambas las redes neuronales, algunas veces llamadas redes neuronales artificiales (*Artificial Neuronal Networks*, ANN por sus siglas en inglés), y las redes neuronales convolucionales (*Convolutional Neural Network*, CNN por sus siglas en inglés) han revolucionado el campo de visión por computadora. Ambas se inspiran en la estructura del cerebro humano donde las capas de aprendizaje de estas redes simulan las neuronas del mismo cerebro, procesando la información que pasa por ellas generando salidas que son utilizadas por otras neuronas o que finalmente llevan a la toma de decisiones. A esto se el conoce como proceso de propagación hacia adelante normalmente es seguido por una dinamización de una función de pérdida. Esto permite que las ANNs sean capaces de aprender patrones complejos de los datos etiquetados en un proceso de aprendizaje supervisado (Iglesias y cols., 2021).

Las CNNs, han demostrado ser eficaces para procesar datos visuales aprovechando las propiedades espaciales de las imágenes aplicando filtros convolucionales que actuan como detectores de características. Estos filtros permiten que dicha red sea capaz de señalar los patrones en las imágenes como sus bordes, texturas y colores que es algo vital en aplicaciones variadas como el reconocimiento facial, la automoción o la medicina (S. Cong y Zhou, 2023).

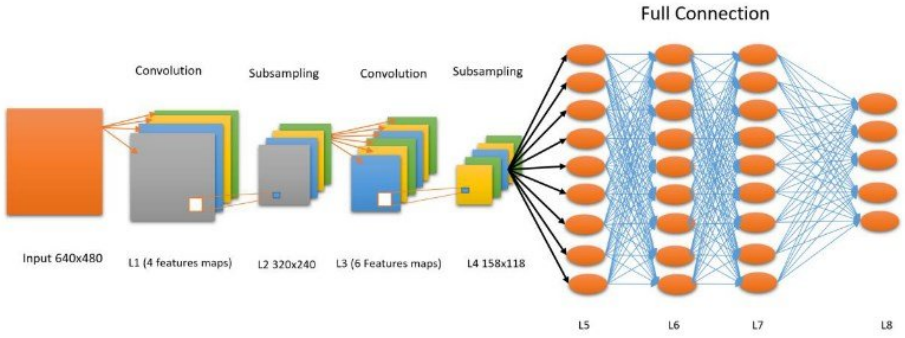

Figura 4.6: Arquitectura convencional de una Red neuronal convolucional.

(Contreras y la Rosa, 2016)

Arquitecturas complejas como ResNet e Inception han dado su origen de las CNNs, mejorando la capacidad de generalización y reconocimiento de patrones en conjuntos de datos de gran tamaño. Esto ha sido fundamental en aplicaciones hoy en día que van desde la medicina, la conducción de vehículos autónomos, seguridad en el tránsito, agricultura y muchos otros más haciendo así que las CNNs ayuden a procesar y analizar problemas del mundo real (A. Khan, Sohail, Zahoora, y Qureshi, 2020).

4.4. Modelos de detección de objetos

Se ha observado una tendencia en los últimos años que permite realizar una división de los enfoques de los modelos de detección de objetos. Por un lado tenemos a los modelos de una etapa como YOLO y SSD combinando la detección y clasificación de objetos en un proceso. Esto permite que sean rápidos haciéndolos una buena alternativa pra aplicaciones que requieren análisis en tiempo real. Por el otro lado tenemos los modelos de dos etapas, como *Faster R-CNN*, que pueden llegar a tener una precisión mayor pero su rapidez no es rápida haciéndolos una buena opción para analizar una tarea que requiere de mucha precisión en un ámbito que no obliga a dar respuestas en tiempo real (Zaidi y cols., 2022). En este modelo se hace la separación de generación de regiones y la clasificación dentro de estas mismas, dándole así una precisión más alta a costa de mayor complejidad computacional (R.-B. Wang, Yan, y Xu, 2021).

La evaluación del rendimiento de estos modelos en distintas aplicaciones es algo importante y para ello contamos con métricas como precisión media (*Average Precision*, AP por sus siglas en inglés) y la intersección sobre unión (*Intersection over Union*, IoU por sus siglas en inglés). Como AP entendemos la métrica que mide el rendimiento de un modelo para la detección de objetos considerando los umbrales de confianza para de precisión y *recall*, donde un valor alto es señal de un mejor rendimiento del modelo. De una forma similar la métrica de IoU evalúa que tan bien se superponen las cajas que delimitan a los objetos. Esta se calcula dividiendo el área de superposición entre las cajas por el área total combinada (R.-B. Wang y cols., 2021).

4.4.1. *You Only Look Once* (YOLO)

El modelo de *You Only Look Once* (YOLO, por sus siglas en inglés) ha mostrado una diferencia a los enfoques anteriores que involucraban múltiples pasos al aplicar el proceso como una tarea de regresión única. Se procesa una imagen completa en una sola pasada haciendo la división en cuadrícula, prediciendo simultáneamente las cajas delimitadora (*bounding boxes*) y asignando una probabilidad a cada celda para determinar el objeto de dicha celda (Terven, Córdova-Esparza, y Romero-González, 2023). Esta metodología que usa el modelo permite que la detección en tiempo real se encuentre en equilibrio con una alta velocidad y precisión, lo cual es crucial para aplicaciones como la vigilancia y la conducción autónoma (Önler y Köycü, 2024).

Se han desarrollado varias iteraciones de dicho YOLO, como por ejemplo YOLOv3, YOLOv5 y más recientemente YOLOv8. Cada uno ha tenido aspectos puntuales que mejoran la precisión y capacidad de generalización manteniendo o disminuyendo la necesidad computacional para lograrlo (Terven y cols., 2023). Estas mejoras permiten que cada vez más estos modelos sean considerados una opción principal para aplicaciones en diversas industrias donde se caracterizan por brindar una solución con una alta velocidad y eficiencia a bajos costos (Hussain, 2023).

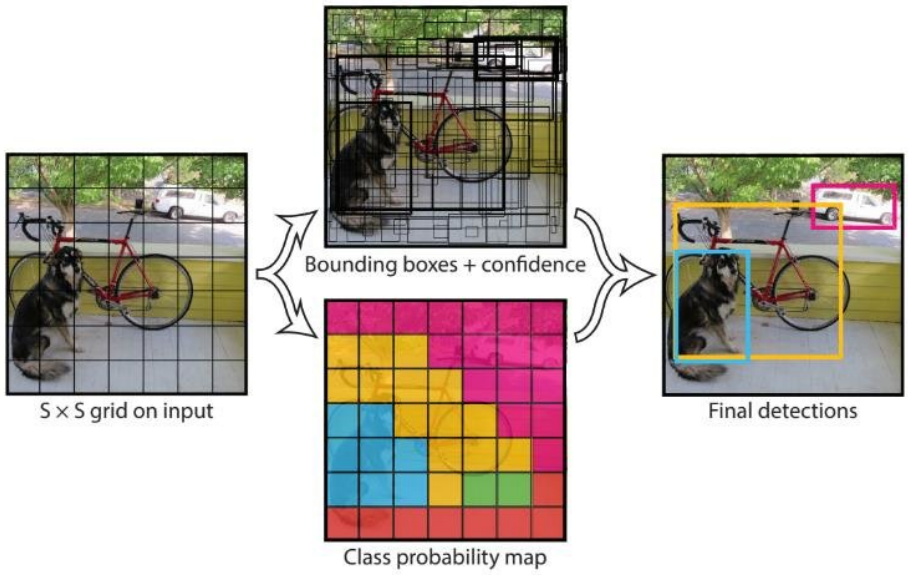

Figura 4.7: Arquitectura de modelo YOLO.

(Ivanov y Chivarov, 2020)

Arquitectura y funcionamiento

La arquitectura del modelo YOLO es característica por procesar imágenes en tiempo real con una alta eficiencia (Terven y cols., 2023). El proceso busca dividir la imagen en cuadrícula y hacer una predicción individual de cada una de las celdas (Hussain, 2023). Esto busca simplificar el proceso de detección al combinar la localización y clasificación en un solo paso en lugar de en dos etapas (Vijayakumar y Vairavasundaram, 2024).

Cada versión de YOLO ha presentado cambios en su arquitectura. Por ejemplo, YOLOv1 utiliza 24 capas convolucionales seguidas de dos capas completamente conectadas, lo que permite capturar características clave en las imágenes mientras mantiene una estructura relativamente simple (Terven y cols., 2023). En versiones más recientes, como YOLOv8, se han incorporado mejoras significativas en la capacidad de detección y la precisión, adaptándose a aplicaciones que requieren un balance entre velocidad y exactitud. Su principal característica es el uso de CSPDarknet (*Cross Stage Partial Network*) como su *backbone* que le da una mejor extracción de los patrones y características en múltiples escalas mejorando la detección de objetos de gran o pequeño tamaño. Luego como *neck* tenemos el uso de *Path Agreggation Network* (PAN, por sus siglas en inglés) o *Feature Pyramid Network* (FPN, por sus siglas en inglés) dandole la capacidad de procesar imágenes con alto o bajo detalle. Esto le da un total de 53 capas a la versión de YOLOv8 (Vijayakumar y Vairavasundaram, 2024).

Adicionalmente, la octava versión de este modelo cuenta con sub variantes conocidas como YOLOv8n (nano), YOLOv8s (*small*), y YOLOv8x (*extra-large*). Donde cada una muestra variaciones que le permiten balancear el tiempo de procesamiento y precisión para lograr adaptarse a aplicaciones especificas del problema en cuestión (Yaseen, 2024).

Ventajas y desventajas

Algunas de las ventajas más importantes a destacar de YOLO es la capacidad de procesar imágenes en tiempo real dado el enfoque de detección de una sola pasada. Esto permite que sea una opción de alto valor al momento de ejecutar tareas donde la velocidad es crítica (Vijayakumar y Vairavasundaram, 2024). Adicional a esto el tener una estructura relativamente simple le da la capacidad de poder ser implementado en ambientes de bajos recursos computaciones (Terven y cols., 2023).

En cuanto a las desventajas, su precisión se puede ver comprometida en algunas ocasiones cuando se debe de analizar una gran cantidad objetos o en objetos de un tamaño reducido (Vijayakumar y Vairavasundaram, 2024). De igual forma el mismo hecho de tener una arquitectura sencilla puede afectar en la detección de objetos con detalles finos o de muy alta precisión (X. Cong, Li, Chen, Liu, y Meng, 2023).

4.4.2. *Single Shot MultiBox Detector* (SSD)

Single Shot MultiBox Detector (SSD, por sus siglas en inglés) es un método en el cual mediante el uso de una única red neuronal se identifican objetos en una escena de una imagen. Este enfoque toma y transforma el proceso de detección de objetos al definir un conjunto predefinido de cajas de varios tamaños y proporciones en puntos específicos dentro del mapeo de la imagen. Al momento de hacer las predicciones la red neuronal asigna una puntuación a cada caja para determinar que objeto y bajo que probabilidad se encuentra el mismo en ese espacio. Luego ajusta la forma de la caja para alinearlas de tal manera que se vaya detectando el objeto dado por las probabilidades unitarias de las cajas (Jiang, Xu, Zhang, y Fang, 2019).

Así como otros modelos de detección de objetos SSD tambien es capaz de trabajar en tiempo real. Al realizar la detección en una sola pasada por la red, brinda respuestas rápidas que le permiten ser tomado en consideración en tareas donde la velocidad de respuesta es importante (Jiang y cols., 2019).

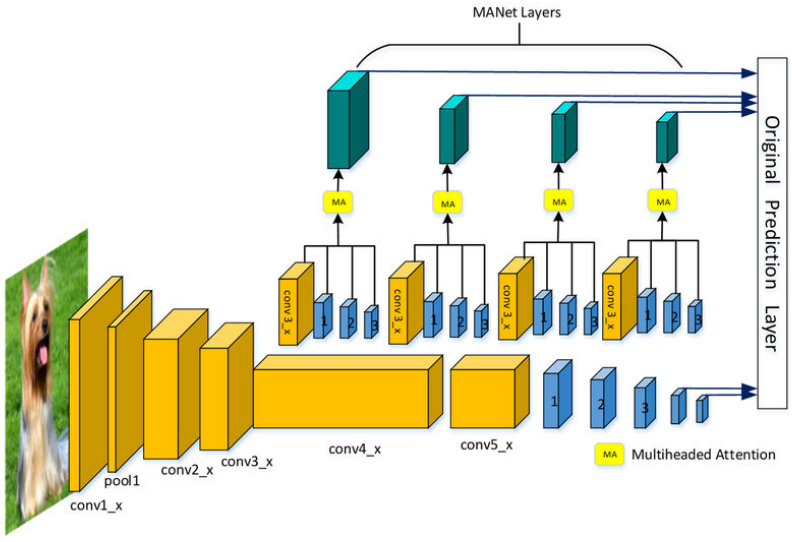

Figura 4.8: Arquitectura de modelo SSD.

(Jiang y cols., 2019)

Arquitectura y funcionamiento

Este es un modelo donde su arquitectura utiliza múltiples mapas de características de diferentes resoluciones para predecir las cajas delimitadoras y las respectivas clases, permitiendo que sin importar la escala pueda realizar la detección (Liu y cols., 2016). A diferencia de los modelos de dos etapas, SSD evita la separación de generación de propuestas haciéndolo más rápido y eficiente en aproximadamente 29 capas(Hyams y Malowany, 2020). La utilización de diferentes relaciones de aspecto y escalas para lograr analizar diversad formas y tamaños de objetos hacen que la flexibilidad y precisión en la detección de objetos de este modelo aporte gran valor (Liu y cols., 2016).

Ventajas y desventajas

Entre las ventajas de mayor valor, donde se logra destacar su velocidad y eficiencia computacional, lo que lo convierte en una opción ideal para aplicaciones en tiempo real este modelo es capaz de realizar detecciones rápidas gracias a su arquitectura de una sola etapa, lo que minimiza el cómputo redundante y permite que el proceso de inferencia sea más ágil en comparación con los modelos de dos etapas, como *Faster R-CNN* (Hyams y Malowany, 2020). Sin embargo, una de las desventajas de SSD es su menor precisión en la detección de objetos pequeños, ya que su capacidad para capturar detalles finos en imágenes de baja resolución puede ser limitada (Liu y cols., 2016).

4.4.3. *Faster* R-CNN (*Regions with Convolutional Neural Networks*)

Arquitectura y funcionamiento

Faster R-CNN (*Regions with Convolutional Neural Networks*) es un modelo de detección de objetos que da mejoras significativas a sucesores como R-CNN y Fast R-CNN al usar una Red de Propuestas de Regiones (*Region Proposal Network*, RPN por sus siglas en inglés) (Ren, He, Girshick, y Sun, 2016). Este es un componente de gran importancia y valor que busca generar propuestas de regiones directamente desde las características convolucionales compartidas con la red de detección. Esto hace que no sea necesario usar algoritmos externos como la Búsqueda Selectiva (Fawzy, 2020). El proceso de detección se lleva a cabo en dos etapas generando primero las regiones propuestas por RPN y luego refinandolas por la red *Fast R-CNN* (Fawzy, 2020).

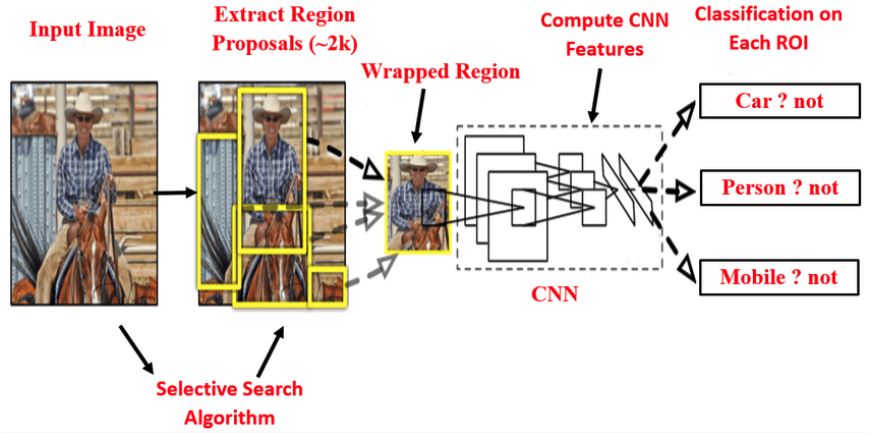

Figura 4.9: Arquitectura de modelo RCNN.

(Murthy, Hashmi, Bokde, y Geem, 2020)

Ventajas y desventajas

La alta precisión en la detección de objetos debido al uso de RPN es una de las ventajas más notorias de este modelo (Ren y cols., 2016). Adicional a esto, el hacer uso de la comparación de características entre la RPN y la red de detección reduce la carga computacional, mejorando la eficiencia del proceso haciendo que su rendimiento en situaciones de tiempo real sea sobresaliente (Fawzy, 2020). Sin embargo, el no ser un modelo de una etapa hace que *Faster R-CNN* sea relativamente más lento y complejo en comparación de modelos como YOLO (Ren y cols., 2016).

4.4.4. Comparación de modelos de detección de objetos

Tanto YOLO, SSD y *Faster R-CNN* cumplen la función de detección de objetos destacando por sus peculiaridades, ventajas y desventajas. Sin embargo sobre estos es mayormente más reconocido YOLO por su capacidad de detección en tiempo real al mostrar una mejor eficiencia debido a su arquitectura sencilla. Muestra una buena efectividad al poder mantener una alta velocidad y eficiencia en detección de objetos de distinta escala a comparación de SSD (Joiya, 2022). Por el otro lado *Faster R-CNN* ofrece una mayor precisión y capacidad de detección de detalles finos en los objetos de una escena lo hace a costa de una mayor complejidad computacional que lo separan ligeramente de la competencia en tareas de tiempo real (Sree, Bharadwaj, y Neelima, 2021).

4.5. Modelos de clasificación de objetos

4.5.1. Definición y diferencia entre detección y clasificación

La clasificación de objetos en visión por computadora tiene el enfoque de asignar etiquetas específicas a una imagen. Esto implica llevar un proceso donde se identifica la clase dominante para luego asignar la etiqueta correspondiente. Este enfoque es esencial para aplicaciones donde el objetivo principal es el reconocimiento del objeto presente en la imagen como lo es en la aplicación de los vehículos autónomos y la detección de otros tipos de vehículos, obstáculos o señales de tránsito (D. Wang, Wang, y Xu, 2021).

Contrariamente, la detección de objetos no solo involucra el proceso de clasificación pero tambien la identificación de la posición exacta de dicho elemento en la imagen mediante las cajas delimitadoras. Las cajas delimitadoras (*bounding boxes*) son rectángulos ajustados alrededor de los objetos en la imagen señalizando su ubicación exacta (Aditi y Dureja, 2021). Para ello se requieren algoritmos capaces de manejar la localización y la clasificación de manera simultanea donde muchas veces se busca llevar a cabo para múltiples objetos en la imagen. Esto es fundamental para aplicaciones, como la automoción, donde no solo debemos de poder identificar el objeto sino saber la ubicación exacta del mismo en tiempo real (Shen y cols., 2024). Algunos modelos comúnmente utilizados para clasificación y detección son VGG y ResNet.

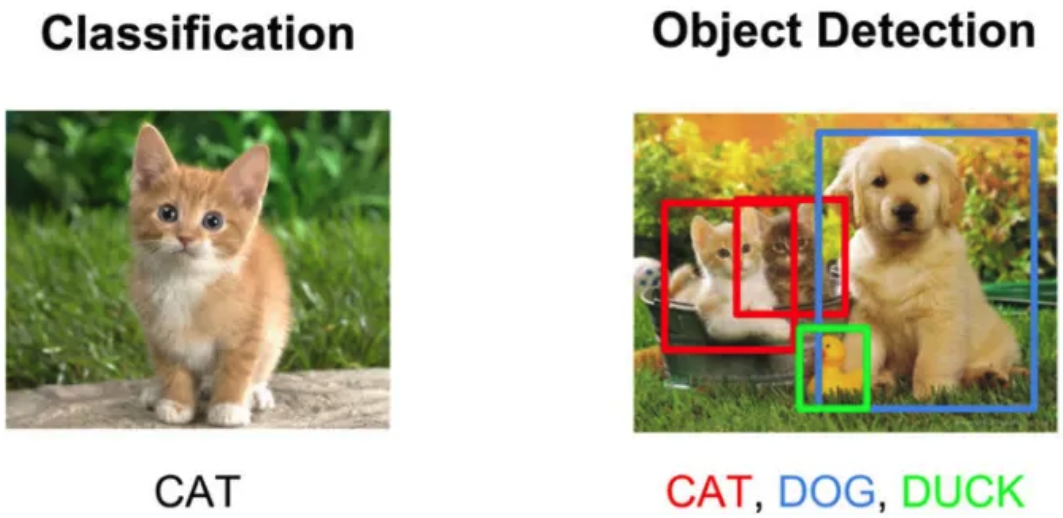

Figura 4.10: Diferencia entre clasificación (izquierda) y detección (derecha) de objetos.

(Martinez, 2022)

4.5.2. Modelos de clasificación VGG y ResNet

El modelo de red neuronal profunda *Visual Geometry Group* (VGG, por sus siglas en inglés) se destaca por una arquitectura simple que usa múltiples capas convolucionales con filtros pequeños de 3x3. VGG ha demostrado ser eficiente en la clasificación de imágenes con estructuras bien definidas (Angelina y Ulfitria, 2024).

En cuanto a *Residual Network* (ResNet, por sus siglas en inglés) presenta una innovación al introducir çonexiones residuales.[en] su arquitectura. Esto permite que la red sea mucho más profunda sin sufrir en precisión. Esto facilita que el entrenamiento con cientos de capas ayude a mejorar la capacidad del modelo para retener y aprender caracteristicas complejas y detalles de grandes conjuntos de datos (Umeaduma, 2024).

4.5.3. Técnicas de mejora de rendimiento *data augmentation* y *transfer learning*

Data augmentation implica la generación de nuevas muestras de datos a partir del existente conjunto de datos mediante transformaciones como la rotación, escalado y volteo. Dicho proceso ayuda a aumentar la diversidad de los datos previniendo el sobreajuste y mejorando la capacidad de generalización del modelo (Celin, Vijayalakshmi, y Nagarajan, 2023). Esto es efectivo cuando los datos disponibles son limitados pues logramos hacer que el proceso de entrenamiento se expanda sin la necesidad de captar más datos (Alomar, Aysel, y Cai, 2023). Por otro lado, *transfer learning* se centra en utilizar modelos preentrenados con grandes conjuntos de datos. Esto luego permite que luego podamos tomar dichos modelos y adaptarlos a tareas especificas. Esto reduce el tiempo de entrenamiento de manera significativa, mejorando la precisión y ayudando a enfocarse en los objetivos de la tarea (Celin y cols., 2023).

Al combinar estas dos técnicas se consigue que los modelos de clasificación de objetos tengan una mayor robustez, precisión y brinden más valor aún cuando hay cierta limitación de datos en la tarea empleada (Alomar y cols., 2023).

4.6. Integración de datos de sensores

4.6.1. Tipos de sensores utilizados en vehículos terrestres

Los vehículos terrestres autónomos integran sensores para lograr percibir de forma precisa el entorno garantizando así una navegación segura para el usuario tanto dentro del mismo como a las personas que se encuentran en el ámbito que se desempeña. El uso de sensores como LIDAR, radar y cámaras juegan roles complementarios que introducen valor dando distintos tipos de información para la toma de decisiones (Khare y Raghavendra, 2024).

LIDAR (*light detection and ranging*)

La tecnología de LIDAR, usando pulsos de luz láser para medir distancias con precisión, es algo esencial para los vehículos autónomos pues permite crear representaciones tridimensionales del mundo tangible a su alrededor. Cabe mencionar que dicho sensor se desempeña con exactitud aún cuando las condiciones son de poca visibilidad o el entorno es complejo (Hälker y Barth, 2018). Por esto mismo la detección de obstáculos se lleva a cabo permitiendo mantener una seguridad y fiabilidad en los sistemas autónomos.

Radar (detección y medición de distancias mediante ondas radioeléctricas)

El radar se basa en la emisión y recepción de ondas de radio para lograr la detección de la distancia, velocidad y dirección de los objetos (Hälker y Barth, 2018). Aún cuando las condiciones meteorológicas son adversas se muestra como este sensor garantiza la seguridad del vehículo y sus pasajeros.

Cámaras

En el proceso de percepción visual es indispensable el uso de cámaras, pues con estas se capturan imágenes que son posteriormente procesadas usando algoritmos de visión por computadora para identificar señales de tránsito, peatones, otros vehículos u obstáculos en el camino (Hälker y Barth, 2018). Con la información recopilada podemos ver a detalle el color, textura y otras características que ayudan en el proceso de detección de mundo real. Un punto negativo de este sensor es que se puede ver afectado por las condiciones climáticas dado que la visibilidad disminuye a medida que el entorno se vuelve más complejo.

4.6.2. Tipos de sensores utilizados en dispositivos móviles para locomoción terrestre

En la actualidad el uso de dispositivos móviles para la locomoción terrestre es algo común. Desde aplicaciones para determinar la mejor ruta, medidores de velocidad o para situaciones de riesgo la precisión en la navegación es algo vital. Los sensores comúnmente empleados en los dispositivos para

dichas tareas con el velocímetro y el acelerómetro, que al combinarlos le damos a los dispositivos móviles la capacidad de operar con precisión en la tarea de la locomoción terrestre.

Velocímetro

El velocímetro mide la velocidad lineal del vehículo, proporcionando datos esenciales para mantener una velocidad adecuada y segura. Esto es fundamental en aplicaciones de navegación autónoma, donde es necesario ajustar la velocidad para evitar obstáculos y cumplir con las normas de tráfico. Los velocímetros modernos a menudo están integrados con sistemas Sistema de Posicionamiento Global (GPS) para mejorar la precisión de la velocidad, lo que es crucial para la sincronización en sistemas de control de velocidad (Andreasson, Grisetti, Stoyanov, y Pretto, 2023).

Acelerómetro

El acelerómetro mide la aceleración del dispositivo en diferentes ejes, detectando tanto las fuerzas estáticas como la gravedad, así como las fuerzas dinámicas resultantes del movimiento de un objeto. Este sensor es esencial en la detección de cambios en la velocidad y dirección, lo que permite al sistema hacer ajustes en el control de la tracción y la estabilidad del vehículo. En la navegación autónoma, el acelerómetro es muy importante para mantener la estabilidad, especialmente durante maniobras rápidas o en terrenos irregulares. También es crucial para el cálculo de la orientación y la inclinación del vehículo (Andreasson y cols., 2023). Además, los acelerómetros son utilizados para detectar impactos y colisiones, activando sistemas de seguridad pasiva como las bolsas de aire en caso de accidente (Andreasson y cols., 2023).

4.6.3. Procesamiento y fusión de datos de sensores

Los datos obtenidos de múltiples sensores debe de ser fusionado y luego procesado para poder fungir como información de entrada en aplicaciones de visión por computadora (Zhang y cols., 2023).Al fusionar estamos obteniendo todas las fortalezas de cada tipo de sensor logrando por ejemplo adquirir mapas en tercera dimensión detallados, información visual de una cámara y condiciones ambientales del radar; Dándonos así la base robusta necesaria para poder tomar decisiones acertadas en tiempo real (Zhang y cols., 2023).

Una de las técnicas más usadas hoy en día es el uso de aprendizaje profundo y estadística para lograr la fusión de datos de una forma eficiente para mejorar la detección y clasificación de objetos. En los sistemas ADAS es algo primario el uso de sensores y que estos sean fusionados y agrupados para ser procesados por la computadora abordo del vehículo ya sea este autónomo o no (Zhang y cols., 2023).

4.6.4. Beneficios de la integración de datos de sensores con visión computacional

Entre los principales beneficios que obtenemos al integrar sensores en visión computacional está el enriquecimiento y lograr una representación del mundo tangible más robusta y precisa (Chai, 2020). Esto ayuda a que un sistema tenga una mejor precisión y percepción de ámbito que lo rodea aún se encuentre en condiciones complejas, con poca visibilidad o con gran cantidad de obstáculos (Varriale, Cammarano, Michelino, y Caputo, 2023). Así mismo permite que la toma de decisiones en tiempo real sea mucho más acertada y rápida lo cual en el ámbito de la automoción es vital (Varriale y cols., 2023).

4.7. Aplicaciones de visión por computadora en sistemas de asistencia al conductor

4.7.1. Sistemas avanzados de asistencia al conductor (ADAS)

En los sistemas avanzados de asistencia al conductor (*Advanced Driver Assistance Systems*, ADAS) son tecnologías que logran mejorar la seguridad y la experiencia de conducción usando la automatización y adaptación explotando las capacidades de un vehículo (Dong y Cappuccio, 2023). Dichos sistemas utilizan una combinación de sensores como LIDAR, cámaras, radar y otros. Adicionalmente se utilizan sistemas internos de visión por computadora que procesan la información capturada por los sensores y procesar los datos en tiempo real, permitiendo a hacer tareas como la detección de mantenimiento de carril, detección de objetos y frenado de emergencia (Dong y Cappuccio, 2023).

Una de las aplicaciones más destacadas de ADAS es la detección de objetos y obstáculos en la carretera, donde la visión por computadora juega un papel crucial al analizar las imágenes capturadas por las cámaras del vehículo (Hemaanand y cols., 2020). Mediante algoritmos avanzados de procesamiento de imágenes y aprendizaje profundo, estos sistemas pueden identificar y clasificar objetos como otros vehículos, peatones y señales de tráfico, mejorando significativamente la seguridad del conductor y los pasajeros (Dong y Cappuccio, 2023).

Además, los ADAS basados en visión por computadora se han beneficiado de los recientes avances en tecnologías de aprendizaje profundo, que han mejorado la precisión y robustez de estos sistemas en condiciones de conducción difíciles, como poca luz o condiciones meteorológicas adversas. Estos sistemas no solo aumentan la seguridad, sino que también contribuyen al desarrollo de tecnologías más avanzadas hacia la conducción autónoma (Dong y Cappuccio, 2023).

4.7.2. Detección y reconocimiento de señales de tránsito en ADAS en vehículos terrestres del mercado

Los vehículos autónomos y semi-autónomos interpretan correctamente la señalización de tránsito y respondan adecuadamente a las condiciones del entorno con el uso de los sistemas ADAS. Los métodos más nuevos, para la detección de señales de tránsito utilizan algoritmos de aprendizaje profundo. Un ejemplo son las redes neuronales convolucionales (CNN), que han demostrado ser altamente eficaces en el reconocimiento de patrones complejos bajo diversas condiciones de iluminación y en presencia de otros obstáculos en la carretera. Estos sistemas son capaces de identificar diferentes tipos de señales, con una alta precisión (Triki, Karray, y Ksantini, 2023).

La integración de la detección de señales de tránsito con los sistemas ADAS no solo permite que el vehículo reconozca las señales, sino que también dar una respuesta y accionar mecanismos sobre la detección de ellas. Algunos ejemplos de esto es ajustar la velocidad del vehículo en respuesta a una señal de límite de velocidad o cambiando de carril de manera segura cuando se detecta una señal de advertencia. Esto reduce significativamente el riesgo de accidentes causados por errores humanos al conducir (Lim y cols., 2023).

4.7.3. Detección de peatones y otros vehículos en ADAS en vehículos terrestres del mercado

La capacidad de detección de peatones y otros vehículos es una parte primordial en los sistemas ADAS ya que con esto buscamos garantizar la seguridad en la conducción autónoma y semi-

autónoma. Los algoritmos de detección de objetos han mejorado significativamente la capacidad para detectar dichos objetos en diversas situaciones aún cuando las condiciones sean difíciles debido a la iluminación o por oclusión (Galvao, Abbod, Kalganova, Palade, y Huda, 2021).

Se debe de tomar en cuenta factores como la variabilidad de la apariencia de los peatones y que al ser seres humanos estos siguen un patrón de movimiento relativamente impredecible. Sin embargo los avances en el campo de los sistemas ADAS a mostrado una gran capacidad para lograr la detección de personas haciendo que el riesgo de accidentes sea bajo (Iftikhar y cols., 2022). De igual forma el detectar otros objetos y vehículos es crucial para la toma de decisiones en cuestión de mili segundos de los sistemas ADAS pues de esto puede por ejemplo tomarse la decisión de accionar el freno de emergencia y posiblemente salvar la vida de un peatón. Al estar diseñados para operar en diversas condiciones y ámbitos variados la navegación con estos sistemas brindan una protección al usuario constantemente (Iftikhar y cols., 2022).

4.7.4. Alertas y recomendaciones para el conductor en ADAS en vehículos terrestres del mercado

Los ADAS en vehículos terrestres modernos integran alertas y recomendaciones para el conductor con el objetivo de mejorar la seguridad y la eficiencia en la conducción. Estas alertas se generan a partir de la interpretación de los datos obtenidos en los múltiples sensores que monitorean constantemente el entorno del vehículo. Dentro de las alertas podemos mencionar advertencias sobre la proximidad de otros vehículos, cambios de carril no intencionados, o la presencia de peatones y obstáculos en la carretera. Por ejemplo, en situaciones de tráfico denso o durante maniobras de estacionamiento, los ADAS pueden emitir alertas visuales o auditivas para prevenir colisiones al detectar la proximidad del vehículo a un objeto (Samantaray, 2023). Además, los sistemas ADAS pueden ofrecer recomendaciones al conductor, como reducir la velocidad o mantener una distancia segura con el vehículo de adelante, basadas en la información procesada en tiempo real (Grant, 2024).

Estas alertas son efectivas de acorde a la precisión con la que los sensores capturan y procesan los datos, así como de la capacidad del sistema para interpretar correctamente el contexto de conducción. Los avances recientes en visión por computadora y aprendizaje profundo han mejorado significativamente la capacidad de los ADAS para emitir alertas más precisas y oportunas, contribuyendo así a una conducción más segura y a la reducción de accidentes (Samantaray, 2023).

4.8. Desafíos y soluciones en visión computacional

4.8.1. Condiciones adversas de iluminación y clima

En la visión computacional, uno de los mayores desafíos es el manejo de condiciones adversas de iluminación y clima, como baja iluminación, niebla, lluvia intensa o nieve. Estas condiciones pueden disminuir a gran escala la calidad de las imágenes capturadas por los sensores, lo que dificulta la correcta interpretación y procesamiento de la información por parte de los algoritmos de visión computacional (Özdenizci y Legenstein, 2023). Para abordar estos desafíos, se han desarrollado diversas soluciones. Entre ellas, destacan los modelos de restauración de imágenes basados en aprendizaje profundo, como los transformadores de visión, que recuperan detalles importantes en imágenes degradadas (Özdenizci y Legenstein, 2023). Estos modelos pueden manejar múltiples tipos de degradaciones climáticas y de iluminación en un solo marco de trabajo, mejorando así la precisión de la detección y reconocimiento de objetos en condiciones adversas (Gao, Zou, Li, y Guo, 2024).

4.8.2. Variabilidad en las señales de tránsito y entornos

La variabilidad en las señales de tránsito y los entornos representa un desafío significativo para los sistemas de visión computacional en vehículos autónomos y sistemas avanzados ADAS (Azfar y cols., 2024). Esta variabilidad incluye diferencias en el diseño, formato de las señales, lenguaje utilizado en las mismas y hasta la posición y condición que se encuentre la misma. También podemos mencionar cambios en las condiciones de iluminación, y la presencia de elementos distractores en el entorno como podría ser un camino obstaculizado por un árbol caído, una señal de tránsito manchada o luz excesiva de otros vehiculos siendo detectada por los sensores (Azfar y cols., 2024). Estos factores pueden afectar la precisión con la que los sistemas reconocen e interpretan las señales de tránsito, lo que es crucial para la seguridad y eficacia de la conducción autónoma (Yu y Ye, 2023).

Para abordar estos desafíos, se han desarrollado técnicas avanzadas basadas en el aprendizaje profundo que son capaces de mejorar la robustez del reconocimiento de señales de tránsito al adaptarse a las variaciones en las condiciones ambientales y de diseño (Lim y cols., 2023). Además, la creación y utilización de conjuntos de datos más diversos y representativos, que incluyan una amplia gama de escenarios y condiciones, es esencial para entrenar modelos que puedan generalizar mejor y mantener un alto rendimiento en entornos reales (Lim y cols., 2023).

4.8.3. Problemas de escalabilidad y eficiencia

En el ámbito de la visión computacional, los problemas de escalabilidad y eficiencia representan desafíos significativos, especialmente cuando se implementan en sistemas que deben manejar grandes volúmenes de datos o funcionar en dispositivos con recursos limitados, como los dispositivos móviles o los sistemas embebidos. La escalabilidad se refiere a la capacidad de un sistema para adaptarse a un aumento en la carga de trabajo sin perder eficiencia, mientras que la eficiencia se relaciona con el uso óptimo de los recursos computacionales para mantener un rendimiento adecuado (Y. Wang y cols., 2023).

Uno de los mayores desafíos en la escalabilidad de la visión computacional, particularmente en el aprendizaje profundo, es el aumento exponencial en la complejidad computacional a medida que los modelos se vuelven más grandes y sofisticados. Esto es particularmente evidente en las redes neuronales profundas, donde el número de parámetros puede llegar a ser extremadamente alto, aumentando la demanda de datos y recursos computacionales (Singh, 2021). Además, la implementación de estos modelos en dispositivos con limitaciones de recursos, como teléfonos móviles o drones, presenta un desafío adicional debido a las restricciones en el poder de procesamiento y la memoria (Y. Wang y cols., 2023).

Las técnicas de aprendizaje multitarea y el uso de múltiples hilos de procesamiento (*threads*) de procesamiento permiten compartir recursos entre diferentes tareas, lo que mejora la eficiencia y la escalabilidad de los sistemas de visión computacional (Singh, 2021). Estos avances son cruciales para garantizar que estos sistemas puedan escalar efectivamente y operar eficientemente en una amplia gama de aplicaciones y dispositivos, manteniendo un rendimiento robusto incluso en condiciones limitadas.

4.8.4. Estrategias de mitigación y mejoras tecnológicas

Las estrategias de mitigación y mejoras tecnológicas son cruciales para superar los desafíos presentes en la visión computacional, especialmente en aplicaciones como los sistemas ADAS. Uno de los principales enfoques es la optimización de modelos, donde se utilizan técnicas como el *pruning* y la cuantización para reducir el tamaño y la complejidad de los modelos sin sacrificar significativamente el rendimiento (Riggs y cols., 2023). Estas técnicas permiten que los modelos sean más

eficientes en términos de uso de memoria y procesamiento, lo cual es esencial para su implementación en dispositivos con recursos limitados, como los sistemas embebidos en vehículos.

Otra estrategia clave es la fusión de datos multisentoriales, que integra la información proveniente de diferentes tipos de sensores para mejorar la precisión y robustez de los sistemas. Esta integración permite compensar las debilidades de un tipo de sensor con las fortalezas de otro, mejorando la capacidad del sistema para operar en condiciones ambientales adversas (Hu, 2023). Además, el uso de técnicas de aprendizaje profundo, como los transformadores de visión y el entrenamiento para reconocer patrones complejos y realizar tareas de visión computacional con alta precisión es una esencial para mejorar y mitigar los problemas de escalabilidad y eficiencia de los modelos.

4.9. Relevancia y futuro de la visión computacional en la seguridad vial

4.9.1. Impacto de la tecnología en la reducción de accidentes de tráfico

El impacto significativo en la reducción de accidentes de tráfico gracias a los sistemas ADAS es posible debido a su alto nivel de detección en tiempo real de posibles peligros en la carretera. Dichos sistemas pueden tomar medidas preventivas llevando a que se tomen decisiones en tiempo real reduciendo significativamente el riesgo de colisiones (Aleksa y cols., 2024). Según la *National Highway Traffic Safety Administration* (NHTSA por sus siglas en inglés), la implementación de sistemas ADAS en vehículos ha disminuido la frecuencia de accidentes en aproximadamente un 20 % a un 25 % en situaciones de tráfico urbano, donde los riesgos son más altos debido a la congestión y la interacción constante con peatones y otros vehículos (on Crash Reporting, 2022).

Según la *National Highway Traffic Safety Administration* (NHTSA por sus siglas en inglés), la implementación de ADAS en vehículos ha reducido la frecuencia de accidentes en un 20 por ciento a 25 por ciento en situaciones de tráfico urbano, donde los riesgos son más altos debido a la congestión y la interacción constante con peatones y otros vehículos (on Crash Reporting, 2022). Además, las estadísticas indican que los sistemas de advertencia de colisión y frenado automático han disminuido la gravedad de los accidentes en un 30 por ciento a 40 por ciento, reduciendo tanto el número de víctimas como la magnitud de los daños (Antony y Whenish, 2021).

El impacto de estos sistemas se ha observado en mercados como Europa y América del Norte, donde la adopción de ADAS ha contribuido a una disminución significativa en las tasas de mortalidad por accidentes de tráfico, según datos de la European Transport Safety Council (ETSC) (for Transport, 2018). Esto consolida la importancia de seguir desarrollando e implementando estas tecnologías a nivel global.

CAPÍTULO 5

Marco metodológico

5.1. Definición general del sistema

Se desarolló un sistema que integra diversos modelos de aprendizaje automático y procesos automáticos de procesamiento de datos que permitió la detección y análisis de señales de tránsito, junto con datos simulados que emulan los sensores de un vehículo en una simulación con agentes. Estos componentes interactuaron de manera coordinada para formar un sistema unificado capaz de recibir imágenes seleccionadas aleatoriamente del set de datos recopilado, procesarlas mediante técnicas de visión por computadora y complementarlas con datos de sensores como la aceleración y la velocidad del vehículo, los cuales, dada la naturaleza de este proyecto, se simulan. El objetivo primordial del sistema fue la generación de alertas personalizadas basadas tanto en las señales detectadas como en la información proporcionada por los sensores simulados.

Sin embargo, el sistema no contempla el desarrollo de una interfaz gráfica de usuario (GUI, por sus siglas en inglés). Todas las operaciones se llevaron a cabo en el nivel *backend*, mediante el uso de *scripts* y *notebooks* implementados en entornos de programación de Python. El sistema no se probó en un entorno físico o real, sino exclusivamente en un entorno simulado. Esto establece las bases de lo que en un futuro podría ser un sistema avanzado de asistencia a la conducción (ADAS, por sus siglas en inglés) físicamente instalado en cualquier vehículo terrestre aún cuando este no cuente con la tecnología moderna para ello. Las pruebas y simulaciones realizadas permitieron evaluar la capacidad y el rendimiento de los modelos, sin necesidad de implementar el sistema en un vehículo real ni someterlo a condiciones de tráfico reales, lo cual evitó cualquier riesgo para el usuario o conductor durante las fases de prueba.

Finalmente, el sistema no utilizó una base de datos externa, a excepción del conjunto de datos unificado generado previamente para entrenar y validar los modelos. Todo el procesamiento de los datos se realizaron de forma local, gestionando tanto las imágenes como los datos simulados dentro de la estructura de archivos del proyecto la cual pudo ser replicada al correr los *notebooks* correspondientes. Esta decisión permitió simplificar el diseño del sistema, eliminando dependencias externas, y concentrar los recursos en el ajuste fino de los modelos, así como en el desarrollo de un *pipeline* eficiente para la detección y clasificación de señales de tránsito y la respuesta ante los *inputs* de los sensores.

5.2. Conjunto de datos

El conjunto de datos utilizado en este proyecto fue construido con la unificación de dos flujos de fuentes principales. La primera proviene de un set de datos proporcionado por el usuario Tesis en la plataforma Roboflow, conteniendo una variedad de imágenes de señales de tránsito, semáforos peatonales, pasos de cebra y demás objetos característicos en el ámbito colombiano. Estos datos fueron preprocesados y organizados en clases. La segunda fuente fue una recopilación propia de imágenes capturadas manualmente en escenarios reales de conducción, además de videos obtenidos de la plataforma YouTube del canal CityCorners World Walkers donde se realiza un tour por las calles de la ciudad Guatemala en vehículo. Esta combinación permitió enriquecer el conjunto de datos, añadiendo diversidad y un mayor rango de condiciones reales, adaptándolo al contexto urbano.

La unificación de estos dos conjuntos de datos fue un proceso que permitió aumentar la diversidad como el volumen del set de datos utilizados en el proceso de desarrollo. Se debe mencionar que la unificación de las fuentes implicó la normalización de etiquetas entre ambas fuentes para asegurar la consistencia en las categorías. Para mejorar aún más la capacidad del sistema de reconocer señales de tránsito en diversas condiciones, se aplicaron técnicas de preprocesamiento que incluyeron la homogeneización de las dimensiones de las imágenes. Además, se utilizaron técnicas de aumento de datos (data augmentation), tales como ajustes de brillo, rotaciones controladas, redimensionamiento y la adición de ruido, con el fin de simular variaciones ambientales y condiciones adversas, como iluminación deficiente o malas condiciones climáticas.

5.2.1. Entradas del modelo de detección de objetos

El sistema desarrollado en este proyecto utilizó diversos tipos de entradas (*inputs*) que alimentan los modelos y procesos involucrados en la detección de señales de tránsito y la simulación de las condiciones del vehículo. Entre los principales *inputs* se encuentran las imágenes estáticas y los videos de conducción. Cada tipo de *input* pasa por un proceso de transformación y análisis específico, lo que permitió al sistema integrar múltiples fuentes de información para generar respuestas precisas y personalizadas.

Los videos de conducción, son procesados mediante un *pipeline* que los transforma en secuencias de imágenes fijas. El proceso consiste en dividir los videos en cuadros (*frames*) individuales, de los cuales se extraen imágenes a intervalos regulares (por ejemplo, cada 20 cuadros). Estas imágenes son posteriormente analizadas por un modelo de detección de objetos de YOLOv5, que identifica los objetos presentes en cada imagen. Cabe mencionar que se ha tomó una versión preentrenada de dicho modelo aprovechando la capacidad del mismo para generalizar múltiples categorías. A partir de este análisis, se pudo determinar la clase del objeto, su posición y tamaño dentro de la imagen lo que permitió etiquetar dicha información especifica. Con esto luego se hizo un recorte únicamente tomando a aquellos datos de clases de interés donde se destacó señal de alto, semáforo y otras.

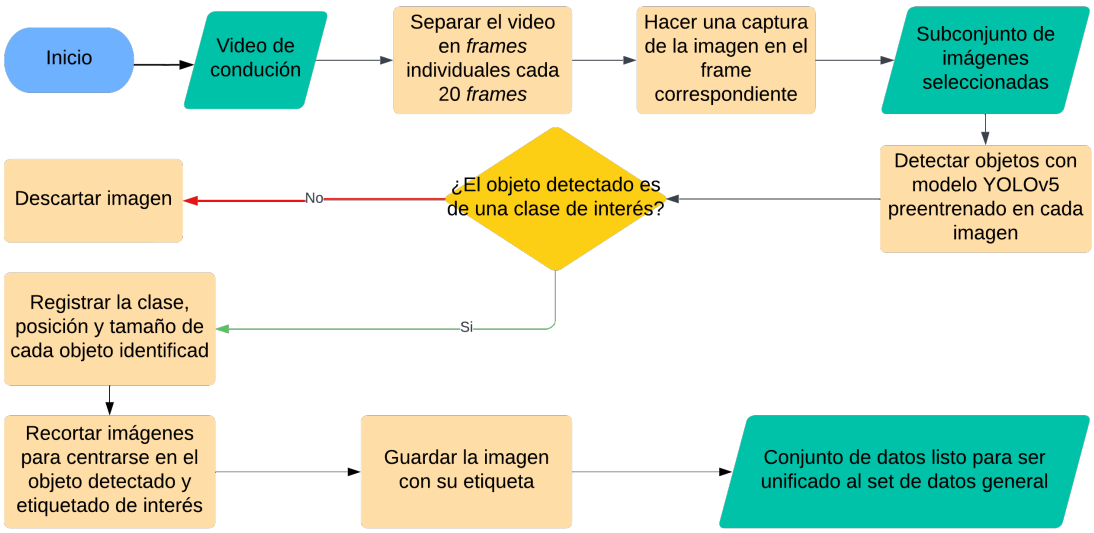

Figura 5.1: *Pipeline* de procesamiento de videos de conducción a imágenes etiquetadas mediante detección de objetos con YOLOv5.

(Elaboración propia)

En este proyecto los datos obtenidos de los sensores fueron simulados aceleración y velocidad del vehículo, los cuales complementan la información visual para proporcionar un contexto más completo al sistema. Por ejemplo, si el sistema detecta una señal de Alto mientras el sensor simulado indica que el vehículo está desacelerando, se genera una alerta personalizada que refuerza la necesidad de detener el vehículo. Por otro lado, si el vehículo está acelerando y se detecta la misma señal, el sistema genera una alerta de peligro para ayudar al conductor a tomar la decisión adecuada y garantizar su seguridad.

Aunque en este proyecto los datos de los sensores fueron simulados, se exploró la posibilidad de utilizar sensores físicos representados en los archivos Python ubicada en el repositorio. En la carpeta de nombre sensores_input se encuentran varias implementaciones para sensores como el ADXL345, BN055, LSM303D y MPU-6050, lo que permitiría en el futuro integrar estos dispositivos para la obtención de datos reales. A pesar de que estas opciones aún no se han implementado, el sistema está diseñado para ser compatible con diferentes fuentes de datos sensoriales, tanto simulados como reales.

A continuación, se presentan ejemplos de algunas de las clases de señales que el modelo fue capaz de detectar:

- **Alto**
  La señal de Alto indica la necesidad de detener el vehículo por completo ante una intersección o cruce. Es fundamental que el vehículo reduzca su velocidad hasta detenerse por completo, independientemente de la presencia de otros vehículos o peatones.

Figura 5.2: Señal de Alto.

(Elaboración propia)

- **Semáforo peatonal en rojo**
  Esta señal luminosa indica que los peatones no deben cruzar, lo cual también puede requerir que el vehículo esté atento a que ningún peatón cruce antes de avanzar.

Figura 5.3: Semáforo peatonal en rojo.

(Elaboración propia en base a imagen de set de datos de (Tesis, 2022))

- **Semáforo peatonal en verde**
  El semáforo peatonal en verde indica que los peatones tienen permitido cruzar. Para el vehículo, esto implica que debe detenerse para ceder el paso a los peatones.

Figura 5.4: Semáforo peatonal en verde.

(Elaboración propia en base a imagen de set de datos de (Tesis, 2022))

- **Semáforo vehicular en rojo**
  Esta señal luminosa indica que los vehículos deben detenerse por completo, lo que impide su avance hasta que la luz cambie a verde.

Figura 5.5: Semáforo vehicular en rojo tipo 1.

(Elaboración propia)

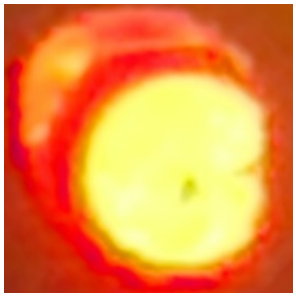

Figura 5.6: Semáforo vehicular en rojo tipo 2.

(Elaboración propia)

- **Semáforo vehicular en verde**
  El semáforo vehicular en verde indica que los vehículos pueden avanzar con normalidad, respetando el flujo del tráfico.

Figura 5.7: Semáforo vehicular en verde.

(Elaboración propia en base a imagen recopilada de video de (CityCorners, 2024))

- **Semáforo vehicular en amarillo**
  Esta señal indica que el semáforo pronto cambiará a rojo, por lo que los vehículos deben reducir su velocidad y prepararse para detenerse.

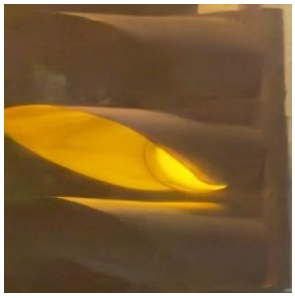

Figura 5.8: Semáforo vehicular en amarillo.

(Elaboración propia)

5.2.2. Visualización y análisis previo del set de datos

La visualización de los datos existentes en el conjunto de datos unificado fue de mucho valor para comprender el mismo. Inicialmente, se realizó una extracción de las dimensiones de las imágenes para obtener una visión clara de los tamaños y proporciones de las mismas. Estas dimensiones fueron procesadas y visualizadas mediante un histograma para graficar el ancho y alto de las imágenes del conjunto. Cabe mencionar que en el proceso de *data augmentation* de datos se aplicaron ciertos variaciones a las imágenes normalizando tres posibles tamaños para las mismas. Como se observa en la figura 5.9 la mayoría de las imágenes tienen dimensiones concentradas en un rango alrededor de los 256 píxeles. Se observó también dos picos principales: uno en 200 píxeles y otro en torno a los 300 píxeles.

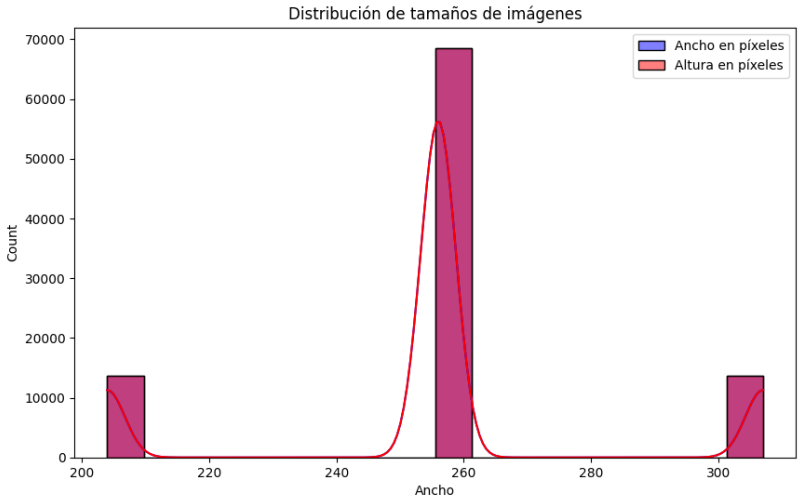

Figura 5.9: Distribución de tamaños de imágenes en términos de su ancho y alto dentro del conjunto de datos.

(Elaboración propia)

Además de la visualización de las dimensiones de las imágenes, se generó el gráfico en la figura 5.10 que muestra la distribución inicial de clases dentro del conjunto de datos. Para ello, se realizó un conteo de las etiquetas presentes en los archivos asociados a cada imagen. Este análisis permitió verificar el equilibrio del *dataset* para determinar si existe una representación adecuada de cada clase. Las clases que tienen el mayor número de imágenes etiquetadas son pedestrian_traffic_light_green, y pedestrian_traffic_light_red, con más de 25,000 y 20,000 imágenes respectivamente. Estas clases representan las señales de semáforo peatonal en verde y en rojo. Por otro lado, las clases traffic_light_red, traffic_light_green, y stop tienen menos ejemplos. Finalmente, la clase con menor presencia fue traffic_light_yellow, con menos de 5,000 imágenes etiquetadas.

Con esto se determinó que existía cierto desequilibrio entre las clases por lo que se tomó la decisión de llevar a cabo el proceso *data augmentation* unicamente a las clases que contienen pocos elementos y posteriormente submuestreo para igualar la cantidad de elementos existentes por cada clase del conjunto de datos.

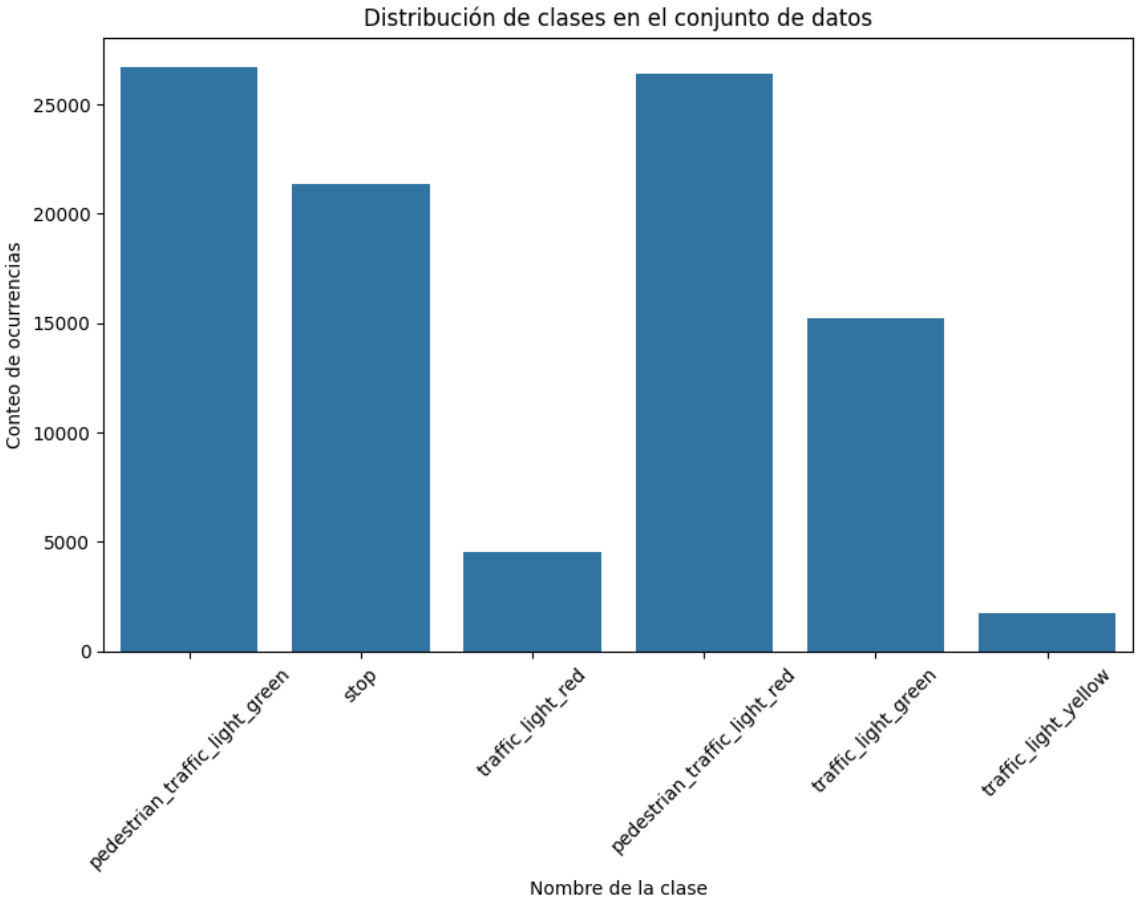

Figura 5.10: Distribución inicial de clases dentro del conjunto de datos.

(Elaboración propia)

Finalmente, se realizó un ultimo histograma (5.11) lo cual indica que casi todas las imágenes tienen únicamente un objeto etiquetado

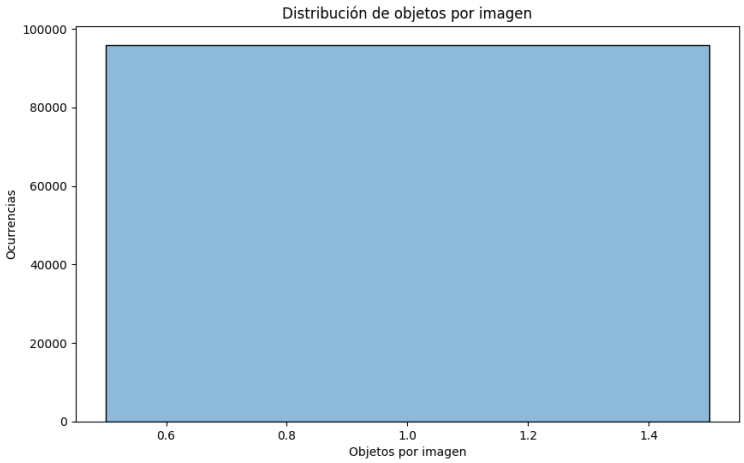

Figura 5.11: Distribución inicial de objetos en el conjunto de datos.

(Elaboración propia)

5.3. Preprocesamiento

El preprocesamiento de los datos fue una etapa importante en el desarrollo del sistema, ya que permitió transformar el conjunto de datos para luego analizarlo y utilizarlo en el entrenamiento del modelo de detección de objetos. Se realizaron diversas técnicas de preprocesamiento para tener la mejor calidad de las imágenes así como la representación de las etiquetas correspondientes a cada clase. El primer paso fue normalizar los nombres de archivo de las imágenes en el conjunto de datos unificado, asegurando que no contuvieran caracteres especiales que dificultaran la lectura de los mismos. Luego, se ajustaron todas las imágenes a una resolución estándar de 256x256 píxeles, lo cual ayudó a evitar sesgos causados por variaciones en el tamaño de las imágenes y aseguró una entrada consistente para el modelo. Adicionalmente, se generaron archivos de texto (.txt) con las etiquetas asignadas a cada objeto detectado por YOLOv5 en el conjunto de datos propio, facilitando la identificación y clasificación de los objetos en cada imagen.

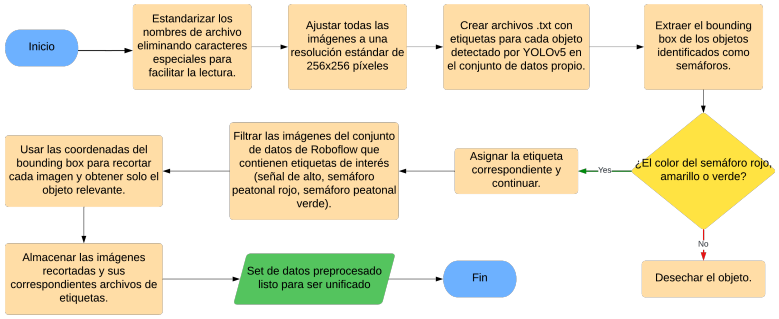

Figura 5.12: Diagrama de flujo del proceso de preprocesamiento del conjunto de datos.

(Elaboración propia)

Para los objetos identificados como semáforos, se implementó un análisis de color para determinar su estado (rojo, amarillo o verde). Este proceso consistió en analizar las áreas del semáforo detectado

mediante un *pipeline* que evaluaba los rangos de color en el promedio de todos los píxeles en la imagen. Los valores de color se compararon con rangos predefinidos, donde cada rango correspondía a un color específico: rojo, amarillo o verde. Si los valores de color predominante coincidían con los rangos establecidos para uno de estos colores, se asignaba la etiqueta correspondiente al semáforo en cuestión. Este enfoque de detección de color fue esencial para mejorar la precisión en la clasificación de semáforos, permitió que el modelo diferenciara los estados de los mismos y en la generación de alertas relevantes en situaciones simuladas de tráfico.

Además, se tomaron las imágenes y etiquetas de las carpetas de entrenamiento, validación y test del conjunto de datos de Roboflow y se realizó un mapeo de sus etiquetas, donde se identificó que en el archivo data.yaml, el código 0 representa una señal de alto, el código 2 representa un semáforo peatonal en rojo y el código 4 representa un semáforo peatonal en verde. Se seleccionaron únicamente las imágenes que contenían estas etiquetas y, utilizando las coordenadas del *bounding box* incluidas en los archivos de etiquetas, se recortaron las imágenes para obtener exclusivamente el objeto de interés. Estas imágenes recortadas, junto con sus nuevas etiquetas en archivos de texto, fueron almacenadas para definir con precisión los objetos relevantes para el proyecto, para luego ser unificadas con el set de datos generado.

5.3.1. Técnicas de *Data augmentation* aplicadas a set de datos

El uso de técnicas de aumento de datos permitió incrementar la variabilidad del conjunto de datos sin la necesidad de recolectar nuevas imágenes. Estas técnicas consistieron en aplicar transformaciones sobre las imágenes originales, generando variaciones controladas que mejoran la capacidad del modelo para generalizar a situaciones del mundo real. A continuación, se describen las principales técnicas de *data augmentation* que fueron implementadas.

- **Imagen original**
  Se muestra la imagen de una señal de alto sin ningún filtro o transformación para referencia del proceso de *data augmentation*.

Figura 5.13: Imagen original sin filtros o transformaciones aplicadas.

(Elaboración propia)

- **Rotación**
  En la primera fase, se realizaron ajustes de rotación en la imagen para simular las variaciones en donde una señal de tránsito o semáforo pueden estar torcidas en el ámbito urbano.

Figura 5.14: Imagen rotada 90 grados.

(Elaboración propia)

Figura 5.15: Imagen rotada 180 grados.

(Elaboración propia)

Figura 5.16: Imagen rotada 270 grados.

(Elaboración propia)

- **Ajustes de brillo**
  En la segunda fase, se realizaron ajustes de brillo para simular condiciones de iluminación variables. Estos ajustes incluyeron brillo alto y bajo.

Figura 5.17: Imagen con brillo alto.

(Elaboración propia)

Figura 5.18: Imagen con brillo bajo.

(Elaboración propia)

- **Ruido para Simular Lluvia**
  Para representar condiciones adversas de lluvia, se añadió ruido a las imágenes, imitando gotas de agua en la cámara.

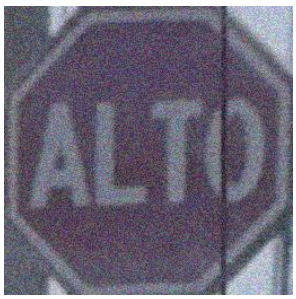

Figura 5.19: Imagen con ruido para simular lluvia.

(Elaboración propia)

- **Redimensionamiento**
  Se realizaron ajustes de tamaño, aumentando y disminuyendo las dimensiones de la imagen, para simular variaciones en la distancia y perspectiva de los objetos.

Figura 5.20: Imagen redimensionada con aumento de tamaño.

(Elaboración propia)

Figura 5.21: Imagen redimensionada con reducción de tamaño.

(Elaboración propia)

El proceso de preprocesamiento y aumento de datos observado en la Figura 5.23 se llevó a cabo en dos fases para maximizar la variabilidad del conjunto de datos. Inicialmente, se comenzó con 4,569 imágenes, las cuales fueron sometidas a una primera fase de rotaciones (90°, 180° y 270°). Posteriormente, en la segunda fase, se aplicaron transformaciones adicionales, como ajustes de brillo, adición de ruido para simular condiciones adversas y redimensionamientos para simular distintas perspectivas. Estas técnicas permitieron que el tamaño del conjunto de datos creciera significativamente, alcanzando un total de 42,351 imágenes finales. Durante este proceso, se prestó especial atención a las imágenes etiquetadas como traffic_light_yellow, a las cuales se les aplicaron augmentaciones adicionales para mejorar la representatividad de esta clase específica. Este incremento en la cantidad y diversidad de las imágenes fue crucial para mejorar la capacidad del modelo de generalizar en condiciones diversas y complejas del mundo real. Posterior a este proceso se puede observar en la Figura 5.22 se observó que las clases como pedestrian_traffic_light_green y pedestrian_traffic_light_red tienen la mayor representación, con más de 11,000 imágenes cada una. La clase traffic_light_green también está bien representada con más de 6,500 imágenes, mientras que las clases traffic_light_red y traffic_light_yellow, que inicialmente tenían una menor representación, se han visto ligeramente aumentadas, alcanzando cerca de 1,950 imágenes cada una.

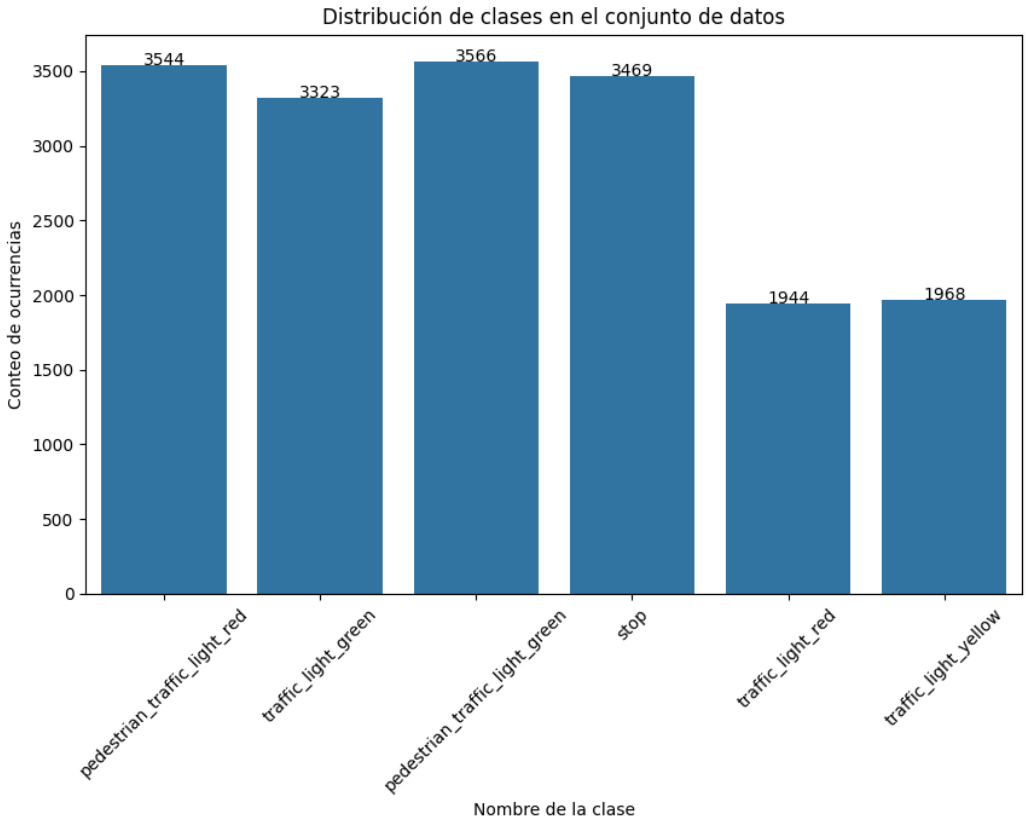

Figura 5.22: Distribución de clases en el conjunto de datos después de la aplicación de un proceso de data augmentation.

(Elaboración propia)

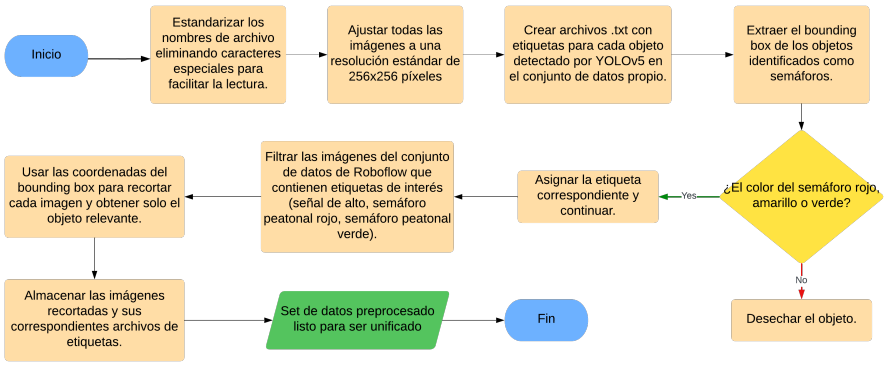

Figura 5.23: Diagrama de flujo del proceso de *data augmentation* del conjunto de datos.

(Elaboración propia)

5.3.2. Técnicas de *Undersampling* aplicadas a set de datos

El proceso de *undersampling* observado en la figura 5.24 se implementó para equilibrar el conjunto de datos reduciendo la cantidad de elementos en las que mostraron mayor representación. Primero, se recorrieron las etiquetas de cada imagen para clasificar las imágenes según la clase a la que pertenecen. Luego, se identificó la clase con menor número de imágenes, estableciendo esta cantidad como el límite para todas las demás clases. A continuación, las clases con más imágenes fueron submuestreadas aleatoriamente para igualar el número mínimo, buscando que ninguna clase tuviera una representación desproporcionada.

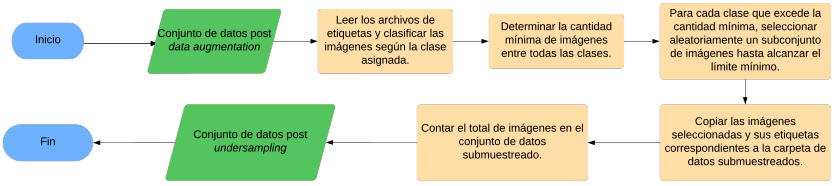

Figura 5.24: Diagrama de flujo del proceso de *undersampling* del conjunto de datos.

(Elaboración propia)

5.3.3. Separación del set de datos en conjuntos de entrenamiento, validación y verificación

El proceso de separación del conjunto de datos, observado en la figura 5.25, se inició leyendo las imágenes y etiquetas donde se aplicó un método de estratificación, asegurando que cada conjunto (entrenamiento, validación y prueba) contenga representaciones equitativas de todas las clases. En primer lugar, el conjunto se dividió en un 70 % para entrenamiento y un 30 % para validación y prueba. Posteriormente, el 30 % restante se separó en partes iguales, asignando un 15 % de los datos al conjunto de validación y otro 15 % al conjunto de prueba. Se optó por esta distribución pues se buscaba maximizar la cantidad de datos para el proceso de entrenamiento del modelo siempre manteniendo un porción representativa para el proceso de validación y prueba.

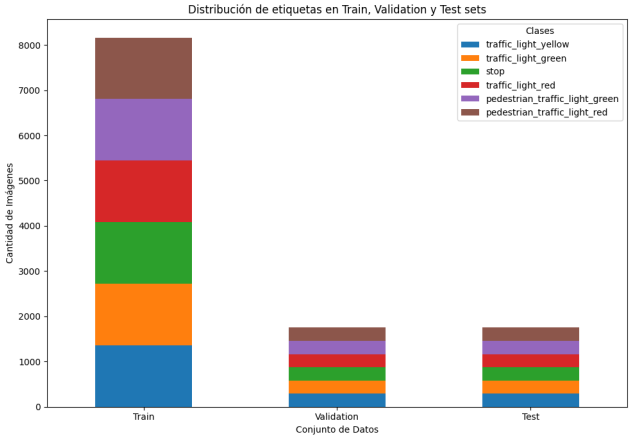

Figura 5.25: Distribución de clases en los conjuntos de entrenamiento, validación y verificación.

(Elaboración propia)

5.4. Sistema de asistencia de manejo de un vehículo terrestre primera versión

La versión inicial del sistema de asistencia de manejo utilizó un modelo de detección de objetos basado en YOLOv8 entrenado para identificar señales de tránsito y semáforos. El modelo se complementa con un simulador de sensores de velocidad que imitan condiciones de aceleración y desaceleración de un vehículo en situaciones de tráfico en un entorno urbano. La integración de ambos elementos permitió al sistema generar alertas cuando detecta objetos peligrosos mientras el vehículo se desplaza en el entorno simulado. El modelo de YOLOv8, seleccionado por su eficiencia en aplicaciones en tiempo real, ofreció detección de objetos cuadro a cuadro, lo cual fue esencial para responder a eventos de manera inmediata.

5.4.1. Modelo de detección de objetos empleado en primera versión del sistema

Se empleó el modelo YOLOv8n para realizar detección de señales de tráfico en tiempo real. Este modelo fue seleccionado debido a su bajo requerimiento de recursos, lo cual fue ideal para aplicaciones en dispositivos de baja potencia. Durante el proceso de entrenamiento, se emplearon cinco épocas y un tamaño de lote de 8, ajustando los hiperparámetros para maximizar la eficiencia sin comprometer la precisión. La red YOLOv8n fue entrenada con el conjunto de datos generado previamente para detectar señales de alto y semáforos. Esto optimizó el rendimiento para situaciones de tráfico urbano en Guatemala. Tras el entrenamiento, se evaluó el modelo en el conjunto de prueba, lo que permitió verificar su precisión y su capacidad para generalizar en imágenes nuevas.

5.4.2. Simulación de sensores empleada en primera versión del sistema

En la simulación de sensores de la primera versión del sistema se incluyó una función que simula la velocidad del vehículo, imitando los patrones de aceleración y desaceleración típicos en un entorno urbano. El simulador aplicó una lógica para imitar detenciones aleatorias en semáforos o señales de alto, haciendo que el vehículo se desacelere gradualmente antes de detenerse completamente. Esta simulación permitió que el sistema reaccione a velocidades cambiantes, ajustándose a condiciones realistas de tráfico. De igual forma, en el simulador se incluyó una pequeña fluctuación aleatoria en la velocidad para reflejar las variaciones comunes en la conducción real. La combinación de estas características permitió que el sistema emita alertas relevantes cuando el vehículo sobrepasa el límite de velocidad o cuando se aproxima a un objeto peligroso.

5.5. Sistema de asistencia de manejo de un vehículo terrestre segunda versión

La segunda versión del sistema de asistencia de manejo incorporó cambios en el modelo de detección de objetos como en el simulador de sensores de velocidad. Se añadieron alertas detalladas que varían según el tipo de objeto detectado y el contexto de la velocidad, ofreciendo al usuario instrucciones más específicas en función de cada situación. A diferencia de la versión anterior, que solo generaba una alerta simple al detectar un objeto peligroso, esta versión fue capaz de diferenciar entre una alerta crítica y una advertencia, dependiendo del objeto y la velocidad del vehículo. Esta funcionalidad nueva permitió alertas precisas para señales de alto y semáforos rojos, sugiriendo al usuario reducir la velocidad o detenerse según sea necesario. La salida de la segunda versión muestra

mensajes de alerta detallados, como advertencias para semáforos rojos y semáforos peatonales, a diferencia de la simple alerta "stop sign"de la primera versión. Este sistema actualizado proporciona un registro detallado de cada evento, mejorando el análisis de comportamiento del vehículo en el entorno simulado.

5.5.1. Modelo de detección de objetos empleado en segunda versión del sistema

Para la segunda versión del sistema se utilizó el modelo YOLOv8n con el mismo proceso de entrenamiento que la primera versión del sistema elaborada, con un enfoque en la identificación rápida y eficiente de señales de tráfico. La precisión y el bajo costo computacional de YOLOv8n permitieron que el sistema funcione en tiempo real, capturando y clasificando objetos de interés en cada cuadro de video haciéndolo una opción que brinda valor como componente de esta nueva versión del sistema.

5.5.2. Simulación de sensores empleada en segunda versión del sistema

La simulación de sensores de velocidad en la segunda versión del sistema se mejoró para reflejar situaciones de tráfico más realistas, incluyendo congestión o alta afluencia vehicular. Además de simular las paradas en semáforos y señales de alto, el sistema ahora tambien poseía la capacidad de reducir la velocidad máxima cuando detecta condiciones de tráfico congestionado, añadiendo una capa de realismo. En esta función se incorporó un temporizador aleatorio para activar períodos de congestión que limitan la velocidad a un umbral seguro, imitando las fluctuaciones del tráfico urbano en la ciudad de Guatemala. La simulación también incluyóvariaciones aleatorias en la aceleración y desaceleración para reflejar los cambios comunes en la conducción, especialmente en áreas con alta densidad de tráfico. Este enfoque garantiza que las alertas generadas por el sistema tengan en cuenta tanto la velocidad como el contexto de tráfico, ofreciendo mensajes adaptados a cada situación. Con esta funcionalidad, la segunda versión proporciona una experiencia de simulación más completa y realista, ayudando al sistema a responder de manera más precisa a las condiciones de la vía.

5.6. Sistema de alerta final

Para el sistema final de asistencia de manejo se utilizó una combinación de modelos YOLOv8, uno preentrenado y otro afinado manualmente, para identificar señales de tránsito y objetos peligrosos en entornos urbanos. Esta estructura híbrida fue diseñada para mejorar la precisión y reducir los tiempos de respuesta del sistema. El modelo preentrenado se centra en la detección efectiva de la señal de alto, lo cual aporta un reconocimiento más confiable en este tipo de señal específica. Por otro lado, el modelo personalizado ha mostrado mejor desepeño específicamente para reconocer distintos estados de semáforos (verde, rojo, y amarillo), logrando una precisión considerablemente alta en estas categorías. La detección y análisis de objetos se complementan con simulaciones de velocidad que emulan un entorno de conducción realista, permitió generar alertas basadas tanto en el contexto de la velocidad como en el tipo de objeto detectado. Además, el sistema registra cada instancia de alerta en un archivo JSON, capturando detalles sobre la velocidad, los objetos detectados y el tiempo de respuesta promedio. La estructura de la función de alertas final se muestra a continuación:

```
def alert_system(input_type='camera', input_source=None):
    current_speed = 0
    max_speed = 70
    acceleration = 5.0
    deceleration = 7.0
```

```
stop_deceleration = 15.0
time_between_stops = 7
stop_chance = 0.1

response_times = []

for frame, timestamp in process_input(input_type, input_source):
    start_time = time.time()
    speed_data = simulate_speed_verbose(
        current_speed,
        max_speed,
        acceleration,
        deceleration,
        stop_deceleration,
        time_between_stops,
        stop_chance
    )
    current_speed = speed_data[0]

    # Mostrar el frame
    cv2.imshow("Frame", frame)

    # Detectar todos los objetos presentes y ausentes en el frame
    frame_analysis = detect_objects(frame)
    detected_objects = frame_analysis["detected"]
    missing_objects = frame_analysis["missing"]

    # Generar alertas para cada objeto detectado
    alert_messages = []
    for obj in detected_objects:
        if obj != "No objects detected":
            alert_message = generate_alert(obj, current_speed)
            alert_messages.append(alert_message)

    # Crear un diccionario para la entrada del log
    log_entry = {
        "timestamp": timestamp,
        "speed": current_speed,
        "detected_objects": detected_objects,
        "missing_objects": missing_objects,
        "alert_messages": alert_messages
    }

    # Guardar el log en formato JSON linea por linea
    with open(log_filename, 'a') as log_file:
        json.dump(log_entry, log_file)
        log_file.write("\n")

    # Guardar el frame donde se detecta la alerta
    if any("ALERTA" in msg for msg in alert_messages):
        alert_frame_name=f"alert_frames/frame_{int(timestamp)}.jpg"
        if not os.path.exists("alert_frames"):
            os.makedirs("alert_frames")
        cv2.imwrite(alert_frame_name, frame)
```

```
            end_time = time.time()  # Fin del tiempo de procesamiento
            response_time = (end_time - start_time) * 1000
            response_times.append(response_time)

            if cv2.waitKey(1) & 0xFF == ord('q'):
                break
            time.sleep(0.1)

    cv2.destroyAllWindows()
    return np.mean(response_times)
```

Este enfoque final ofreció mejoras en la estructura y el flujo del procesamiento, donde el sistema evalúa cada cuadro de video capturado para determinar la presencia de objetos peligrosos. Se observa una mayor eficiencia en el tiempo de respuesta, optimizando el análisis de cada cuadro en milisegundos. Las constantes para la velocidad urbana y los tipos de objetos considerados peligrosos han sido ajustadas para un entorno específico de conducción urbana, logrando un equilibrio entre la precisión de detección y el tiempo de procesamiento. Al combinar modelos preentrenados con uno especializado, el sistema ofreció mayor precisión sin comprometer la velocidad de procesamiento, algo crucial para un sistema de asistencia de conducción en tiempo real. Esta configuración también facilitó un análisis profundo de los resultados mediante la generación de métricas de evaluación (como precisión, recall, y F1 score) y una matriz de confusión para entender mejor el desempeño y ajustar el sistema según sea necesario.

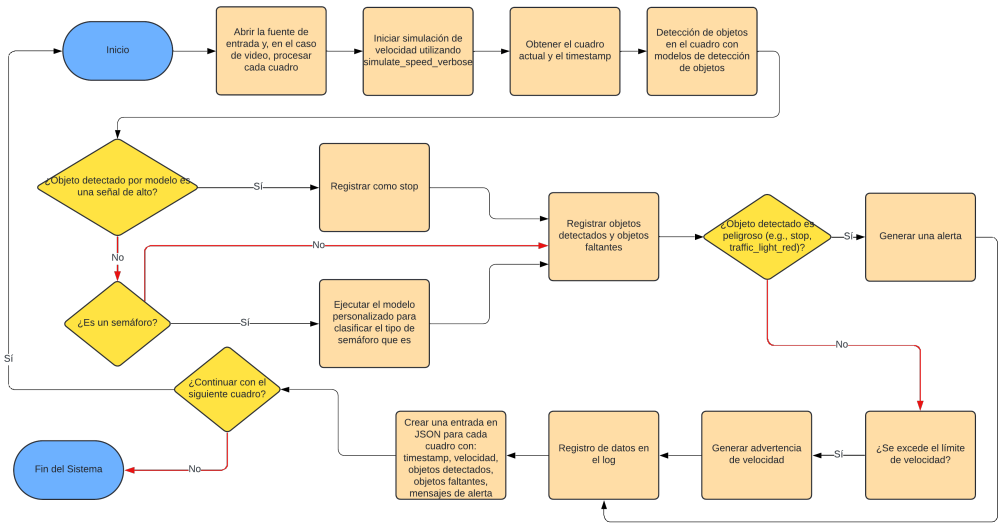

Figura 5.26: Diagrama de flujo de sistema de alerta en su versión final.

(Elaboración propia)

5.6.1. Modelo de detección de objetos empleado en versión final del sistema

El modelo de detección de señales de tránsito implementado en el sistema en su versión final utilizó la arquitectura YOLOv8 5.27. Este modelo se compone de múltiples capas convolucionales y bloques de procesamiento avanzados, como C2f y Bottleneck, los cuales permitieron capturar carac-

terísticas visuales de manera eficiente. A través de estas capas, el modelo pudo reducir la resolución de las imágenes y extraer patrones complejos que facilitan la identificación de señales de tránsito con alta precisión. Además, las capas de Concat y Upsample integran información de distintas escalas, logrando una detección efectiva de objetos de diferentes tamaños en la escena. Durante el entrenamiento, el modelo se ajusta con datos específicos de señales de tránsito, permitió su adaptación a las necesidades del sistema de asistencia de conducción. La configuración del entrenamiento se diseñó para encontrar un balance entre la precisión y la velocidad, con parámetros como el tamaño de imagen de 416x416 píxeles y una tasa de aprendizaje inicial de 0.001. Asimismo, el modelo utilizó el optimizador AdamW, que mejora la estabilidad y velocidad de convergencia al ajustar los pesos en cada iteración. La estructura de esta red profunda está diseñada para minimizar los tiempos de respuesta y maximizar la precisión en la detección, un factor crucial para el sistema de asistencia en carretera.

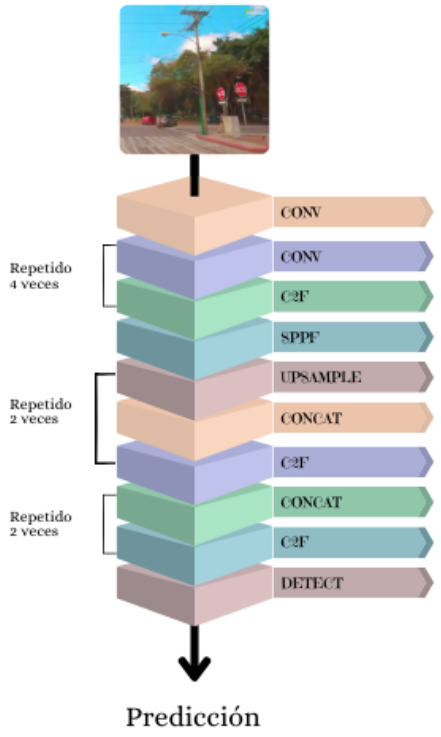

Figura 5.27: Arquitectura del modelo YOLOv8 entrenado para implementación en sistema de alertas.

(Elaboración propia)

Una vez entrenado, el modelo se evaluó en un conjunto de prueba que simula condiciones de tráfico real, permitió analizar su rendimiento en escenarios complejos. Las métricas de evaluación incluyeron precisión, recall, F1 score y el mAP (Mean Average Precision), calculado en los umbrales de IoU de 0.50 y 0.95 para medir su efectividad general. Adicionalmente, se generó una matriz de confusión que ayuda a identificar las clases de señales que el modelo podría confundir, como los distintos estados de los semáforos. Esta matriz visualiza los errores y las aciertos del modelo, proporcionando información útil para futuras mejoras. A lo largo del entrenamiento, se monitorearon gráficos de evolución de precisión y recall por época, asegurando que el modelo no cayera en sobreajuste y

lograra una generalización óptima. La implementación final del modelo en el sistema permitieron realizar inferencias en tiempo real, procesando cada cuadro de video y detectando objetos relevantes en milisegundos. De esta manera, el sistema pudo emitir alertas al conductor de forma inmediata, mejorando la seguridad y asistencia en la conducción. En resumen, la metodología del modelo combina arquitectura avanzada, entrenamiento especializado y métricas de evaluación robustas para alcanzar un balance ideal entre precisión y rapidez en la detección de señales de tránsito.

5.6.2. Simulación de sensores empleada en versión final del sistema

La versión final de la simulación de sensores introduce una metodología considerablemente más compleja y realista en comparación con la versión inicial. En la versión inicial, el comportamiento del vehículo estaba limitado a funciones básicas como aceleración, desaceleración y detención en situaciones específicas como semáforos o señales de alto. La nueva versión ofreció factores adicionales como congestión de tráfico, zonas de velocidad controlada, pendientes o curvas, y condiciones climáticas, enriqueciendo el modelo. Estos cambios permitieron capturar una mayor diversidad de situaciones reales que afectan la velocidad y el comportamiento del vehículo en un entorno simulado, añadiendo realismo al ajustar parámetros como la velocidad máxima, aceleración y desaceleración según condiciones de lluvia o niebla.

Además, la versión final introduce temporizadores y probabilidades para activar eventos específicos, como zonas de velocidad controlada, adelantamientos y pendientes o curvas, aumentando la variabilidad en la simulación. Por ejemplo, en una zona de velocidad controlada, la velocidad del vehículo se reduce temporalmente hasta que el temporizador expire, reflejando comportamientos que tendría en zonas escolares o de construcción. También se han añadido transiciones fluidas entre estados de aceleración, desaceleración y detención, permitió respuestas dinámicas a eventos externos como tráfico o condiciones de carretera. Esta metodología avanzada, que combina reglas probabilísticas, temporizadores y condiciones ambientales, ofreció una simulación robusta y adaptable, acercándola a escenarios de conducción realista.

Adicionalmente, la simulación de sensores en este código recrea el comportamiento de un vehículo en un entorno urbano utilizando Pygame 5.28 para la visualización en tiempo real. El vehículo respondió a distintas condiciones, como zonas de velocidad controlada, señales de alto, congestión de tráfico, y pendientes o curvas. Además, se aplican efectos climáticos que alteran la aceleración, desaceleración y velocidad máxima, aportando realismo al entorno. En la interfaz, el vehículo se desplaza horizontalmente y puede cambiar de carril para adelantar, mientras que señales visuales, como una señal de alto o de zona escolar, indican su estado actual. La velocidad y el estado del vehículo se muestran en tiempo real en pantalla, facilitando la observación de su comportamiento. Esta representación gráfica permitió comprender de manera clara y visual cómo distintas condiciones influyen en la conducción.

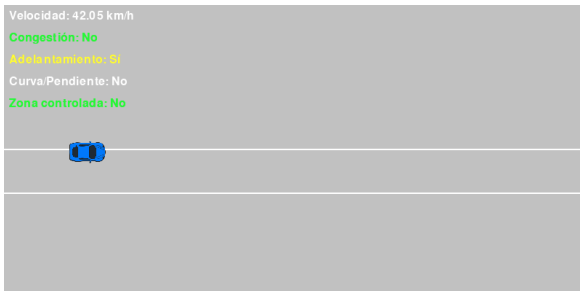

Figura 5.28: Visualización en tiempo real de simulación de sensores en Pygame.

(Elaboración propia)

CAPÍTULO 6

Resultados

6.1. Evaluación del modelo de detección de objetos en su versión inicial y final

6.1.1. Gráficos de entrenamiento y validación del modelo de detección

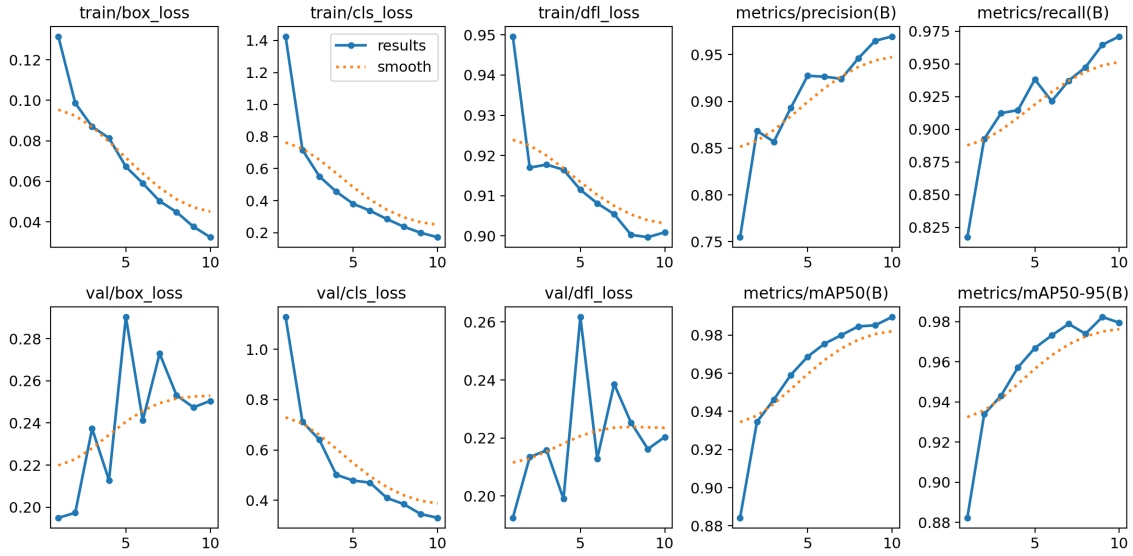

Figura 6.1: Resultados de entrenamiento y validación del modelo de detección de objetos utilizado en la primera y segunda versión del sistema de alertas, mostrando la evolución de pérdidas y métricas de precisión y recall a lo largo de las épocas.

(Elaboración propia)

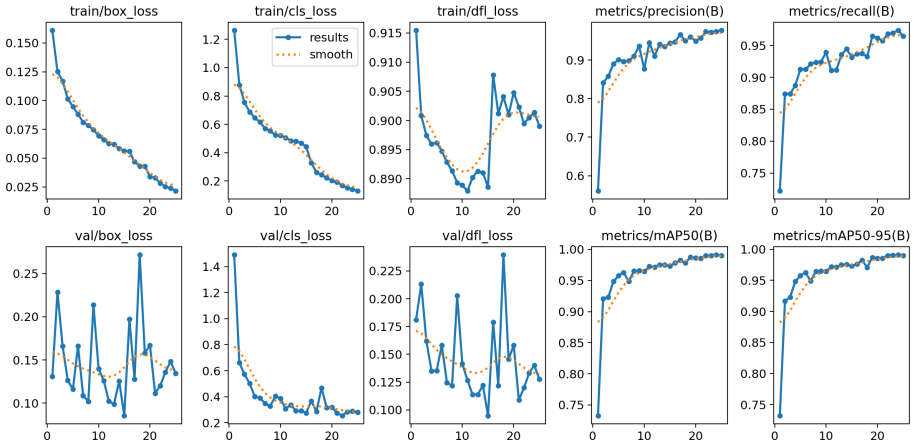

Figura 6.2: Resultados de entrenamiento y validación del modelo de detección de objetos utilizado en la versión final del sistema de alertas, mostrando la evolución de pérdidas y métricas de precisión y recall a lo largo de las épocas.

(Elaboración propia)

6.1.2. Matriz de confusión normalizada

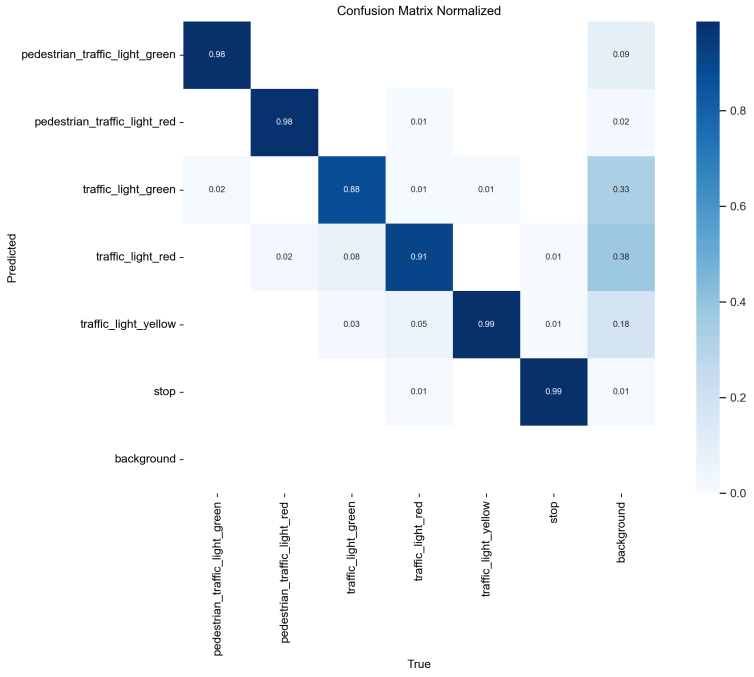

Figura 6.3: Matriz de confusión normalizada del modelo de detección de objetos utilizado en la primera y segunda versión del sistema de alertas, mostrando la precisión en la clasificación de cada clase en condiciones reales de prueba.

(Elaboración propia)

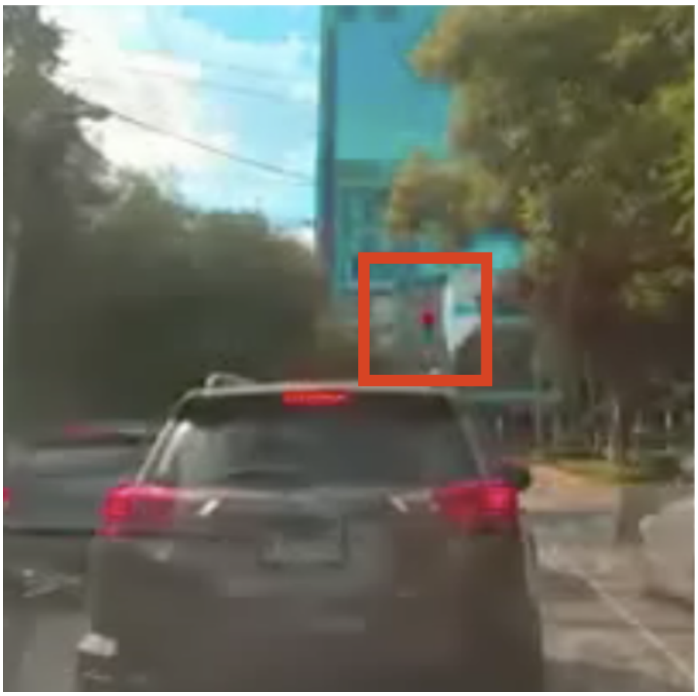

Figura 6.4: Ejemplo de dificultad en detección para el objeto semáforo rojo.

(Elaboración propia)

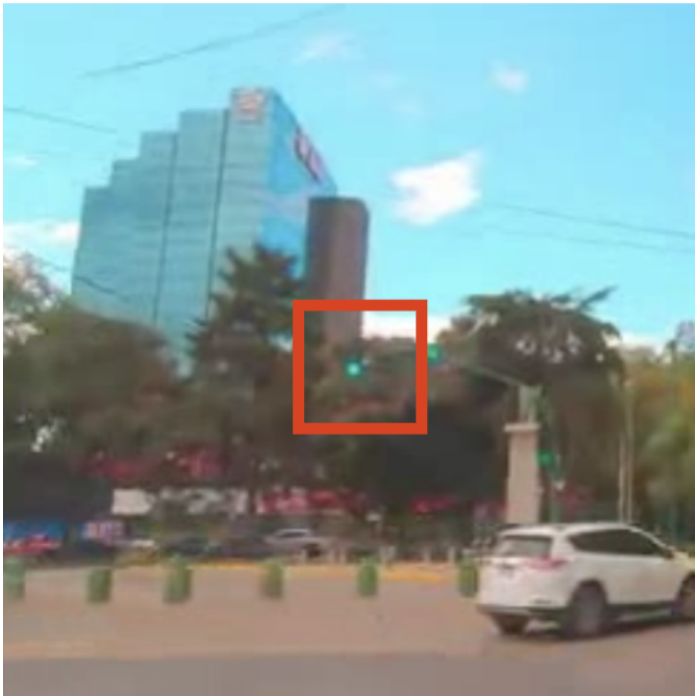

Figura 6.5: Ejemplo de dificultad en detección para el objeto semáforo verde.

(Elaboración propia)

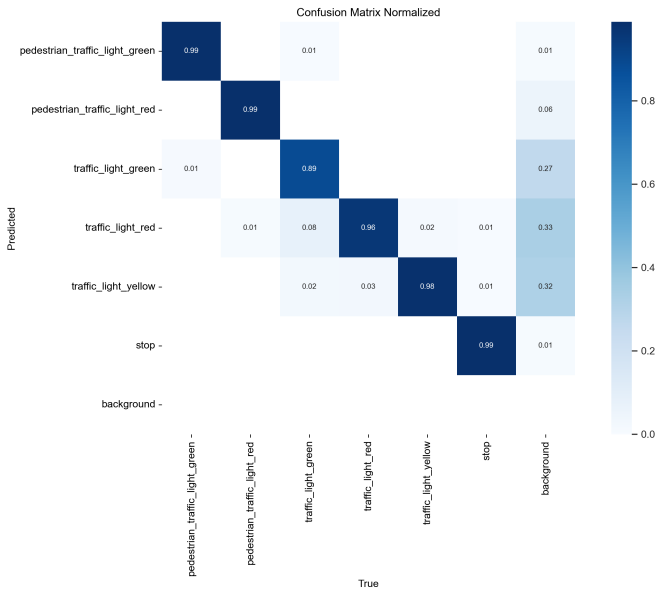

Figura 6.6: Matriz de confusión normalizada del modelo de detección de objetos utilizado en versión final del sistema de alertas, mostrando la precisión en la clasificación de cada clase en condiciones reales de prueba.

(Elaboración propia)

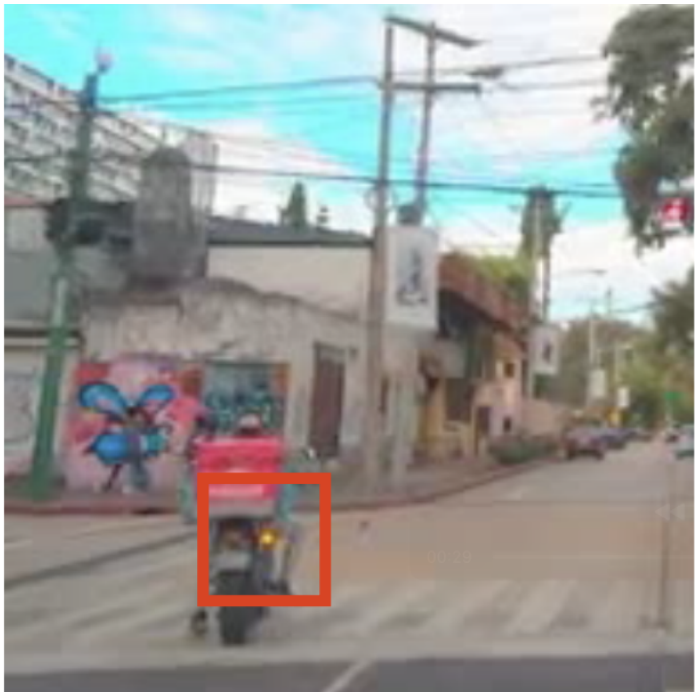

Figura 6.7: Ejemplo de dificultad en detección para el objeto semáforo amarillo.

(Elaboración propia)

6.1.3. Curvas de desempeño del modelo de detección de objetos

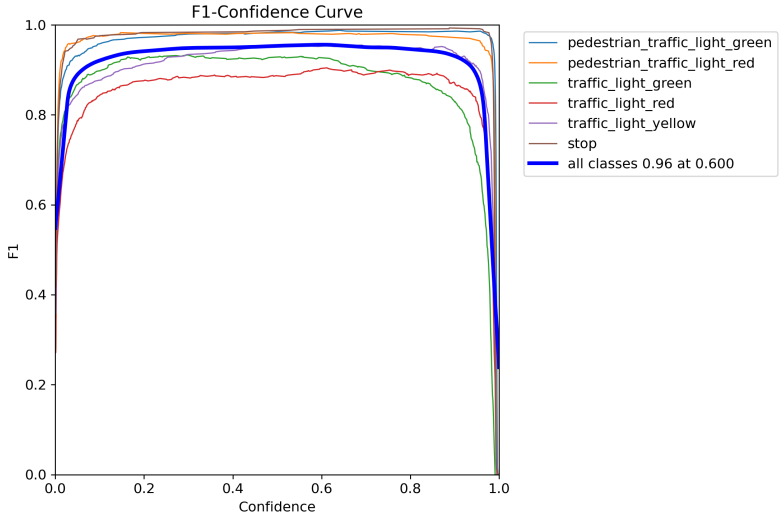

Figura 6.8: Curva F1-Confianza del modelo de detección de objetos utilizado en la primera y segunda versión del sistema de alertas, mostrando el equilibrio entre precisión y sensibilidad a diferentes niveles de confianza para cada clase.

(Elaboración propia)

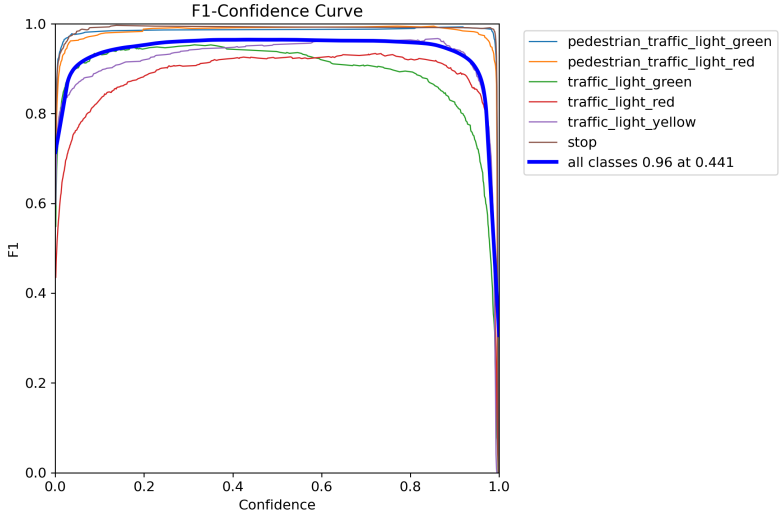

Figura 6.9: Curva F1-Confianza del modelo de detección de objetos utilizado en versión final del sistema de alertas, mostrando el equilibrio entre precisión y sensibilidad a diferentes niveles de confianza para cada clase.

(Elaboración propia)

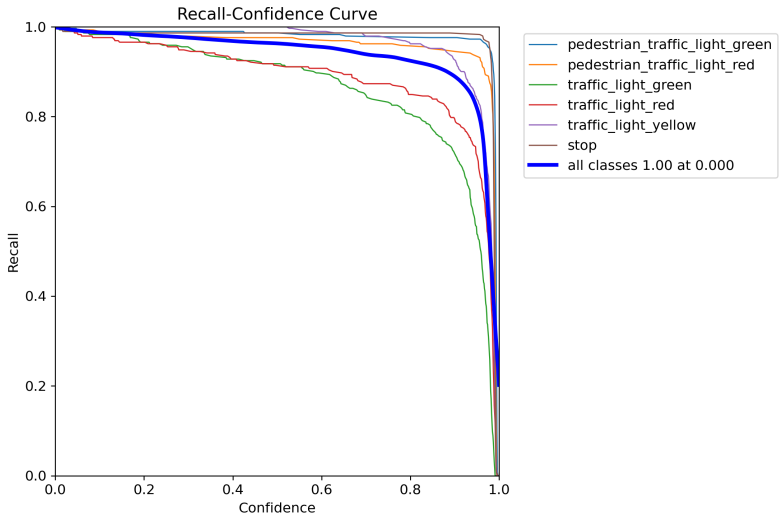

Figura 6.10: Curva Recall-Confianza del modelo de detección de objetos utilizado en la primera y segunda versión del sistema de alertas, mostrando la sensibilidad del modelo en función de diferentes niveles de confianza para cada clase.

(Elaboración propia)

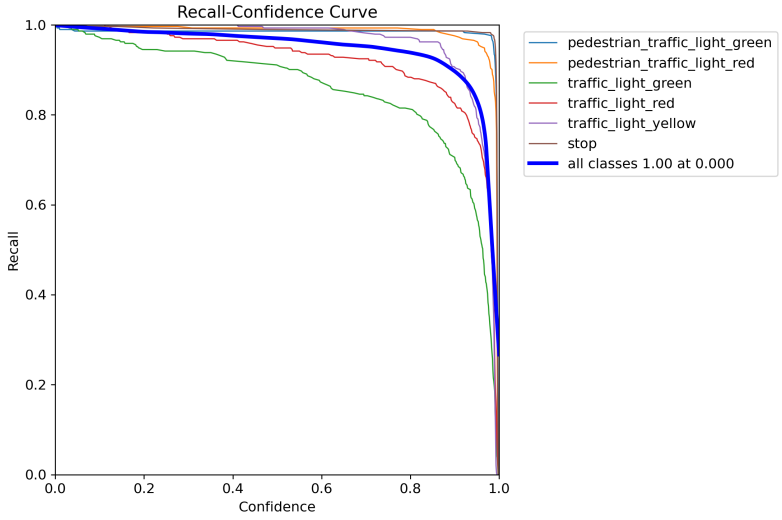

Figura 6.11: Curva Recall-Confianza del modelo de detección de objetos utilizado en la versión final del sistema de alertas, mostrando la sensibilidad del modelo en función de diferentes niveles de confianza para cada clase.

(Elaboración propia)

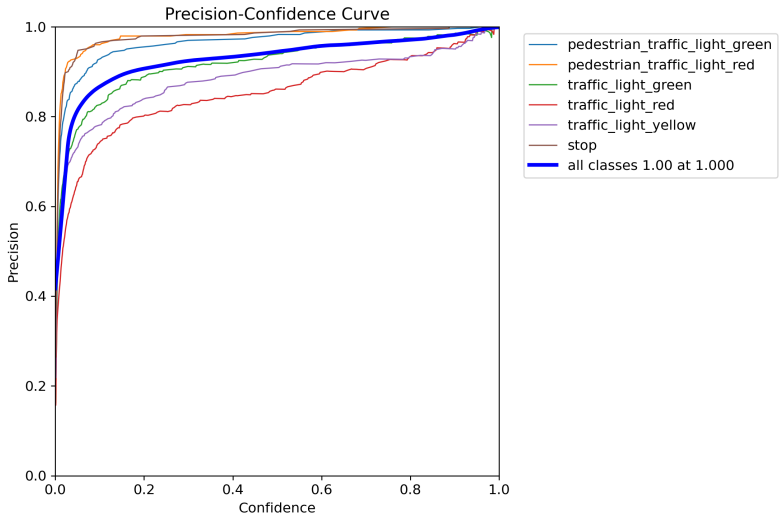

Figura 6.12: Curva de Precisión-Confianza del modelo de detección de objetos utilizado en la primera y segunda versión del sistema de alertas, ilustrando la precisión alcanzada en función de diferentes niveles de confianza para cada clase.

(Elaboración propia)

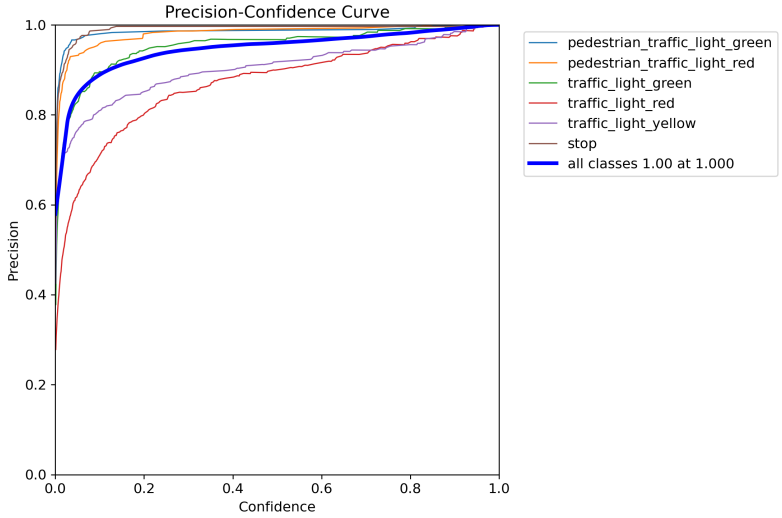

Figura 6.13: Curva de Precisión-Confianza del modelo de detección de objetos utilizado en la versión final del sistema de alertas, ilustrando la precisión alcanzada en función de diferentes niveles de confianza para cada clase.

(Elaboración propia)

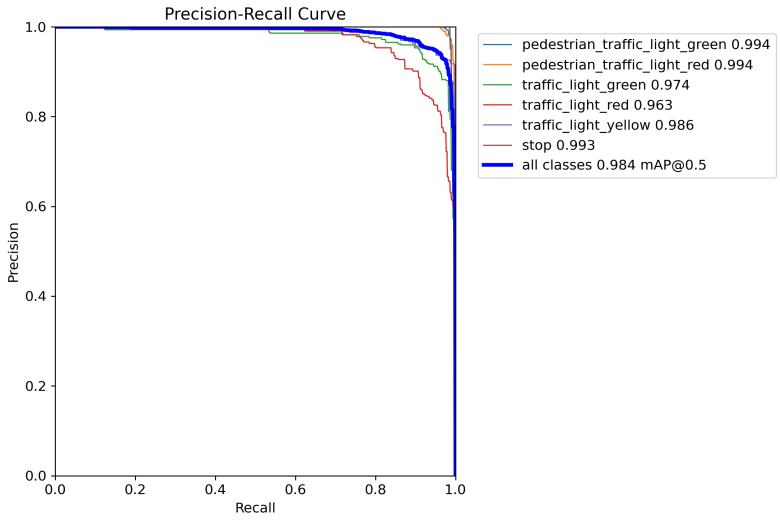

Figura 6.14: Curva de Precisión-Recall del modelo de detección de objetos utilizado en la primera y segunda versión del sistema de alertas, mostrando la relación entre precisión y recall para cada clase detectada.

(Elaboración propia)

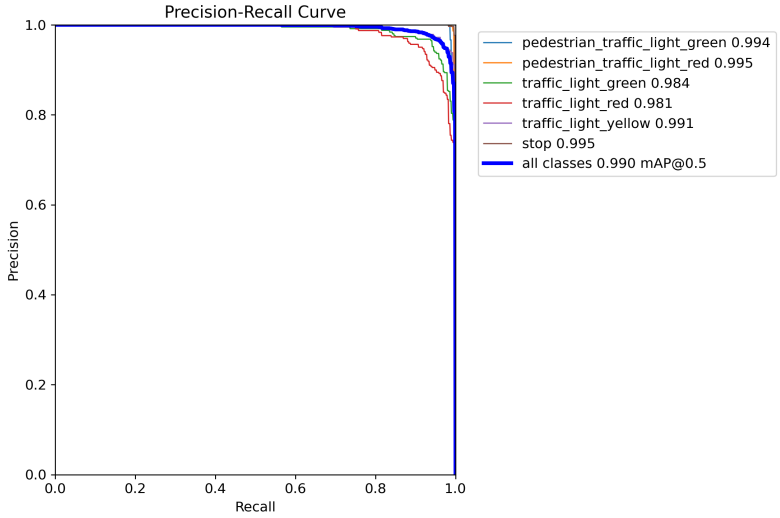

Figura 6.15: Curva de Precisión-Recall del modelo de detección de objetos utilizado en la versión final del sistema de alertas, mostrando la relación entre precisión y recall para cada clase detectada.

(Elaboración propia)

6.2. Tipos de alertas configuradas y criterios de activación

Tipo de Alerta	Condición de Activación
ALERTA CRÍTICA (Detenerse)	Objeto detectado en lista de peligro y velocidad > 0
ALERTA (Semáforo rojo)	Semáforo en rojo detectado y velocidad > 0
Advertencia (Exceso de velocidad)	Velocidad excede el límite permitido
Velocidad adecuada	Velocidad dentro del límite permitido

Tabla 6.1: Tabla de tipos de alerta y sus respectivas condiciones de activación en el sistema de asistencia al conductor.

(Elaboración propia)

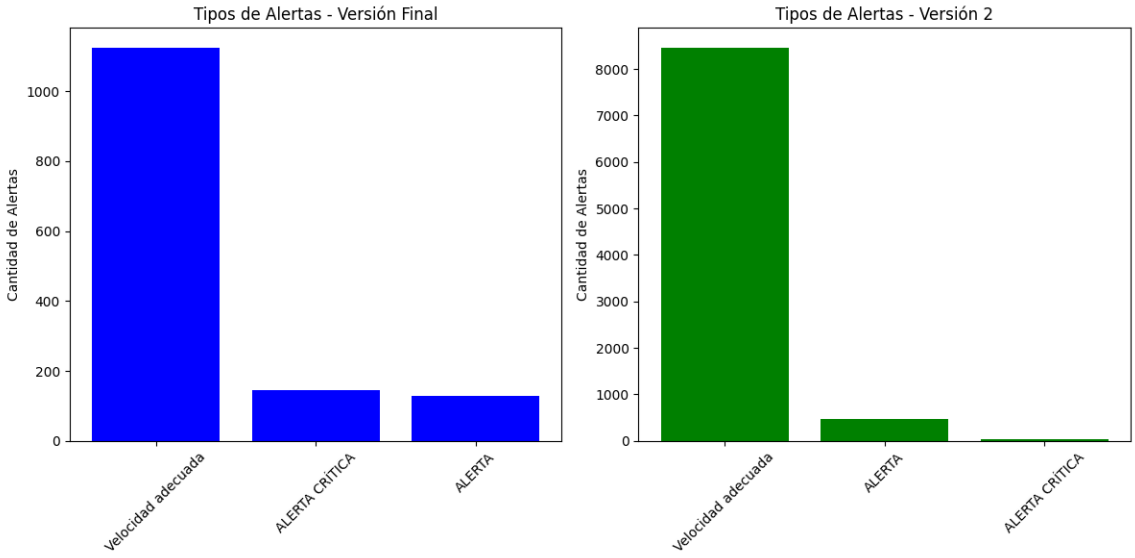

Figura 6.16: Gráfico comparativo de la cantidad de alertas generadas por tipo en las versiones final y segunda versión del sistema de asistencia al conductor.

(Elaboración propia)

6.3. Comparativa de desempeño entre segunda versión y última versión del Sistema de Alertas

6.3.1. Gráfico de barras de métricas generales

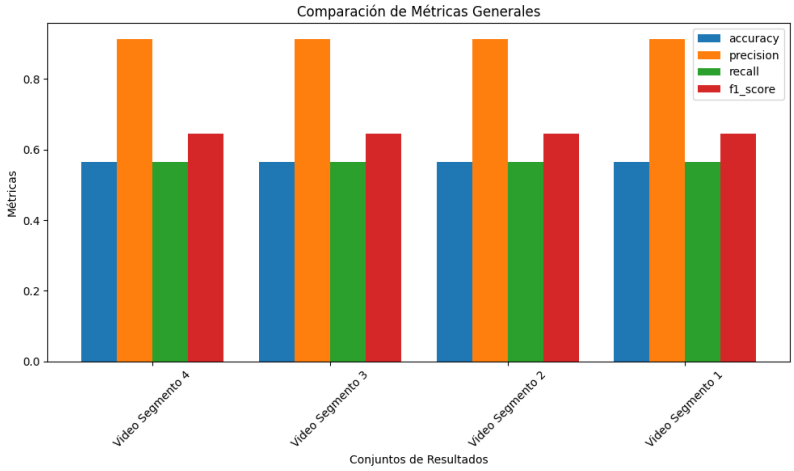

Figura 6.17: Gráfico comparativo de métricas generales (accuracy, precision, recall, f1_score) en distintos segmentos de video para evaluar el rendimiento de la segunda versión del sistema en cada caso.

(Elaboración propia)

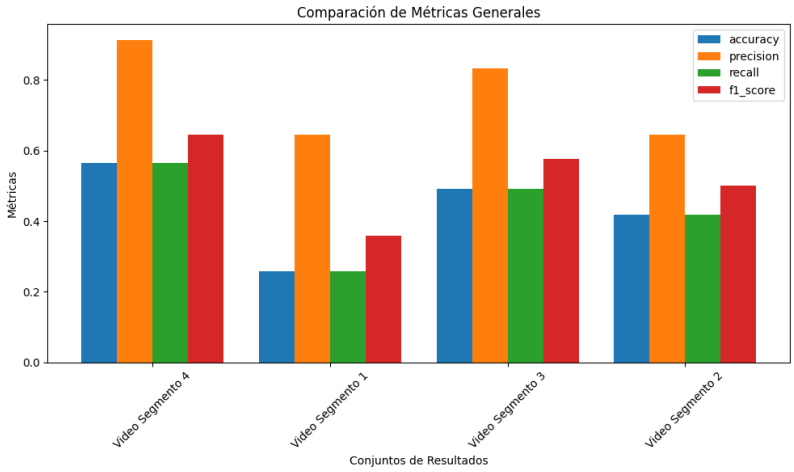

Figura 6.18: Gráfico comparativo de métricas generales (accuracy, precision, recall, f1_score) en distintos segmentos de video para evaluar el rendimiento de la versión final del sistema en cada caso.

(Elaboración propia)

6.3.2. Relación entre F1 Score y Tiempo de Respuesta

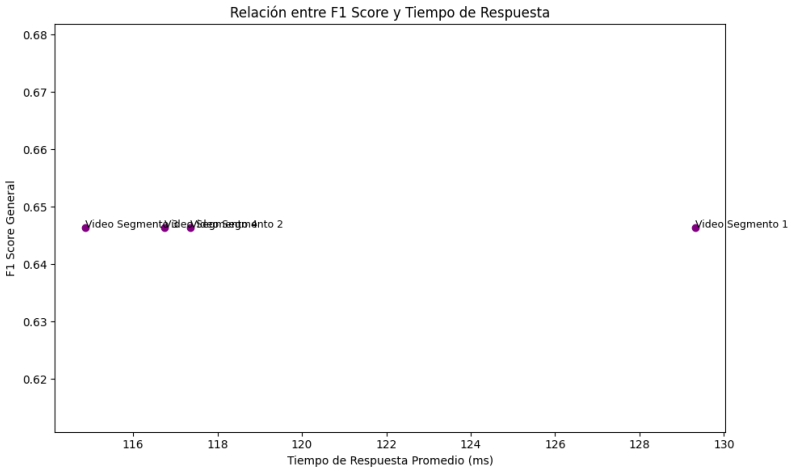

Figura 6.19: Gráfico de relación entre el F1 Score general y el tiempo de respuesta promedio en milisegundos para cada segmento de video, evaluando la efectividad y eficiencia del sistema de alertas en su segunda versión.

(Elaboración propia)

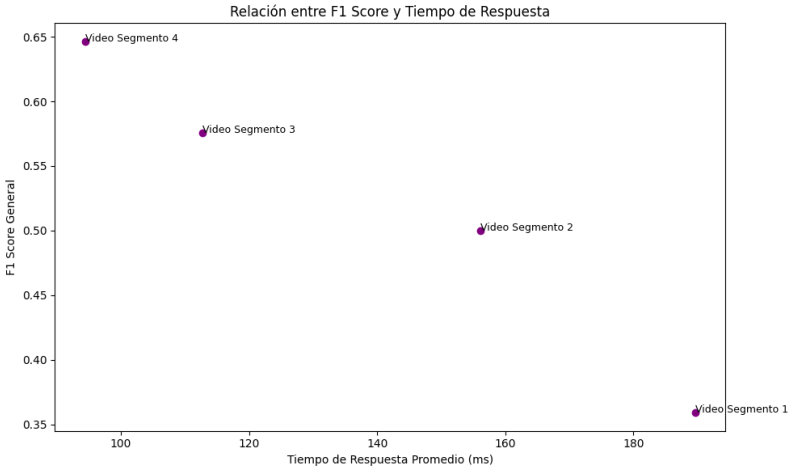

Figura 6.20: Gráfico de relación entre el F1 Score general y el tiempo de respuesta promedio en milisegundos para cada segmento de video, evaluando la efectividad y eficiencia del sistema de alertas en su versión final.

(Elaboración propia)

6.3.3. Gráfico de F1 Score y Tiempo de Respuesta Promedio

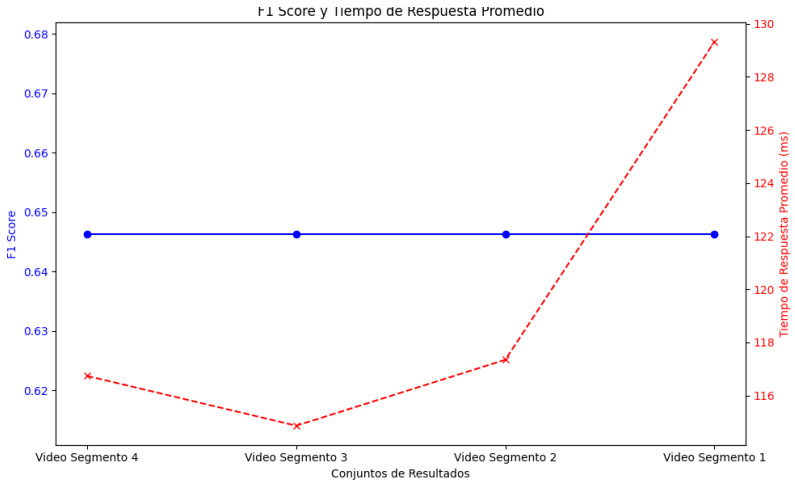

Figura 6.21: Gráfico comparativo de F1 Score y tiempo de respuesta promedio del sistema de alerta en su segunda versión en milisegundos para distintos segmentos de video, mostrando consistencia en F1 Score y variabilidad en tiempo de respuesta entre los segmentos.

(Elaboración propia)

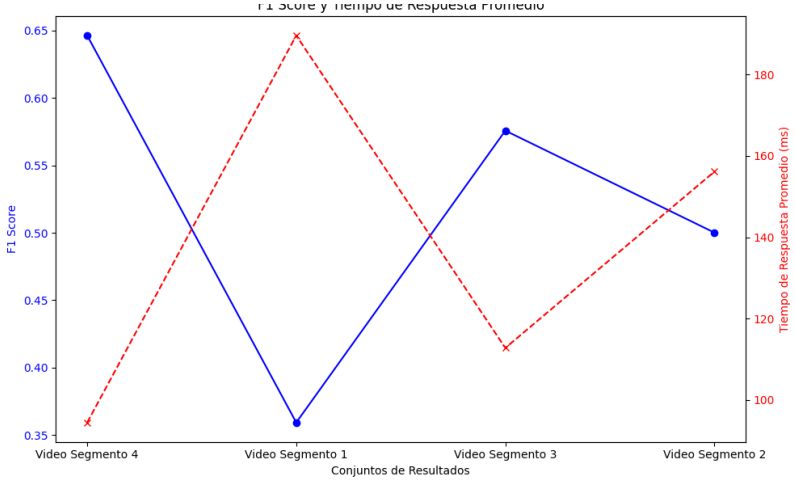

Figura 6.22: Gráfico comparativo de F1 Score y tiempo de respuesta promedio de la versión final del sistema de alerta en milisegundos para distintos segmentos de video, mostrando consistencia en F1 Score y variabilidad en tiempo de respuesta entre los segmentos.

(Elaboración propia)

6.3.4. Matriz de falsos positivos, falsos negativos, verdaderos positivos y verdaderos negativos

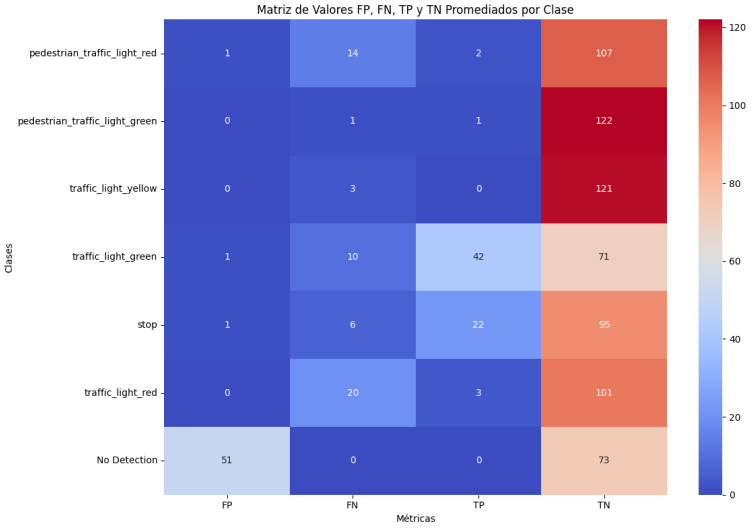

Figura 6.23: Matriz de valores promedio de Falsos Positivos (FP), Falsos Negativos (FN), Verdaderos Positivos (TP) y Verdaderos Negativos (TN) por clase en la segunda versión del sistema de alertas.

(Elaboración propia)

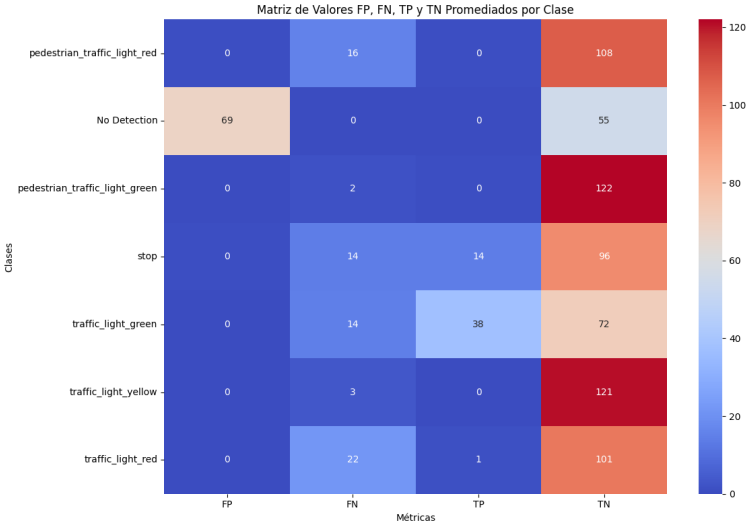

Figura 6.24: Matriz de valores promedio de Falsos Positivos (FP), Falsos Negativos (FN), Verdaderos Positivos (TP) y Verdaderos Negativos (TN) por clase en la versión final del sistema de alertas.

(Elaboración propia)

6.4. Evaluación entre segunda versión y última versión del Sistema en Segmentos de Video

6.4.1. Desempeño del Sistema de Alertas por Segmento de Video

Video	Exactitud (%)	Precisión (%)	Recall (%)	F1 Score (%)	Tiempo de Respuesta Promedio (ms)
Video Segmento 4	56.45%	91.32%	56.45%	64.63%	116.74
Video Segmento 3	56.45%	91.32%	56.45%	64.63%	114.87
Video Segmento 2	56.45%	91.32%	56.45%	64.63%	117.35
Video Segmento 1	56.45%	91.32%	56.45%	64.63%	129.32

Tabla 6.2: Tabla de desempeño del sistema de alertas en la segunda versión, mostrando exactitud, precisión, recall, F1 Score y tiempo de respuesta promedio en ms para cada segmento de video evaluado.

(Elaboración propia)

Video	Exactitud (%)	Precisión (%)	Recall (%)	F1 Score (%)	Tiempo de Respuesta Promedio (ms)
Video Segmento 4	56.45%	91.32%	56.45%	64.63%	94.46
Video Segmento 1	25.81%	64.52%	25.81%	35.93%	189.60
Video Segmento 3	49.19%	83.35%	49.19%	57.57%	112.80
Video Segmento 2	41.94%	64.52%	41.94%	50.01%	156.14

Tabla 6.3: Tabla de desempeño del sistema de alertas en la versión final, mostrando exactitud, precisión, recall, F1 Score y tiempo de respuesta promedio en ms para cada segmento de video evaluado.

(Elaboración propia)

CAPÍTULO 7

Discusión de resultados

Mediante este trabajo, se buscaba desarrollar e implementar un sistema de asistencia al conductor que integrara tecnologías de visión por computadora y análisis de datos de sensores con el fin de detectar y clasificar elementos que fueran críticos en las calles y carreteras como los son las señales de tránsito verticales. Contribuyendo así a la experiencia del conductor y su percepción de seguridad y eficiencia en la conducción cumpliendo con las leyes de tránsito establecidas en la República de Guatemala.

7.1. Conjunto de datos generado

La construcción del conjunto de datos se realizó con el objetivo de abarcar un amplio espectro de situaciones de tráfico urbano, de modo que el modelo de detección de objetos fuera capaz de generalizar su desempeño en entornos reales. Se recopiló una gran variedad de imágenes y videos en condiciones diversas de iluminación, incluyendo tanto escenarios diurnos como nocturnos. Esto permitió que el sistema de detección se adaptara a cambios de luz y condiciones atmosféricas tal como se puede visualizar en las Figuras 5.17 y 5.19, mejorando su utilidad en diferentes momentos del día. Estas variaciones de iluminación pueden afectar la precisión del modelo, por lo que su inclusión fue clave para robustecer el sistema y considerar todos los casos posibles del ámbito urbano guatemalteco que puedan existir.

En cuanto a la categorización, cada objeto fue etiquetado para que el modelo de detección de objetos tuviera la capacidad de aprender a diferenciar entre señales de tráfico y otros elementos críticos. Se dio especial importancia a las señales que en la ley de tránsito se definen como señales verticales en el Título 1, Artículo 2, numeral 60 de la Ley de Tránsito de Guatemala (de la República de Guatemala, 1996). Dentro de ellos se consideró las señales que indican al conductor realizar una parado o conducir con precaución, tales como el semáforo en rojo y la señal de alto. Estos fueron relevantes para la toma de decisiones del sistema al momento de actuar sobre algún tipo de *input* en la vía. Para mejorar el equilibrio en el aprendizaje del modelo, sería útil incrementar la cantidad de muestras de los objetos que no eran de interés para poder enseñarle al modelo de detección a clasificarlo y generar un conjunto de datos más robusto y representativo.

La diversidad geográfica también juega un papel fundamental en la efectividad del conjunto de datos. Se incluyeron diferentes entornos urbanos, pero las variaciones geográficas específicas, como

condiciones de tráfico en zonas rurales o montañosas, aún están limitadas. La incorporación de datos de estas áreas podría ampliar la capacidad del sistema para adaptarse a una mayor variedad de entornos. Esto ayudaría a que el modelo sea funcional no solo en ciudades, sino también en áreas suburbanas y rurales, donde los elementos viales pueden ser diferentes. Es importante también mencionar el hecho de que la incorporación de un conjunto de datos de señales de tránsito recopilado en otro ámbito distinto al guatemalteco ayudó a enriquecer a que el modelo generalizara más los datos al momento de su análisis y detección del mundo tangible.

Las condiciones climáticas presentes en el conjunto de datos abarcan escenarios de clima despejado, nublado y lluvioso, ofreciendo cierta robustez ante variaciones meteorológicas. No obstante, faltan ejemplos en condiciones extremas, como la nieve, que en el caso de Guatemala no se tomó en cuenta pues la misma ubicación geográfica hace que esto sea muy poco común. La adaptabilidad del sistema a distintas condiciones climáticas fue crucial, dado que el clima afecta directamente la visibilidad y la seguridad en la conducción y por ende la capacidad del modelo de poder analizar los patrones y características de los datos. Dado esto la decisión de aumentar la cantidad de datos mediante *Data Augmentation* brinda una muestra más grande y valor al set de datos.

El balance entre las clases de objetos en el conjunto de datos fue otro aspecto importante considerado en el proceso de desarrollo. La alta frecuencia de algunos objetos, como los semáforos peatonales, facilitó su aprendizaje. Por otro lado, habían objetos menos comunes que presentaron mayores dificultades. Esta disparidad en la frecuencia de los datos fue detectada en el análisis previo del conjunto de datos y se modificó para evitar que afectara la precisión del modelo al detectar clases menos representadas. Aplicar técnicas de aumento de datos para estas clases ayudaría a equilibrar el rendimiento del sistema, sin necesidad de aumentar la cantidad total de datos. Se realizó un submuestreo de las clases que presentaban una mayor cantidad de elementos para lograr que sin importar la clase se tuviera la misma cantidad de elementos logrando una distribución más uniforme permitiendo que el sistema fuera más justo y preciso en su detección.

Otro desafío encontrado fue la consistencia en el etiquetado de los datos, lo cual es esencial para el aprendizaje correcto del modelo. La calidad de las etiquetas influye directamente en la precisión de detección de los objetos pues esto es la forma en la que el modelo detecta un objeto en una imagen y lo categoriza. Durante el proceso de etiquetado, se establecieron criterios rigurosos, pero la complejidad de algunos entornos ocasionó inconsistencias. Esto se encontró principalmente al momento de unificar los conjuntos de datos y buscar que todos los elementos de los mismos tuvieran homogeneidad en el formato de sus etiquetas. Por lo tanto, la elaboración del conjunto de datos analizado, curado y procesado permitió que se tuviera una cantidad de datos representativa para cada una de las clases. Dando así la robustez y generalidad que se buscaba para poder entrenar un modelo de detección de objetos basado en el algoritmo de YOLOv8.

7.2. Modelo de detección de objetos basado en el algoritmo de YOLOv8

El modelo de detección de objetos basado en YOLOv8 fue seleccionado por su capacidad de balancear precisión y velocidad. La versión YOLOv8 incorpora mejoras en la arquitectura de la red que optimizan el procesamiento de imágenes en tiempo real. Se logró alcanzar valores promedios de tiempo de respuesta para la versión final del sistema entre los 94.46 ms a 189.60 ms tal como se puede observar en el cuadro 6.3.Esto muestra una mejora entre la versión inicial de dicho modelo de detección pues en la primera iteración se obtuvo valores promedio de tiempo de respuesta entre los 114.87 ms a 129.32 ms tal como se puede ver en el cuadro 6.2. Estas mejoras permiten detectar múltiples objetos en fracciones de segundo, una característica crucial para entornos de conducción que requieren decisiones rápidas. Las pruebas realizadas muestran que el modelo responde bien en escenarios con pocos elementos, pero presenta ciertos desafíos en escenas densas.

En a las métricas de *F1-Confidence* y *Recall-Confidence*, se observó que el modelo de detección basado en YOLOv8 mostró un rendimiento de clasificación elevado en la mayoría de las clases analizadas. La curva *F1-Confidence* observada en la Figura 6.9 indicó que el modelo mantuvo valores altos de F1 en todo el rango de confianza, especialmente alrededor de la región de 0.6 a 0.8, lo cual sugiere una buena precisión y exhaustividad en las detecciones. En el caso de la curva *Recall-Confidence* observada en la Figura 6.11, el modelo mantuvo altos niveles de *recall* para la mayoría de las clases hasta un umbral de confianza elevado, lo cual es indicativo de la capacidad del modelo para capturar la mayoría de las verdaderas instancias de las clases objetivo. Sin embargo, se detectaron ligeras caídas en *recall* que no fueron significativas pues estuvieron por debajo del 80 %.

Los gráficos de *Precision-Confidence Curve* y *Precision-Recall Curve* presentaron una alta precisión en la detección de objetos para cada clase relevante del sistema de alertas basado en YOLOv8. La curva de precisión se mantuvo cerca de valores óptimos en todos los niveles de confianza, lo cual indica que el modelo logró distinguir adecuadamente las distintas señales de tráfico y señales peatonales, incluso en condiciones variadas de confianza. Además, las métricas de precisión alanzaron cifras superiores al 98 % en la mayoría de las clases, lo que sugiere una baja tasa de falsos positivos al clasificar los objetos detectados. La curva de *Precision-Recall*, mostró una inclinación muy ajustada, confirmando que la precisión del modelo se mantuvo alta sin sacrificar el *recall*. La clase *stop* y los semáforos, tanto peatonales como de tráfico, presentaron un comportamiento consistente en la detección, indicando que el modelo tiene una alta efectividad en clasificar estas señales críticas para la seguridad en carretera. A través del uso de la curva F1 también se corroboró que el equilibrio entre precisión y *recall* está en un nivel óptimo, reflejando una gran capacidad del modelo para emitir alertas en condiciones reales de tráfico. Esto significa que el sistema es capaz de identificar y clasificar con precisión las señales y objetos en tiempo real, minimizando los errores y garantizando una mayor confiabilidad en las alertas proporcionadas al conductor.

En cuanto la relación entre el *F1 Score* y el tiempo de respuesta promedio del sistema en diferentes segmentos de video observado en la Figura 6.22. Se observó que el *F1 Score* varía considerablemente entre los segmentos, alcanzando su valor máximo en el Segmento 4 y su mínimo en el Segmento 1. Esta fluctuación sugirió que la precisión del sistema para detectar y clasificar objetos fue sensible a las condiciones de cada segmento de video. En paralelo, el tiempo de respuesta promedio mostró un comportamiento inverso en algunos casos, siendo menor en el Segmento 3 y mayor en el Segmento 4. Esto mostró que en los segmentos donde el sistema tuvo mayor precisión (*F1 Score* alto), el tiempo de respuesta fue más elevado, lo que podría deberse a la complejidad de los objetos detectados o al procesamiento necesario para mantener la precisión. En segmentos con menor *F1 Score*, el tiempo de respuesta fue más bajo, lo que sugiere que el sistema pudo haber priorizado la velocidad sobre la precisión en ciertas condiciones. En general, el equilibrio entre *F1 Score* y tiempo de respuesta es clave para optimizar el rendimiento del sistema en diferentes escenarios.

Finalmente, en la Figura 6.3, se observó que en la versión inicial del modelo existían ciertas dificultades para detectar correctamente los semáforos verdes como se puede observar en la Figura 6.5 cuando el fondo presentaba tonalidades similares, como árboles u otros elementos naturales. Asimismo, se encontró que los semáforos rojos también presentaban desafíos de detección en situaciones donde un vehículo en el fondo tenía sus luces traseras de freno activadas tal como se aprecia en la Figura 6.4, ya que estas luces rojas interferían con la clasificación del modelo. Estos resultados reflejaron una necesidad de ajustar la sensibilidad del modelo para mejorar la precisión en estos contextos específicos. Además, en esta versión inicial, algunas clases, como *pedestrian_traffic_light_red* y *pedestrian_traffic_light_green*, mostraron valores altos de verdaderos negativos, indicando una correcta identificación de la ausencia de estos objetos en la mayoría de las instancias, probablemente debido a su carácter eventual en situaciones normales de manejo.

Por otro lado, en la Figura 6.6, las mejoras implementadas en el modelo lograron disminuir parcialmente las dificultades para la detección de los semáforos verdes en escenarios con fondos similares. Sin embargo, se identificó una nueva dificultad más pronunciada en la detección de semáforos amarillos como se puede observar en la Figura 6.7. Esta problemática se atribuye a que, en

ciertos casos, las luces direccionales amarillas de los vehículos fueron percibidas erróneamente como semáforos amarillos, lo que afectó la precisión en la clasificación. A pesar de estas dificultades, otras clases, como *traffic_light_green* y *stop*, mostraron un balance razonable entre verdaderos positivos y verdaderos negativos, lo que sugiere que el sistema fue capaz de identificar estas señales con precisión aceptable. Los falsos positivos fueron mínimos en la mayoría de las clases, lo cual es beneficioso para evitar alertas innecesarias. Estas mejoras refuerzan la capacidad del modelo para integrarse en un sistema de alertas confiable que cumpla con las normas de la Ley de Tránsito de Guatemala.

7.3. Mecanismo de incorporación de una simulación de sensores de velocidad y aceleración

El mecanismo de simulación de sensores de velocidad y aceleración en el proyecto permitió evaluar la respuesta del sistema bajo diferentes condiciones de tráfico y clima. Esta simulación incorporó una variedad de factores, como la congestión del tráfico, las curvas o pendientes, y las zonas de velocidad controlada, lo cual facilitó la observación de cómo estos elementos influyeron en la aceleración, desaceleración y velocidad del vehículo. La adición de condiciones climáticas, como lluvia y niebla, proporcionó un escenario más realista, limitando la velocidad máxima y ajustando los parámetros de aceleración y desaceleración. Estas condiciones lograron simular el impacto directo del clima en la dinámica del vehículo, replicando escenarios críticos para la seguridad del conductor. El mecanismo también permitió variaciones en la respuesta del sistema mediante el uso de probabilidades y temporizadores, simulando eventos como las paradas repentinas o adelantamientos con cambio de carril. Este nivel de realismo fue una parte importante para evaluar el rendimiento del modelo de detección en condiciones complejas y cambiantes. Sin embargo, la naturaleza probabilística de algunos eventos pudo introducir variabilidad en los resultados, lo cual sugiere la necesidad de ajustar ciertos parámetros para garantizar la consistencia en escenarios de prueba. En general, este enfoque permitió probar el sistema de alertas y detección en un contexto cercano al uso real, mostrando fortalezas en situaciones simuladas.

La integración de probabilidades y temporizadores en la simulación logró un balance entre eventos controlados y aleatorios, permitiendo una mayor variedad de escenarios para el análisis. Por ejemplo, las probabilidades de detenerse o adelantar en ciertas condiciones añadieron realismo, replicando decisiones de conducción que dependen de factores externos y condiciones momentáneas. Al incluir zonas de velocidad controlada y pendientes, el sistema pudo responder a situaciones específicas que requerían ajustes de velocidad y posicionamiento del vehículo en el entorno. Esta flexibilidad permitió al modelo de detección de objetos operar en diferentes velocidades y con distintas configuraciones de sensor, evaluando su capacidad de adaptación. A pesar de los beneficios, algunos eventos, como las zonas de velocidad controlada, dependían de temporizadores fijos, lo cual podría ser mejorado para adaptarse más a la realidad del tráfico cambiante. Asimismo, la simulación de frenado gradual y paradas repentinas aportó variabilidad en las pruebas, aunque estos eventos pudieran haber afectado la estabilidad de los resultados de la detección de objetos. En conjunto, estos ajustes permitieron que el sistema respondiera a distintas situaciones de manera realista buscando acercarse a las peculiaridades que existen en el ámbito urbano de Guatemala.

En el desarrollo del sistema, se incluyeron posibles funciones para detectar sensores comunes de vehículos, tales como acelerómetros y giroscopios, aunque estos no se utilizaron en pruebas en un vehículo en movimiento. Se tomó la decisión de no realizar dichas pruebas en la vía pública para evitar infracciones a la ley de tránsito y, fundamentalmente, para resguardar la vida e integridad tanto del desarrollador como de los demás usuarios de la vía. Esto pues el hecho de no obedecer un alto o un semáforo rojo representa un gran peligro. En cambio, las pruebas fueron limitadas a simulaciones y entornos seguros donde no existía el riesgo de causar accidentes o interrupciones en el tráfico. Se recomienda, como trabajo futuro, realizar evaluaciones en un entorno controlado, como una pista de prueba o un espacio cerrado, donde puedan replicarse condiciones de movimiento sin

poner en riesgo la seguridad. Estas pruebas en un entorno seguro permitirían verificar la capacidad del sistema para reaccionar a eventos reales y a datos de sensores en tiempo real. Asimismo, este enfoque proporcionaría información más precisa sobre el rendimiento y confiabilidad del sistema en un contexto de conducción realista.

7.4. Sistema de asistencia al conductor que integra tecnologías de visión por computadora y análisis de datos de sensores

El análisis de la comparación de tipos de alertas permitió observar el impacto de los ajustes en la configuración del sistema sobre la frecuencia de alertas generadas. Los cambios realizados lograron incrementar las alertas de velocidad adecuada, mientras que se mantuvo controlada la emisión de alertas críticas y generales. Esto sugirió que el sistema fue capaz de alertar eficazmente al conductor sin crear una sobrecarga de notificaciones que pudieran resultar en distracciones innecesarias. La configuración final favoreció la seguridad del conductor al enfocarse en alertas relevantes y reducir aquellas que no requerían acción inmediata. A través de esto, se buscó maximizar la percepción de seguridad sin sacrificar la atención del conductor ante falsas alarmas o notificaciones poco críticas. El balance obtenido indicó que el sistema estaba bien ajustado para las condiciones previstas en el entorno de prueba. Esto también evidenció la importancia de la calibración en sistemas de asistencia, donde un exceso de alertas podría reducir su efectividad y generar fatiga en el usuario. Los resultados de esta optimización pueden observarse en la Figura 6.16, donde se muestra la comparación de las alertas generadas en ambas versiones del sistema.

La Comparación de métricas observada generales en la Figura 6.18, mostró cómo variaron las métricas de precisión, *recall* y *F1 Score* en los diferentes segmentos de video, lo cual reflejó el rendimiento del sistema en distintos escenarios de tráfico. A pesar de las variaciones observadas entre segmentos, el modelo basado en YOLO mantuvo una consistencia aceptable en la mayoría de los casos. Esta consistencia evidenció la capacidad del modelo para adaptarse y detectar correctamente los objetos críticos en diversas condiciones, lo cual es esencial para su aplicación en un entorno de conducción en el ámbito urbano guatemalteco. Los resultados sugirieron que el sistema respondía de manera eficiente ante cambios en el contexto de tráfico, logrando mantener una precisión elevada incluso en escenarios más complejos. Sin embargo, se identificaron algunos casos en los que el *recall* fue menor, indicando una posible omisión en la detección de ciertos elementos en condiciones específicas. La variabilidad en el *F1 Score* también reflejó la capacidad del sistema para equilibrar precisión y *recall*, aunque en ciertos segmentos se observaron ligeras caídas en estos valores. En general, el rendimiento del modelo fue satisfactorio, demostrando su efectividad para asistir al conductor sin grandes fluctuaciones en la precisión global.

El conjunto de *inputs* visuales y de los sensores, permite al sistema simular una conducción autónoma básica en un entorno controlado. Esto da las bases para que el usuario no arriesgue su vida ni infrinja las leyes de tránsito de Guatemala al evitar pruebas en calles reales de la ciudad. Según el Título 5, Capítulo I, Artículo 46 de las normas de tránsito guatemaltecas, los usuarios de la vía pública deben comportarse de manera que no entorpezcan la circulación ni causen peligro a otros usuarios (de la República de Guatemala, 1996). Además, el Artículo 47 especifica que los conductores deben actuar con diligencia y precaución para evitar daños propios o ajenos, y que deben estar en todo momento en condiciones de controlar su vehículo, evitando conductas negligentes o temerarias (de la República de Guatemala, 1996). Estas disposiciones son particularmente importantes en situaciones que conllevan riesgos, como pasarse una señal de alto o un semáforo en rojo. La simulación permitió eliminar estos riesgos, verificando la efectividad de las alertas sin poner en peligro la seguridad del usuario ni violar las leyes de tránsito evitando también las sanciones establecidas en el Artículo 196, inciso 50 donde se define la multa de 10 salarios mínimos del campo (smc) por no obedecer las señales de tránsito.

En cuanto al impacto del modelo YOLO en el sistema, se mostró la capacidad del modelo para detectar diferentes tipos de objetos con precisión. En la matriz de confusión normalizada que se observa en la Figura 6.6 permitió determinar qué clases fueron mejor identificadas y dónde hubo confusiones, especialmente entre clases visualmente similares. Las Curvas *F1-Confidence* 6.9 y *Precision-Recall* 6.15 fueron esenciales para evaluar el tiempo de respuesta del sistema de alertas, mostrando cómo variaban la precisión y el *recall* según el nivel de confianza del modelo. Estas curvas permitieron identificar los puntos óptimos de confianza donde el sistema alcanzó un buen equilibrio entre precisión y *recall*, optimizando así la efectividad de las alertas en tiempo real. Sin embargo hay potenciales mejoras que fueron detectadas en la matriz de falsos positivos, falsos negativos, verdaderos negativos y verdaderos positivos en la Figura 6.24 con el cual se facilitó el análisis detallado de cada clase en términos de errores de detección, permitiendo identificar las clases con mayores incidencias de falsos positivos y negativos. El sistema presentó una alta tasa de falsos positivos en detecciones de semáforos, especialmente en las clases *traffic_light_red*, *traffic_light_yellow*, y *traffic_light_green*, al ser confundidas con el fondo. Esto sugiere que ciertos ajustes en el entrenamiento del modelo podrían ser necesarios para mejorar la precisión en la clasificación de semáforos y reducir las confusiones con el fondo. La presencia de falsos positivos en ciertas clases podría generar alertas innecesarias, distrayendo al conductor y disminuyendo la eficiencia del sistema. Por otro lado, la reducción de falsos negativos es crucial para asegurar que los elementos críticos siempre sean detectados, especialmente en situaciones de riesgo. Es recomendable realizar un proceso de entrenamiento de tal modo que el modelo de detección de objetos sepa identificar no solo los elementos objetivo pero también los que no son para poder aprender a categorizar mejor los que son de importancia para el sistema.

Se demostró cómo el sistema alcanzó un balance adecuado entre precisión y *recall* en diferentes niveles de confianza, lo cual fue fundamental para cumplir con los objetivos de seguridad y eficiencia planteados. Las Figuras de *F1-Confidente Curve* 6.9 y *Precision-Recall Curve* 6.15 permitieron identificar los umbrales de confianza óptimos en los que el sistema mantuvo un alto nivel de precisión, minimizando el riesgo de falsas alarmas, y un *recall* adecuado, asegurando la detección de elementos críticos en el entorno de conducción. A niveles de confianza más altos, se observó una mayor precisión, lo cual redujo la probabilidad de distracciones innecesarias para el conductor. Sin embargo, un *recall* ligeramente reducido en estos niveles indicó que algunos elementos menos críticos podrían no haber sido detectados, lo cual es aceptable dentro de los parámetros de seguridad establecidos. En niveles de confianza más bajos, el *recall* se incrementó, permitiendo al sistema detectar una mayor variedad de elementos, aunque a expensas de una precisión ligeramente menor. Este balance entre precisión y *recall* respaldó el objetivo de mejorar la seguridad y eficiencia del conductor, permitiendo una reacción oportuna a eventos de importancia sin generar una sobrecarga de alertas.

CAPÍTULO 8

Conclusiones

Luego de un análisis de los resultados obtenidos, se llegó a las siguientes conclusiones de este proyecto:

1. El sistema de asistencia al conductor integra tecnologías de visión por computadora y análisis de datos de sensores, proporcionando detección y clasificación precisa de elementos críticos en el entorno vial contribuyendo significativamente al brindar la información necesaria en la toma de acciones para la seguridad y eficiencia en la conducción, con un tiempo de respuesta promedio de hasta 94.46 milisegundos (ms) y una precisión del 91.32 % en la generación de alertas.

2. El modelo de *machine learning*, optimizado con el algoritmo de YOLOv8, demostró ser altamente eficaz en la identificación y clasificación de señales de tránsito y otros elementos críticos en flujos de video en tiempo real con logrando la detección con una precisión del 91.32 %, lo que evidencia su capacidad para reconocer patrones de forma rápida y precisa, contribuyendo a la fiabilidad del sistema en condiciones diversas del entorno urbano vial guatemalteco.

3. La incorporación de datos de sensores de velocidad y aceleración mediante una simulación otorgan al sistema de asistencia al conductor la capacidad de ajustar las diversas alertas y recomendaciones relevantes que consideren con base en la dinámica de conducción del piloto permitiendo comunicar la información analizada para contribuir en la experiencia, seguridad y eficiencia de manejo.

4. El conjunto de alertas y recomendaciones en el sistema informan al conductor sobre potenciales peligros, por ejemplo una señal de alto, en tiempo real brindado la información necesaria para realizar los cambios en la conducción obedeciendo las señales de tránsito.

CAPÍTULO 9

Recomendaciones

Luego de la realización del trabajo se sugieren las siguientes recomendaciones para futuros trabajos o continuaciones de este proyecto:

1. Utilizar un entorno de pruebas seguro y controlado para evidenciar que la integración de sensores de velocidad y aceleración en un vehículo permiten realizar pruebas con datos no simulados y obtener métricas que demuestren con mayor precisión el comportamiento del sistema en condiciones de tráfico reales, garantizando así una validación efectiva de su desempeño y seguridad.

2. Entrenar el modelo de detección de objetos para ampliar el reconocimiento de los elementos de interés, sino también para identificar aquellos que no son relevantes pero se encuentran en el contexto de conducción. Este enfoque mejorará la capacidad del sistema para discriminar entre objetos críticos y no críticos, reduciendo los falsos positivos y asegurando que las alertas sean pertinentes para el conductor.

3. Implementar un sistema de almacenamiento para todas las entradas visuales que sean procesadas por el sistema de detección de objetos con la finalidad de recopilar datos que permitan crear una base de datos visual diversa, facilitando el reentrenamiento continuo del modelo. Manteniendo y mejorando su precisión en la identificación y clasificación de elementos en diferentes contextos y condiciones ambientales al contar con un conjunto de datos en constante crecimiento.

4. Incorporación de más componentes que robustezcan la funcionalidad y amplíen la ayuda que brinda el sistema. Por ejemplo, se pueden integrar modelos de detección y mantenimiento de carriles, monitoreo de proximidad con vehículos, visión estéreo y otras tecnologías que complementen las capacidades actuales del sistema, incrementando su utilidad y efectividad en entornos de conducción complejos.

CAPÍTULO 10

Bibliografía

Referencias

Aditi, y Dureja, A. (2021). A review: Image classification and object detection with deep learning. En (p. 69-91). doi: 10.1007/978-981-33-4604-8_6

Aleksa, M., Schaub, A., Erdelean, I., Wittmann, S., Soteropoulos, A., y Fürdös, A. (2024, 6). Impact analysis of advanced driver assistance systems (adas) regarding road safety – computing reduction potentials. *European Transport Research Review*, *16*, 39. doi: 10.1186/s12544-024-00654-0

Alomar, K., Aysel, H. I., y Cai, X. (2023, 2). Data augmentation in classification and segmentation: A survey and new strategies. *Journal of Imaging*, *9*, 46. doi: 10.3390/jimaging9020046

Alvarado, A. V. (2009, 8). El neocognitrón. *Universidad Mayor de San Andres*.

Anand, S., y Priya, L. (2019). *A guide for machine vision in quality control.* Chapman and Hall/CRC. doi: 10.1201/9781003002826

Andreasson, H., Grisetti, G., Stoyanov, T., y Pretto, A. (2023). Sensors for mobile robots. En (p. 1-22). Springer Berlin Heidelberg. doi: 10.1007/978-3-642-41610-1_159-1

Angelina, C., y Ulfitria, R. U. (2024). Classification of skin cancer using resnet and vgg deep learning network. En *Proceedings of the 11th international applied business and engineering conference, abec 2023, september 21st, 2023, bengkalis, riau, indonesia.* EAI. doi: 10.4108/eai.21-9-2023.2342881

Antony, M. M., y Whenish, R. (2021). Advanced driver assistance systems (adas). En (p. 165-181). doi: 10.1007/978-3-030-59897-6_9

Azfar, T., Li, J., Yu, H., Cheu, R. L., Lv, Y., y Ke, R. (2024, 4). Deep learning-based computer vision methods for complex traffic environments perception: A review. *Data Science for Transportation*, *6*, 1. doi: 10.1007/s42421-023-00086-7

Bakshee, R. (2021). *Edge detection in computer vision | by rahul bakshee | datadriveninvestor.* Descargado de https://medium.datadriveninvestor.com/edge-detection-in-computer-vision-6c2c25100bca

Barik, D., y Mondal, M. (2010, 6). Object identification for computer vision using image segmentation. En *2010 2nd international conference on education technology and computer* (p. V2-170-V2-172). IEEE. doi: 10.1109/ICETC.2010.5529412

Bertasius, G., Shi, J., y Torresani, L. (2015, 6). Deepedge: A multi-scale bifurcated deep network for top-down contour detection. En *2015 ieee conference on computer vision and pattern recognition (cvpr)* (p. 4380-4389). IEEE. doi: 10.1109/CVPR.2015.7299067

Burger, W., y Burge, M. J. (2022). *Digital image processing.* Springer International Publishing. doi: 10.1007/978-3-031-05744-1

Cadena, L., Zotin, A., Cadena, F., Korneeva, A., Legalov, A., y Morales, B. (2017). Noise reduction techniques for processing of medical images. En *World congress on engineering.*

Celin, T. A. M., Vijayalakshmi, P., y Nagarajan, T. (2023, 1). Data augmentation techniques for transfer learning-based continuous dysarthric speech recognition. *Circuits, Systems, and Signal Processing*, *42*, 601-622. doi: 10.1007/s00034-022-02156-7

Chai, Y. (2020, 3). In-sensor computing for machine vision. *Nature*, *579*, 32-33. doi: 10.1038/d41586-020-00592-6

CityCorners. (2024). *Recorriendo ciudad de guatemala en vehículo (parte 1) - guatemala 2024.* Youtube. Descargado de https://www.youtube.com/watch?v=oJvDaMmIa-Y

Cong, S., y Zhou, Y. (2023, 3). A review of convolutional neural network architectures and their optimizations. *Artificial Intelligence Review*, *56*, 1905-1969. doi: 10.1007/s10462-022-10213-5

Cong, X., Li, S., Chen, F., Liu, C., y Meng, Y. (2023, 6). A review of yolo object detection algorithms based on deep learning. *Frontiers in Computing and Intelligent Systems*, *4*, 17-20. doi: 10.54097/fcis.v4i2.9730

Contreras, S., y la Rosa, F. D. (2016, 9). Aplicación de deep learning en robótica móvil para exploración y reconocimiento de objetos basados en imágenes. En *2016 ieee 11th colombian computing conference (ccc)* (p. 1-8). IEEE. Descargado de http://ieeexplore.ieee.org/document/7750800/ doi: 10.1109/ColumbianCC.2016.7750800

Cork, D. (2020). Object recognition - computer vision. *Cerberus Nuclear*. Descargado de https://cerberusnuclear.com/object-recognition-computer-vision/

de Estadística, I. N. (2024). *Estadísticas continuas: Accidentes de tránsito.* Descargado de https://www.ine.gob.gt/bases-de-datos/accidentes-de-transito/

de la República de Guatemala, C. (1996, 6). *Ley de transito y su reglamento: Decreto 132-96.* Descargado de https://transito.gob.gt/wp-content/uploads/2015/06/Ley-y-Reglamento-Transito.pdf

Dhanya, V., Subeesh, A., Kushwaha, N., Vishwakarma, D. K., Kumar, T. N., Ritika, G., y Singh, A. (2022). Deep learning based computer vision approaches for smart agricultural applications. *Artificial Intelligence in Agriculture*, *6*, 211-229. doi: 10.1016/j.aiia.2022.09.007

Dong, X., y Cappuccio, M. L. (2023, 11). Applications of computer vision in autonomous vehicles: Methods, challenges and future directions. *School of Engineering and Technology, UNSW Canberra, Canberra, Australia*. doi: 10.13140/RG.2.2.28479.89766

E.R., D. (2018). *Computer vision* (Fifth Edition ed.). Elsevier. doi: 10.1016/C2015-0-05563-0

Esteva, A., Chou, K., Yeung, S., Naik, N., Madani, A., Mottaghi, A., ... Socher, R. (2021, 1). Deep learning-enabled medical computer vision. *npj Digital Medicine*, *4*, 5. doi: 10.1038/s41746-020-00376-2

Fang, W., Ding, L., Love, P. E., Luo, H., Li, H., Peña-Mora, F., ... Zhou, C. (2020, 2). Computer vision applications in construction safety assurance. *Automation in Construction*, *110*, 103013. doi: 10.1016/j.autcon.2019.103013

Fawzy, A. (2020). Faster r-cnn explained for object detection tasks. *Paperspace*.

for Transport, D. G. (2018). *Advanced driver assistance systems.* Descargado de https://road-safety.transport.ec.europa.eu/document/download/93f6a003-83dc-4e66-ae14-852d93236e57_en?filename=ersosynthesis2018-adas.pdf

Galvao, L. G., Abbod, M., Kalganova, T., Palade, V., y Huda, M. N. (2021, 10). Pedestrian and vehicle detection in autonomous vehicle perception systems—a review. *Sensors*, *21*, 7267. doi: 10.3390/s21217267

Gao, M., Zou, G., Li, Y., y Guo, X. (2024, 7). Recent advances in computer vision: Technologies and applications. *Electronics*, *13*, 2734. doi: 10.3390/electronics13142734

Grant, A. (2024). *Feature: Driver monitoring systems – trends and developments.* Descargado de https://www.autonomousvehicleinternational.com/features/feature-driver-monitoring-systems-trends-and-developments.html

Grigorescu, S., Trasnea, B., Cocias, T., y Macesanu, G. (2020, 4). A survey of deep learning techniques for autonomous driving. *Journal of Field Robotics*, *37*, 362-386. doi: 10.1002/

rob.21918

Han, X., Zhong, Y., Cao, L., y Zhang, L. (2017, 8). Pre-trained alexnet architecture with pyramid pooling and supervision for high spatial resolution remote sensing image scene classification. *Remote Sensing*, *9*, 848. doi: 10.3390/rs9080848

Hemaanand, M., Chowdary, P. R., Darshan, S., Jagadeeswaran, S., Karthika, R., y Parameswaran, L. (2020). Advanced driver assistance system using computer vision and iot. En (p. 768-778). doi: 10.1007/978-3-030-37218-7_85

Hladnik, A., y Gabrijelčič, H. (2016, 8). Surf: Detection, description and matching of local features in 3d computer graphics..

Hoiem, D., y Savarese, S. (2011, 8). Representations and techniques for 3d object recognition and scene interpretation. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, *5*, 1-169. doi: 10.2200/S00370ED1V01Y201107AIM015

Horgan, J., Hughes, C., McDonald, J., y Yogamani, S. (2015, 9). Vision-based driver assistance systems: Survey, taxonomy and advances. En *2015 ieee 18th international conference on intelligent transportation systems* (p. 2032-2039). IEEE. doi: 10.1109/ITSC.2015.329

Hu, L. (2023, 12). *Computer vision: 2023 recaps and 2024 trends*. Descargado de https://towardsai.net/p/l/computer-vision-2023-recaps-and-2024-trends

Hussain, M. (2023, 6). Yolo-v1 to yolo-v8, the rise of yolo and its complementary nature toward digital manufacturing and industrial defect detection. *Machines*, *11*, 677. doi: 10.3390/machines11070677

Hyams, G., y Malowany, D. (2020). The battle of speed vs. accuracy: Single-shot vs two-shot detection meta-architecture. *Clear ML*. Descargado de https://clear.ml/blog/the-battle-of-speed-accuracy-single-shot-vs-two-shot-detection

Hälker, J., y Barth, H. (2018, 8). Lidar as a key technology for automated and autonomous driving. *ATZ worldwide*, *120*, 70-73. doi: 10.1007/s38311-018-0102-z

Iftikhar, S., Zhang, Z., Asim, M., Muthanna, A., Koucheryavy, A., y El-Latif, A. A. A. (2022, 10). Deep learning-based pedestrian detection in autonomous vehicles: Substantial issues and challenges. *Electronics*, *11*, 3551. doi: 10.3390/electronics11213551

Iglesias, L. L., Bellón, P. S., del Barrio, A. P., Fernández-Miranda, P. M., González, D. R., Vega, J. A., ... Blanco, J. A. P. (2021, 12). A primer on deep learning and convolutional neural networks for clinicians. *Insights into Imaging*, *12*, 117. doi: 10.1186/s13244-021-01052-z

Ivanov, A., y Chivarov, N. (2020, 6). Methods for object recognition and classification for tele-controlled service robots. *IOP Conference Series: Materials Science and Engineering*, *878*, 012005. Descargado de https://iopscience.iop.org/article/10.1088/1757-899X/878/1/012005 doi: 10.1088/1757-899X/878/1/012005

Jiang, J., Xu, H., Zhang, S., y Fang, Y. (2019, 5). Object detection algorithm based on multiheaded attention. *Applied Sciences*, *9*, 1829. Descargado de https://www.mdpi.com/2076-3417/9/9/1829 doi: 10.3390/app9091829

Joiya, F. (2022, 9). Object detection: Yolo vs faster r-cnn. *International Research Journal of Modernization in Engineering Technology and Science*. doi: 10.56726/IRJMETS30226

Khan, A., Sohail, A., Zahoora, U., y Qureshi, A. S. (2020, 12). A survey of the recent architectures of deep convolutional neural networks. *Artificial Intelligence Review*, *53*, 5455-5516. doi: 10.1007/s10462-020-09825-6

Khan, A. A., Laghari, A. A., y Awan, S. A. (2021, 4). Machine learning in computer vision: A review. *EAI Endorsed Transactions on Scalable Information Systems*, *8*(32). doi: 10.4108/eai.21-4-2021.169418

Khare, M. D., y Raghavendra, R. (2024). Exploring sensor technologies and automation levels in autonomous vehicles. En (p. 293-304). doi: 10.1007/978-981-99-8135-9_26

Konstantinidis, F. K., Mouroutsos, S. G., y Gasteratos, A. (2021, 8). The role of machine vision in industry 4.0: an automotive manufacturing perspective. En *2021 ieee international conference on imaging systems and techniques (ist)* (p. 1-6). IEEE. doi: 10.1109/IST50367.2021.9651453

Lim, X. R., Lee, C. P., Lim, K. M., Ong, T. S., Alqahtani, A., y Ali, M. (2023, 5). Recent advances in traffic sign recognition: Approaches and datasets. *Sensors*, *23*, 4674. doi: 10.3390/s23104674

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., y Berg, A. C. (2016). Ssd: Single shot multibox detector. En (p. 21-37). doi: 10.1007/978-3-319-46448-0_2

Lowe, D. (1999). Object recognition from local scale-invariant features. En *Proceedings of the seventh ieee international conference on computer vision* (p. 1150-1157 vol.2). IEEE. doi: 10.1109/ICCV.1999.790410

Makone, A. (2020). *Download citation of the landscape of deep learning based object detection techniques in computer vision.* Descargado de https://www.researchgate.net/publication/347439520_The_Landscape_of_Deep_Learning_Based_Object_Detection_Techniques_in_Computer_Vision/citation/download doi: 10.13140/RG.2.2.12679.42401

Martinez, J. (2022). *Guía de iniciación a la detección de objetos para proyectos de computer vision - datasmarts.* Descargado de https://www.datasmarts.net/guia-de-iniciacion-a-la-deteccion-de-objetos-para-proyectos-de-computer-vision/

Montserrat, D. M., Lin, Q., Allebach, J., y Delp, E. J. (2017, 1). Training object detection and recognition cnn models using data augmentation. *Electronic Imaging*, *29*, 27-36. doi: 10.2352/ISSN.2470-1173.2017.10.IMAWM-163

Murthy, C. B., Hashmi, M. F., Bokde, N. D., y Geem, Z. W. (2020, 5). Investigations of object detection in images/videos using various deep learning techniques and embedded platforms—a comprehensive review. *Applied Sciences*, *10*, 3280. Descargado de https://www.mdpi.com/2076-3417/10/9/3280 doi: 10.3390/app10093280

on Crash Reporting, S. G. O. (2022). *Standing general order on crash reporting | nhtsa.* Descargado de https://www.nhtsa.gov/laws-regulations/standing-general-order-crash-reporting

Patel, R., y Patel, S. (2020, 8). A comprehensive study of applying convolutional neural network for computer vision. *International Journal of Advanced Science and Technology*, *6*, 2161-2174.

Poojari, S. (2023). Histograms of oriented gradients for object detection a comprehensive review. *Irjmets*.

Ren, S., He, K., Girshick, R., y Sun, J. (2016). *Faster r-cnn: Towards real-time object detection with region proposal networks.* Microsoft Research.

Rezaei, M., y Klette, R. (2017). *Computer vision for driver assistance* (Vol. 45). Springer International Publishing. doi: 10.1007/978-3-319-50551-0

Riggs, H., Tufail, S., Parvez, I., Tariq, M., Khan, M. A., Amir, A., ... Sarwat, A. I. (2023, 4). Impact, vulnerabilities, and mitigation strategies for cyber-secure critical infrastructure. *Sensors*, *23*, 4060. doi: 10.3390/s23084060

Samantaray, R. (2023, 3). Adas sensor data handling in the world of autonomous mobility.. doi: 10.4271/2023-01-0993

Sarker, I. H. (2021, 5). Machine learning: Algorithms, real-world applications and research directions. *SN Computer Science*, *2*, 160. doi: 10.1007/s42979-021-00592-x

Shen, Q., Zhang, L., Zhang, Y., Li, Y., Liu, S., y Xu, Y. (2024, 8). Distracted driving behavior detection algorithm based on lightweight stardl-yolo. *Electronics*, *13*, 3216. doi: 10.3390/electronics13163216

Shorten, C., y Khoshgoftaar, T. M. (2019, 12). A survey on image data augmentation for deep learning. *Journal of Big Data*, *6*, 60. doi: 10.1186/s40537-019-0197-0

Singh, C. (2021). Applications and challenges of deep learning in computer vision. En (p. 223-233). doi: 10.1007/978-3-030-90885-0_20

Sree, B. B., Bharadwaj, V. Y., y Neelima, N. (2021). An inter-comparative survey on state-of-the-art detectors—r-cnn, yolo, and ssd. En (p. 475-483). doi: 10.1007/978-981-33-4443-3_46

Szeliski, R. (2022). *Computer vision:algorithms and applications 2nd edition.* Springer International Publishing. doi: 10.1007/978-3-030-34372-9

Terven, J., Córdova-Esparza, D.-M., y Romero-González, J.-A. (2023, 11). A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas. *Machine Learning and Knowledge Extraction*, *5*, 1680-1716. doi: 10.3390/make5040083

Tesis. (2022, 11). *Señales de tránsito dataset.* Roboflow. Descargado de https://universe.roboflow.com/tesis-lhuim/senales-de-transito (Visited on 2024-08-11)

Toshev, A. (2011). Shape representations for object recognition. *University of Pennsylvania*.

Triki, N., Karray, M., y Ksantini, M. (2023, 4). A real-time traffic sign recognition method using a new attention-based deep convolutional neural network for smart vehicles. *Applied Sciences*, *13*, 4793. doi: 10.3390/app13084793

Tseng, Y.-H., y Jan, S.-S. (2018, 4). Combination of computer vision detection and segmentation for autonomous driving. En *2018 ieee/ion position, location and navigation symposium (plans)* (p. 1047-1052). IEEE. doi: 10.1109/PLANS.2018.8373485

Umeaduma, L. I. (2024, 1). Survey of image classification models for transfer learning. *World Journal of Advanced Research and Reviews*, *21*, 373-383. doi: 10.30574/wjarr.2024.21.1.0006

Varriale, V., Cammarano, A., Michelino, F., y Caputo, M. (2023, 11). Critical analysis of the impact of artificial intelligence integration with cutting-edge technologies for production systems. *Journal of Intelligent Manufacturing*. doi: 10.1007/s10845-023-02244-8

Velez, G., y Otaegui, O. (2015, 8). Embedded platforms for computer vision-based advanced driver assistance systems: a survey.. doi: https://doi.org/10.48550/arXiv.1504.07442

Vijayakumar, A., y Vairavasundaram, S. (2024, 3). Yolo-based object detection models: A review and its applications. *Multimedia Tools and Applications*. doi: 10.1007/s11042-024-18872-y

Viola, P., y Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. En *Proceedings of the 2001 ieee computer society conference on computer vision and pattern recognition. cvpr 2001* (p. I-511-I-518). IEEE Comput. Soc. doi: 10.1109/CVPR.2001.990517

Wang, D., Wang, J.-G., y Xu, K. (2021, 11). Deep learning for object detection, classification and tracking in industry applications. *Sensors*, *21*, 7349. doi: 10.3390/s21217349

Wang, R.-B., Yan, D.-K., y Xu, L. (2021). A brief survey on recent advances of object detection with deep learning. En (p. 348-356). doi: 10.1007/978-981-33-6420-2_43

Wang, S., Ge, F., y Liu, T. (2006, 12). Evaluating edge detection through boundary detection. *EURASIP Journal on Advances in Signal Processing*, *2006*, 076278. doi: 10.1155/ASP/2006/76278

Wang, Y., Han, Y., Wang, C., Song, S., Tian, Q., y Huang, G. (2023). *Computation-efficient deep learning for computer vision: A survey.*

Yaseen, M. (2024, 8). What is yolov8: An in-depth exploration of the internal features of the next-generation object detector. *National University of Computer and Emerging Sciences, Lahore 54770, Pakistan*.

Yu, Z., y Ye, T. (2023, 12). Autonomous traffic sign detection for self-driving car system using convolutional neural network algorithm. *Journal of Optics*. doi: 10.1007/s12596-023-01518-x

Zaidi, S. S. A., Ansari, M. S., Aslam, A., Kanwal, N., Asghar, M., y Lee, B. (2022, 6). A survey of modern deep learning based object detection models. *Digital Signal Processing*, *126*, 103514. doi: 10.1016/j.dsp.2022.103514

Zhang, X., Li, J., Li, Z., Liu, H., Zhou, M., Wang, L., y Zou, Z. (2023). *Multi-sensor fusion for autonomous driving*. Springer Nature Singapore. doi: 10.1007/978-981-99-3280-1

Önler, E., y Köycü, N. D. (2024, 8). Wheat powdery mildew detection with yolov8 object detection model. *Applied Sciences*, *14*, 7073. doi: 10.3390/app14167073

Özdenizci, O., y Legenstein, R. (2023, 8). Restoring vision in adverse weather conditions with patch-based denoising diffusion models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *45*, 10346-10357. doi: 10.1109/TPAMI.2023.3238179