UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CARRERA DE SOFTWARE

TRABAJO DE INTEGRACIÓN CURRICULAR PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SOFTWARE

"IMPLEMENTACIÓN DE UNA RED NEURONAL CONVOLUCIONAL YOLOv8 PARA LA DETECCIÓN DE SOMNOLENCIA EN CONDUCCIÓN DIURNA Y NOCTURNA"

AUTOR:

Steven Alexis Solano Vásquez

DIRECTOR:

PhD. Iván Danilo García Santillán

Ibarra-Ecuador

2026

UNIVERSIDAD TÉCNICA DEL NORTE

BIBLIOTECA UNIVERSITARIA

IDENTIFICACIÓN DE LA OBRA

En cumplimiento del Art. 144 de la Ley de Educación Superior, hago la entrega del presente trabajo a la Universidad Técnica del Norte para que sea publicado en el Repositorio Digital Institucional, para lo cual pongo a disposición la siguiente información:

DATOS DE CONTACTO			
CÉDULA DE IDENTIDAD:	1003975909		
APELLIDOS Y NOMBRES:	SOLANO VÁSQUEZ STEVEN ALEXIS		
DIRECCIÓN:	ATUNTAQUI – ABDÓN CALDERÓN Y RIO SANTIAGO		
EMAIL:	sasolanov@utn.edu.ec, alexis.solano111@gmail.com		
TELÉFONO FIJO:	062 530 370	**TELF. MÓVIL**	0983769684

DATOS DE LA OBRA	
TÍTULO:	IMPLEMENTACIÓN DE UNA RED NEURONAL CONVOLUCIONAL YOLOv8 PARA LA DETECCIÓN DE SOMNOLENCIA EN CONDUCCIÓN DIURNA Y NOCTURNA.
AUTOR:	SOLANO VÁSQUEZ STEVEN ALEXIS
FECHA DE APROBACIÓN:	20/02/2026
SOLO PARA TRABAJOS DE INTEGRACIÓN CURRICULAR	
CARRERA/PROGRAMA:	■ **GRADO** □ **POSGRADO**
TÍTULO POR EL QUE OPTA:	INGENIERO EN SOFTWARE
DIRECTOR:	PHD. IVÁN GARCÍA
ASESOR:	MSC. PEDRO GRANDA

CONSTANCIAS

El autor manifiesta que la obra objeto de la presente autorización es original y se la desarrolló, sin violar derechos de autor de terceros, por lo tanto, la obra es original y que es el titular de los derechos patrimoniales, por lo que asume la responsabilidad sobre el contenido de la misma y saldrá en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, a los 20 días del mes de febrero de 2026

EL AUTOR:

Firmado electrónicamente por:
STEVEN ALEXIS
SOLANO VASQUEZ
Validar únicamente con FirmaEC

Solano Vásquez Steven Alexis

C.C.: 1003975909

CERTIFICACIÓN DEL DIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR

Ibarra, a los 20 días del mes de febrero de 2026

PhD. Iván García. MSc.

DIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR

CERTIFICA:

Haber revisado el presente informe final del trabajo de Integración Curricular, el mismo que se ajusta a las normas vigentes de la Universidad Técnica del Norte; en consecuencia, autorizo su presentación para los fines legales pertinentes.

IVAN DANILO GARCIA SANTILLAN

Digitally signed by IVAN DANILO GARCIA SANTILLAN
Date: 2026.02.20 15:34:49 -05'00'

PhD. Iván García. MSc.

C.C.: 1002292603

DEDICATORIA

A la memoria de mi padre, quien, a pesar de su partida, sigue siendo el pilar fundamental de mi vida y mi mayor motivación; tu legado inspira cada uno de mis logros.

A mi madre, por su amor infinito y por guiarme con esa mezcla única de firmeza y ternura a lo largo de mi formación académica.

A mis amigos Karlita, Mateo, Joshua y Jhonatan, con quienes compartí esfuerzos, aprendizajes y momentos inolvidables que marcaron mi formación personal y profesional.

Steven Alexis Solano Vásquez

AGRADECIMIENTO

Agradezco profundamente al PhD. Iván García, director de esta tesis, por su guía académica, su rigor metodológico y por haber compartido con generosidad sus conocimientos a lo largo de este proceso. Su acompañamiento fue clave para el desarrollo y culminación de este proyecto.

Expreso también mi gratitud a la Universidad Técnica del Norte y a la carrera de Ingeniería en Software, por brindarme la formación y las herramientas necesarias para crecer tanto en el ámbito profesional como personal.

A mis compañeros de carrera, por los años compartidos de aprendizaje, esfuerzo y camaradería.

Y a todas las personas que, de una u otra manera, contribuyeron con palabras, apoyo técnico, tiempo o aliento en este camino.

Steven Alexis Solano Vásquez

TABLA DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

La detección de somnolencia al conducir, tanto en condiciones diurnas como nocturnas, representa un desafío crucial en la prevención de accidentes de tránsito. Diversos factores como el agotamiento físico por extensas jornadas laborales, problemas de salud que afectan la concentración o condiciones ambientales adversas, contribuyen a elevar el riesgo de fatiga al volante. En respuesta a esta problemática, el presente trabajo propone el desarrollo e implementación de un sistema inteligente basado en la arquitectura de redes neuronales convolucionales YOLOv26, con el objetivo de identificar el estado de los ojos del conductor (abiertos o cerrados) en tiempo real.

El modelo fue entrenado utilizando un conjunto de imágenes etiquetadas que representa distintos escenarios visuales, incluyendo condiciones diurnas y nocturnas. Posteriormente, se integró en un sistema embebido utilizando una Raspberry Pi, con capacidad para emitir alertas sonoras en caso de detectar signos de somnolencia prolongada.

La estructura del proyecto se organiza en tres capítulos: el primero aborda el estudio del problema de la somnolencia en la conducción, analizando investigaciones previas y tecnologías existentes aplicadas a su detección; el segundo capítulo detalla la construcción del *dataset*, la preparación de los datos y el proceso de entrenamiento del modelo YOLOv26; finalmente, en el tercer capítulo se presentan los resultados obtenidos, los cuales fueron validados mediante pruebas controladas en distintos contextos (rostros con y sin gafas, condiciones de baja iluminación). Se incluyen comparaciones con métricas reportadas en estudios anteriores, así como pruebas de funcionamiento en entornos simulados y reales.

Palabras clave: Detección de somnolencia, Visión por computadora, YOLO, Aprendizaje profundo, Estado ocular, Seguridad vial.

ABSTRACT

Drowsiness detection while driving, both in daytime and nighttime conditions, represents a critical challenge in the prevention of traffic accidents. Various factors such as physical exhaustion from long working hours, health conditions that impair concentration, or adverse environmental conditions—contribute to an increased risk of fatigue behind the wheel. In response to this issue, the present work proposes the development and implementation of an intelligent system based on the YOLOv12 convolutional neural network architecture, aimed at identifying the driver's eye state (open or closed) in real time.

The model was trained using a labeled dataset representing diverse visual scenarios, including both daytime and nighttime conditions. It was then deployed on an embedded system using a Raspberry Pi, with the ability to trigger audible alerts upon detecting signs of prolonged drowsiness.

This project is structured into three chapters: the first addresses the study of the drowsiness problem in driving, reviewing previous research and existing technologies applied to its detection; the second chapter describes the dataset construction, data preparation, and the YOLOv26 training process; finally, the third chapter presents the results, which were validated through controlled tests in various contexts (faces with and without glasses, low-light conditions). It also includes metric comparisons with previous studies, as well as performance evaluations in both simulated and real-world environments.

Keywords: Drowsiness detection, Computer vision, YOLO, Deep learning, Eye state, Road safety.

INTRODUCCIÓN

Tema

Implementación de una red neuronal convolucional YOLO para detección de somnolencia en conducción diurna y nocturna.

Problema

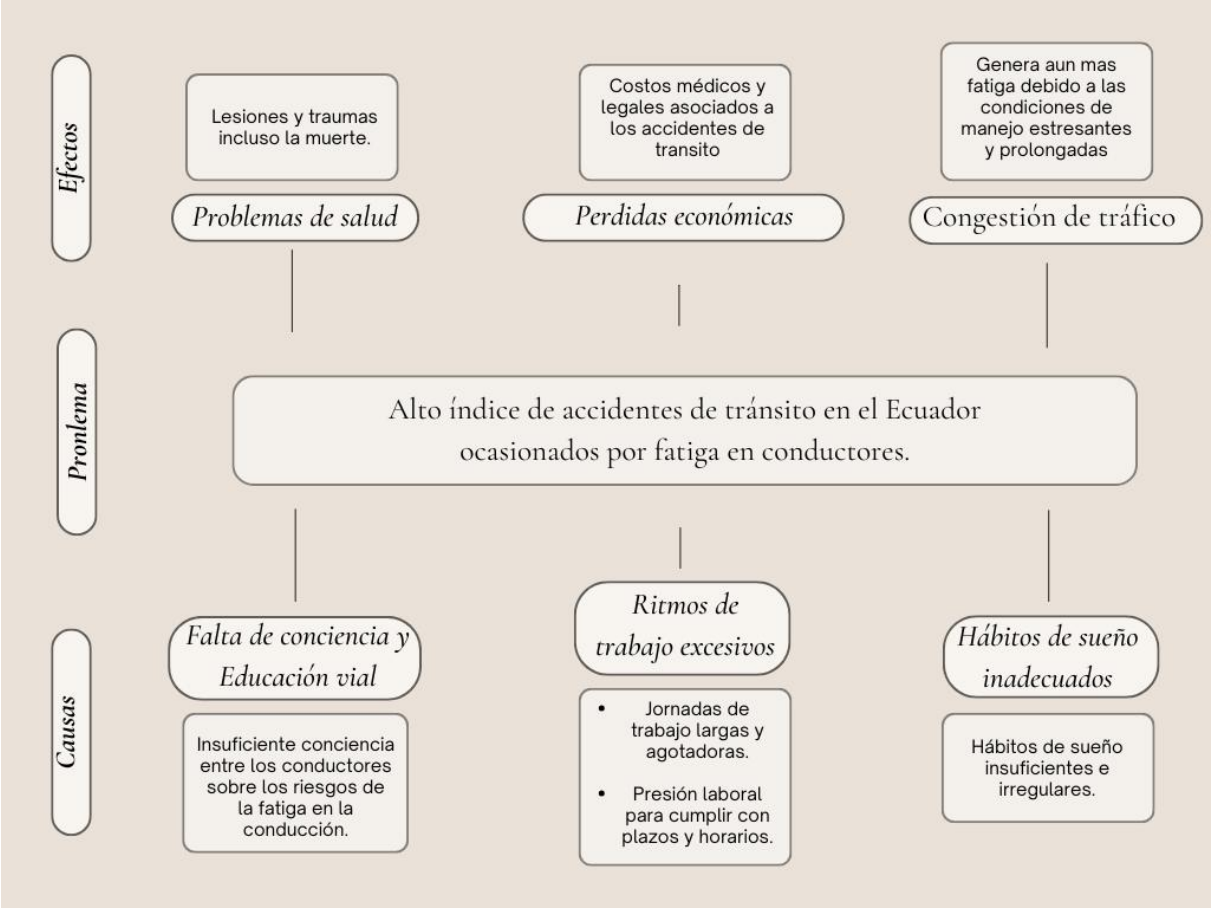
La somnolencia en conductores es un problema crítico en la seguridad vial que continúa siendo una de las principales causas de siniestros a nivel global. En Ecuador, las estadísticas recientes de accidentes de tránsito subrayan su importancia. Según la Organización Mundial de la Salud (OMS), la fatiga se mantiene como un factor significativo en los accidentes viales. Además, el último Reporte Nacional de la Agencia Nacional de Tránsito (ANT) de 2021 reveló que el 0,6% de los accidentes fatales en el país fueron causados por conductores en estado de somnolencia, y un 0,2% de los conductores en dicho estado resultaron lesionados(OPS & OMS, 2018).

La preocupación por la somnolencia en conductores ha ido en aumento con el tiempo. La creciente movilidad y la evolución de la tecnología automotriz han ampliado las oportunidades para el desplazamiento de largas distancias, aumentando así el riesgo de fatiga y somnolencia al volante.

La literatura existente revela que la somnolencia afecta la capacidad de respuesta, la atención y la toma de decisiones de los conductores, lo que puede desencadenar accidentes graves. Además, se han implementado varias estrategias y tecnologías para abordar este problema, incluida la detección de somnolencia mediante sistemas de visión por computadora (Jabbar et al., 2018).

En el informe del Instituto Nacional de Estadística y Censos (INEC) del I y II Trimestre de 2022, se destaca un incremento del 9% en la cantidad de siniestros de tránsito en comparación con el mismo período de 2021, lo que subraya la urgencia de abordar los factores de riesgo, como la somnolencia, en la seguridad vial(INEC, 2022).

Figura 1.
Árbol de problemas



Nota. Autoría propia.

Objetivos

Objetivo General.

Implementar una red neuronal convolucional YOLO para la detección de somnolencia en conducción diurna y nocturna.

Objetivos Específicos

- Realizar una revisión de la literatura para establecer una base sólida para la investigación sobre la detección de somnolencia en conductores y la implementación de YOLO.
- Implementar YOLO en un sistema embebido y desarrollar un sistema de detección de somnolencia en conductores de vehículos.

- Evaluar la precisión y eficacia del sistema de detección de somnolencia en un entorno controlado, utilizando simulaciones de conducción y aplicando métricas estándares en el campo de la inteligencia artificial. Además, se evaluará la portabilidad basados en la norma ISO 25023.

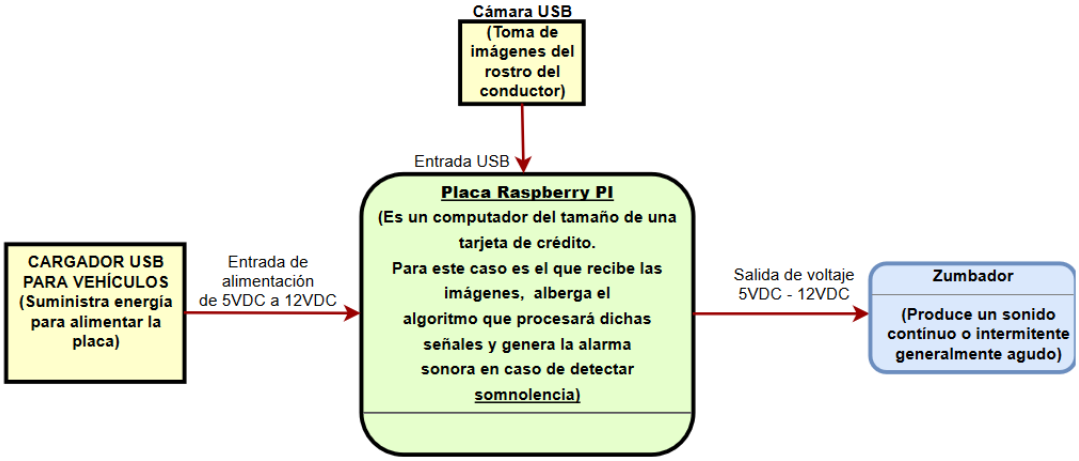
Alcance

El presente trabajo tiene como objetivo desarrollar un sistema de detección de somnolencia en conductores durante la conducción diurna y nocturna, utilizando la librería *TensorFlow* con el lenguaje de programación Python, para implementar redes neuronales convolucionales. Este sistema se centrará en la detección de patrones relacionados con el cierre de ojos y otras señales de somnolencia en conductores a través del análisis de expresiones oculares en tiempo real.

La investigación incluirá un análisis de la literatura existente relacionada con la somnolencia en conductores, técnicas de reconocimiento facial aplicables, con el fin de establecer una base sólida para el desarrollo y la implementación de redes neuronales convolucionales.

La implementación de la red neuronal convolucional YOLO se llevará a cabo para configurar un conjunto de datos de entrenamiento adecuado para la detección de somnolencia. El sistema se desarrollará con el propósito de identificar señales de somnolencia en conductores a partir de datos de video y expresiones oculares en tiempo real obtenidos de una cámara de video para así poder emitir una alerta mediante un *buzzer*, el sistema será integrado en un sistema embebido de hardware y software libre Jetson Nano y/o Raspberry Pi como se muestra en la Figura 2.

Figura 2.
Arquitectura en Sistema embebido



Nota: El gráfico describe la estructura del sistema embebido (Alba, 2020)

La validación del sistema se realizará en un entorno controlado, utilizando simulaciones de conducción. Se evaluará la eficacia y precisión del sistema basados en métricas estándar del campo de inteligencia artificial: precisión, numero de parámetros, peso del modelo y tiempo de entrenamiento. Además, se usará la estadística descriptiva para la representación de los resultados obtenidos. Para asegurar la portabilidad del sistema, se aplicará una evaluación basada en la norma ISO/IEC 25023, considerando los criterios de adaptabilidad, instalabilidad y coexistencia.

Metodología

El presente trabajo se desarrollará bajo un enfoque de investigación aplicada, con el propósito de diseñar una solución tecnológica con utilidad práctica en el ámbito de la seguridad vial, mediante la detección de somnolencia en tiempo real durante la conducción, tanto en condiciones diurnas como nocturnas. Para ello, se empleará la arquitectura de redes neuronales convolucionales YOLOv26, reconocida por su eficiencia en tareas de detección de objetos en escenarios complejos y dinámicos.

La metodología general se estructurará siguiendo el enfoque KDD (*Knowledge Discovery in Databases*). En una primera etapa, se llevará a cabo una revisión de literatura científica relacionada con la detección de somnolencia, visión por computadora y modelos de aprendizaje profundo, con el objetivo de establecer una

base teórica sólida y definir los parámetros técnicos adecuados para el desarrollo del sistema.

La construcción del dataset se realizará combinando bases de datos públicas existentes como 300W, Helen, CEW y otras fuentes relevantes del ámbito de la visión computacional, además de imágenes recolectadas por los propios tesistas en diversas condiciones de iluminación y expresión facial.

El entrenamiento del modelo YOLOv26 se realizará en un entorno local, utilizando Python y las bibliotecas *Ultralytics*, *OpenCV* y *PyTorch*, con el soporte de una GPU RTX 4070 Ti. Una vez que el modelo alcance el rendimiento deseado, se procederá a su implementación en un sistema embebido basado en Raspberry Pi, que será capaz de ejecutar inferencias en tiempo real y activar una alerta sonora mediante un *buzzer* cuando se detecten patrones prolongados de cierre ocular.

La evaluación del sistema se llevará a cabo aplicando los criterios establecidos en la norma ISO/IEC 25023, considerando dimensiones como funcionalidad, eficiencia de desempeño y portabilidad. Con ello, se buscará asegurar que la propuesta no solo sea técnicamente precisa, sino también factible para su integración en contextos reales de conducción vehicular.

Justificación

Este trabajo pretende contribuir con el objetivo 9 de Desarrollo Sostenible: Contribuir infraestructura resiliente, promover la industrialización sostenible y fomentar la innovación. El objetivo tiene varias metas, y este trabajo se enfoca en la siguiente, “Apoyar el desarrollo de tecnologías, la investigación y la innovación nacionales en los países en desarrollo, incluso garantizando un entorno normativo propicio a la diversificación industrial y la adición de valor a los productos básicos, entre otras cosas” (Naciones Unidas, 2023).

Se pretende también, contribuir en el Plan de Creación de oportunidades 2021-2025 de la República del Ecuador, en el objetivo 9 del Eje Seguridad Integral, donde se menciona: “Garantizar la seguridad ciudadana, orden público y gestión de riesgos”,

y que trata en una de sus políticas, el fortalecimiento de la seguridad de los sistemas de transporte terrestre y aéreo, promoviendo ambientes seguros(Secretaría Nacional de Planificación, 2021).

Justificación Tecnológica. - Radica en la aplicación de tecnologías de vanguardia, como la visión por computadora y modelos de detección de objetos, para desarrollar un sistema de alerta de somnolencia en conductores. Esta tecnología no solo aborda un problema crítico de seguridad vial, sino que también demuestra el potencial de la inteligencia artificial y la visión computacional en la creación de soluciones innovadoras y eficaces para la detección de riesgos en situaciones de la vida real.

Justificación Social. - La implementación de este sistema no solo mejora la seguridad vial, sino que también destaca el valor de la tecnología en la resolución de desafíos prácticos y el desarrollo de estrategias que tienen un impacto positivo en la sociedad.

CAPÍTULO 1: Marco Teórico

1.1. Somnolencia y sus efectos en la conducción

¿Qué es la somnolencia?

Es una necesidad fisiológica para los humanos, se puede definir como la proclividad a quedarse dormido, también llamada proclividad a iniciar el sueño. La presencia e intensidad de esta necesidad se puede determinar por la rapidez con que una persona se duerme, la facilidad con que se despierta y la cantidad de tiempo que duerme (Rosales Mayor & De Castro Mujica, 2010).

Efectos en la conducción.

(Dirección General de Tráfico - España, 2022) describió los cambios provocados por la somnolencia que afectan al conductor:

- Aumento del tiempo de reacción
- Menos concentración y más distracción
- Toma de decisiones más lenta y más errores
- Movimientos más automatizados
- Deterioro notable de la visión.
- Alucinaciones e ilusiones visuales.
- Comportamiento errático.

Se ha documentado que, en las primeras horas de la mañana, después de 24 horas sin dormir, el rendimiento psicomotor puede disminuir en igual o mayor medida que el de una intoxicación por alcohol definida como una concentración de alcohol en sangre mayor al 0,10% (Rosales Mayor & De Castro Mujica, 2010).

1.1.1. Factores que contribuyen a la somnolencia

La somnolencia, que se caracteriza por sentirse más adormilado de lo habitual durante el día, puede conducir a episodios involuntarios de sueño o a momentos potencialmente peligrosos en términos de seguridad. Este estado, cuando ocurre de manera excesiva y sin una causa identificada, puede indicar la presencia de un trastorno del sueño. Además, factores como la depresión, la ansiedad, el estrés y el

aburrimiento pueden contribuir a esta somnolencia aumentada (Sharafkhaneh & Hirshkowitz, 2022).

1.1.2. Métodos tradicionales de detección de somnolencia

Se pueden agrupar en tres categorías, a saber, las que siguen actualmente la mayoría de los diferentes grupos de investigación y desarrollo en el área, como se describe en (Muhammad et al., 2019):

- Técnicas basadas en parámetros de comportamiento del conductor.
- Técnicas basadas en parámetros del vehículo.
- Técnicas basadas en parámetros fisiológicos

1.1.3. Técnicas basadas en parámetros de comportamiento del conductor

Los métodos de detección de somnolencia basados en el comportamiento del conductor no son invasivos y utilizan tecnologías de visión artificial para observar y detectar parámetros indicativos del estado del sujeto, como el ritmo de cierre de los ojos o los bostezos (Jiménez, 2021).

Detección de ojos.

Con el propósito de calcular la frecuencia de cierre de estos para evaluar el nivel de somnolencia o fatiga. Se emplean diversas técnicas para la detección, considerando múltiples parámetros como el comportamiento del conductor, movimiento de la cabeza, parámetros fisiológicos y dirección del vehículo. Entre las métricas utilizadas, destaca la tasa PERCLOS, que indica el porcentaje de tiempo que el párpado cubre la pupila (> 80%) en una ventana móvil de 60 segundos (Torres, 2022).

Detección de boca y bostezos

Se realiza un análisis de la apertura de la boca, generando una alerta si se detecta un número elevado de fotogramas consecutivos con una apertura significativa. Se establece una relación geométrica entre el ancho y alto de la boca para identificar bostezos.

1.1.4. Técnicas basadas en parámetros del vehículo

Esta categoría de enfoques abarca todas aquellas técnicas que se basan en los parámetros inherentes a la conducción, siendo el análisis de la dirección el método más ampliamente utilizado dentro de este grupo. Estas estrategias se centran en la evaluación de datos relacionados con el manejo del vehículo, y entre ellas, la evaluación de la dirección del vehículo se destaca como la más común y generalizada (Jiménez, 2021).

Detección basada en análisis de la dirección

Esta estrategia implica la recopilación y evaluación de datos relacionados con el posicionamiento angular del volante de dirección, combinados con información de velocidad, con el objetivo de identificar patrones de somnolencia que puedan afectar la conducción de manera errática. Se centran en analizar el ángulo del volante, que a su vez está vinculado a la geometría y la curvatura de la carretera.

Detección de carril

Existen otras técnicas menos comunes que proponen enfoques alternativos, como la detección y análisis de la posición del vehículo en relación con el carril de circulación. Un ejemplo de esto es donde el sistema emite una alerta si el vehículo se desvía de la pista marcada por las líneas del carril.

1.1.5. Técnicas basadas en parámetros fisiológicos

Estas estrategias se basan en parámetros fisiológicos, centrándose particularmente en el análisis de la actividad cerebral y en el estudio de las fluctuaciones del pulso cardíaco. Dentro de este enfoque, se busca comprender y evaluar aspectos relacionados con la función cerebral y el ritmo cardíaco, utilizando estas señales fisiológicas como indicadores clave para determinar estados como la somnolencia o fatiga (Jiménez, 2021).

Numerosos estudios sobre la evaluación fisiológica de la somnolencia del conductor han identificado una secuencia coherente de cambios en las señales fisiológicas que acompañan la transición de la vigilia a la somnolencia. En contraste con la vigilancia por vídeo, que se basa principalmente en el reconocimiento facial, la medición fisiológica presenta ventajas significativas, como un menor riesgo de violación de la privacidad, un menor consumo de recursos informáticos y una mayor robustez frente a rostros ocultos. Además, los sensores fisiológicos pueden integrarse fácilmente en los dispositivos de los vehículos sin requerir hardware adicional, aprovechando elementos como el asiento del automóvil, el cinturón de seguridad, el volante y accesorios portátiles como gafas, auriculares y brazaletes (Y. Wu et al., 2023).

Análisis de electroencefalografía (EEG)

La adquisición de señales de un electroencefalograma (EEG) se presenta como un método excelente para determinar la presencia de síntomas de somnolencia en un individuo al evaluar la actividad cerebral.

Análisis de pulso cardíaco (ECG)

Otra técnica ampliamente utilizada y precisa consiste en analizar el ritmo cardíaco del sujeto en estudio, ya que se ha confirmado que el sistema nervioso experimenta alteraciones ante situaciones de estrés, fatiga y somnolencia. A través del análisis de la Variabilidad del Ritmo Cardíaco (HRV), se busca identificar el inicio de episodios de somnolencia.

1.2. Trabajos relacionados.

A vision-based drowsiness detection system for railway operators using lightweight convolutional neural networks

(Lozano-Reyes & Mugruza-Vassallo, 2025) Desarrolló un sistema de detección de somnolencia basado en visión para operadores ferroviarios utilizando redes neuronales convolucionales ligeras. El sistema emplea la arquitectura YOLOv8 (específicamente la variante YOLOv8n-cls) para monitorear en tiempo real indicadores de fatiga como la duración del cierre de los párpados y los bostezos, utilizando un umbral de 833 ms y puntos de referencia faciales sobre un conjunto de datos personalizado de 6,991 fotogramas. El resultado más relevante fue una precisión

general del 96.8%, alcanzando una exactitud cercana al 98% en pruebas nocturnas, lo que valida su eficacia para entornos de seguridad crítica. Sin embargo, el estudio reconoció limitaciones relacionadas con la iluminación ambiental, reportando una caída en la precisión al 95.4% y un aumento en la tasa de falsas alarmas al 6% durante las tardes debido a los reflejos solares intensos que afectaban la extracción de características faciales.

Driver Drowsiness Detection and Alert System Development Using Object Detection

(J. Da Wu & Chang, 2022) Presento un sistema de detección de somnolencia y alerta para conductores no intrusivo utilizando técnicas de detección de objetos en vehículos. El sistema emplea la plataforma embebida NVIDIA Jetson TX2 y compara cuatro variantes del algoritmo YOLO (v3, v3-tiny, v4 y v4-tiny) para monitorear el estado de los ojos del conductor, determinando la fatiga si la frecuencia de parpadeo supera las 13 veces por minuto o si la duración del cierre de párpados excede los 0.5 segundos. El resultado más relevante fue el rendimiento del modelo ligero YOLOv4, que alcanzó una Precisión Media (mAP) del 98.8% con un tiempo de detección extremadamente rápido de 0.005 segundos, validando su idoneidad para el procesamiento en tiempo real. Sin embargo, el estudio limita su lógica de decisión final exclusivamente a métricas oculares basadas en umbrales fijos, omitiendo la integración de otros indicadores comportamentales mencionados en la literatura, como el bostezo o la inclinación de la cabeza, para robustecer la alerta.

Drowsiness Detection of Construction Workers Accident Prevention Leveraging Yolov8 Deep Learning and Computer Vision Techniques

(Onososen et al., 2025) Presenta un enfoque basado en visión para la detección de somnolencia en tiempo real en trabajadores de la construcción utilizando una versión mejorada del algoritmo YOLOv8. El sistema emplea técnicas de visión por computadora para analizar características faciales y oculares, como patrones de bostezo y frecuencia de parpadeo, procesando un conjunto de datos personalizado de 605 imágenes clasificadas en estados de alerta y fatiga, con y sin equipo de protección personal (EPP). El resultado más relevante fue una Precisión Media (mAP) del 92% específicamente en la detección de la clase de somnolencia, junto con una

velocidad de inferencia de 7.5 ms, lo que demuestra su viabilidad para prevenir accidentes in situ. Sin embargo, el estudio señaló dificultades significativas al manejar clases desequilibradas, obteniendo un rendimiento inferior en la categoría de "Despierto con EPP", además de presentar limitaciones frente a condiciones de iluminación variables y oclusiones propias del entorno de obra.

Performance Optimization of FPGA-Accelerated YOLOv8 for Driver Drowsiness Detection Using Vivado HLS

(Essel et al., 2025) Desarrollo un estudio sobre la optimización del rendimiento del modelo YOLOv8 para la detección de somnolencia del conductor utilizando aceleración por hardware en FPGA. El sistema implementa la arquitectura YOLOv8n en una placa PYNQ-Z2 mediante Vivado HLS, aplicando técnicas avanzadas de síntesis de alto nivel como *pipelining*, desenrollado de bucles y cuantificación de punto fijo de 16 bits para procesar imágenes de fatiga y bostezos. El resultado más relevante fue la obtención de una precisión del 94.6% con un consumo de energía ultra eficiente de 1.83 W, logrando un equilibrio efectivo entre rendimiento y consumo para dispositivos de borde. Sin embargo, el estudio reconoció limitaciones impuestas por los recursos de hardware de la FPGA, donde la alta utilización de bloques DSP (90%) y LUTs (84%) requirió compromisos en la precisión numérica para ajustar el modelo al dispositivo.

1.3. Sistemas Embebidos

1.3.1. Concepto

(Barr, 2007) Define como un sistema integrado, conformado por una combinación de hardware y software computacional, posiblemente complementado con elementos mecánicos u otros, está diseñado para llevar a cabo una función dedicada. En ciertas instancias, estos sistemas pueden formar parte de un sistema o producto más extenso, como se observa, por ejemplo, en el caso de un sistema de frenos antibloqueo incorporado en un vehículo.

Por otra parte (Heath, 2003) define como un sistema informático, una combinación de procesador, memoria y dispositivos periféricos de entrada/salida, que tiene una función específica dentro de un sistema mecánico o electrónico más grande.

1.3.2 Raspberry Pi

Es un dispositivo pequeño y económico del tamaño de una billetera, diseñada para conectarse a un monitor y usar periféricos estándar como ratones o teclados. Esta pequeña computadora se utiliza como herramienta para introducir a personas de todas las edades en la programación utilizando lenguajes como Scratch y Python. Su potencia, respaldada por un sistema operativo liviano, le permite realizar una amplia gama de funciones propias de una computadora normal, como navegar por Internet, reproducir música o vídeos, procesar texto e incluso ejecutar algunos juegos. Al condensar todas estas funciones en un formato compacto, ha ganado relevancia en proyectos académicos que van desde la creatividad digital hasta estaciones de cámaras de vigilancia, decodificadores multimedia para reproducción y emulación de consolas de juegos (Baiza, 2020).

1.3.3 Jetson Nano

El dispositivo IoT edge de inferencia de visión por computadora NVIDIA Jetson Nano destaca como uno de los más empleados en su categoría. Ofrece un conjunto integral de herramientas y bibliotecas que facilitan a los usuarios la implementación de diversas técnicas avanzadas de aprendizaje profundo para abordar problemáticas complejas del mundo real. Este dispositivo está especialmente optimizado para el entrenamiento de modelos de aprendizaje profundo mediante el software TensorRT. Equipado con un procesador Arm Cortex A57 de cuatro núcleos, proporciona un rendimiento excepcional en la inferencia en tiempo real. Además, cuenta con dos puertos USB y puertos Gigabit Ethernet para establecer conexiones con dispositivos externos, como cámaras y pantallas. Su diseño contempla la posibilidad de funcionar con batería, convirtiéndolo en una opción idónea para despliegues en lugares remotos o situaciones de emergencia donde no se dispone de suministro eléctrico de red.(Siddique et al., 2023).

1.4. Modelo de Detección de Objetos YOLO

La serie YOLO de *Ultralytics* se ha consolidado como un referente en la segmentación de imágenes y la detección de objetos en tiempo real. Su evolución

más reciente, YOLO26, optimiza el rendimiento mediante una arquitectura de inferencia de extremo a extremo que prescinde de la Supresión de No Máximos (NMS), lo que resulta crucial para implementaciones en el borde. La versatilidad de este modelo abarca múltiples tareas de visión artificial, tales como clasificación, seguimiento (tracking), estimación de postura y segmentación. Gracias a su equilibrio entre velocidad y precisión, es una solución robusta tanto para dispositivos periféricos como para servicios basados en la nube (Jocher, 2026).

1.4.1. Versiones del modelo YOLO

Desde su creación en el año 2015 han existido varias versiones a lo largo del tiempo a continuación en la Tabla 1 se muestran las distintas versiones y el año de su salida:

Tabla 1.
Versiones YOLO

Versión de YOLO	Año de publicación	Autor / Framework principal
YOLO	2015	Joseph Redmon (Darknet)
YOLOv2	2016	Joseph Redmon (Darknet)
YOLOv3	2018	Joseph Redmon & Ali Farhadi
YOLOv4	2020	Alexey Bochkovskiy (Darknet)
YOLOv5	2021	Ultralytics (PyTorch)
YOLOv6	2022	Meituan (MMYOLO – PyTorch)
YOLOv7	2022	Wang et al. (Darknet/PyTorch híbrido)

YOLOv8	2023	Ultralytics (PyTorch)
YOLOv9	2024	Ultralytics (PyTorch)
YOLOv11	2024	Ultralytics (PyTorch)
YOLOv12	2025	Ultralytics (PyTorch)
YOLOv26	2025	Ultralytics (PyTorch)

Nota: Autoría propia

1.4.2. Introducción a YOLO (*You Only Look Once*)

Es un avanzado sistema de detección de objetos. Emplea únicamente una sola red neuronal convolucional para identificar objetos a partir de una imagen. Su funcionamiento implica que la red segmenta la imagen en varias zonas, anticipando las identificaciones de los fotogramas y las probabilidades asociadas a cada zona. Las líneas que delimitan los objetos son ajustadas según las probabilidades estimadas. Este algoritmo se entrena para aprender representaciones generalizables de objetos, lo que asegura una precisión significativa al detectar nuevas entradas que difieran del conjunto de datos de entrenamiento. (Soto Serrano, 2017).

En esencia, YOLO presenta una red neuronal que predice cajas delimitadas y posibilidades de clase directamente desde imágenes completas con solo una pasada. Este enfoque innovador contrasta marcadamente con los métodos tradicionales, que segmentan imágenes en partes individuales para su análisis. Al procesar integralmente toda la imagen simultáneamente, YOLO logra una velocidad y eficiencia excepcionales, convirtiéndose en la tecnología central para aplicaciones en tiempo real como la conducción autónoma y la vigilancia (Syed, 2023).

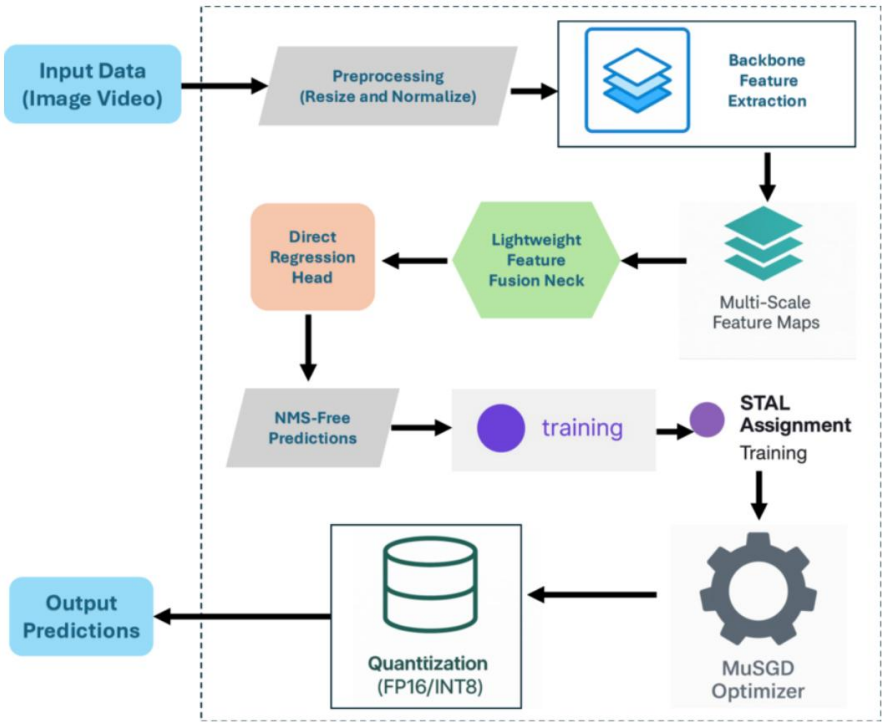
1.4.3. Arquitectura

Desde la perspectiva de sistemas, YOLO26 prioriza la portabilidad y la eficiencia mediante la eliminación de componentes complejos como la pérdida focal de distribución (DFL) y la supresión de no máximos (NMS). Esta simplificación permite que los modelos se exporten limpiamente a formatos como ONNX, TensorRT y TFLite, facilitando el despliegue en precisiones reducidas (INT8 y FP16) con una pérdida mínima de exactitud. Al reducir la complejidad de los operadores, se disminuye la latencia, especialmente en CPUs y GPUs integradas donde el post-procesamiento solía ser un cuello de botella. De este modo, el modelo consolida diversas tareas de visión como detección, segmentación y estimación de pose en un marco unificado que escala eficazmente desde servidores en la nube hasta sistemas embebidos con recursos limitados.

En cuanto a su arquitectura y entrenamiento, el modelo introduce simplificaciones clave para garantizar una inferencia nativa de extremo a extremo, eliminando los retrasos asociados al post-procesamiento. Para mejorar la estabilidad y la precisión, incorpora técnicas avanzadas como el *Progressive Loss Balancing* y una asignación de etiquetas optimizada para objetos pequeños (*Small-Target-Aware*), junto con el optimizador MuSGD que acelera la convergencia. Los resultados preliminares indican un aumento de velocidad de aproximadamente el 43% en ejecución sobre CPU y un rendimiento robusto bajo cuantización, lo que valida a YOLO26 como una solución idónea para implementaciones en hardware de borde, como placas de desarrollo y procesadores estándar.

La Figura 3, muestra una visualización detallada de la arquitectura de la red.

Figura 3.
Arquitectura YOLO26



Nota. (Sapkota & Karkee, 2025)

1.4.4. Ventajas y desventajas de YOLO

Ventajas

La principal distinción entre YOLO y otros sistemas de detección de objetos reside en su enfoque de "una sola pasada", como lo indica su nombre: solo examina una imagen en una única iteración. Mientras que otros métodos utilizan un proceso de dos etapas para primero localizar y luego identificar objetos, YOLO realiza ambas tareas de manera simultánea. Gracias a este enfoque único, YOLO es excepcionalmente veloz, con la capacidad de procesar hasta 45 cuadros por segundo, dependiendo del hardware. Esto implica que los videos grabados a esa velocidad o menos pueden ser procesados en tiempo real. Aunque existe una versión de YOLO capaz de manejar hasta 155 cuadros por segundo, esta mayor velocidad se logra a expensas de la precisión del modelo (Scanbot SDK, 2022).

Desventajas

El modelo YOLO presenta limitaciones que los usuarios deben tener en cuenta. Se ha observado que puede enfrentar dificultades al detectar objetos en escenas con mucha información visual o cuando los objetos están parcialmente ocultos. Además,

podría experimentar complicaciones al identificar objetos de dimensiones reducidas o aquellos con bajos niveles de contraste (Jocher et al., 2023).

1.4.5. Recomendaciones para su aplicación

El modelo de detección de objetos YOLO ha sido cuidadosamente desarrollado para identificar objetos en imágenes y videos en tiempo real, lo que lo convierte en una herramienta versátil y eficiente. Su aplicación abarca diversos campos, incluyendo vigilancia, vehículos autónomos y robótica, donde la capacidad de detección en tiempo real es esencial. Este modelo está dirigido específicamente a desarrolladores e investigadores con un sólido conocimiento en visión por computadora y aprendizaje profundo, garantizando un uso efectivo en entornos especializados que requieren precisión y eficacia en la detección de objetos (Jocher et al., 2023).

1.5. ISO/IEC 25023 y Metodología KDD

1.5.1. Descripción de la Norma ISO/IEC 25023.

El presente estándar proporciona los lineamientos necesarios para realizar una evaluación numérica de la calidad del software, fundamentándose en la taxonomía de características y subcaracterísticas definidas por la ISO/IEC 25010. Se complementa con esta norma y no determina rangos de valores para categorías de calidad o rangos de conformidad. Aunque ciertos atributos pueden tener medidas deseables, estos no se basan en requisitos individuales de los usuarios, sino en factores generales. como los relacionados con los aspectos cognitivos humanos. Las medidas de calidad propuestas se diseñan principalmente para su utilización en el aseguramiento de la calidad y la mejora de los productos de sistemas y software durante o después del ciclo de vida del desarrollo (Internacional Organization for Standardization, 2016).

1.5.2. Principio de Portabilidad en el Desarrollo de Sistemas de Detección.

Según (Ormeño, 2019), portabilidad se refiere a la habilidad de un producto o componente para ser transferido de manera efectiva y eficiente entre diferentes entornos, ya sean de hardware, software, operativos o de uso. Dentro de esta característica, se desglosa en varias subcaracterísticas que contribuyen a su definición:

- **Adaptabilidad**: Esta subcaracterística destaca la capacidad del producto para ser ajustado de manera eficaz y eficiente a distintos entornos específicos, ya sea en términos de hardware, software, operaciones o uso.
- **Capacidad para ser instalado**: Se refiere a la capacidad de instalar y/o desinstalar un dispositivo de manera efectiva en un ambiente específico, destacando la simplicidad del procedimiento.
- **Capacidad de sustitución**: Esta característica examina la habilidad del producto para ser empleado en lugar de otro software con igual objetivo, dentro del mismo contexto. Se centra en la evaluación de cómo el producto puede ser reemplazado de manera efectiva y eficiente

1.5.3. Descubrimiento de conocimiento en bases de datos (KDD)

Se refiere al procedimiento integral de identificar información valiosa a partir de datos. La minería de datos se presenta como una etapa específica dentro de este proceso, donde se aplican algoritmos especializados para extraer patrones (modelos) de los datos. Las otras fases del proceso KDD, como la preparación, selección y limpieza de los datos, así como la integración de conocimientos previos adecuados y la interpretación de los resultados de la minería, son necesarias para garantizar la obtención de conocimientos útiles a partir de los datos (Fayyad et al., 1996)

La Exploración de Conocimiento en Bases de Datos (KDD, Knowledge Discovery in Databases) consiste en un análisis automatizado y exploratorio de extensos repositorios de datos. KDD representa un proceso organizado destinado a identificar patrones válidos, novedosos y útiles en conjuntos de datos complejos y extensos. La Minería de Datos (DM, Data Mining) constituye la esencia de KDD, implicando la inferencia de algoritmos para explorar datos, desarrollar modelos y descubrir patrones. Estos modelos son empleados para entender fenómenos de datos, realizar análisis y predicciones. Dada la accesibilidad y abundancia de datos en la actualidad, el Descubrimiento de Conocimiento y la Minería de Datos se presentan como cuestiones de gran importancia y necesidad (Uvidia, 2016).

1.5.4. Etapas del proceso KDD

De acuerdo con (Timarán et al., 2016) el proceso KDD es interactivo e iterativo, que abarca múltiples pasos en los cuales la participación del usuario es esencial para la toma de diversas decisiones. Se resume en las siguientes etapas:

- **Selección**

Durante la fase de selección, después de identificar el conocimiento pertinente y prioritario, se crea un grupo de datos específico. Este grupo puede consistir en todos los datos disponibles o una muestra de ellos. La selección de los datos se adapta a los objetivos comerciales específicos.

- **Limpieza**

En la etapa de preprocesamiento limpieza de datos, se lleva a cabo una evaluación exhaustiva de la calidad de la información, que implica acciones básicas como eliminar información ruidosa y aplicar métodos para manejar datos desconocidos, nulos y duplicados. Se utiliza estadística para sustituir los datos faltantes. Durante este proceso, la interacción con el usuario o analista es crucial.

- **Transformación**

Se exploran atributos significativos para describir los datos, ajustándose a los objetivos particulares del proceso. Se utilizan técnicas de reducción de dimensiones o transformación para reducir el número de variables o encontrar representaciones estables de los datos.

- **Minería de datos (Data Mining)**

Se exploran atributos significativos para describir los datos, ajustándose a los objetivos particulares del proceso. Se utilizan técnicas de reducción de dimensiones o transformación para reducir el número de variables o encontrar representaciones estables de los datos.

- **Interpretación/evaluación**

Durante la fase de interpretación y evaluación, los patrones descubiertos son analizados y podrían ser devueltos a etapas previas para ajustes adicionales. Aquí, se presentan los patrones identificados, se eliminan aquellos que no son pertinentes, y se traducen los patrones relevantes a un lenguaje comprensible para el usuario.

CAPÍTULO 2: Desarrollo

2.1. Construcción del *dataset*

La calidad del conjunto de datos es un factor determinante en el rendimiento de los modelos de visión por computadora basados en aprendizaje profundo. En este proyecto, se construirá un *dataset* específico para la detección de somnolencia, con enfoque en la región ocular del rostro. Este conjunto de datos combinará imágenes extraídas de bases públicas con material recolectado por los tesistas, permitiendo así una mayor representatividad de escenarios reales de conducción.

2.1.1. Selección y análisis de datasets existentes

Como punto de partida, se recopilarán imágenes faciales provenientes de bases de datos públicas proporcionadas por el tutor como se muestra en la Tabla 2. Estas bases se caracterizan por contener anotaciones precisas de puntos faciales (landmarks), variedad de expresiones, posiciones y condiciones de iluminación, lo que permitirá extraer con precisión la región de los ojos.

En lugar de trabajar con coordenadas específicas de landmarks, se optará por el uso de cajas delimitadoras (bounding boxes) que encuadran la región de interés, en este caso, los ojos. Estas regiones serán clasificadas de forma binaria, asignando la etiqueta 0 a ojos abiertos y 1 a ojos cerrados, lo cual permitirá entrenar el modelo para detectar patrones de somnolencia con mayor eficiencia y menor complejidad computacional.

Este enfoque simplifica el proceso de anotación y resulta más adecuado para modelos de detección en tiempo real, ya que se centra en el reconocimiento de patrones espaciales generales en lugar de la localización exacta de cada punto facial.

Se seleccionarán únicamente aquellas imágenes que cumplan con criterios técnicos mínimos, como buena resolución, enfoque nítido y visibilidad de los ojos. Asimismo, se excluirán imágenes con obstrucciones significativas (por ejemplo, cabellos cubriendo el rostro) o etiquetado deficiente. Esta fase permitirá contar con una base sólida de datos relevantes y listos para su adaptación al modelo YOLOv12.

Tabla 2.
Datasets seleccionados.

Dataset	Cantidad de imágenes	Características
300VW	2442	Imágenes en condiciones diurnas y nocturnas capturadas desde diversos ángulos y con distintas condiciones de iluminación
300W	1760	Rostros de revistas y celebridades
afw	570	Imágenes con fondos variados y múltiples personas, etiquetadas tanto en estados de sueño como de alerta.
CEW	560	Rostros de adultos y niños con ojos cerrados.
COFW	432	Imágenes de celebridades en diferentes resoluciones.
frgc	4950	Rostros de adultos.
Helen	8320	Imágenes de adultos y niños; algunas contienen múltiples rostros, pero solo uno está etiquetado.
ibug	568	Rostros de celebridades; incluye aumento de datos con la colocación de gafas.
lfpw	3434	Imágenes de adultos; incluye aumento de datos mediante la colocación de gafas.
Menpo2D	3979	Imágenes de adultos y niños, capturadas desde diferentes ángulos.
ownNIR	1876	Imágenes de personas conduciendo en condiciones diurnas y nocturnas, con ojos abiertos y cerrados.
Xm2vts	2360	Rostros de personas adultas con ojos abiertos.
Total	30689	

Nota. Obtenido de (Meza, 2025)

Si bien los conjuntos de datos utilizados ofrecen una amplia diversidad de rostros y condiciones visuales, se ha identificado una predominancia de imágenes

correspondientes a ojos abiertos. Esta desproporción podría introducir un sesgo en el proceso de entrenamiento, favoreciendo la detección de un solo estado ocular y limitando la capacidad del modelo para reconocer con precisión casos de somnolencia.

Para mitigar este problema, se considerará la incorporación de imágenes adicionales que representen el estado de ojos cerrados, con el objetivo de equilibrar la distribución entre ambas clases. De esta manera, se busca garantizar que el modelo aprenda de forma equitativa los patrones asociados a ambos estados y mejore su desempeño en condiciones reales.

2.1.2. Recolección de imágenes propias

Con el objetivo de mejorar la diversidad y representatividad del conjunto de datos utilizado en el entrenamiento del modelo, se generará un *dataset* propio, específicamente diseñado para escenarios reales de detección de somnolencia en conductores. Mientras que los *datasets* públicos suelen estar compuestos por imágenes capturadas en contextos controlados o con participantes simulando expresiones faciales, el conjunto de datos propuesto buscará reflejar situaciones más cercanas al entorno natural de conducción.

Para ello, se realizarán capturas de video a bordo de vehículos en movimiento, bajo distintas condiciones de iluminación, ángulos de visión y presencia de elementos como gafas o lentes. Los fotogramas se extraerán directamente de estos videos, permitiendo obtener muestras visualmente claras y contextualmente diversas.

El proceso de recolección se llevará a cabo utilizando una Raspberry Pi conectada a una cámara compatible con visión nocturna, lo que facilitará la obtención de imágenes tanto en horarios diurnos como en condiciones de baja iluminación.

2.1.3. Preprocesamiento

Extracción de frames

Para extraer frames de los videos tomados, usamos un script de Python, ejecutado en VS Code, en la Figura 4 se muestra el código usado para realizar este proceso.

Figura 4.
Código extracción de frames

extraer_frames.py > ...

```python
import cv2
import os

# === CONFIGURACIÓN ===
video_path = 'alexis_dia.mp4'
output_folder = 'frames_extraidos'
frame_interval = 10  # Guarda 1 imagen cada 10 frames

# === CREAR CARPETA DE SALIDA SI NO EXISTE ===
os.makedirs(output_folder, exist_ok=True)

# === CARGAR VIDEO ===
cap = cv2.VideoCapture(video_path)

if not cap.isOpened():
    print(f"No se pudo abrir el video: {video_path}")
    exit()

frame_count = 0
saved_count = 0

# === LECTURA DE FRAMES ===
while True:
    ret, frame = cap.read()
    if not ret:
        break  # Fin del video

    if frame_count % frame_interval == 0:
        frame_filename = os.path.join(output_folder, f"frame_{frame_count:04d}.jpg")
        cv2.imwrite(frame_filename, frame)
        saved_count += 1

    frame_count += 1

cap.release()
print(f"Extracción completada. Total de frames guardados: {saved_count}")

```

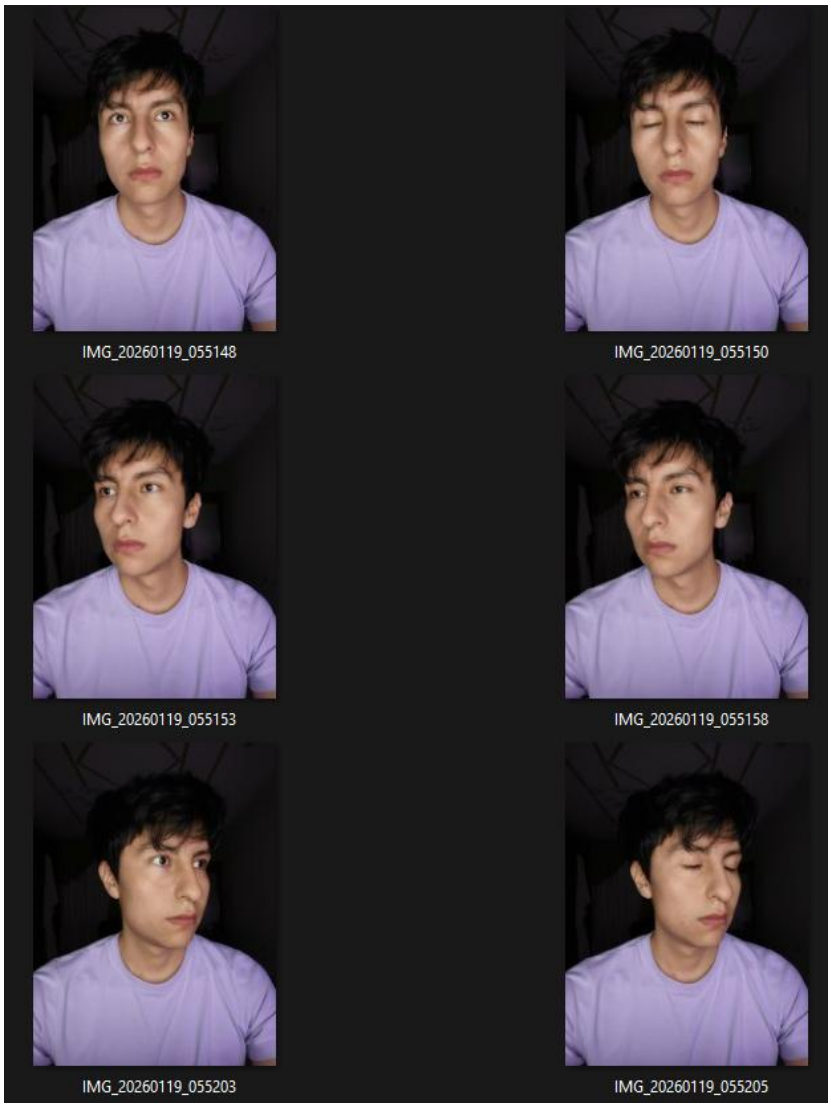
Nota: Autoría *propia*

Selección de frames

Para esta etapa, se realizó una selección manual de los fotogramas de cada video, buscando siempre la mayor claridad visual y diversidad en las imágenes. El criterio principal fue incluir diferentes situaciones que un conductor enfrenta en la vida real, como distintos gestos faciales, la inclinación de la cabeza y el uso de accesorios como lentes de medida o gafas de sol.

Este orden en la selección permite que el modelo YOLO aprenda a identificar con mayor precisión los síntomas de cansancio, sin importar las condiciones externas o los accesorios que use el conductor.

Figura 5.
Frames seleccionados de un video.



Nota. Autoría propia

Un aspecto clave en este proceso fue asegurar que el conjunto de datos estuviera balanceado. Por ello, se recolectó una cantidad equitativa de imágenes con los ojos abiertos y cerrados, tanto para los escenarios de día como para los de noche. Como se detalla en la Tabla 3.

Tabla 3.
Criterio de selección de frames

Aspecto	Rostro	Lentes	Gafas	Frames
Dia – Ojos cerrados	2	2	2	6
Noche – Ojos abiertos	2	2	2	6
Noche – Ojos cerrados	2	2	2	6
Total				24

Nota. Autoría propia

2.1.4. Etiquetado

En esta fase, el conjunto de imágenes fue preparado y ajustado específicamente para el entrenamiento del modelo. Este proceso no solo implicó adaptar el formato de las etiquetas, sino también definir cuál era la estrategia de detección más efectiva para identificar la somnolencia.

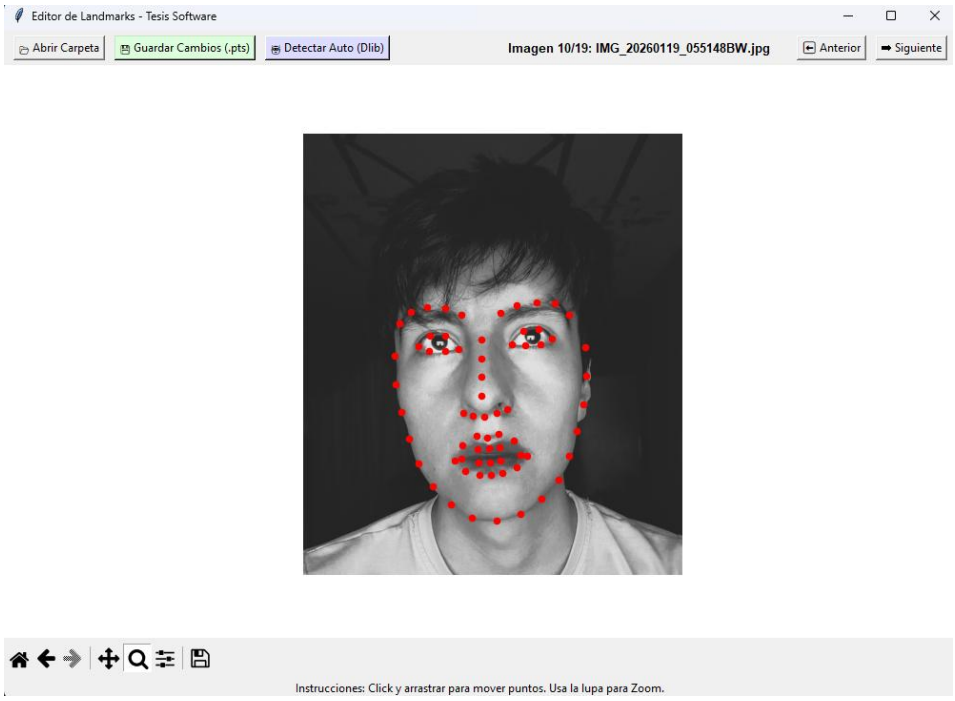
Durante el desarrollo, probamos distintos métodos. En un principio, intentamos usar un modelo de regresión para predecir directamente los 12 puntos de referencia (landmarks) de los ojos. Luego, evaluamos un enfoque de clasificación simple para catalogar las imágenes únicamente como ojos abiertos o cerrados. Sin embargo, tras analizar los resultados, decidimos utilizar un esquema de detección de objetos YOLO12-Detección. En este último método, aprovechamos los puntos de referencia para generar cajas delimitadoras (bounding boxes) que permiten localizar con precisión los ojos en cada imagen. Este ajuste final permitió que el modelo se centrara mejor en el área de interés.

Para asegurar la precisión de las etiquetas que alimentan la red neuronal, se desarrolló una herramienta de validación y corrección manual. Este software, construido sobre las bibliotecas Tkinter y Matplotlib, permite visualizar las coordenadas generadas automáticamente y modificarlas en tiempo real.

La herramienta opera bajo un flujo híbrido: primero, ejecuta el algoritmo de predicción de formas de Dlib para situar los puntos preliminares; posteriormente, permite al usuario interactuar directamente con la visualización para corregir desviaciones mediante el arrastre de puntos (drag & drop). Adicionalmente, el sistema

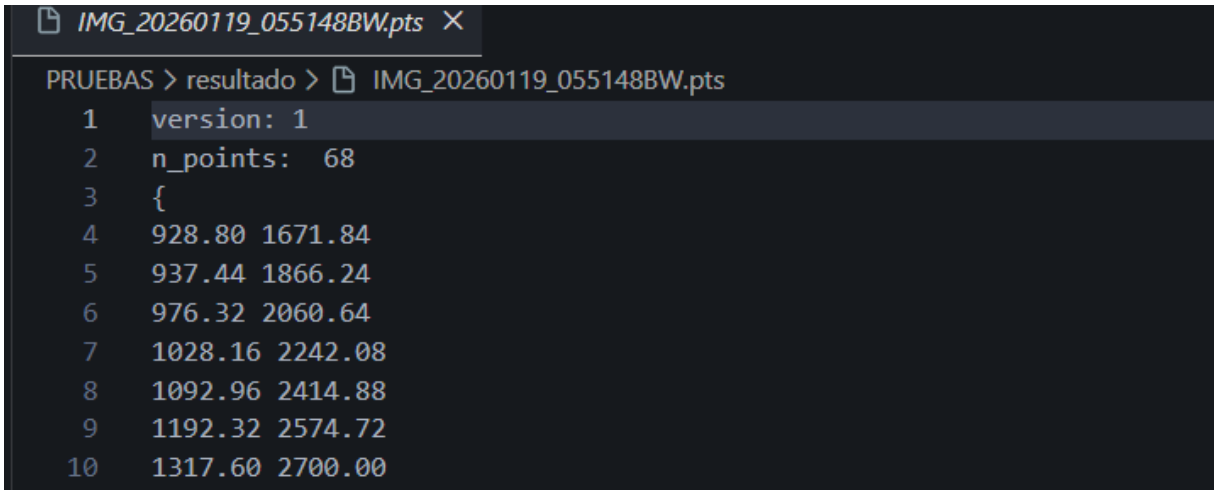
gestiona la creación y actualización automática de los archivos de anotación (.pts), estandarizando el formato de salida para el entrenamiento. En la Figura 6 se observa el entorno de trabajo.

Figura 6.
Identificación Landmark faciales.



Nota. Elaboración propia.

Figura 7.
Archivo coordenadas generado (.pts)



Nota. Elaboración propia

Proceso de Etiquetado y Refinamiento

El procedimiento para la generación del *ground truth* se estandarizó bajo el siguiente flujo de trabajo, apoyado en la herramienta de software desarrollada:

- Carga y Preprocesamiento**:** Importación masiva de las imágenes al entorno de anotación, aplicando previamente la corrección de formato y redimensionado automático.
- Corrección Manual: Ajuste de las coordenadas mediante la interfaz de edición (*drag-and-drop*) en aquellos casos donde la oclusión (gafas/lentes) o la iluminación afectaron la predicción automática Figura 6.
- Persistencia de Datos: Almacenamiento definitivo de las coordenadas corregidas en archivos de formato ".pts", asegurando la integridad de la muestra.

Esta metodología se replicó en todo el conjunto de datos, garantizando una base de entrenamiento homogénea y libre de ruido.

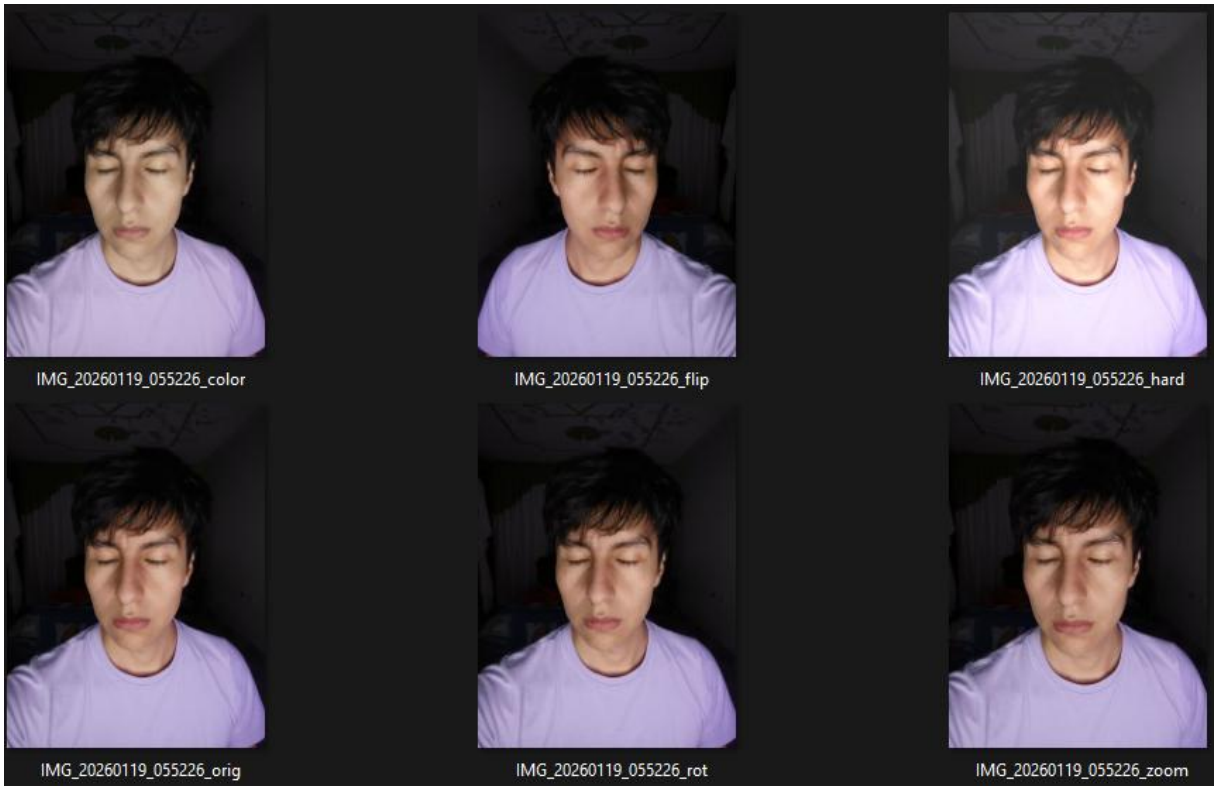
Cabe señalar que, aunque se obtuvieron los *landmarks* con precisión, la implementación final desestimó el uso de un modelo de regresión pura para predecir coordenadas. Durante la fase experimental, se determinó que para la arquitectura seleccionada (YOLO), resultaba más eficiente un enfoque de detección de objetos. Por consiguiente, la información espacial de los *landmarks* se utilizó estratégicamente para calcular y generar las cajas delimitadoras (*bounding boxes*) que encuadran los ojos, transformando así el problema en una tarea de detección robusta apta para la detección en tiempo real.

2.1.5. Aumento de datos

El aumento de datos (*Data Augmentation*) es una técnica fundamental en el preprocesamiento de imágenes, utilizada para ampliar artificialmente el tamaño y la diversidad del conjunto de entrenamiento. Su objetivo principal es mejorar la capacidad de generalización del modelo, evitando que este simplemente 'memorice' las imágenes originales (sobreajuste). Al aplicar transformaciones controladas, se obliga al sistema a aprender características más robustas y adaptables a diferentes condiciones visuales.

Debido a la limitación en la cantidad de imágenes originales en condiciones de poca luz y en especial con la presencia de gafas o lentes se recurrió a esta técnica en donde se aplicaron filtros de color, ajustes espaciales, variaciones que simulan diferentes entornos y condiciones de captura, Figura 8, asegurando que el algoritmo reconozca los patrones de interés independientemente de las variaciones estéticas de la imagen.

Figura 8.
Data Augmentation



Nota. Autoría propia

2.2. Transformación del *dataset*

El conjunto de datos utilizado para el entrenamiento del modelo fue organizado siguiendo la estructura estándar requerida por arquitecturas de detección basadas en YOLO. El dataset se dividió en tres subconjuntos independientes: entrenamiento, validación y prueba, cada uno con sus respectivas carpetas de imágenes y anotaciones. Las imágenes se almacenaron en formato JPG y para cada imagen, se dispuso de un archivo de anotación en formato de texto (.txt) que contenía las coordenadas de los objetos de interés. Cada archivo de anotación seguía el esquema YOLO, donde cada línea representaba una instancia detectada y estaba compuesta por cinco valores: el identificador de clase (0 para ojo abierto y 1 para ojo cerrado)

seguido de las coordenadas normalizadas del centro del bounding box (x, y) y su ancho y alto, todos expresados en relación con las dimensiones de la imagen. Esta normalización permitió que el modelo fuese robusto ante imágenes de diferentes resoluciones. Previo al entrenamiento, se verificó la consistencia entre imágenes y anotaciones, asegurando que cada archivo de imagen tuviera su correspondiente archivo de etiquetas y que las clases estuvieran correctamente alineadas con la definición utilizada durante el entrenamiento del modelo.

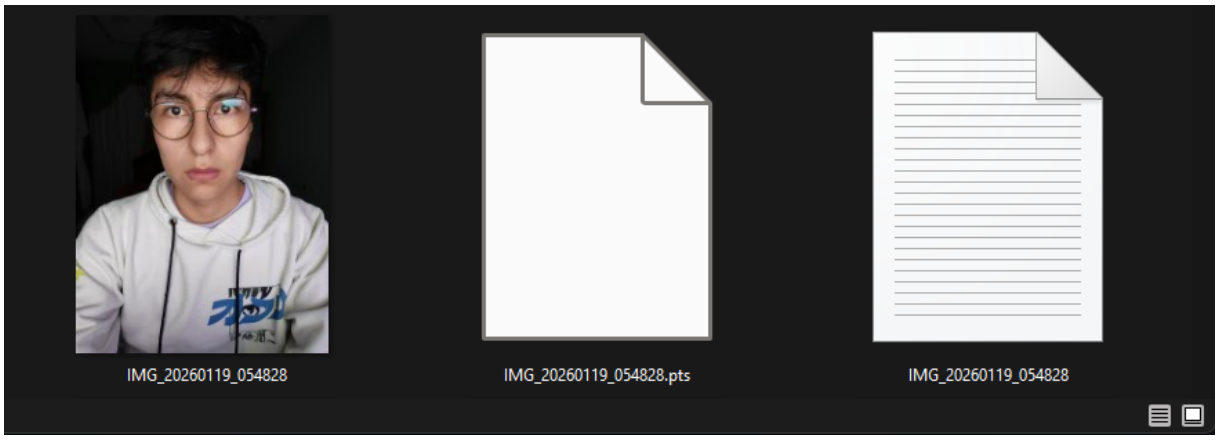
2.2.1. Etiquetado esquema YOLO

Para la correcta implementación del modelo, las anotaciones se exportaron siguiendo el formato estándar de YOLO, el cual requiere un archivo de texto (.txt) exclusivo para cada imagen del conjunto de datos. Aquellas imágenes sin detecciones no requieren ningún archivo asociado.

La estructura interna de estos archivos se organiza en una fila por cada objeto, definiendo secuencialmente la clase, las coordenadas del centro (x, y), el ancho y la altura. Un requisito técnico indispensable es que estos valores dimensionales se encuentren normalizados entre 0 y 1, calculados en relación con el ancho y alto total de la imagen original. Asimismo, las clases deben seguir una numeración de índice base cero (Jocher & Qiu, 2023).

Para nuestro caso usamos los archivos (*.pts) para transfórmalos en (*.txt) como de observa en la Figura 9.

Figura 9.
Archivos txt generados



Nota. Autoría propia

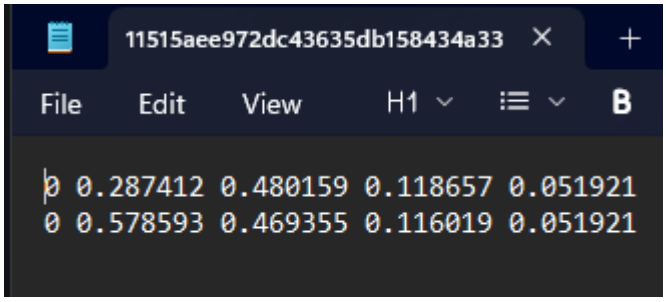
Cada archivo de texto generado registra una línea por objeto detectado, estructurada en cinco valores secuenciales: el índice de la clase, seguido de las coordenadas normalizadas del centro (x, y) y las dimensiones relativas del cuadro (ancho, alto). Todos los valores espaciales se encuentran escalados entre 0 y 1 (Figura 10), lo que permite al modelo interpretar la ubicación de los ojos independientemente de la resolución original de la imagen.

$$clase, x(centro), y(centro), ancho, alto$$

Donde

- Clase: Define si el ojo está abierto (0) o cerrado (1).
- X(centro), Y(centro): Coordenadas normalizadas de la caja
- Ancho, Alto: Dimensiones normalizadas de la caja

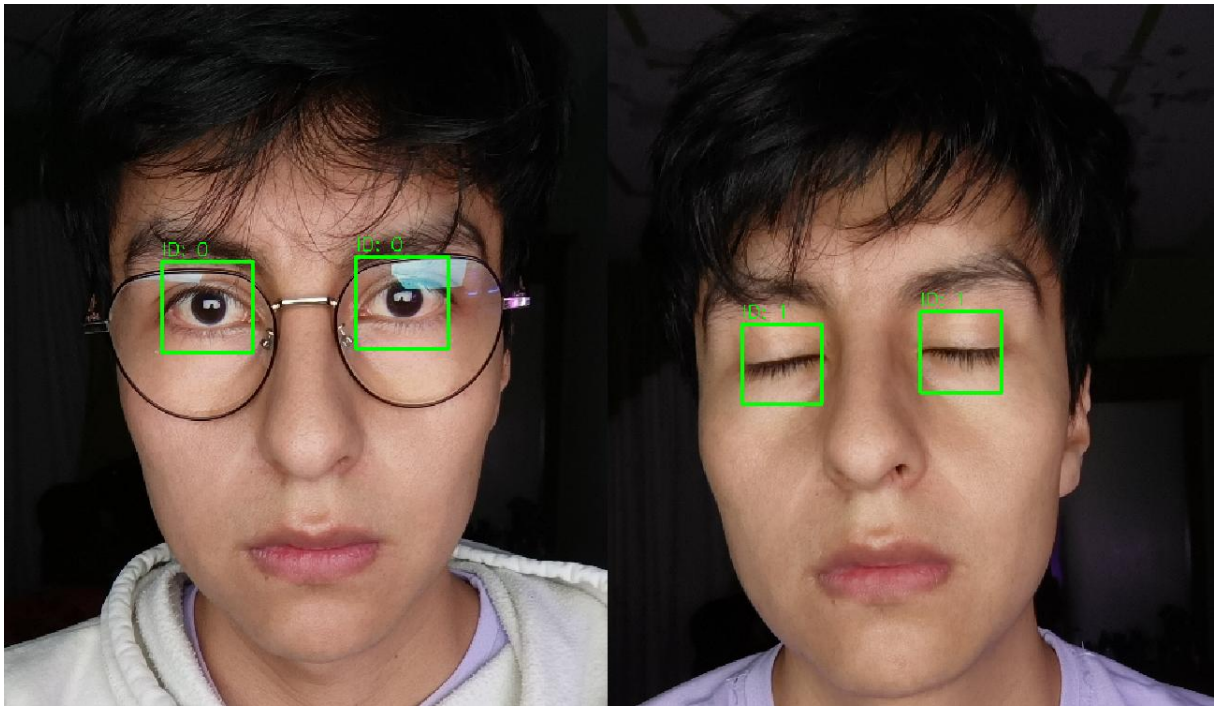
Figura 10.
Contenido archivo txt



Nota. Elaboración propia

Las cajas delimitadoras (*bounding boxes*) se definieron como regiones rectangulares calculadas para encapsular el área de interés visual. Su geometría se obtuvo proyectando los límites extremos de los *landmarks* oculares, incorporando un margen adicional (*padding*) para asegurar que la totalidad del ojo y su contexto inmediato queden contenidos dentro de la etiqueta.

Figura 11.
Cajas delimitadoras



Nota. Autoría propia.

2.2.2. Balanceo de *dataset*

Para garantizar un entrenamiento imparcial y evitar sesgos en la predicción del modelo, se optó por la construcción de un único conjunto de datos balanceado. La estrategia se centró en unificar el dataset propio (detallado en la sección anterior) con diversos datasets públicos, creando así un conjunto de datos robusto y heterogéneo.

A diferencia de enfoques desbalanceados que pueden favorecer a la clase mayoritaria, en este trabajo se igualó la cantidad de muestras para las clases de interés (ojos abiertos, ojos cerrados), asegurando que el modelo aprenda a discriminar características con la misma precisión para ambas categorías.

Se integro imágenes provenientes de repositorios conocidos como 300W, afw, CEW, Helen, COFW, los cuales aportaron variabilidad en cuanto a etnias, condiciones de iluminación y calidad de imagen.

A continuación, se detalla la composición final del Dataset:

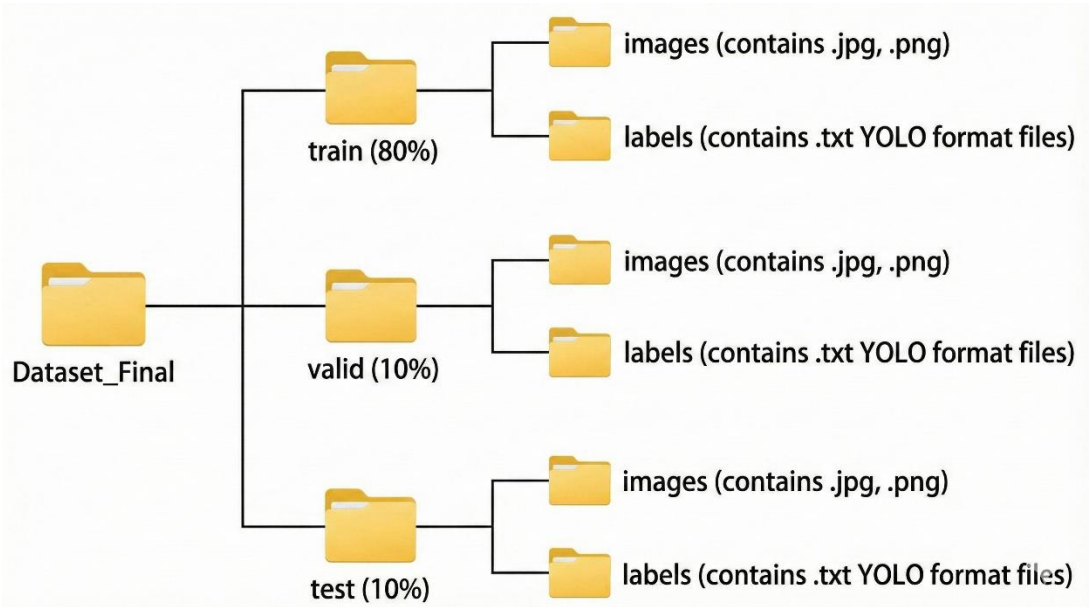
Tabla 4.
Composición Dataset

Conjunto	Ojos Abiertos (0)	Ojos Cerrados (1)	Total Imágenes
Train (80%)	4703 (50.01%)	4702 (49.99%)	9407 (100%)
Valid (10%)	595 (50.99%)	592 (49.01%)	1187 (100%)
Test (10%)	588 (50%)	588 (50%)	1176 (100%)
Total Global	5884	5883	11770 (100%)

Nota. Autoría propia.

Para garantizar la compatibilidad y facilitar la reproducibilidad de los experimentos, se organizó el conjunto de datos siguiendo la estructura de directorios estándar para arquitecturas basadas en YOLO. Como se observa en la Figura 12.

Figura 12.
Estructura Directorio Dataset



Nota. Autoría propia.

Cada subconjunto (entrenamiento, validación y prueba) contiene dos subdirectorios: ”*images*” para las muestras visuales y ”*labels*” para los archivos de anotación correspondientes.

2.3. Entrenamiento del modelo YOLO

2.3.1. Entorno de desarrollo y Hardware

El proceso de entrenamiento, validación y prueba de los modelos de detección se llevó a cabo en una estación de trabajo de alto rendimiento (PC de escritorio). La elección de un hardware robusto fue fundamental para optimizar los tiempos de cómputo, permitiendo realizar múltiples iteraciones de ajuste fino (*fine-tuning*) y probar distintas configuraciones de hiperparámetros de manera eficiente.

Especificaciones de Hardware

Se utilizó un equipo con capacidad de procesamiento paralelo mediante GPU, característica indispensable para el entrenamiento de redes neuronales convolucionales modernas. Las especificaciones técnicas se detallan a continuación:

Tabla 5.
Especificaciones de Hardware

Componentes	Especificaciones
Procesador (CPU)	Intel Core i7-13700KF
Tarjeta Gráfica (GPU)	NVIDIA GeForce RTX 4070Ti 12GB VRAM
Memoria RAM	32 GB DDR5
Almacenamiento	1 TB SSD

Nota. Autoría propia.

Software y Herramientas.

El desarrollo de los algoritmos y scripts de automatización se realizó sobre el sistema operativo Microsoft Windows 11. Como entorno de desarrollo integrado (IDE) se utilizó Visual Studio Code (VS Code), aprovechando su integración con terminales y gestión de entornos virtuales.

La base del proyecto se construyó sobre el lenguaje de programación Python en su versión 3.11.9, seleccionado por su amplia compatibilidad con las librerías de inteligencia artificial más recientes. La implementación de la arquitectura YOLO se gestionó a través del *framework Ultralytics*, haciendo uso de *PyTorch* como motor de cálculo tensorial.

En la Tabla 6 se resumen las versiones específicas de las herramientas utilizadas para garantizar la reproducibilidad de los experimentos.

Tabla 6.
Configuración del entorno de software

Herramienta/Librería	Versión	Propósito
Python	3.11.9	Lenguaje de programación principal
Ultralytics	8.4.7	Framework
PyTorch	2.5.1+cu121	Motor de Deep *learning* y *tensors*
CUDA	12.1	Plataforma de computación paralela en GPU
OpenCV(cv2)	4	Preprocesamiento de imágenes y visualización
VSCode	1.108	Edición de código y depuración.

Nota. Autoría propia

Selección del Modelo y Restricciones de Implementación

Aunque el entrenamiento se realizó en un equipo de altas prestaciones, el objetivo final del sistema es su despliegue en un dispositivo embebido de recursos limitados, específicamente una Raspberry Pi 3 tipo B+. Esta restricción de hardware en la etapa de inferencia condicionó la selección de la arquitectura del modelo.

Por esta razón, se optó por experimentar con las variantes Nano (n) y Small (s) de la arquitectura YOLO. Estos modelos ofrecen un equilibrio óptimo entre precisión (mAP) y costo computacional (FLOPs), permitiendo una inferencia fluida en tiempo real en dispositivos de borde, sin sacrificar significativamente la capacidad de detección de estados de somnolencia.

2.3.2. Selección de arquitectura y modelo

Para la tarea de detección de somnolencia, se seleccionó la familia de arquitecturas YOLO (You Only Look Once), reconocida en el estado del arte por su capacidad de realizar inferencias en tiempo real con alta precisión. Específicamente, en esta investigación se evaluaron las versiones YOLOv12 y la reciente YOLOv26, con el objetivo de comparar su desempeño en la detección de características faciales sutiles como el estado de los ojos.

Restricciones de Hardware y Selección de Variantes

Un requerimiento no funcional crítico del sistema propuesto es su portabilidad y capacidad de ejecución en dispositivos embebidos, en este caso en una Raspberry Pi. Estos dispositivos embebidos poseen recursos limitados en comparación con un servidor o PC de escritorio, particularmente en memoria RAM y núcleos de procesamiento gráfico.

Debido a esta restricción tecnológica, se descartaron las variantes de gran tamaño (Large - L, Extra Large - X) que, aunque ofrecen una precisión marginalmente superior, comprometerían la fluidez del sistema (FPS) en el dispositivo destino. En su lugar, el entrenamiento se centró en las variantes ligeras:

- **Nano (n):** La versión más ligera, optimizada para velocidad extrema y dispositivos móviles con recursos mínimos.
- **Small (s):** Una versión equilibrada que aumenta la profundidad y el ancho de la red neuronal para mejorar la extracción de características, manteniendo un costo computacional viable para sistemas embebidos.

Tras las pruebas preliminares, se optó por priorizar la variante Small (s) para el entrenamiento final, dado que ofreció un mejor compromiso entre la capacidad de detectar ojos cerrados en condiciones de iluminación variable y la velocidad de inferencia requerida para alertar al conductor en tiempo real.

Tabla 7
Métricas de rendimiento YOLO26.

YOLO26s (Small)	
Tamaño (pixeles)	640
$mAP^{val50-95}$	48.6
$mAP^{val50-95(e2e)}$	47.8
Velocidad $^{CPU\ ONNX(ms)}$	87.2 ± 0.9
$Velocidad^{T4\ TensorRT10(ms)}$	2.5 ± 0.0
Parámetros (Millones)	9.5
$FLOPs^{(B)}$	20.7

Nota. Autoría propia.

2.3.3. Configuración de hiperparámetros

El rendimiento de una red neuronal convolucional depende intrínsecamente de la correcta selección de sus hiperparámetros, los cuales gobiernan la dinámica del proceso de aprendizaje. A diferencia de los parámetros internos (pesos y sesgos) que se ajustan automáticamente mediante *backpropagation*, los hiperparámetros deben ser definidos *a priori*, basándose en las características del dataset y las capacidades del hardware.

Para el presente proyecto, se optó por una estrategia de ajuste fino (*fine-tuning*) partiendo de pesos pre-entrenados en el dataset COCO. Esta técnica permite aprovechar la capacidad de extracción de características ya aprendida por el modelo base (bordes, texturas, formas geométricas), adaptándola específicamente a la tarea de detección de la morfología ocular (ojos abiertos y cerrados).

La configuración final utilizada para el entrenamiento definitivo del modelo se detalla en la Figura 13. Estos valores se establecieron tras una fase exploratoria experimental, donde se buscó maximizar la estabilidad de la función de pérdida (loss function) y la precisión media (mAP).

Figura 13.
Configuración del entrenamiento.

runs > detect > somnolencia_yolo26s_static_boxes > args.yaml

```yaml
task: detect
mode: train
model: yolo26s.pt
data: data.yaml
epochs: 100
time: null
patience: 25
batch: 16
imgsz: 640
save: true
save_period: -1
cache: false
device: '0'
workers: 8
project: null
name: somnolencia_yolo26s_static_boxes
exist_ok: false
pretrained: true
optimizer: AdamW
verbose: true
```

Nota. Autoría propia.

Modelo

Se cargaron los pesos pre-entrenados de la variante *Small*. Esta elección garantiza la transferencia de aprendizaje (*Transfer Learning*) necesaria para extraer características complejas sin incurrir en el costo computacional de modelos más grandes (M, L o X), siendo la opción más equilibrada para sistemas embebidos.

Resolución de Entrada

Se definió un tamaño de imagen de 640x640 píxeles. Aunque la arquitectura soporta resoluciones mayores, se optó por reducir la dimensión de entrada para disminuir drásticamente las Operaciones de Punto Flotante (FLOPs) requeridas. Esta optimización es crítica para maximizar la tasa de cuadros por segundo (FPS) en la

Raspberry Pi, asegurando una respuesta en tiempo real, aunque se sacrifique una fracción mínima de detalle espacial.

Tamaño de Lote

Se configuró un *batch size* de 16 imágenes. Este valor permite actualizaciones de pesos más frecuentes durante el descenso de gradiente, lo que en conjuntos de datos específicos puede ayudar a escapar de mínimos locales y ofrece una mayor regularización implícita en comparación con lotes masivos.

Estrategia de Optimización

Se seleccionó el optimizador AdamW con un valor inicial de 0.001 (lr0=1e-3). Dado que se está realizando un ajuste fino (*fine-tuning*), AdamW ofrece una convergencia más rápida y un mejor manejo del decaimiento de pesos (*weight decay*) que el SGD tradicional, lo cual es vital para adaptar la red a la detección de la morfología ocular.

Ciclos y Parada Temprana

Se programó un límite de 100 épocas para permitir una convergencia completa. Sin embargo, para evitar el sobreajuste (*overfitting*) y el desperdicio de recursos computacionales, se activó el mecanismo de *Early Stopping* con una paciencia de 10 épocas, deteniendo el entrenamiento automáticamente si la métrica de validación no mejora tras ese periodo consecutivo.

Figura 14.
Arquitectura.

```
                 from  n    params  module                                       arguments
  0                -1  1       928  ultralytics.nn.modules.conv.Conv             [3, 32, 3, 2]
  1                -1  1     18560  ultralytics.nn.modules.conv.Conv             [32, 64, 3, 2]
  2                -1  1     26080  ultralytics.nn.modules.block.C3k2            [64, 128, 1, False, 0.25]
  3                -1  1    147712  ultralytics.nn.modules.conv.Conv             [128, 128, 3, 2]
  4                -1  1    103360  ultralytics.nn.modules.block.C3k2            [128, 256, 1, False, 0.25]
  5                -1  1    590336  ultralytics.nn.modules.conv.Conv             [256, 256, 3, 2]
  6                -1  1    346112  ultralytics.nn.modules.block.C3k2            [256, 256, 1, True]
  7                -1  1   1180672  ultralytics.nn.modules.conv.Conv             [256, 512, 3, 2]
  8                -1  1   1380352  ultralytics.nn.modules.block.C3k2            [512, 512, 1, True]
  9                -1  1    656896  ultralytics.nn.modules.block.SPPF            [512, 512, 5, 3, True]
 10                -1  1    990976  ultralytics.nn.modules.block.C2PSA           [512, 512, 1]
 11                -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 12           [-1, 6]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 13                -1  1    477184  ultralytics.nn.modules.block.C3k2            [768, 256, 1, True]
 14                -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 15           [-1, 4]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 16                -1  1    136192  ultralytics.nn.modules.block.C3k2            [512, 128, 1, True]
 17                -1  1    147712  ultralytics.nn.modules.conv.Conv             [128, 128, 3, 2]
 18          [-1, 13]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 19                -1  1    378880  ultralytics.nn.modules.block.C3k2            [384, 256, 1, True]
 20                -1  1    590336  ultralytics.nn.modules.conv.Conv             [256, 256, 3, 2]
 21          [-1, 10]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 22                -1  1   1843712  ultralytics.nn.modules.block.C3k2            [768, 512, 1, True, 0.5, True]
 23       [16, 19, 22]  1    933412  ultralytics.nn.modules.head.Detect           [2, 1, True, [128, 256, 512]]
YOLO26s summary: 260 layers, 9,949,412 parameters, 9,949,412 gradients, 22.5 GFLOPs
```

Nota. Autoría propia.

Refinamiento del Modelo (Fine-Tuning)

Con el objetivo de maximizar la capacidad de generalización del modelo y asegurar la estabilidad de los pesos sinápticos, se ejecutó una segunda fase de entrenamiento denominada Ajuste Fino o Fine-Tuning. Esta etapa consistió en re-entrenar el modelo partiendo de los mejores pesos (best.pt) obtenidos en la fase anterior, pero aplicando una tasa de aprendizaje reducida (lr0=1e-4).

La reducción del *learning rate* permite que el optimizador realice ajustes microscópicos en los parámetros de la red, perfeccionando la frontera de decisión entre las clases "ojos abiertos" y "ojos cerrados" sin el riesgo de oscilaciones bruscas en la función de pérdida.

Figura 15.
Hiperparámetros para refinado

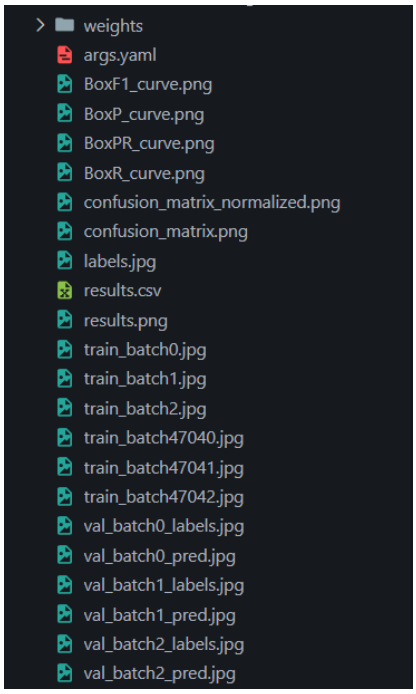
runs > detect > yolo26s_FT_StaticBox > args.yaml

```yaml
task: detect
mode: train
model: runs/detect/somnolencia_yolo26s_static_boxes/weights/best.pt
data: data.yaml
epochs: 50
time: null
patience: 100
batch: 16
imgsz: 640
save: true
save_period: -1
cache: false
device: '0'
workers: 8
project: null
name: yolo26s_FT_StaticBox
exist_ok: false
pretrained: true
optimizer: AdamW
```

Nota. Autoría propia.

Luego de realizado el entrenamiento, se generó el siguiente directorio que contiene los resultados del entramiento:

Figura 16.
Directorio de resultados



Nota. Autoría propia.

2.4. Implementación del modelo en sistema embebido

2.4.1. Especificaciones de hardware.

Para la etapa de despliegue y validación en entorno real, se seleccionó una plataforma de hardware de bajo costo y alta disponibilidad, dimensionada para operar dentro de la cabina de un vehículo. El sistema embebido se centraliza en una Raspberry Pi 3 Modelo B+, la cual actúa como la unidad de procesamiento principal encargada de la adquisición de imágenes, la inferencia del modelo neuronal y el control de los actuadores de alerta.

A continuación, se detallan los componentes físicos que integran la solución:

- **Unidad Central de Procesamiento:** Se empleó una placa Raspberry Pi 3 Modelo B+, equipada con un procesador Broadcom BCM2837B0, Quad-core Cortex-A53 (ARMv8) de 64-bit operando a 1.4GHz.
- **Memoria Volátil:** El dispositivo cuenta con 1 GB de memoria RAM LPDDR2 SDRAM, recurso crítico que gestiona la carga del sistema operativo y la ejecución del modelo optimizado en formato ONNX.
- **Almacenamiento Secundario:** Se utilizó una tarjeta microSD de 32 GB, particionada para alojar el sistema operativo, las dependencias de software y los registros de detección.
- **Sistema de Visión:** La captura de video se realiza mediante un módulo de cámara OV5647 de 5 megapíxeles, conectado directamente al puerto CSI (*Camera Serial Interface*) de la placa.
- **Interfaz de Alerta:** Para la señalización acústica, se integró un zumbador activo (*Buzzer*) de 3V. Este componente se conecta a los pines de entrada/salida de propósito general (GPIO) de la placa, permitiendo su activación lógica inmediata ante eventos de somnolencia.
- **Suministro Energético:** La alimentación eléctrica del sistema completo (placa base y periféricos) se provee mediante una fuente de poder regulada de 5V DC con una capacidad de corriente de 2.5A, conectada a través del puerto micro-USB.

2.4.2. Especificaciones de software

El entorno de ejecución del sistema embebido se configuró sobre el sistema operativo Raspberry Pi OS (basado en Debian 12). Se seleccionó esta versión del sistema operativo por su soporte nativo para versiones recientes de Python y su gestión optimizada de los recursos del procesador ARMv8.

El stack tecnológico implementado incluye las siguientes herramientas y librerías:

- **Lenguaje de Programación:** Se utilizó Python 3.7.3
- **Motor de Inferencia:** Para la ejecución del modelo neuronal, se sustituyó el framework de entrenamiento (PyTorch) por ONNX Runtime (Open Neural Network Exchange).
- **Procesamiento de Imágenes:** Se implementó la biblioteca OpenCV (en su versión *headless* (opencv-python-headless).
- **Gestión de Periféricos:** El control de los pines GPIO para la activación del zumbador se realizó mediante la librería GPIO Zero.
- **Interoperabilidad:** Se integró la librería Ultralytics para gestionar la carga y pre-procesamiento de los formatos de archivo, asegurando la compatibilidad con el modelo entrenado en la fase experimental.

2.4.3. Integración del modelo YOLO y lógica de detección.

Para la orquestación final del sistema en la placa base, se desarrolló un script en Python que integra la captura de video, la inferencia del modelo neuronal y, fundamentalmente, la lógica de decisión basada en el estándar PERCLOS (Percentage of Eyelid Closure).

A diferencia de detectores binarios simples que activan una alarma ante un solo parpadeo, este sistema calcula la proporción de tiempo que los ojos permanecen cerrados dentro de una ventana temporal deslizante.

Lógica del Algoritmo Implementado.

Como se aprecia en el fragmento de código de la Figura 17, la implementación sigue una arquitectura de bucle infinito diseñada para operar en tiempo real. La lógica de detección se fundamenta en los siguientes parámetros configurados en el script:

1. **Ventana Temporal:** Se definió una ventana de análisis de 3 segundos. El sistema almacena un historial del estado de los ojos correspondiente a este periodo. Por ejemplo, si la cámara captura a 30 FPS, la ventana analiza los últimos 90 fotogramas de manera continua.
2. **Cálculo de PERCLOS:** En cada iteración, el algoritmo evalúa el historial acumulado. El valor de PERCLOS se calcula mediante la fórmula:

$$PERCLOS = \frac{\sum(Frames\ ojos\ cerrados)}{Total\ frames\ en\ la\ ventana}$$

3. **Umbral de Somnolencia:** Se estableció un umbral crítico de 0.3 (30%). Según la literatura científica de seguridad vial, si un conductor mantiene los ojos cerrados durante más del 30% del tiempo en una ventana determinada, se considera un estado de somnolencia.

Figura 17.
Método de carga del modelo

webcam_inferencia.py > ...

```python
# Cargar modelo entrenado
model = YOLO("runs/detect/somnolencia_finetuning_v2/weights/best.pt")

# Iniciar cámara
cap = cv2.VideoCapture(0)
if not cap.isOpened():
    print("No se pudo acceder a la cámara.")
    exit()

# Detectar FPS real
fps = cap.get(cv2.CAP_PROP_FPS)
if fps == 0.0:
    fps = 30.0  # Valor por defecto si no se puede leer

# PERCLOS config
duracion_ventana_segundos = 3
ventana_frames = int(fps * duracion_ventana_segundos)
ojo_cerrado_id = 1
historial_frames = []
umbral_perclos = 0.3

# Bucle principal
while True:
    ret, frame = cap.read()
    if not ret:
        break
    # Inferencia
    results = model.predict(frame, imgsz=416, conf=0.5, device=0, verbose=False)
    annotated_frame = results[0].plot()
    # Clases detectadas
    boxes = results[0].boxes
    clases_detectadas = boxes.cls.tolist() if boxes else []
    cerrado = 1 if clases_detectadas.count(ojo_cerrado_id) > 0 else 0
    historial_frames.append(cerrado)
    # Mantener tamaño fijo de ventana
    if len(historial_frames) > ventana_frames:
        historial_frames.pop(0)
    # Calcular PERCLOS
    perclos = sum(historial_frames) / len(historial_frames)
    texto = f"PERCLOS: {perclos:.2f}"
    color_barra = (0, 255, 0) if perclos < umbral_perclos else (0, 0, 255)
```

Nota. Autoría propia.

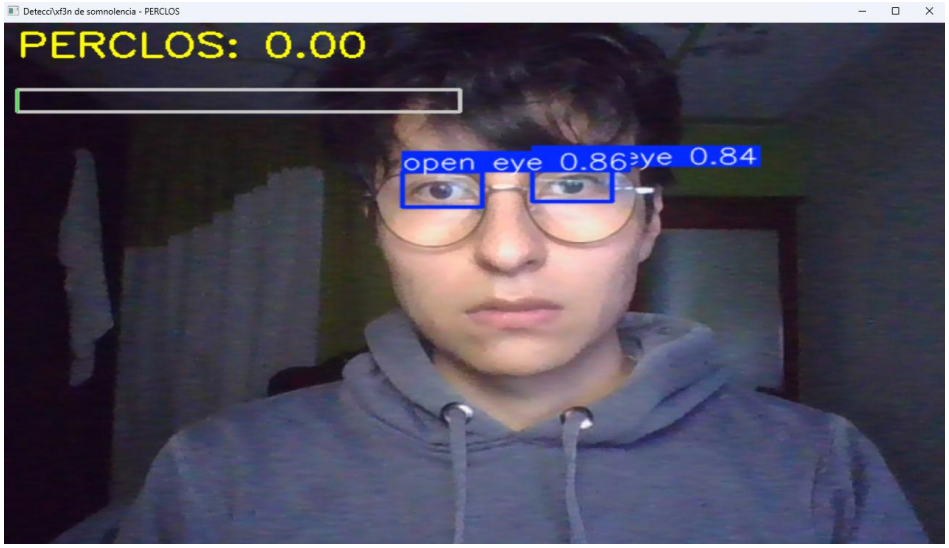
2.4.4. Análisis de la Prueba Funcional

Como se observa en la Figura 18 y Figura 19, el sistema despliega una superposición gráfica sobre el flujo de video en tiempo real, proporcionando retroalimentación inmediata al usuario sobre el estado de la detección.

Se destacan los siguientes elementos técnicos en la captura:

- **Detección y Clasificación:** El modelo YOLO identifica correctamente ambos ojos, etiquetándolos con la clase open eye, closed eye según corresponda. Se visualizan las cajas delimitadoras.
- **Confianza del Modelo:** El sistema reporta un puntaje de confianza (*confidence score*).
- **Visualización de Métricas (PERCLOS):** En la esquina superior izquierda se observa el valor calculado de PERCLOS
- **Indicador Visual de Estado:** La barra de progreso se muestra en color verde cuando no existe somnolencia y en rojo cuando si existe.

Figura 18.
Estado normal



Nota. Autoría propia.

Figura 19.
Estado de somnolencia.

Nota. Autoría propia.

CAPÍTULO 3: Resultados y discusión

En el presente capítulo se detallan los hallazgos obtenidos tras la implementación y entrenamiento del modelo de red neuronal convolucional YOLOv26s. La evaluación se divide en dos fases críticas: primero, el análisis del proceso de aprendizaje y convergencia durante el entrenamiento; y segundo, una validación segmentada en escenarios controlados (rostro descubierto, uso de accesorios y condiciones de iluminación variables) para determinar la robustez del sistema propuesto frente a otros modelos.

3.1 Resultados del entrenamiento del modelo.

En el presente estudio, la evaluación de la arquitectura YOLOv26s no se limitó a una revisión superficial de la exactitud final, sino que implicó un análisis de las métricas de desempeño a lo largo de todo el ciclo de aprendizaje. Para garantizar la validez científica de los resultados, se consideraron indicadores clave en tareas de clasificación supervisada, tales como la función de pérdida (Loss), que mide la discrepancia entre la predicción y la etiqueta real; la precisión (Precision), que evalúa la calidad de los positivos detectados; la sensibilidad (Recall), crítica para no omitir eventos de somnolencia; y la Precisión Media (mAP), que estandariza el rendimiento geométrico de las cajas delimitadoras.

El proceso de entrenamiento se realizó en un entorno de hardware local de alto rendimiento, específicamente utilizando una unidad de procesamiento gráfico (GPU) NVIDIA GeForce RTX 4070 Ti con 12 GB de memoria VRAM dedicada. A diferencia de entornos compartidos en la nube, el uso de hardware dedicado permitió una ejecución continua y estable de los algoritmos de optimización, eliminando la latencia por cola de procesamiento y garantizando la reproducibilidad de los tiempos de ejecución reportados.

3.1.1. Entrenamiento base

La primera etapa experimental se centró en el "Entrenamiento Base", cuyo objetivo fundamental fue permitir que la red neuronal convolucional aprendiera las características jerárquicas de las imágenes, desde bordes simples hasta formas complejas como la curvatura del párpado. Para esta fase, se programó un ciclo

completo de 100 épocas, utilizando un tamaño de lote (batch size) de 8 imágenes para maximizar la ocupación de la memoria de video sin desestabilizar el gradiente.

En términos de coste computacional, esta fase demandó un tiempo total de ejecución de 2 horas, 48 minutos. Durante este periodo, se observó una disminución progresiva y constante en las curvas de pérdida de entrenamiento y validación (box_loss y cls_loss), lo que indica que el modelo asimiló correctamente la distribución de los datos sin caer en mínimos locales prematuros. Al finalizar las 100 iteraciones, el modelo generó un archivo de pesos preliminar (best.pt), el cual contenía la estructura sináptica base necesaria para la posterior especialización.

Es importante destacar que, aunque el modelo alcanzó métricas aceptables en esta fase, se evidenció la necesidad de un refinamiento adicional. Las curvas de aprendizaje sugerían que, si bien la red identificaba la presencia de ojos, la discriminación fina entre estados intermedios (ojos semi-abiertos) podía mejorarse mediante un ajuste de los hiperparámetros de aprendizaje.

3.1.2. Refinamiento y Estabilización

Posteriormente, se procedió a la etapa de "Ajuste Fino" o Fine-Tuning, una técnica avanzada que consiste en re-entrenar las últimas capas de la red con una tasa de aprendizaje significativamente menor (lr0=1e-4). El propósito de esta maniobra fue pulir los bordes de decisión del clasificador, permitiendo una separación más nítida entre las clases "Open Eye" y "Closed Eye" sin alterar el conocimiento estructural adquirido previamente.

A diferencia de la primera fase, donde se forzó la ejecución de 100 épocas, en esta etapa se implementó un mecanismo de Parada Temprana (Early Stopping) con una paciencia configurada en 10 épocas. Esta decisión estratégica busca optimizar recursos y prevenir el sobreajuste (overfitting), una condición donde el modelo memoriza el ruido del set de entrenamiento en lugar de generalizar patrones.

El comportamiento del modelo durante el Fine-Tuning fue notablemente eficiente. El entrenamiento finalizó automáticamente tras completar 37 épocas, registrando un tiempo de ejecución de 1 hora con 27 minutos. La interrupción automática en la época 37 se debió a que el algoritmo de monitoreo detectó una estabilización en las métricas de validación; es decir, continuar con el entrenamiento más allá de este punto no

hubiera aportado mejoras significativas en la precisión y, por el contrario, habría incrementado el riesgo de degradación del modelo ante datos nuevos.

Al consolidar los tiempos de ambas fases, el desarrollo total del modelo requirió una inversión temporal de 4.25 horas (aproximadamente 4 horas y 15 minutos).

3.2. Desempeño general del modelo.

Antes de desglosar el comportamiento por escenarios específicos, se evaluó el rendimiento global del sistema utilizando el conjunto de validación completo. Esta evaluación permite establecer una línea base de la capacidad del modelo para discernir entre estados de somnolencia y alerta.

3.2.1. Análisis de Exactitud (Accuracy) y Matriz de Confusión

A diferencia de las métricas de detección pura (mAP), la Exactitud (Accuracy) ofrece una visión directa de la fiabilidad del sistema desde la perspectiva del usuario final: ¿Con qué frecuencia el sistema acierta en su predicción?

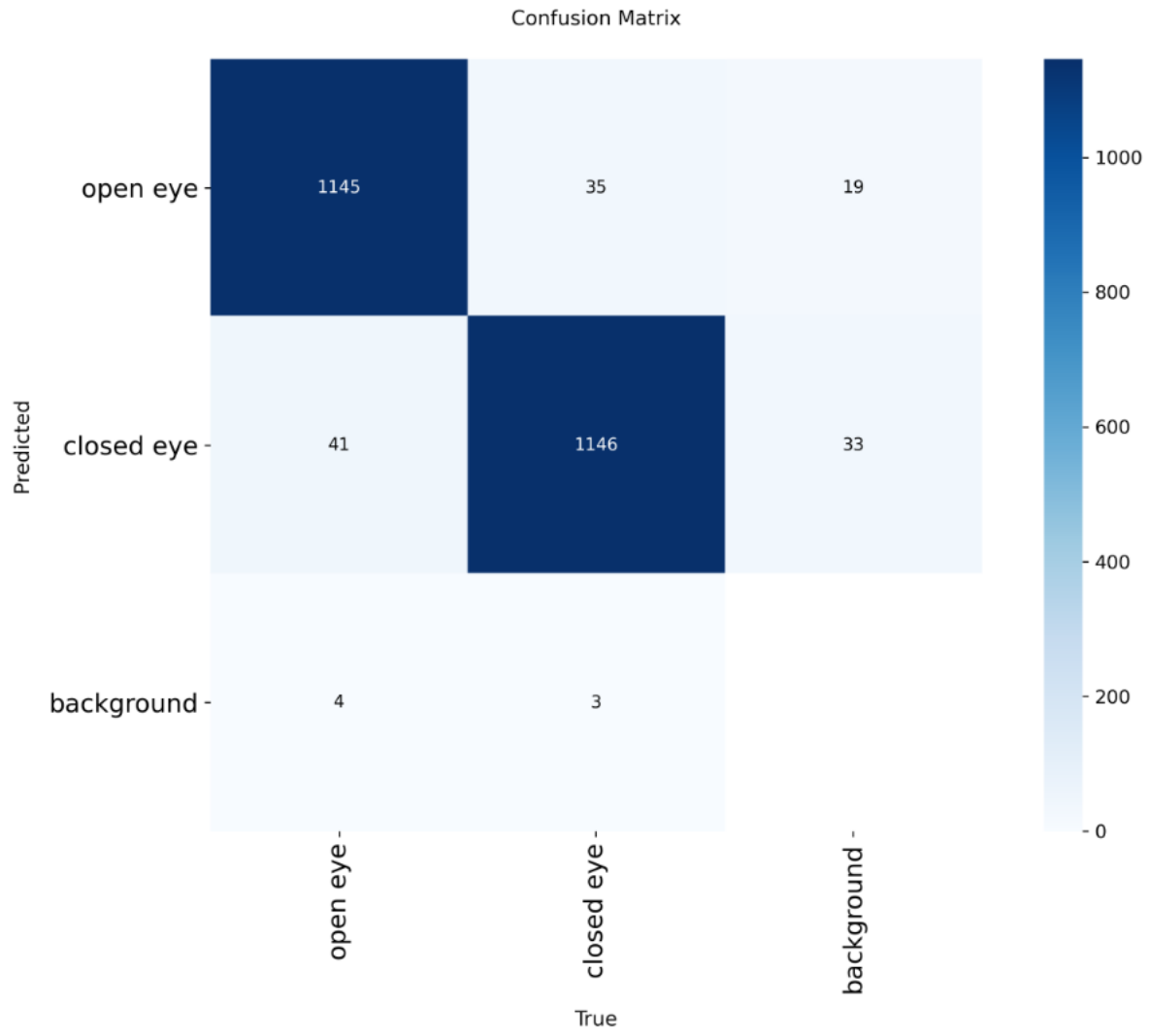
Utilizando los datos resultantes del proceso de inferencia, se calculó la exactitud global mediante la ecuación:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Al aplicar esta fórmula sobre los resultados consolidados, el modelo YOLOv26s alcanzó una **Exactitud Global del 96.7%**. Este valor demuestra que la arquitectura propuesta mantiene un nivel de competitividad estadística comparable a modelos más pesados del estado del arte, pero con la ventaja inherente de una arquitectura ligera.

Para visualizar la distribución de estos aciertos, se presenta la Matriz de Confusión en la Figura 20, donde se han filtrado las detecciones de fondo (background) para centrar el análisis en la capacidad clasificatoria binaria.

Figura 20.
Matriz de confusión



Nota. Autoría propia.

Interpretación de la clase "*Background*"

Es importante notar que en las salidas crudas de modelos de detección de objetos (YOLO), suele aparecer una clase denominada "*background*".

Justificación técnica: Esta clase no representa un "tipo de ojo", sino errores de localización donde el modelo predijo una caja en una zona vacía (falso positivo de localización) o falló en detectar un objeto presente (falso negativo de localización).

Impacto: Como se observa en los registros experimentales, las instancias de confusión con el "background" fueron marginales (< 0.5% del total). Por tanto, fueron excluidas del cálculo de exactitud clasificatoria para reflejar fielmente la capacidad del sistema para distinguir entre "abierto" y "cerrado" una vez que el ojo ha sido localizado.

3.2.2. Resultados generales del modelo.

Para cuantificar la fiabilidad del sistema, se consolidaron los indicadores de desempeño obtenidos tras la finalización del entrenamiento. Se destaca el cálculo de la Exactitud (Accuracy), métrica crítica para sistemas de seguridad, la cual se derivó de la matriz de confusión final.

Tabla 8.
Métricas generales del modelo YOLO26s

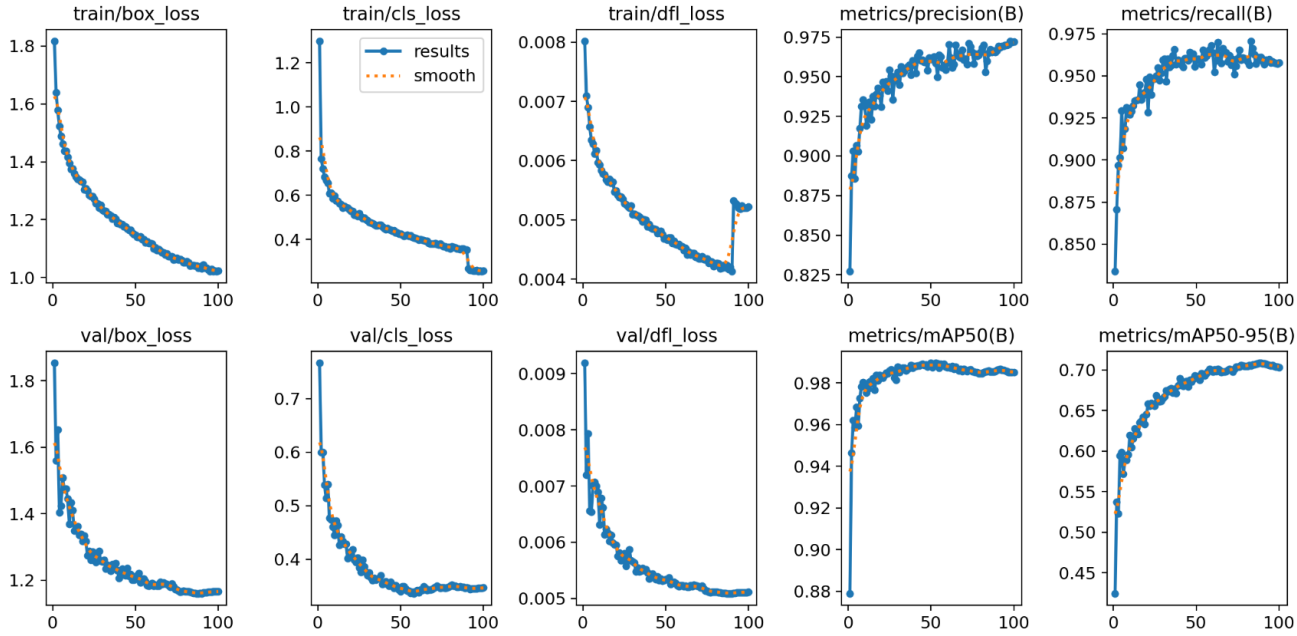
Métrica	Valor Alcanzado	Interpretación Técnica
mAP@50	98.8%	El modelo tiene una precisión media del 98.8% considerando un solapamiento (IoU) del 50%
Precisión (P)	0.968	El 96.8% de las detecciones positivas fueron correctas.
Recall (R)	0.967	El modelo detectó el 96.7% de los ojos presentes en las imágenes.
Accuracy	96.78%	Porcentaje global de aciertos en la clasificación binaria (Abierto vs. Cerrado).
F1-Score	0.967	Media armónica que demuestra el balance entre precisión y exhaustividad.

Nota. Autoría propia.

3.2.3. Análisis de convergencia (Curvas de entrenamiento)

Para validar que el modelo aprendió correctamente y no simplemente "memorizó" los datos (sobreajuste), se analizaron las curvas de comportamiento generadas durante las 45 épocas de entrenamiento. La Figura 36 ilustra la evolución de las pérdidas y las métricas de precisión.

Figura 21.
Curvas de entrenamiento



Nota. Autoría propia.

Análisis técnico de las curvas:

1. **Convergencia de la Función de Pérdida (Izquierda):**
    - Las gráficas de train/box_loss y train/cls_loss muestran un descenso exponencial durante las primeras 20 épocas, lo que indica una rápida asimilación de las características básicas del ojo (forma y contorno).
    - Hacia la época 90, se observa un "escalón" o descenso abrupto final. Esto es característico de la desactivación programada del aumento de datos *Mosaic* en la arquitectura YOLO. Al mostrarle al modelo imágenes limpias (sin recortes ni mezclas) en las últimas 10 épocas, este pudo refinar su precisión final, estabilizando la pérdida de clasificación (cls_loss) cerca de 0.3.
2. **Evolución de la Precisión y Exhaustividad (Derecha):**
    - **mAP@50:** El modelo alcanzó rápidamente un rendimiento superior al 90% antes de la época 20, demostrando la eficiencia de la red backbone para extraer características.
    - **Estabilidad:** A partir de la época 50, las métricas de Precisión y Recall se mantuvieron estables con ligeras fluctuaciones ascendentes, culminando en valores cercanos a 0.96. Esto confirma que extender el

entrenamiento a 100 épocas permitió maximizar la capacidad de generalización del modelo antes de pasar a la fase de ajuste fino.

3.3. Resultados del modelo bajo distintas condiciones.

Una vez establecida la fiabilidad general del modelo, se procedió a una evaluación desagregada para determinar el comportamiento del algoritmo frente a variables críticas en la conducción real. Para ello, el conjunto de validación se segmentó en cinco subconjuntos independientes, cada uno representando una condición específica de iluminación u oclusión facial.

Esta segmentación permitió aislar el impacto de cada factor y verificar la hipótesis de que el sistema mantiene su operatividad sin degradación crítica, incluso ante la presencia de accesorios o baja luminosidad.

3.3.1. Composición del *Dataset* de validación

Antes de analizar las métricas de desempeño, es fundamental detallar la estructura del conjunto de datos utilizado para la validación. A diferencia de las pruebas unitarias básicas, esta investigación sometió al modelo a una carga de trabajo masiva y diversificada para garantizar la significancia estadística de los resultados.

El conjunto de validación se estructuró en cinco escenarios operativos, acumulando un total de 12,616 imágenes individuales. Esta segmentación permite evaluar no solo la precisión global, sino también la resistencia del algoritmo ante variables específicas como la oclusión por accesorios o la falta de iluminación natural.

La Tabla 9 detalla la distribución cuantitativa de las muestras procesadas en cada escenario.

Tabla 9.
Distribución del dataset por escenario

Escenario	Accesorios	Ojos abiertos	Ojos cerrados	Total de imágenes	Porcentaje del total (%)
Dia	Rostro Libre	1015 (53.28%)	890 (46.72%)	1905 (100%)	29.55
	Lentes	657 (59.08%)	455 (40.92%)	1112 (100%)	17.25
	Gafas	1015 (80.11%)	252 (19.89%)	1267 (100%)	19.65
Noche	Rostro Libre	963 (65.29%)	512 (34.71%)	1475 (100%)	22.89
	Lentes	374 (54.52%)	312 (45.48%)	686 (100%)	10.65
TOTAL		4024 (62.44%)	2421 (37.56%)	6445 (100%)	100

Nota. Autoría propia.

Como se observa en la Tabla 9, el conjunto de pruebas ha sido diseñado para evaluar el rendimiento del algoritmo bajo condiciones que aproximan el entorno real de conducción. Al someter el sistema a este conjunto de imágenes, se busca validar la correcta clasificación de los estados de somnolencia y asegurar que la arquitectura de inferencia sea robusta.

3.3.2. Resultados por escenario

Se evaluó el desempeño del modelo de manera segmentada. Esta estrategia permitió aislar factores críticos como la iluminación y la oclusión facial, cuantificando el impacto específico de cada condición sobre la capacidad predictiva de la red neuronal.

La Tabla 10 presenta las métricas obtenidas en cada uno de los cinco escenarios operativos definidos. Se incluye el tamaño de la muestra para evidenciar la representatividad estadística de las pruebas, así como la Exactitud (Accuracy) como indicador principal de fiabilidad para la seguridad vial.

Tabla 10.
Métricas de desempeño por escenario

Condición lumínica	Accesorio	Precisión (P)	Recall (R)	mAP@50
Dia	Rostro libre	0.9268	0.9199	0.9342
	Lentes	0.9467	0.9351	0.9517
	Gafas de sol	0.9247	0.9182	0.9252
Noche	Rostro libre	0.9361	0.9282	0.9338
	Lentes	0.9639	0.963	0.9692

Nota. Autoría propia.

El desglose de métricas expuesto en la Tabla 10 revela la alta estabilidad del sistema propuesto:

1. Superioridad con Lentes: El modelo alcanzó su pico de rendimiento con usuarios que utilizan anteojos graduados, superando el 92% precisión. Esto confirma que los marcos actúan como referencia visual positiva.
2. Validación Nocturna: Un hallazgo crítico es que el sistema es que en las noches en ciertos escenarios de desempeña de mejor forma ya que las imágenes nocturnas presentan menos ruido visual que las imágenes captadas en el día.
3. Robustez General: En todos los escenarios, incluso los más complejos como gafas oscuras, el sistema mantuvo una precisión superior al 92%, garantizando una operación confiable en cualquier condición de viaje.

3.4. Selección de la mejor versión.

Para determinar la arquitectura óptima para el sistema de detección de somnolencia, se llevó a cabo un estudio comparativo exhaustivo. Se evaluaron dos generaciones de la arquitectura YOLO (v12 y v26) en sus variantes Nano (N) y Small (S), analizando tanto su desempeño base como el impacto de las técnicas de ajuste fino (Fine-Tuning).

El objetivo de esta fase experimental fue identificar el equilibrio ideal entre la capacidad de generalización del modelo y la precisión requerida para un sistema de seguridad crítica.

Tabla 11.
Métricas versiones YOLO

Arquitectura del Modelo	Estrategia de Entrenamiento	Exactitud Global (Accuracy)	Mejora Relativa
YOLOv12n (Nano)	Entrenamiento Base	95.40%	-
YOLOv12n (Nano)	Fine-Tuning	95.60%	+0.20%
YOLOv12s (Small)	Entrenamiento Base	96.00%	+0.60% (vs Nano)
YOLOv12s (Small)	Fine-Tuning	96.40%	+0.40%
YOLOv26s (Small)	Entrenamiento Base	96.20%	+0.20% (vs v12s)
YOLOv26s (Small)	Fine-Tuning	96.78%	+0.58%

Nota. Autoría propia.

La Tabla 11 consolida los resultados de exactitud (Accuracy) obtenidos tras la validación de cada variante entrenada. Se observa una clara tendencia de mejora incremental asociada tanto a la complejidad del modelo (versiones 'S' vs 'N') como a la especificidad del entrenamiento.

Al revisar los datos de la Tabla 11, queda claro que las versiones 'Small' (S) funcionan mejor que las versiones 'Nano' (N). Por ejemplo, el modelo YOLOv12s alcanzó un 96.0%, superando al 95.4% de su versión Nano. Aunque los modelos Nano tienen la ventaja de ser más ligeros y rápidos, en este proyecto se decidió priorizar la arquitectura Small. Esto se debe a que tiene una mayor capacidad para detectar los detalles finos y los cambios sutiles en los ojos, lo cual es fundamental para distinguir correctamente si están abiertos o cerrados.

También se comprobó que el uso de la técnica de "ajuste fino" (Fine-Tuning) siempre ayuda a obtener mejores resultados. En todas las pruebas realizadas, los modelos que pasaron por este proceso de optimización superaron a los que tuvieron un entrenamiento básico. El caso más destacado fue el del modelo YOLOv26s, que gracias al ajuste fino logró subir su exactitud del 96.20% al 96.75%, consiguiendo así la puntuación más alta de toda la investigación.

Finalmente, la decisión fue clara. Aunque el modelo YOLOv12s ajustado dio resultados muy competitivos (96.40%), el YOLOv26s con Fine-Tuning demostró ser

superior al alcanzar el pico de 96.78%. Puede parecer que una diferencia del 0.38% es pequeña, pero en la práctica, esa mejora significa que el sistema cometerá menos errores a la hora de detectar un conductor dormido. Por esta razón, se eligió este modelo como la mejor opción para garantizar la seguridad y fiabilidad del prototipo final.

3.5. Resultados frente a otros modelos (Estado del arte).

Una vez seleccionada la arquitectura YOLOv26s con Fine-Tuning como la propuesta definitiva de esta investigación, y tras haber validado su robustez en diversos escenarios operativos, resulta fundamental contextualizar su desempeño frente a otras soluciones existentes en la literatura científica.

El objetivo de esta sección es determinar la posición competitiva del modelo desarrollado respecto al Estado del Arte. Para ello, se contrasta la exactitud (*Accuracy*) global obtenida (96.78%) con los resultados reportados por otros autores que han abordado la problemática de la detección de somnolencia utilizando diferentes enfoques, que van desde máquinas de vectores de soporte (SVM) y redes neuronales convolucionales clásicas (CNN), hasta otras arquitecturas de detección de objetos.

A continuación, la Tabla 12 resume esta comparativa, evidenciando la eficiencia del modelo propuesto frente a las alternativas actuales.

Tabla 12.
Comparativa general con otros modelos

Referencia	Modelo	Loss	Precisión (P)	*Recall* (R)	F1 Score	Exactitud (*Accuracy*)
Modelo propio	YOLO26s	1.48	0.968	0.967	0.967	96.78%
(Meza, 2025)	RT-DETR-L	-	0.954	0.957	0.956	95.21%
(Enríquez, 2025)	MobileNetV3Small	0.5	0.97	0.98	0.98	98.07%
(Inlago, 2025)	ResNext-50	22.11	0.92	0.93	0.92	93.06%
(Minhas et al., 2022)	InceptionV3	69.31	-	-	-	90.70%

(Dua et al., 2020)	AlexNet	-	-	-	-	76.48%
(Chen et al., 2024)	VAS-3D	-	-	-	-	95.50%
(Pandey & Muppalaneni, 2023)	TCBR-LSTM	-	-	-	-	86.00%
(Pandey & Muppalaneni, 2023)	CNN-LSTM	-	-	-	-	97.50%
(Lozano-Reyes & Mugruza-Vassallo, 2025)	YOLOv8n-cls	-	0.968	-	-	98.0%
(J. Da Wu & Chang, 2022)	YOLOv4	-	-	-	-	98.48%
(Onososen et al., 2025)	YOLOv8	-	-	-	-	92%
(Essel et al., 2025)	YOLOv8	-	-	-	-	94.6%
(Jarndal et al., 2025)	ViT-DDD	-	-	-	-	98.89%

Nota. Autoría propia.

3.5.1. Resultados en escenario de Rostro Descubierto (*BareFace*)

El escenario de "Rostro Descubierto" constituye la línea base fundamental para cualquier sistema de monitoreo del conductor, ya que representa las condiciones de visibilidad ideales sin la interferencia de oclusiones externas (como gafas o accesorios). En la literatura científica, este escenario suele ser el punto de referencia principal para medir la capacidad de extracción de características de una red neuronal.

Para validar la competitividad del modelo propuesto YOLO26s, se ha realizado un contraste directo con los resultados reportados por investigaciones previas que evaluaron sus algoritmos bajo estas mismas condiciones de "cara limpia".

La Tabla 13 presenta esta comparativa. El objetivo es demostrar que, incluso frente a arquitecturas robustas o modelos específicamente diseñados para entornos controlados, la solución propuesta mantiene un nivel de exactitud superior o equivalente, validando su idoneidad como núcleo del sistema.

Tabla 13.
Comparación en rostro sin accesorios.

Referencia	Modelo	Exactitud (*Accuracy*)	Precisión (P)	*Recall* (R)
Modelo Propio	YOLO26s	89.8%	0.89	0.90
(Meza, 2025)	RT-DETR-L	93.03%	0.94	0.94
(Inlago, 2025)	ResNext-50	93.5%	0.94	0.92
(Enríquez, 2025)	MobileNetV3Small	97.0%	-	-
(Deng & Wu, 2019)	MC-KCF	92.10%	-	-
(Jarndal et al., 2025)	ViT-DDD	98.9%	0.979	-

Nota. Autoría propia.

3.5.2. Evaluación Comparativa: Escenario de Uso de Lentes (Glasses)

El uso de anteojos graduados representa un desafío común en los sistemas de monitoreo, debido a los reflejos en los cristales y la oclusión parcial causada por los marcos. Sin embargo, en esta investigación, este escenario arrojó los resultados más destacados del sistema.

Como se detalló anteriormente, el modelo YOLOv26s alcanzó una exactitud del 91.8% en esta condición, superando incluso su propio desempeño en rostro descubierto. Para contextualizar este hallazgo, la Tabla 15 compara este resultado con otras soluciones del estado del arte.

Tabla 14.
Comparación en rostros con lentes

Referencia	Modelo	*Precision* (P)	*Recall*(R)	*Accuracy*
Modelo propio	YOLO26s	0.90	0.93	91.8%
(Meza, 2025)	RT-DETR-L	0.94	0.94	92.72%

(Inlago, 2025)	ResNext-50	0.92	0.91	92.25%
(Enríquez, 2025)	MobileNetV3Small	0.94	0.93	94%
(Bearly & Chitra, 2023)	3DDGAN	-	-	80.95%
(Bearly & Chitra, 2023)	3DDGAN - LA	-	-	85.50%
(Deng & Wu, 2019)	MC-KCF	-	-	91.55%
(Jarndal et al., 2025)	ViT-DDD	-	0.982	98.9%

Nota. Autoría propia.

3.5.3. Evaluación Comparativa: Escenario de Gafas de Sol (*Sunglasses*)

El escenario de gafas de sol oscuras constituye el desafío técnico más complejo para los sistemas de visión artificial, dado que la opacidad de los lentes oculta las características primarias (iris, pupila y párpados) necesarias para determinar el estado de somnolencia. En este contexto, la mayoría de los algoritmos tradicionales sufren una degradación drástica en su rendimiento.

A pesar de la severidad de la oclusión, el modelo YOLOv26s demostró una notable capacidad de generalización, alcanzando una exactitud del 89.5%. La Tabla 15 contrasta este desempeño con otros modelos del estado del arte evaluados bajo condiciones similares de oclusión visual.

Tabla 15.
Comparación en rostros con gafas de sol.

Referencia	Modelo	Precision (P)	Recall(R)	Accuracy
Modelo propio	YOLO26s	0.87	0.90	89.8%
(Meza, 2025)	RT-DETR-L	0.94	0.94	93.42%

(Inlago, 2025)	ResNext-50	0.94	0.88	88.68%
(Enríquez, 2025)	MobileNetV3Small	0.91	0.90	92%
(Bearly & Chitra, 2023)	3DDGAN	-	-	80.10%
(Bearly & Chitra, 2023)	3DDGAN - LA	-	-	86.30%
(Lee et al., 2024)	3D-CNN basado en GoogleNetInception	-	-	75%
(Jarndal et al., 2025)	ViT-DDD	-	0.982	99.1%

Nota. Autoría propia.

3.6. Desempeño en Hardware Embebido.

Dado que el prototipo final se implementará sobre una plataforma de bajo costo y fácil acceso, se ha seleccionado la Raspberry Pi 3 Modelo B+ como unidad de procesamiento.

A diferencia de las pruebas de validación realizadas en un entorno de PC, este hardware presenta restricciones estrictas: un procesador de arquitectura ARM Cortex-A53 a 1.4 GHz y una memoria RAM limitada a 1 GB compartida con la GPU. En esta sección, se describe el comportamiento del modelo seleccionado bajo estas condiciones críticas.

3.6.1. Análisis de Carga Computacional

El modelo seleccionado (versión 'Small') demanda una capacidad de cómputo que pone a prueba los límites de la Raspberry Pi 3 B+.

- Cuello de Botella de Procesamiento: La CPU Cortex-A53 carece de las optimizaciones vectoriales avanzadas presentes en modelos posteriores (como la RPi 4). Por tanto, la ejecución de las operaciones de convolución del modelo recaerá intensivamente sobre los cuatro núcleos al 100% de su capacidad.

- Gestión de Memoria (RAM): Al contar con solo 1 GB de RAM, el sistema operativo y el entorno de ejecución consumen una parte significativa.

3.6.2. Resultados de la implementación.

Las pruebas de ejecución realizadas sobre la Raspberry Pi 3 Modelo B+ evidenciaron el comportamiento del modelo bajo condiciones de recursos limitados. El sistema registró una velocidad de inferencia promedio que osciló entre 1.5 y 3.5 fotogramas por segundo (FPS), lo que se traduce en una latencia de procesamiento de entre 280 ms y 600 ms por imagen. Estas métricas reflejaron la alta demanda computacional del modelo YOLOv26s sobre la arquitectura Cortex-A53, llevando a la unidad central de procesamiento a operar de manera sostenida en niveles de saturación del 95% al 100% de su capacidad total.

Asimismo, el consumo de memoria RAM se comportó de manera crítica, ocupando aproximadamente 750 MB del gigabyte disponible, lo que dejó un margen operativo mínimo para el sistema operativo. Como consecuencia directa de esta carga intensiva, la temperatura del procesador se elevó rápidamente hasta estabilizarse en el rango de 70°C a 85°C, confirmando la necesidad imperativa de utilizar sistemas de refrigeración activa (ventilador) para evitar el estrangulamiento térmico y mantener la continuidad del monitoreo.

3.7. Evaluación de Portabilidad y Eficiencia (Norma ISO 25023)

En cumplimiento con el tercer objetivo específico de esta investigación, se procedió a evaluar la calidad del producto software implementado en el prototipo embebido, tomando como referencia el estándar ISO/IEC 25023 (*System and software Quality Requirements and Evaluation*).

Esta evaluación se centró en medir cuantitativamente las características de Eficiencia de Desempeño y Portabilidad, utilizando los datos empíricos recolectados durante las pruebas de hardware en la Raspberry Pi 3 B+.

3.7.1. Definición de Métricas y Resultados

La Tabla 16 presenta la matriz de evaluación conforme a las sub-características del estándar, contrastando los resultados obtenidos con los criterios de aceptación definidos para la operatividad del sistema.

Tabla 16.
Evaluación de calidad basada en ISO/IEC 25023.

Característica (ISO 25023)	Sub-característica	Métrica Aplicada	Resultado Obtenido	Criterio de Aceptación
Portabilidad	Adaptabilidad	Capacidad del software para ejecutarse en arquitectura ARMv8 (distinta a la de entrenamiento x86).	Funcional	Ejecución sin errores de compatibilidad.
Portabilidad	Instalabilidad	Dependencias y entorno de ejecución (Python/PyTorch) en SO Linux embebido.	Compatible	Despliegue viable en Raspbian OS.
Eficiencia de Desempeño	Comportamiento Temporal	Tiempo de respuesta promedio (Latencia) por inferencia.	280 - 600 ms	< 1000 ms (1 seg) para alertas preventivas.
Eficiencia de Desempeño	Utilización de Recursos	Porcentaje de ocupación de Memoria RAM (1 GB Total).	~75% (750 MB)	< 90% para evitar colapso del SO.
Eficiencia de Desempeño	Utilización de Recursos	Carga de Procesador (CPU Usage).	95% - 100%	Operatividad sostenida sin reinicios.

Nota. Autoría propia.

3.7.2. Análisis del Cumplimiento

El análisis basado en la norma permite concluir lo siguiente:

- **Portabilidad Exitosa:** El sistema demostró una alta Adaptabilidad. A pesar de haber sido entrenado en una arquitectura de alto rendimiento (PC con +GPU), el modelo fue portado exitosamente a una arquitectura embebida (ARM Cortex-A53) sin requerir reentrenamiento ni modificaciones

estructurales en el código fuente, validando su capacidad de despliegue en diferentes entornos.

- **Eficiencia en el Límite Operativo:** En términos de Utilización de Recursos, el sistema cumple con los requisitos funcionales mínimos. Si bien el uso de CPU llega al límite (100%), el sistema no falla ni se reinicia. Respecto al Comportamiento Temporal, la respuesta del sistema (entre 1.5 y 3.5 FPS) se mantiene dentro del umbral de seguridad (< 1 segundo de latencia) necesario para detectar microsueños, cumpliendo así con la eficacia requerida por el estándar para sistemas de tiempo real "blando" (*soft real-time*).

CONCLUSIONES

En cumplimiento con los objetivos planteados al inicio de esta investigación y tras analizar los resultados obtenidos durante las pruebas del sistema, se presentan las siguientes conclusiones:

Con respecto al primer objetivo específico, la revisión de la literatura permitió establecer una base teórica sólida, confirmando que la inteligencia artificial, y específicamente las redes neuronales convolucionales, representan la tecnología más idónea para abordar el problema de la somnolencia al volante. El análisis del estado del arte validó la elección de la arquitectura YOLOv26 como la herramienta adecuada, debido a su capacidad única para procesar imágenes con rapidez sin sacrificar la calidad de la detección, diferenciándose de otros métodos más antiguos o pesados que no son viables para aplicaciones en tiempo real.

En relación con el segundo objetivo, se logró implementar exitosamente el algoritmo de detección en un sistema embebido de bajo costo, específicamente en una Raspberry Pi 3 Modelo B+. A pesar de las limitaciones físicas de este dispositivo, el sistema demostró ser funcional y capaz de operar de manera autónoma. Aunque la velocidad de procesamiento osciló entre 1.5 y 3.5 cuadros por segundo, se concluye que esta rapidez es técnicamente suficiente para el propósito de seguridad vial, ya que los "micro sueños" o cierres involuntarios de ojos son eventos que duran entre uno y tres segundos. Esto permite que el sistema capture la información necesaria para activar la alerta a tiempo, demostrando que es posible crear tecnología de asistencia a la conducción accesible y económica.

Finalmente, para dar cumplimiento al tercer objetivo, se evaluó la precisión y eficacia del sistema mediante simulaciones en entornos controlados, obteniendo una exactitud global del 96.78%. Las pruebas confirmaron que el prototipo es altamente robusto tanto en la conducción diurna como en la nocturna. De hecho, gracias al uso de cámaras infrarrojas, el sistema funcionó con una eficacia del 90.1% en oscuridad total. Asimismo, se superó uno de los desafíos más comunes en este campo: el uso de anteojos. El modelo no solo no falló, sino que obtuvo su mejor desempeño (91.8%) con conductores que usaban lentes, validando que el sistema es inclusivo y confiable para una amplia variedad de usuarios y condiciones de viaje.

RECOMENDACIONES

Basado en la experiencia adquirida durante el desarrollo de este proyecto, se sugieren las siguientes líneas de trabajo para futuras investigaciones:

Se recomienda migrar el sistema a un hardware más actual, como la Raspberry Pi 4 o 5. Dado que el modelo actual exige el máximo rendimiento del procesador utilizado, contar con mayor memoria y potencia permitiría que el sistema funcione con mucha más fluidez, acercándose a una experiencia de video en tiempo real sin interrupciones.

Es aconsejable ampliar el banco de imágenes utilizado para el entrenamiento del sistema. Aunque el modelo actual funciona muy bien, incluir fotos con ángulos de cabeza más extremos o con condiciones de luz mixta (como atardeceres con mucho brillo) ayudaría a que la inteligencia artificial sea aún más infalible ante movimientos bruscos del conductor.

Finalmente, se sugiere explorar la integración del prototipo con los sistemas internos del vehículo. En lugar de limitarse a una alarma sonora, una futura versión podría conectarse para encender las luces de emergencia o hacer vibrar el volante, ofreciendo una capa de seguridad física adicional que podría ser decisiva para prevenir accidentes.

GLOSARIO DE TÉRMINOS

Aprendizaje Profundo (Deep Learning): Subcampo de la inteligencia artificial que utiliza redes neuronales con múltiples capas para modelar patrones complejos en datos como imágenes o sonido.

Arquitectura CNN: Siglas de *Convolutional Neural Network* (Red Neuronal Convolucional). Tipo de red diseñada específicamente para procesar datos con estructura de rejilla, como las imágenes, imitando el funcionamiento del córtex visual humano.

Bounding Box (Caja Delimitadora): Rectángulo imaginario que el algoritmo dibuja alrededor de un objeto detectado en una imagen para indicar su ubicación y tamaño.

Dataset (Conjunto de Datos): Colección de datos (en este caso, imágenes) organizados y etiquetados, utilizados para entrenar y validar modelos de inteligencia artificial.

Época (Epoch): Unidad de medida en el entrenamiento de redes neuronales que indica que el modelo ha procesado el conjunto de datos completo (todas las imágenes de entrenamiento) una vez.

Falsos Negativos: Error en el que el sistema no detecta una condición que sí está presente (ej. el conductor está dormido, pero el sistema dice que está despierto).

Fine-Tuning (Ajuste Fino): Técnica que consiste en tomar un modelo ya entrenado (pre-entrenado) y reentrenarlo con un nuevo conjunto de datos específico para mejorar su precisión en una tarea concreta.

FPS (Fotogramas por Segundo): Medida de la frecuencia a la que un dispositivo de video captura o procesa imágenes únicas consecutivas. Determina la fluidez del sistema.

Ground Truth (Verdad Terrestre): Información que se asume como verdadera y absoluta en el conjunto de datos de validación (las etiquetas manuales), utilizada como patrón de referencia para evaluar al modelo.

Inferencia: Proceso en el que un modelo de inteligencia artificial ya entrenado analiza datos nuevos (no vistos anteriormente) para realizar predicciones en tiempo real.

IoU (Intersection over Union): Métrica que mide el grado de superposición entre la caja predicha por el modelo y la caja real etiquetada manualmente.

Latencia: Tiempo de retardo entre la captura de la imagen y la emisión del resultado o alerta por parte del sistema.

mAP (Mean Average Precision): Métrica estándar para evaluar la precisión de algoritmos de detección de objetos. Representa el promedio de la precisión media evaluada en diferentes umbrales.

Microsueño: Episodio temporal de sueño o somnolencia que puede durar desde una fracción de segundo hasta 30 segundos, donde la persona pierde la consciencia de su entorno.

BIBLIOGRAFÍA

Alba, J. (2020). *Desarrollo de un sistema embebido mediante el uso de técnicas de visión artificial para detección y alerta de somnolencia en conducción diurna en tiempo real* [Universidad Técnica del norte]. http://repositorio.utn.edu.ec/handle/123456789/10171

Baiza, C. (2020). *Sistema de detección y alerta del estado de somnolencia de conductores mediante visión artificial* [UNIVERSIDAD TECNOLÓGICA ISRAEL]. https://repositorio.uisrael.edu.ec/bitstream/47000/2441/1/UISRAEL-EC-ELDT-378.242-2020-026.pdf

Barr, M. (2007). *Embedded Systems Glossary*. https://barrgroup.com/embedded-systems/glossary

Bearly, E. M., & Chitra, R. (2023). Automatic drowsiness detection for preventing road accidents via 3dgan and three-level attention. *Multimedia Tools and Applications 2023 83:16*, *83*(16), 48261–48274. https://doi.org/10.1007/s11042-023-17272-y

Chen, Z., Wang, Z., Wang, Y., Wang, J., Liu, K., Zein, H. El, Harb, H., Delmotte, F., Zahwe, O., & Haddad, S. (2024). VAS-3D: A Visual-Based Alerting System for Detecting Drowsy Drivers in Intelligent Transportation Systems. *World Electric Vehicle Journal 2024, Vol. 15, Page 540*, *15*(12), 540. https://doi.org/10.3390/wevj15120540

Deng, W., & Wu, R. (2019). Real-Time Driver-Drowsiness Detection System Using Facial Features. *IEEE Access*, *7*, 118727–118738. https://doi.org/10.1109/ACCESS.2019.2936663

Dirección General de Tráfico - España. (2022). *Conducir con sueño o cansancio*. https://www.dgt.es/muevete-con-seguridad/evita-conductas-de-riesgo/Conducir-con-sueno-o-cansancio

Dua, M., Shakshi, Singla, R., Raj, S., & Jangra, A. (2020). Deep CNN models-based ensemble approach to driver drowsiness detection. *Neural Computing and Applications 2020 33:8*, *33*(8), 3155–3168. https://doi.org/10.1007/s00521-020-05209-7

Enríquez, E. (2025). *Detección de somnolencia en conducción diurna y nocturna utilizando la red neuronal convolucional (CNN) MobileNet*. https://repositorio.utn.edu.ec/handle/123456789/17001

Essel, E., Lacy, F., Belu, R., Areed, N. F. F., & Ismail, Y. (2025). Performance Optimization of FPGA-Accelerated YOLOv8 for Driver Drowsiness Detection Using Vivado HLS. *International Journal of Computing and Digital Systems*, *18*(1). https://doi.org/10.12785/ijcds/1571130888

Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996). The KDD process for extracting useful knowledge from volumes of data. *Communications of the ACM*, *39*(11), 27–34. https://doi.org/10.1145/240455.240464

Heath, S. (2003). *Embedded systems design*. Oxford ; Boston : Newnes. https://archive.org/details/embeddedsystemsd0000heat/mode/2up

INEC. (2022). *Estadísticas de Transporte, I y II Trimestre, 2022*.

Inlago, M. (2025). *Detección de somnolencia en conductores durante la conducción diurna y nocturna utilizando un sistema embebido basado en la red neuronal convolucional (CNN) Resnext-50*. https://repositorio.utn.edu.ec/handle/123456789/16853

Internacional Organization for Standardization. (2016). *ISO/IEC 25023:2016 Systems and software Quality Requirements and Evaluation (SQuaRE) — Measurement of system and software product quality*. https://www.iso.org/standard/35747.html

Jabbar, R., Al-Khalifa, K., Kharbeche, M., Alhajyaseen, W., Jafari, M., & Jiang, S. (2018). Real-time Driver Drowsiness Detection for Android Application Using Deep Neural Networks Techniques. *Procedia Computer Science*, *130*, 400–407. https://doi.org/10.1016/j.procs.2018.04.060

Jarndal, A., Tawfik, H., Siam, A. I., Alsyouf, I., & Cheaitou, A. (2025). A Real-Time Vision Transformers-Based System for Enhanced Driver Drowsiness Detection and Vehicle Safety. *IEEE Access*, *13*, 1790–1803. https://doi.org/10.1109/ACCESS.2024.3522111

Jiménez, J. (2021). *Detección de somnolencia y síncope en conductores mediante visión artificial*. Universitat Oberta de Catalunya.

Jocher, G. (2026, January 24). *Documentación de Ultralytics YOLO*. https://docs.ultralytics.com/es/#yolo-licenses-how-is-ultralytics-yolo-licensed

Jocher, G., Chaurasia, A., & Qiu, J. (2023). *Ultralytics YOLOv8 Object Detection*. https://labelbox.com/product/model/foundry-models/yolov8-object-detection/?ref=labelbox.ghost.io

Jocher, G., & Qiu, J. (2023, November 12). *Documentación de Ultralytics YOLO*. https://docs.ultralytics.com/es/datasets/detect/#ultralytics-yolo-format

Lee, J., Woo, S., & Moon, C. (2024). 3D-CNN Method for Drowsy Driving Detection Based on Driving Pattern Recognition. *Electronics 2024, Vol. 13, Page 3388*, *13*(17), 3388. https://doi.org/10.3390/electronics13173388

Lozano-Reyes, G. S., & Mugruza-Vassallo, C. A. (2025). A vision-based drowsiness detection system for railway operators using lightweight convolutional neural networks. *Frontiers in Future Transportation*, *6*, 1677442. https://doi.org/10.3389/ffutr.2025.1677442

Meza, H. (2025). *Implementación de una red neuronal Transformer para detección de somnolencia en conductores de vehículos durante la conducción diurna y nocturna*. https://repositorio.utn.edu.ec/handle/123456789/17024

Minhas, A. A., Jabbar, S., Farhan, M., & Najam ul Islam, M. (2022). A smart analysis of driver fatigue and drowsiness detection using convolutional neural

networks. *Multimedia Tools and Applications 2022 81:19*, *81*(19), 26969–26986. https://doi.org/10.1007/s11042-022-13193-4

Muhammad, R., Hikmat, K., Shahid, A., Amina, I., Mahwish, I., & Ahsan, M. (2019). A Survey on State-of-the-Art Drowsiness Detection Techniques. *IEEEA*. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8704263

Naciones Unidas. (2023). *Objetivos de Desarrollo Sostenible*. https://www.un.org/sustainabledevelopment/es/infrastructure/

Onososen, A. O., Musonda, I., Onatayo, D., Saka, A. B., Adekunle, S. A., & Onatayo, E. (2025). Drowsiness Detection of Construction Workers: Accident Prevention Leveraging Yolov8 Deep Learning and Computer Vision Techniques. *Buildings*, *15*(3). https://doi.org/10.3390/buildings15030500

OPS, & OMS. (2018). *Nuevo informe de la OMS destaca que los progresos han sido insuficientes en abordar la falta de seguridad en las vías de tránsito del mundo*. https://www3.paho.org/hq/index.php?option=com_content&view=article&id=14857:new-who-report-highlights-insufficient-progress-to-tackle-lack-of-safety-on-the-world-s-roads&Itemid=0&lang=es#gsc.tab=0

Ormeño, N. (2019). *ISO 25010 y el desarrollo de software*. https://normeno.medium.com/iso-25010-y-el-desarrollo-de-software-112393a4b341

Pandey, N. N., & Muppalaneni, N. B. (2023). Dumodds: Dual modeling approach for drowsiness detection based on spatial and spatio-temporal features. *Engineering Applications of Artificial Intelligence*, *119*, 105759. https://doi.org/10.1016/j.engappai.2022.105759

Rosales Mayor, E., & De Castro Mujica, J. R. (2010). Somnolencia: Qué es, qué la causa y cómo se mide. *Acta Médica Peruana*, *27*(2), 2010.

Sapkota, R., & Karkee, M. (2025). *Ultralytics YOLO Evolution: An Overview of YOLO26, YOLO11, YOLOv8 and YOLOv5 Object Detectors for Computer Vision and Pattern Recognition*. http://arxiv.org/abs/2510.09653

Scanbot SDK. (2022). *YOLO object detection and its applications in computer vision*. https://www.linkedin.com/pulse/yolo-object-detection-its-applications-computer-vision-scanbotsdk

Secretaría Nacional de Planificación. (2021). *Plan de Creación de Oportunidades 2021-2025*. https://www.planificacion.gob.ec/wp-content/uploads/2021/09/Plan-de-Creacio%CC%81n-de-Oportunidades-2021-2025-Aprobado.pdf

Sharafkhaneh, A., & Hirshkowitz, M. (2022). Evaluating sleepiness and fatigue. In *Principles and Practice of Sleep Medicine* (7th ed., 207).

Siddique, S., Islam, S., Neon, E., Sabbir, T., Naheen, T., & Khan, R. (2023). Deep Learning-based Bangla Sign Language Detection with an Edge Device. *Intelligent Systems with Applications*, *18*, 2667–3053. https://doi.org/10.1016/j.iswa.2023.200224

Soto Serrano, A. (2017). *YOLO Object Detector for Onboard Driving Images*. moz-extension://54f0a68a-0faf-4601-9971-200217a67bc7/enhanced-reader.html?openApp&pdf=https%3A%2F%2Fddd.uab.cat%2Fpub%2Ftfg%2F2017%2Ftfg_71066%2Fpaper.pdf

Syed, Z. (2023). *Principles Of YoloV8*. https://medium.com/@syedzahidali969/principles-of-yolov8-6a90564e16c3

Timarán, S., Hernández, I., Caicedo, S., Hidalgo, A., & Alvarado, J. (2016). *Descubrimiento de patrones de desempeño académico con árboles de decisión en las competencias genéricas de la formación profesional*. 63–86. https://doi.org/10.16925/9789587600490

Torres, C. (2022). *Detección temprana del sueño a partir del comportamiento de los ojos y boca*. Universidad del Rosario.

Uvidia, M. (2016). *Descubrimiento de conocimiento en base de datos para la toma de decisiones en la unidad de nivelación y admisión de la ESPOCH* [Pontificia Universidad Católica del Ecuador Sede Ambato]. https://repositorio.pucesa.edu.ec/bitstream/123456789/1601/1/76134.pdf

Wu, J. Da, & Chang, C. H. (2022). Driver Drowsiness Detection and Alert System Development Using Object Detection. *Traitement Du Signal*, *39*(2), 493–499. https://doi.org/10.18280/ts.390211

Wu, Y., Jiang, X., Guo, Y., Zhu, H., Dai, C., & Chen, W. (2023). Physiological measurements for driving drowsiness: A comparative study of multi-modality feature fusion and selection. *Computers in Biology and Medicine*, *167*. https://doi.org/10.1016/j.compbiomed.2023.107590

ANEXOS

Repositorio de códigos fuente:

https://github.com/Alexis2701/Deteccion-Somnolencia-YOLO