UNIVERSIDAD NACIONAL DE SAN ANTONIO ABAD DEL CUSCO

FACULTAD DE INGENIERÍA ELÉCTRICA, ELECTRÓNICA, INFORMÁTICA Y MECÁNICA

ESCUELA PROFESIONAL DE INGENIERÍA ELECTRÓNICA

TESIS

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DETECTOR DE SOMNOLENCIA EN TIEMPO REAL MEDIANTE VISIÓN COMPUTACIONAL USANDO REDES NEURONALES CONVOLUCIONALES APLICADO A CONDUCTORES

Presentado Por:

Br. Ruben Dario Florez Zela

Para optar el título profesional de:

Ingeniero Electrónico

Asesor:

Dr. Ing. Facundo Palomino Quispe

Cusco - Perú

2024

INFORME DE ORIGINALIDAD

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

El que suscribe, **Asesor** del trabajo de investigación/tesis titulada: DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DETECTOR DE SOMNOLENCIA EN TIEMPO REAL MEDIANTE VISIÓN COMPUTACIONAL USANDO REDES NEURONALES CONVOLUCIONALES APLICADO A CONDUCTORES

presentado por: Ruben Dario Florez Zela con DNI Nro.: 72038792

presentado por: ..... con DNI Nro.: .....

para optar el título profesional/grado académico de Ingeniero Electrónico

.....

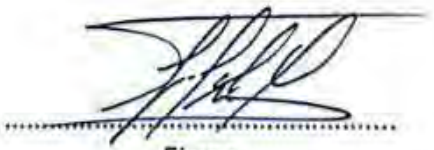
Informo que el trabajo de investigación ha sido sometido a revisión por 2 veces, mediante el Software Antiplagio, conforme al Art. 6° del ***Reglamento para Uso de Sistema Antiplagio de la UNSAAC*** y de la evaluación de originalidad se tiene un porcentaje de 8 %.

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

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

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

Cusco, 11 de enero de 2024



Firma

Post firma Dr. Ing. Facundo Palomino Quispe

Nro. de DNI 00435194

ORCID del Asesor 0000-0002-5947-6682

Se adjunta:

1. Reporte generado por el Sistema Antiplagio.
2. Enlace del Reporte Generado por el Sistema Antiplagio: https://unsaac.turnitin.com/viewer/submissions/oid:27259:303174006?locale=es-MX

NOMBRE DEL TRABAJO
Tesis_Ruben_Dario_Florez_Zela_compressed.pdf

AUTOR
Ruben Dario Florez Zela

RECUENTO DE PALABRAS
50329 Words

RECUENTO DE CARACTERES
260857 Characters

RECUENTO DE PÁGINAS
247 Pages

TAMAÑO DEL ARCHIVO
3.8MB

FECHA DE ENTREGA
Jan 11, 2024 11:57 AM GMT-5

FECHA DEL INFORME
Jan 11, 2024 12:00 PM GMT-5

8% de similitud general

El total combinado de todas las coincidencias, incluidas las fuentes superpuestas, para cada base d

- 6% Base de datos de Internet
- Base de datos de Crossref
- 4% Base de datos de trabajos entregados
- 1% Base de datos de publicaciones
- Base de datos de contenido publicado de Crossr

Excluir del Reporte de Similitud

- Material bibliográfico
- Material citado
- Bloques de texto excluidos manualmente
- Material citado
- Coincidencia baja (menos de 10 palabras)

Dedicatoria

La presente investigación se la dedico a mis padres Ruben y Maribel, quienes me brindan su apoyo y amor incondicional, y ser los arquetipos de mi vida; aprendiendo que todo lo que uno se propone, se logra con esfuerzo. A mi hermana, por sus palabras de aliento y apoyo para culminar mis estudio. A mi asesor y amistades, quienes me brindaron motivación constante a lo largo de esta investigación.

Doy gracias a Dios, a la vida y a mis padres por cada nuevo triunfo.

Ruben Dario Florez Zela

Agradecimientos

Quiero expresar mi sincero agradecimiento a todas las personas que hicieron posible la culminación de este proyecto de tesis. En primer lugar, a mis padres y hermana, por su sacrificio y dedicación para brindarme una educación sólida y por ser modelos de esfuerzo y perseverancia.

Agradezco profundamente a mi asesor, Dr. Ing. Facundo Palomino, por su orientación, su paciencia y su compromiso con mi crecimiento académico. Y a la Dr. Ing. Ana Beatriz Alvarez, sus conocimientos y consejos fueron fundamentales en el desarrollo de este trabajo.

A los integrantes del laboratorio LIECAR, mis amigos, quienes compartieron conmigo momentos de estudio, reflexión y diversión. Gracias por su compañía y por ser un gran apoyo en este viaje.

A mis docentes de la Escuela Profesional de Ingeniería Electrónica, quienes compartieron sus vastos conocimientos y valiosas experiencias, contribuyendo significativamente a mi crecimiento académico y profesional.

Y a todas las personas que tuve el privilegio de conocer durante el desarrollo de esta investigación. Sus palabras de aliento y sabios consejos no solo me brindaron apoyo en el camino de la investigación, sino que también me inspiraron a seguir adelante con renovado entusiasmo.

Presentación de la investigación

Durante el desarrollo de la presente investigación, se logró la participación en varios eventos académicos y científicos, que se detallan a continuación en orden cronológico.

- La investigación fue presentada durante el "MES DE LA CIENCIA, ARTE Y CULTURA" en el marco del "II CONGRESO INTERNACIONAL DE INVESTIGACIÓN, INNOVACIÓN Y EMPRENDIMIENTO 2022" organizado por la Universidad Nacional de San Antonio Abad del Cusco, donde obtuve el premio al 1° puesto por la participación con la mejor presentación en categoría STAND con el título **"Sistema Detector de Somnolencia a Conductores en Tiempo Real Mediante Vision por Computador e Inteligencia Artificial"**.

  9 y 10 de Noviembre de 2022.

- La investigación se presentó en forma de seminario en el evento "VII Seminario de Actualización en Electrónica Industrial y Macatrónica Automotriz" organizada por la IESTP Enrique Pablo Mejia Tupayachi (EPAMET), participando como ponente con el título **"Sistema detector de somnolencia en conductores de transporte mediante redes neuronales"**.

  Viernes 26 de Mayo de 2023, 10:00 - 11:00 a.m.

- La investigación se presentó como articulo científico en el **journal Q2** de alto impacto "Applied Sciences", en el numero especial "Computer Vision and Pattern Recognition Based on Deep Learning" bajo el título **"A CNN-Based Approach for**

Driver Drowsiness Detection by Real-Time Eye State Identification" con DOI: app13137849.

Received: 1 June 2023 / Revised: 29 June 2023 / Accepted: 30 June 2023 / Published: 4 July 2023.

- La investigación fue presentada como prototipo en el evento "VIII Semana y Feria de Ciencia Tecnología e Innovación YACHAYNINCHIS CUSCO – 2023" organizado por el Gobierno Regional del Cusco, Oficina de Cooperación Técnica Internacional (OCTI Cusco) y CORCYTEC, bajo el título **"SISTEMA DETECTOR DE SOMNOLENCIA A CONDUCTORES DE UNIDADES VEHICULARES EN TIEMPO REAL MEDIANTE VISIÓN EMBEBIDA E INTELIGENCIA ARTIFICIAL"**.

  7 y 8 de Setiembre de 2023.

Gracias a estas contribuciones, y en particular a la publicación de la tesis como artículo científico, logre el reconocimiento como **"Investigador Renacyt Nivel VII"** por parte del Concytec. Este logro destaca la relevancia y el impacto de la investigación en la comunidad científica y académica, consolidando así un valioso reconocimiento.

Resumen

Actualmente, la seguridad vial es una preocupación global importante debido a los accidentes de tráfico causados por la fatiga del conductor, que representan una de las principales amenazas en las carreteras y provocan la pérdida de vidas, especialmente en países en desarrollo. Para abordar este desafío, esta investigación propone un sistema portátil que detecta la somnolencia del conductor en tiempo real al analizar el estado de la boca y los ojos. El sistema utiliza hardware NVIDIA Jetson Nano junto con una cámara de infrarrojo cercano (NIR). Detecta bostezos mediante el uso del Mouth Aspect Ratio (MAR) y la somnolencia visual a través de redes neuronales convolucionales (CNN) centradas en la región ocular, con una técnica de corrección de la región de interés (ROI) de los ojos. Se evaluaron tres CNN (InceptionV3, VGG16 y ResNet50V2) con transfer learning, así como dos arquitecturas propuestas (DD-AI y DD-AI-G). Las pruebas con voluntarias en condiciones simuladas y reales de conducción demostraron el buen rendimiento del prototipo. La red DD-AI-G sobresalió en las pruebas simuladas y alcanzó un promedio del 91.48 % de precisión y un 86.28 % de tasa de detección de somnolencia visual en un entorno de conducción real, ejecutada en el hardware NVIDIA Jetson Nano.

Palabras clave: sistema detector de somnolencia; detección de bostezos; relación de aspecto de la boca (MAR); detección de somnolencia; red neuronal convolucional (CNN); NVIDIA Jetson Nano.

Abstract

Today, road safety is a major global concern due to traffic accidents caused by motorist's running away, which represent one of the main threats on the roads and cause loss of lives, especially in developing countries. To address this challenge, this research proposes a portable system that detects the motorist's loudness in real time by analyzing the state of the mouth and eyes. The system uses NVIDIA Jetson Nano hardware along with a near infrared (NIR) camera. It detects mouths using mouth aspect ratio (MAR) and visual loudness using convolutional neural networks (CNNs) centered on the odor region with a region of interest (ROI) correction technique for odors. Three CNNs (InceptionV3, VGG16 and ResNet50V2) with transfer learning as well as two proposed architectures (DD-AI and DD-AI-G) were tested. Tests with volunteers in simulated and real steering conditions demonstrated the good performance of the prototype. The DD-AI-G network excelled in simulated tests and achieved an average 91.48 % accuracy and 86.28 % visual sound detection rate in a real steering environment, running on NVIDIA Jetson Nano hardware.

Keywords: drowsiness detection system; yawning detection; mouth aspect ratio (MAR); drowsiness detection; convolutional neural network (CNN); NVIDIA Jetson Nano.

Introducción

En esta investigación se ha diseñado y desarrollado un sistema detector de somnolencia basado en visión computacional y redes neuronales convolucionales (CNN). El objetivo principal de este proyecto fue crear una solución eficaz y versátil para detectar signos de somnolencia en conductores vehiculares en la región de Cusco. El trabajo se estructuró en seis capítulos que abarcaron diversos aspectos del proyecto.

En el Capítulo 1, se presentan los aspectos generales de la investigación, destacando la importancia de abordar el problema de la somnolencia en la conducción y su relevancia en la seguridad vial. Definiendo sus objetivos, alcance y metodología de trabajo.

El Capítulo 2 proporciona el marco teórico necesario para comprender los conceptos y tecnologías clave involucrados en el sistema detector de somnolencia. Se exploran temas como la visión computacional, las redes neuronales convolucionales y los sistemas de alerta de somnolencia existentes.

En el Capítulo 3, se presenta el sistema propuesto con una definición clara de sus requerimientos generales, así como los requerimientos específicos en cuanto a diseño (software) e implementación (hardware). Este capítulo también expone la lógica del sistema, destacando los subsistemas que lo integran y su interacción.

El Capítulo 4 se centra en el diseño del sistema, detallando las arquitecturas de las redes neuronales convolucionales empleadas, con especial énfasis en la presentación de dos arquitecturas propuestas, denominadas DD-AI y DD-AI-G. Además, en este capítulo se propone una técnica de corrección aplicada a la región de interés (ROI) correspondiente a la zona de los ojos, contribuyendo así al estado del arte.

En el Capítulo 5, aborda implementación del sistema, donde se detallan aspectos técnicos cruciales para el proyecto. Analizando el sistema embebido de acuerdo a los requerimientos establecidos, se profundiza en la configuración e implementación del hardware y los componentes fundamentales del sistema.

Finalmente, el Capítulo 6 se enfoca en las pruebas y resultados obtenidos. Se describen las evaluaciones realizadas en condiciones simuladas y en un entorno real de conducción en Cusco. Los datos recopilados y el desempeño del sistema se analizan en detalle.

Esta investigación tiene como objetivo proporcionar una solución efectiva y localizada para la detección de somnolencia en conductores vehiculares, contribuyendo así a la seguridad vial en la región de Cusco, Perú, y sentando las bases para futuros desarrollos en este campo.

Índice general

Agradecimientos II

Presentación de la Investigación IV

Resumen V

Abstract VI

Introducción VIII

Índice de figuras XVII

Índice de tablas XXI

1 Aspectos generales **1**

1.1 Planteamiento del problema 1

1.1.1 Problema general 4

1.1.2 Problemas específicos 4

1.2 Justificación 5

1.2.1 Justificación social 5

1.2.2 Justificación económica 5

1.2.3 Justificación ambiental 5

1.2.4 Justificación tecnológica 6

1.2.5 Justificación profesional/académico y personal 6

1.3 Objetivos 7
1.3.1 Objetivo general 7
1.3.2 Objetivos específicos 7
1.4 Hipótesis 7
1.4.1 Hipótesis general 7
1.4.2 Hipótesis específicas 8
1.5 Variables 8
1.5.1 Variable independiente 8
1.5.2 Variable dependiente 8
1.6 Alcances y limitaciones 8
1.6.1 Alcances 8
1.6.2 Limitaciones 9

2 Marco teórico **11**
2.1 Antecedentes 11
2.1.1 Tesis internacionales 11
2.1.2 Tesis nacionales 13
2.1.3 Tesis regionales 16
2.1.4 Artículos científicos relacionados 16
2.2 Base teórica 20
2.2.1 Estudio de la somnolencia 20
2.2.1.1 Principales factores 20
2.2.1.2 Características visuales de la somnolencia 20
2.2.2 Sistema avanzado de asistencia a la conducción (ADAS) 21
2.2.3 Sistema de monitoreo del conductor (DSM) 22
2.2.3.1 Sistemas comerciales DSM 22
2.2.3.2 Limitaciones de los sistemas comerciales DSM 26
2.2.4 Métodos de detección de somnolencia 26

2.2.4.1 Métodos basados en imágenes 27
2.2.4.2 Métodos basados en datos de bioseñales 28
2.2.4.3 Métodos basados en los vehículos 28
2.2.4.4 Métodos híbridos 29
2.2.5 Métodos basados en el análisis de expresiones faciales mediante imágenes 29
2.2.5.1 Relación de aspecto ocular (EAR) 29
2.2.5.2 Relación de aspecto bucal (MAR) 30
2.2.6 Visión computacional 31
2.2.6.1 Ventajas de la visión computacional 32
2.2.6.2 Aplicaciones de la visión computacional 33
2.2.6.3 Componentes de la visión computacional 33
2.2.7 Visión computacional en el espectro electromagnético 33
2.2.7.1 Métodos en el espectro visible (RGB) 34
2.2.7.2 Métodos en el infrarrojo cercano (NIR) 35
2.2.7.2.1 Aplicaciones NIR por longitud de onda 37
2.2.8 Imagen digital 37
2.2.8.1 Píxel 38
2.2.9 Procesamiento digital de imágenes (PDI) 39
2.2.9.1 Conversión a escala de grises 39
2.2.9.2 Escalamiento 40
2.2.9.3 Región de interés (ROI) 41
2.2.10 Detección de rostros y puntos faciales 41
2.2.10.1 Haar cascade 42
2.2.10.1.1 Haar cascades pre-entrenados 43
2.2.10.2 Dlib 44
2.2.10.2.1 Dlib face detection: 44
2.2.10.2.2 Dlib face landmark: 45

2.2.10.3 MediaPipe 45
2.2.10.3.1 MediaPipe face detector 46
2.2.10.3.2 Mediapipe face mesh 46
2.2.11 Deep learning y redes neuronales profundas 46
2.2.11.1 Métricas de evaluación 48
2.2.12 Redes neuronales convolucionales (CNN) 50
2.2.12.1 Capa de convolución 51
2.2.12.2 Capa de agrupación (pooling) 52
2.2.12.3 Capa totalmente conectada 52
2.2.13 Arquitecturas de CNNs 53
2.2.13.1 InceptionV3: 53
2.2.13.2 VGG-16: 54
2.2.13.3 ResNet50: 54
2.2.14 Transfer learning 55
2.2.15 Base de datos 55
2.2.15.1 Driver drowsiness detection dataset 55
2.2.15.2 UTA real-life drowsiness dataset 56
2.2.15.3 Night-time yawning-microsleep-eyeblink-driver distraction . 57
2.2.16 Cámara RGB-NIR 57
2.2.16.1 ELP-USB3MP01H 58
2.2.16.2 ELP-USBFHD05MT-KL36IR 58
2.2.16.3 ELP-USBFHD08S-KV100IR 59
2.2.17 Sistema embebido 59
2.2.17.1 Raspberry Pi 60
2.2.17.2 NVIDIA Jetson Nano 60
2.2.18 Herramientas computacionales 61
2.2.18.1 Python: 61

2.2.18.2 OpenCV: . . . 62
2.2.18.3 Tensorflow: . . . 62
2.2.18.4 ONNX: . . . 63

3 Planteamiento del sistema 64

3.1 Requerimientos del sistema . . . 64
3.1.1 Requerimientos generales . . . 65
3.1.2 Requerimientos de diseño (software) . . . 65
3.1.3 Requerimientos de implementación (hardware) . . . 66
3.2 Lógica del sistema . . . 66
3.3 Subsistemas del sistema detector de somnolencia . . . 68
3.3.1 Subsistema de monitoreo . . . 69
3.3.2 Subsistema de alerta y alarma . . . 69
3.3.3 Subsistema de precaución . . . 70

4 Diseño del sistema detector de somnolencia 71

4.1 Elección de elementos para el diseño del sistema . . . 71
4.1.1 Elección de la cámara . . . 72
4.1.2 Elección de la base de datos . . . 74
4.1.3 Elección del método de extracción de características faciales . . . 75
4.2 Método del diseño del sistema de detección de somnolencia . . . 77
4.2.1 Fase 1: Adquisición de imágenes . . . 78
4.2.2 Fase 2: Detección de puntos faciales . . . 78
4.2.3 Fase 3: Selección de ROI . . . 79
4.2.4 Fase 4: Extracción de ROI . . . 85
4.2.5 Fase 5: Método aplicado . . . 86
4.2.5.1 Método por relación de aspecto bucal (MAR) . . . 86
4.2.5.2 Método por redes neuronales convolucionales (CNN) . . . 90

4.2.5.2.1 Extracción de frames: . . . 92
4.2.5.2.2 Conversión a gris: . . . 93
4.2.5.2.3 Generación del dataset: . . . 95
4.2.5.2.4 Entrenamiento de las CNNs: . . . 96
4.2.5.2.5 Resultados de los entrenamientos: . . . 101
4.2.5.2.6 Prueba de las CNNs entrenadas: . . . 107
4.2.5.2.7 Resultados visuales de las CNNs: . . . 111
4.2.5.2.8 Identificación de somnolencia visual: . . . 114
4.2.5.2.9 Resultados generales de las CNNs: . . . 116
4.2.6 Fase 6: Activación de alarma . . . 117
4.3 Diagrama de flujo del diseño del sistema . . . 117
4.3.1 Subdiagramas del diagrama principal . . . 120
4.4 Diagramas de flujo complementarios . . . 124

5 Implementación del sistema detector de somnolencia 126

5.1 Elección de elementos para la implementación del sistema . . . 127
5.1.1 Elección del sistema embebido . . . 127
5.1.2 Elección de la pantalla . . . 130
5.1.3 Elección de la fuente de alimentación . . . 131
5.2 Método de la implementación del sistema de detección de somnolencia . . . 132
5.2.1 Subsistema de monitoreo . . . 134
5.2.1.1 Sistema embebido y procesamiento de datos . . . 134
5.2.1.1.1 Configuración del hardware: . . . 134
5.2.1.1.2 Compatibilidad de modelos CNNs entrenados: . . . 140
5.2.1.2 Equipo de adquisición de datos . . . 141
5.2.2 Subsistema de alerta y alarma . . . 142
5.2.2.1 Visualización . . . 142
5.2.2.2 Alerta . . . 143

5.2.3 Subsistema de precaución 145
5.2.4 Bloque de suministro energético 146
5.3 Construcción del sistema de detección de somnolencia 149

6 Pruebas y resultados 154
6.1 Pruebas de los componentes 154
6.1.1 Prueba del sistema embebido y pantalla LCD 155
6.1.2 Prueba de la cámara NIR 155
6.2 Pruebas y resultados de los modelos CNNs en condición simulada 155
6.2.1 Pruebas en el entorno de la computadora 156
6.2.2 Resultados en el entorno de la computadora 157
6.2.3 Pruebas en el sistema embebido 159
6.2.4 Resultados en el sistema embebido 160
6.3 Elección de la red CNN 166
6.4 Pruebas del sistema en un entorno real 166
6.4.1 Instalación del sistema en el vehículo 166
6.4.2 Pruebas de funcionamiento del sistema en el vehículo 168
6.4.2.1 Pruebas diurnas 168
6.4.2.2 Pruebas en la tarde 169
6.4.2.3 Pruebas nocturnas 170
6.5 Detección de fallos del sistema 170
6.6 Resultados de las pruebas del sistema en un entorno real 171
6.6.1 Resultados del sistema en voluntaria 1 (señora conductora) 171
6.6.2 Resultados del sistema en voluntaria 2 (señorita conductora) 174
6.7 Resultados generales del sistema 176

Discusión 179

Conclusiones 181

Recomendaciones **183**

Abreviaciones y Acrónimos **184**

Referencias **194**

Anexos **195**

Índice de figuras

1.1 Tendencia de accidentes de tránsito 2010-2020. 2
1.2 Factores que interviene en siniestros viales, 2019-2022. 3

2.1 sistema de vigilancia del estado de conductor - STONKAM. 23
2.2 Sistema GUARDIAN - Seeing Machines Latin America. 23
2.3 Monitor de fatiga del conductor MR688 - CareDrive. 24
2.4 Medidas de detección de la somnolencia del conductor. 27
2.5 Puntos faciales para EAR. 30
2.6 Puntos faciales para MAR. 31
2.7 similitud entre sistema de visión humana y sistema de visión computacional. 32
2.8 Rango de espectro electromagnético (UV - IR). 34
2.9 Toma de ojos en el espectro visible. 35
2.10 Toma de ojos en el infrarrojo cercano a 850 nm. 36
2.11 Representación de una imagen digital. 38
2.12 Píxel de una imagen digital. 38
2.13 Escalamiento de una imagen. 40
2.14 ROI de una imagen. 41
2.15 Detectores Haar. 43
2.16 Detección de características con Haar cascade. 44
2.17 Detección de características con Dlib face landmark. 45
2.18 Detección de características con mediapipe face mesh. 46
2.19 Representación de una ANN. 47

2.20 Representación de una CNN. 50
2.21 Proceso de convolución. 51
2.22 Proceso de pooling. 52
2.23 Arquitectura de InceptionV3. 53
2.24 Arquitectura de VGG-16. 54
2.25 Arquitectura de ResNet50. 55
2.26 Imagen referencial del dataset NTHU. 56
2.27 Imagen referencial del dataset UTA. 56
2.28 Imagen referencial del dataset NITYMED. 57
2.29 Cámara Web ELP-USB3MP01H. 58
2.30 Cámara Web ELP-USBFHD05MT-KL36IR. 58
2.31 Cámara Web ELP-USBFHD08S-KV100IR. 59
2.32 Raspberry Pi 4 modelo B. 60
2.33 NVIDIA Jetson Nano. 61

3.1 Lógica general del sistema - propuesta híbrida para la detección de somnolencia. 68
3.2 Subsistemas que conforman del sistema detector de somnolencia. 68

4.1 Adquisición de imágenes. 78
4.2 Detección de puntos faciales con MediaPipe. 79
4.3 Corrección de ROI de la zona de los ojos. 81
4.4 ROIs oculares corregidos en diferentes posturas de la cabeza del conductor. . 84
4.5 ROIs de los ojos y la boca del estado del conductor. 85
4.6 Extracción de los puntos de la boca. 86
4.7 Índices de bostezo. 88
4.8 Procedimiento de entrenamiento mediante CNN. 92
4.9 Conversión a escala de grises. 94
4.10 Ejemplos aleatorios de muestras del conjunto de datos. 96

4.11 Arquitectura propuesta con transfer learning. 98
4.12 Arquitectura propuesta DD-AI. 100
4.13 Arquitectura propuesta DD-AI-G. 100
4.14 Resultados Accuracy y Loss de las CNNs. 102
4.15 Matriz de confusión de las CNNs en validación. 103
4.16 Boxplot de dos métricas de evaluación para la validación. 106
4.17 Comportamiento radial de dos métricas para la validación. 106
4.18 Matriz de confusión de las CNNs en testeo. 108
4.19 Boxplot de dos métricas de evaluación para el testeo. 110
4.20 Comportamiento radial de dos métricas para el testeo. 110
4.21 Visualización de las CNNs con Grad-CAM. 113
4.22 Diagrama de flujo del diseño del sistema. 119
4.23 Diagrama de flujo de detección de puntos faciales. 120
4.24 Diagrama de flujo de selección de ROI. 121
4.25 Diagrama de flujo de extracción de ROI. 122
4.26 Diagrama de flujo de redimensionamiento de ROI. 122
4.27 Diagrama de flujo de inferencia del modelo. 123
4.28 Diagrama de flujo de determinación de puntos. 123
4.29 Diagrama de flujo de calculo por MAR. 124
4.30 Diagrama de flujo de la somnolencia visual. 125
4.31 Diagrama de flujo del bostezo. 125

5.1 Esquema del sistema detector de somnolencia. 133
5.2 Actualización del sistema operativo - sistema embebido. 135
5.3 Instalación de memoria swap - sistema embebido. 136
5.4 Instalación de MediaPipe - sistema embebido. 138
5.5 Instalación de OpenCV - sistema embebido. 139
5.6 Esquema del subsistema de monitoreo. 142

5.7 Interfaz gráfica para la visualización del estado del conductor. 143
5.8 Diagrama de bloques del control de la alerta. 144
5.9 Esquema del subsistema de alerta y alarma. 145
5.10 Esquema del bloque de suministro energético. 149
5.11 Componentes del sistema detector de somnolencia. 150
5.12 Puertos asignados del Jetson Nano. 151
5.13 Fuente adaptada para el sistema. 152
5.14 Diagrama de conexiones eléctricas del sistema. 152
5.15 Ensamblaje del sistema detector de somnolencia. 153

6.1 Pruebas en el entorno de la computadora. 157
6.2 Porcentaje de acierto de las cinco CNN en los tres escenarios - computadora. 159
6.3 Porcentaje promedio de las cinco CNN en los tres escenarios - computadora. 159
6.4 Porcentaje de acierto de las cinco CNN en los tres escenarios - jetson nano. . 162
6.5 Porcentaje promedio de las cinco CNN en los tres escenarios - jetson nano. . 162
6.6 Consumo de memoria RAM de las cinco CNNs. 163
6.7 Consumo de GPU de las cinco CNNs. 164
6.8 Consumo de potencia del sistema con las cinco CNNs. 165
6.9 Comprobación del suministro energético. 167
6.10 Sistema instalado en el vehículo. 167
6.11 Ruta de las pruebas. 168
6.12 Pruebas diurnas del sistema. 169
6.13 Pruebas en la tarde del sistema. 169
6.14 Pruebas nocturnas del sistema. 170
6.15 Casos de falsos positivos y falsos negativos. 171
6.16 Tendencia de Accuracy del sistema - voluntaria 1. 174
6.17 Tendencia de Accuracy del sistema - voluntaria 2. 176

Índice de tablas

2.1 Tecnología DSM en compañías de autos. 25
2.2 Aplicaciones NIR por longitud de onda. 37
2.3 Matriz de confusión. 49

4.1 Comparación de las tres cámaras RGB-NIR. 72
4.2 Comparación de las bases de datos. 74
4.3 Comparación de los métodos de extracción de características faciales. 76
4.4 Distribución del dataset. 95
4.5 Parámetros de entrenamiento. 101
4.6 Métricas de evaluación en los datos de validación. 105
4.7 Métricas de evaluación en los datos de testeo. 109
4.8 Resultados generales de las CNNs. 116
4.9 Resultados de consumo de memoria RAM y GPU de las CNNs. 117

5.1 Comparación de los dos sistemas embebidos. 128
5.2 Comparación de los modelos h5 y onnx. 141
5.3 Cálculo de consumo de potencia de los componentes del sistema. 147

6.1 Resultado de acierto en entorno de la computadora. 158
6.2 Resultado de acierto en el sistema embebido. 161
6.3 Resultado de las pruebas diurnas - voluntaria 1. 172
6.4 Resultado de las pruebas en la tarde - voluntaria 1. 172
6.5 Resultado de las pruebas nocturnas - voluntaria 1. 173

6.6 Resultado de las pruebas diurnas - voluntaria 2. 174
6.7 Resultado de las pruebas en la tarde - voluntaria 2. 175
6.8 Resultado de las pruebas nocturnas - voluntaria 2. 175
6.9 Promedio Accuracy y Recall - voluntaria 1. 177
6.10 Promedio Accuracy y Recall - voluntaria 2. 177
6.11 Promedio General del sistema en Accuracy y Recall 177

Capítulo 1

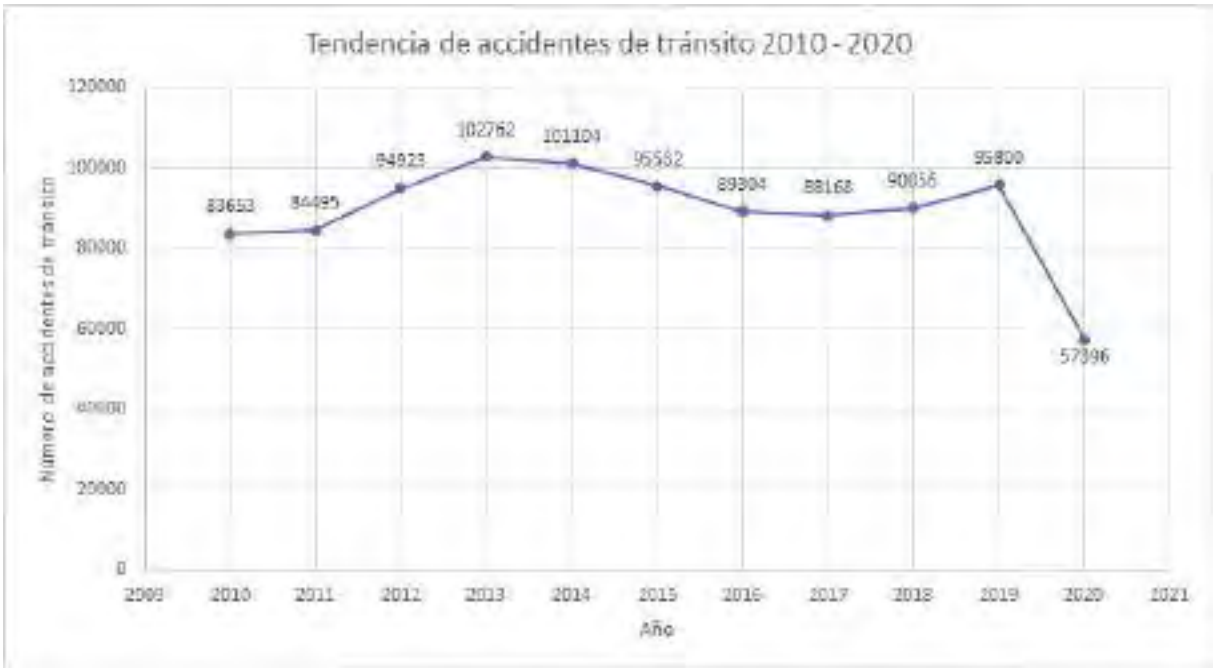
Aspectos generales

1.1. Planteamiento del problema

Según la Organización Mundial de la Salud (OMS), "Cada año se pierden aproximadamente 1,3 millones de vidas a consecuencia de estas lesiones. Entre 20 y 50 millones de personas sufren traumatismos no mortales, y muchos de ellos provocan una discapacidad". Se estima que 13 millones de personas más morirán y 500 millones más resultarán heridas en la próxima década, especialmente en países de bajos y medianos ingresos. Estos números no son aceptables en términos absolutos o relativos. Los accidentes automovilísticos son la principal causa de muerte en el mundo, a pesar de que todas las muertes y lesiones son evitables (OMS, 2022).

En el Perú, en la estadística de la Policía Nacional del Perú (PNP), se registraron 983193 accidentes correspondiente a la última década (2010 - 2020) (Figura 1.1), donde el año que obtuvo mayor cantidad de siniestros viales fue el 2013 con 102762, seguido del año 2014 con 101104 y el año 2019 con 95800 (PNP, 2020). En el primer semestre del presente año 2022, el 38,53 % de los lesionados por accidentes de tránsito pertenece a Lima, mientras que el 61,47 % pertenece al resto de regiones del Perú (CDC-Perú, 2022).

Figura 1.1: Tendencia de accidentes de tránsito 2010-2020.



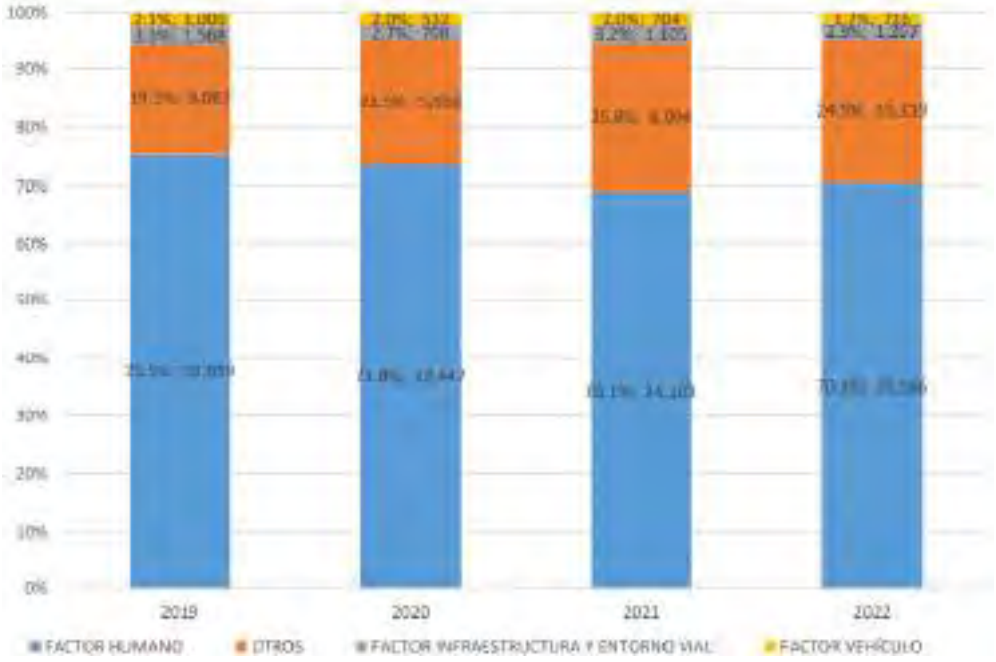
Fuente: PNP (2020).

Según las estadísticas de la Policía Nacional del Perú, entre 2019 y 2022, la región Lima aún presenta la mayor cantidad de accidentes, con un 52,90 % (25184) en 2019, un 48,23 % (19982) en 2022. La región Cusco presento una cantidad de accidentes con un 4.03 % (1920) el 2019, un 4.19 % (1103) el 2020, un 4.21 % (1469) el 2021 y un 4.09 % (1693) el año 2022 durante el primer semestre; siendo Cusco la sexta región con mayor índice de siniestros viales a nivel Nacional (ONSV-Perú, 2022).

Los factores que contribuyeron a la ocurrencia de los siniestros viales fueron: factor humano, factor vehículo, factor infraestructura y entorno vial, causa indetectable y/o poco clara.

Se observa en la Figura 1.2 que, el factor humano concentra las causas de los accidentes de tránsito, desde 75,5 % (35959) en 2019 y 70,4 % (29186) en 2022. En este sentido, en cuanto al factor humano, se demostró que el conductor fue el principal causante de los siniestros viales, siendo para el año 2022 el 93.9 % (27396) de siniestros viales ocasionados por los conductores (ONSV-Perú, 2022).

Figura 1.2: Factores que interviene en siniestros viales, 2019-2022.



Fuente: ONSV-Perú (2022).

Se encontró que las causas dentro del factor humano al conductor son: exceso de velocidad, ebriedad del conductor, imprudencia del conductor, cansancio del conductor. Viendo las causas mencionadas, no es difícil deducir su conexión con el sueño y la fatiga, porque el conductor maneja más rápido cuando está cansado al querer llegar rápido a su destino; un conductor ebrio tiene más probabilidades de quedarse dormido al volante; y un conductor imprudente por lo general elige conducir a pesar de que sabe que está cansado. Así lo observa una publicación del Ministerio de Salud, que muestra que el 30 % de los accidentes en el Perú están relacionados con el cansancio y la somnolencia al volante (MINSA, 2017).

Por lo que, la somnolencia al conducir es un estado de cansancio y agotamiento causado por la falta de sueño, la fatiga, el consumo de alcohol el día anterior, la sobrealimentación antes de conducir o el sobre esfuerzo. Esta y otras causas inhiben la capacidad de respuesta del cuerpo; puede conducir a la tragedia o accidente de tráfico.

Por otra parte, según el objetivo de la Unión Europea, que para el año 2050 no habrá muertes en accidentes de tránsito, donde en el 2022 se dio los primeros pasos para exigir la

inclusión del sistema de seguridad ADAS en todos los vehículos nuevos homologados este año (UE, 2019).

Los sistemas de seguridad ADAS (Advanced Driver Assistance Systems) son asistentes que facilitan la tarea de conducción y reducen el riesgo de accidentes vehiculares, dentro de este se encuentra el sistema DMS (Driver Status Monitoring), que es una ayuda para detectar fatiga, somnolencia, microsueños, comportamientos que distraen y causan accidentes.

Si bien es cierto, un accidente o siniestro vehicular a causa de la somnolencia no se puede evitar por completo, sin embargo, se puede lograr disminuir las probabilidades de ocurrencia de estos.

Considerando que en diversos lugares ya hay prototipos y productos con este fin. Tal es el caso como la implementación de este sistema en los autos automáticos/eléctricos de diversas compañías enfocadas en el mercado Europeo y Asiático, las cuales funcionan en tiempo real utilizando inteligencia artificial. Y productos en el mercado a nivel mundial y Latinoamericano, enfocando la venta a compañías y empresas mineras, flota de transporte, etc., haciendo difícil la adquisición de este sistema para un conductor particular.

1.1.1. Problema general

¿Cómo diseñar e implementar un sistema detector de somnolencia en tiempo real mediante visión computacional usando redes neuronales convolucionales implementado en un sistema embebido robusto, eficiente y rápido para beneficio de los conductores de la ciudad del Cusco?

1.1.2. Problemas específicos

1. ¿Que tipos de arquitecturas de redes neuronales convolucionales son los más adecuados para la detección de somnolencia?.

2. ¿Que metodología de validación usar para evaluar el comportamiento del algoritmo

CNN a ser implementado para la detección de somnolencia?.

3. ¿Cual plataforma de hardware usar para la implementación del sistema embebido en la unidad vehicular?.

4. ¿Cual procedimiento usar para la evaluación del sistema ante la presencia de síntomas de somnolencia en tiempo real para determinar el desempeño del prototipo?.

1.2. Justificación

1.2.1. Justificación social

La presente investigación, puede contribuir en la disminución de ocurrencia de los siniestros viales, ya que este es un factor que perjudica a las personas, dejándolas con secuelas psicológicas como traumas, secuelas físicas como lesiones, discapacitados o incluso causarles la muerte. Esta investigación busca salvaguardar la integridad física del conductor y ocupantes de la unidad vehicular.

1.2.2. Justificación económica

La implementación del sistema propuesto en este trabajo de investigación, puede contribuir en la reducción de gastos económicos, como gastos por parte de las empresas de transporte al evitar pagar las multas ocasionadas por los accidentes, pagar la reparación de sus unidades vehiculares, lo gastos/cuidados hospitalarios a las personas involucradas, gastos a ambientes dañados y también poder evitar la indemnización.

1.2.3. Justificación ambiental

Se sabe que, en la ocurrencia de siniestros viales, también se presenta la contaminación del medio ambiente por parte de vehículos que transportan materiales peligrosos cómo: petroleo,

gasolina u otros inflamables, el cual representa un problema en la limpieza de estos; por otra parte cuando ocurre un siniestro vial, muchas veces quedan restos del accidente en la zona afectando así al medio ambiente tanto a la flora como a la fauna.

En ese sentido, el proyecto de investigación al detectar la somnolencia y alertar al conductor, puede evitar el accidente vehicular que puede ocasionar también una contaminación ambiental.

1.2.4. Justificación tecnológica

En un mundo donde la tecnología avanza a grandes pasos, es necesario buscar una solución a una de las principales causas de siniestros viales como es la somnolencia, haciendo uso de las tecnologías actuales como la inteligencia artificial (IA), el aprendizaje profundo o deep learning (DL), visión computacional o Computer Vision (CV), el procesamiento digital de imágenes (PDI), esta investigación al hacer uso de estas tecnologías, presenta una solución innovadora, como las que se vienen realizando en el ámbito de los carros de conducción autónoma.

1.2.5. Justificación profesional/académico y personal

El presente trabajo de investigación, se justifica desde el punto profesional/académico, debido a que se aplican los conocimientos de ingeniería adquiridos en la E.P. de Ingeniería Electrónica, abarcando los cursos de Inteligencia Artificial, Procesamiento Digital de Imágenes y Sistemas Embebidos a profundidad, pertenecientes al plan curricular.

De igual manera, se justifica desde el punto personal, dado que el presente trabajo de investigación, le brinda al autor, la oportunidad de especializarse en los temas de interés sobre inteligencia artificial y visión computacional y revelar sus habilidades para ganar experiencia en la profesión, cuya investigación también es la base para obtener el titulo profesional de Ingeniero Electrónico.

1.3. Objetivos

1.3.1. Objetivo general

Diseñar e implementar un sistema detector de somnolencia para ser usado en tiempo real mediante visión computacional usando redes neuronales convolucionales implementado en un sistema embebido, eficaz y eficiente.

1.3.2. Objetivos específicos

1. Implementar los tipos de arquitecturas que usan redes neuronales convolucionales para la detección de somnolencia.

2. Validar los resultados obtenidos para evaluar el comportamiento del algoritmo CNN a ser implementado para la detección de somnolencia.

3. Analizar una plataforma de hardware para la implementación del sistema embebido en la unidad vehicular.

4. Realizar pruebas y obtener resultados de funcionamiento para la evaluación del sistema ante la presencia de síntomas de somnolencia en tiempo real para determinar el desempeño del prototipo.

1.4. Hipótesis

1.4.1. Hipótesis general

El diseño e implementación del sistema mediante visión computacional usando redes neuronales convolucionales implementado en un sistema embebido robusto, eficiente y rápido detectará la somnolencia en tiempo real.

1.4.2. Hipótesis específicas

1. La implementación de arquitecturas de redes neuronales convolucionales detectará la somnolencia.

2. La validación de los resultados obtenidos permitirá evaluar el comportamiento del algoritmo CNN para ser implementado en la detección de somnolencia.

3. Con el análisis de una plataforma de hardware, se logrará implementar en un sistema embebido en la unidad vehicular.

4. Las pruebas y resultados de funcionamiento del sistema ante la presencia de síntomas de somnolencia en tiempo real determinarán el desempeño del prototipo.

1.5. Variables

1.5.1. Variable independiente

- Las arquitecturas de las redes neuronales convolucionales.

1.5.2. Variable dependiente

- Desempeño del sistema detector de somnolencia.

1.6. Alcances y limitaciones

1.6.1. Alcances

La investigación contempla el diseño e implementación de un sistema detector de somnolencia a conductores, con el fin de ser un sistema robusto y fácil de emplear en los

vehículos, que detecte y avise a los conductores si presentan inicios de somnolencia/cansancio para poder evitar los siniestros viales.

Haciendo uso de visión computacional o Computer Vision, este permite determinar los rasgos faciales característicos de la somnolencia (estado de los ojos y bostezo) y la detección facial, para poder hacer un análisis de imágenes faciales.

Con el aprendizaje profundo, se puede entrenar una red neuronal convolucional (CNN) que sea capaz de clasificar el estado actual de los ojos cuando el conductor presenta síntomas de somnolencia en los rasgos faciales característicos, esto con el fin de hacer al sistema mas robusto y minimizar los falsos negativos.

Para poder realizar todo el procesamiento del sistema en tiempo real, se necesita ocupar un sistema embebido con características de software y hardware capaces de soportar los algoritmos de visión computacional y aprendizaje profundo.

Con el sistema detector ya implementado, se realizan pruebas a conductores en tiempo real para evaluar el desempeño del sistema y corregir las falencias que se puedan presentar en acción, con el fin de que sea adaptable a cualquier vehículo.

Los resultados obtenidos en la evaluación del sistema, han sido analizados y, ha sido presentado en un articulo científico pubicado en un journal Q2 de alto impacto.

1.6.2. Limitaciones

La investigación esta principalmente enfocado en el diseño e implementación de un sistema detector de somnolencia a conductores, es por ello que esta tesis se limitará en los siguientes aspectos:

- El conjunto de datos para el entrenamiento de la CNN, se hizo respectivamente en regiones de imágenes que contengan ambos ojos.
- En el entrenamiento de la CNN, se hizo respectivamente en un computador con recursos óptimos.

- Para la detección de bostezo, se ha implementado la relación de distancia de puntos para determinar el grado de apertura de la boca.
- Para la presente tesis, no se ha tomado en cuenta la distracción del conductor mediante la estimación de pose de la cabeza.
- En la implementación del sistema detector de somnolencia, este se realizo en un hardware con consumo máximo de potencia de 10W, lo que implica disminuir el rendimiento del sistema.
- En la realización de las pruebas en un entorno real, estas se han realizado en el sector de San Jerónimo.

Capítulo 2

Marco teórico

2.1. Antecedentes

2.1.1. Tesis internacionales

– Suarez Buitrago (2022). **Sistema de detección de somnolencia en conductores de automóviles empleando técnicas de procesamiento de imágenes y machine learning**. [Tesis de grado para optar el título profesional de Ingeniero en Telecomunicaciones, Universidad de Pamplona].

El autor emplea técnicas de machine learning, donde estas técnicas son aplicadas en la clasificación de imágenes para determinar el grado de somnolencia en conductores de automóvil, las técnicas empleadas son: el Algoritmo k-Nearest Neighbor (KNN), el algoritmo Support vector machine (SVM) y una red neuronal convolucional (CNN), empleado en el conjunto de datos denominado MRL EYE Dataset que consta de 84898 imágenes, donde después de realizar los entrenamientos en el conjunto de datos y validando con 1200 imágenes, obtuvo como resultado que, en la predicción en cada modelo de las tres técnicas en la que lo resultados de clasificación favorecieron a la técnica KNN, seguido por la técnica CNN y por último la técnica SVM; luego para probar en tiempo real, uso el clasificador Haar Cascade

para la detección de rostro y ojos, donde el autor concluyo que, la técnica KNN fue la más sensible, la técnica SVM realizaba predicciones en tiempo real de forma correcta, pero en la técnica CNN fue donde se presenció que se trabajaba con mejor robustez en la iluminación.

- Alba Neppas (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**. [Trabajo de grado previo a la obtención del título de Ingeniero en Sistemas Computacionales, Universidad Técnica del Norte].

En la tesis, el autor desarrolla un sistema para la detección y alerta de somnolencia haciendo uso de un raspberry pi 3B+, una cámara web RGB para la obtención de las imágenes, haciendo énfasis en esta, ya que la cámara al poseer una distancia focal de aproximadamente 85mm hacen un mejor acercamiento a la zona de interés (rostro) y descarta el resto de áreas que no tienen relevancia, donde al ubicar los puntos faciales de los ojos mediante Dlib, se hace el procesamiento mediante el índice de aspecto ocular (EAR), para determinar la presencia de somnolencia mediante un umbral establecido, el cual, el autor concluye haber obtenido una exactitud del 88.25 % para ambientes controlados y 87 % para ambientes reales de conducción diurna, donde estos resultados, satisfacen los requisitos de valores de precisión comúnmente aceptados que deben ser iguales o superiores al 85 %, haciendo pruebas en varios voluntarios, en algunos haciendo uso de gorra y lentes; teniendo la limitación el presente trabajo de que, el sistema solo trabaja en modo diurno, donde el autor recomienda hacer uso de una cámara en la banda infrarroja (NIR) para superar ese limitante.

- Montiel et al. (2019). **Sistema embebido de bajo costo para la asistencia al conductor mediante procesamiento de expresiones faciales**. [Tesis para obtener el grado de Maestría en Instrumentación y Control Automático, Universidad Autónoma de Querétaro].

El objetivo de la tesis es centrarse en el desarrollo de un algoritmo de detección de somnolencia que pueden ser utilizados en los sistemas de asistencia al conductor (ADAS), aplicando diversas técnicas y conceptos de visión artificial. Para lograr este objetivo, la autora desarrolló un detector facial basado en descriptores HOG y clasificadores en cascada (Haar Cascade), donde la autora, realiza el entrenamiento para la detección de rostro mediante la función trainCascadeObjectDetector() de MATLAB, la cual soporta la identificación de objetos con ayuda de tres diferentes descriptores: Haar, LBP y HOG, las imágenes usadas en el entrenamiento fueron de 19000 en rostros con ángulos mayores de inclinación y giro, donde estas imágenes se estandarizaron a un tamaño de 576 x 768 píxeles en escala de grises, obteniendo un porcentaje de detección en promedio de 90.65 %; seguido hace uso de Dlib para la extracción de los puntos faciales correspondientes a los ojos y boca, para luego procesarlos mediante la relación de aspecto ocular (EAR) y la relación de aspecto bucal (MAR), finalmente la autora hace la comparación de 3 CPU's del tiempo de procesamiento por cuadro de vídeo en relación con el CPU utilizado a diferentes formatos de píxeles, donde concluye que, el sistema elegido es el raspberry pi 3B, con una entrada de imágenes a 320 x 240 píxeles obteniendo el tiempo promedio de procesamiento de 0.179s.

2.1.2. Tesis nacionales

– Custodio Effio and Tarrillo Nuntón (2021). **Sistema de monitoreo para detección de somnolencia en choferes de rutas interprovinciales de empresas de transporte de Chiclayo**. [Tesis de grado para obtener el título profesional de Ingeniero Electrónico, Universidad Nacional Pedro Ruiz Gallo].

Los autores de la presente tesis, hacen uso del lenguaje de programación Python y la librería de OpenCV para la aplicación en visión artificial, donde, para detectar el rostro y puntos de interés de los ojos, hacen uso de la librería Dlib, Haar Cascade y redes neuronales pre-entrenadas, en el cual, determinan que el uso de Dlib es la mas indicada para su investigación, además, para un correcto funcionamiento y análisis del sistema,

realizan una alineación del rostro a distintos ángulos, con el objetivo de tener la zona de los ojos en una posición central, para luego pasar su análisis mediante la relación de aspecto ocular (EAR), los autores hacen pruebas simuladas a 30 voluntarios en edades que oscilan los 19 a 66 años, sentados a una distancia aproximada de 40 cm a 50 cm de la cámara, obteniendo resultados favorables en condiciones diurnas con buena iluminación, donde concluyen que su sistema tiene una eficacia total de 97.78 %; presentando una limitante en la condición nocturna, el autor hace recomendación del uso de NVIDIA Jetson Nano para la implementación del sistema en un diseño portátil, ya que el sistema demanda una gran cantidad de recursos de procesamiento.

– Yauri Machaca (2021). **Sistema de detección de somnolencia del conductor de vehículo mediante el procesamiento de imágenes usando Matlab**. [Tesis de grado para optar el título profesional de Ingeniero Electrónico con Mención en Telecomunicaciones, Universidad de Ciencias y Humanidades].

Esta tesis propone introducir un sistema de detección de somnolencia en el conductor para reducir los accidentes de tráfico, el sistema consta de una cámara web con una resolución de 1280x720 FHD a 30 fps que se enfocará en la cara del conductor y está conectada a una computadora que ejecuta un algoritmo creado en MATLAB que recibe imágenes en tiempo real y pasa por el procesamiento de imágenes, calculando el número de parpadeos por período durante la conducción y registrando la somnolencia del conductor. Para realizar dicho proceso, la autora hace uso de Viola Jones (Haar Cascade), para la detección de su región de interés que son los ojos, luego realiza la detección del iris con la transformada de Hough, para posteriormente realizar el análisis con la frecuencia de parpadeos durante un determinado tiempo con un umbral establecido de 21 parpadeos por minuto, donde la autora realizo pruebas a 384 conductores de vehículo, en donde 277 conductores de vehículo trabajan en ambos turnos diurno y nocturno, 69 trabajan en el turno diurno y 38 en el turno nocturno, concluyendo que el sistema es efectivo, obteniendo una efectividad de 87.23 % en

la sección del turno diurno, 94.64 % en la sección del turno nocturno y en la sección de ambos turnos se encuentra el 94.74 % en turno diurno y 96.99 % en turno nocturno.

- Arroyo Rodriguez (2021). **Diseño e implementación de sistema de visión artificial para alerta y detección de somnolencia mediante aprendizaje profundo aplicable en conductores de vehículos**. [Tesis de grado para optar el título profesional de Ingeniero Mecatrónico, Universidad Nacional de Trujillo].

El autor de la tesis, indica que los rasgos importantes que presenta la somnolencia son el bostezo, pestañeo y cabeceo, por lo cual las regiones de interés a extraer son la cara, los ojos y la boca, en ese sentido para este propósito hace uso de Haar Cascade. Donde para el entrenamiento de la red neuronal, hizo uso de 4778 imágenes de rostros, 3865 imágenes de ojos y 4705 imágenes de bocas extraídas cuadro por cuadro de algunos videos de la base de datos denominada "A Realistic Dataset and Baseline Temporal Model for Early Drownsiness Detection", En cada una se utilizó el 80 % para entrenamiento y el 20 % del total de imágenes para validar el entrenamiento, dando como resultado en total 3 grupos de modelos de entrenamiento, para este caso el autor uso la red neuronal convolucional del tipo LeNet junto con el optimizador Adam, obteniendo como resultado en la detección de Bostezo un 96 % y en la detección de Pestañeo un 98 %, finalmente el autor implementa todo es sistema en un Raspberry Pi 3 B+, concluyendo con un tiempo de procesamiento aproximadamente de 0.3s y una precisión del 99 % durante la validación de los modelos entrenados, indicando también que, para evitar falsas detecciones de bostezo, añadió medir la duración de bostezo y pestañeo para aumentar la precisión y extraer valores para la determinación del nivel de somnolencia.

2.1.3. Tesis regionales

– Rondón Condori and Paucara Núñez (2013). **Reconocimiento de Somnolencia en Conductores bajo Condiciones Simuladas**. [Tesis de grado para optar el título profesional de Ingeniero Informático y de Sistemas, Universidad Nacional de San Antonio Abad del Cusco].

El objetivo principal de los autores del proyecto es la utilización de algoritmos de detección de objetos para reconocer la somnolencia del conductor, por lo tanto se debe utilizar la información visual del rostro del conductor. Donde usaron una cámara web para capturar imágenes del conductor, evaluando cada fotograma buscando primero detectar un rostro, si se detecta un rostro, entonces se determina el estado de los ojos (abiertos o cerrados), para este caso usaron el algoritmo de detección de objetos de Viola & Jones (Haar Cascade), utilizando la información de posición de los ojos de los últimos 10 cuadros se calcula el porcentaje de ojos cerrados o PERCLOS , para un PERCLOS superior al 40 %, consideraron que el estado de somnolencia del conductor es peligroso y muestran una señal de alarma. Los autores realizaron las pruebas en condiciones simuladas con: luz natural, cámara web de 640x480 píxeles, a una distancia de 40 a 60 cm del conductor, donde obtuvieron un error visual de hasta el 8 % en la clasificación de ojos "abiertos" o "cerrados", concluyendo así con un tiempo promedio de respuesta de 299ms para dar alarma desde que se detectaron los ojos cerrados.

2.1.4. Artículos científicos relacionados

– P Vikranth Reddy, Joshua D'Souza, Shradhya Rakshit, Sahil Bavariya, Priya Badrinath, 2022 **A Survey on Driver Safety Systems using Internet of Things** (P Vikranth Reddy, 2022).

La investigación tiene como objetivo abordar los problemas de los accidentes de tráfico, especialmente aquellos vinculados a la somnolencia del conductor y la conducción distraída,

mediante el desarrollo y perfeccionamiento de métodos de detección. La metodología implica un análisis detallado de 24 estudios que emplean técnicas avanzadas de aprendizaje profundo para identificar la somnolencia y la distracción del conductor, evaluando aspectos como bostezos, movimientos de cabeza, parpadeo y detección de rostros. Los resultados resaltan la existencia de algoritmos eficaces derivados de estos estudios, centrados en señales y mediciones de comportamiento, con énfasis en la importancia de detectar bostezos, movimientos de cabeza, parpadeo y rostros para mejorar la precisión de los sistemas. En las conclusiones, se destaca el desafío persistente de desarrollar un sistema fiable y preciso, reconociendo la necesidad de algoritmos sólidos y adaptables en el contexto del aprendizaje profundo. Se enfatiza la importancia de la investigación continua para superar dificultades y mejorar constantemente la detección de somnolencia en entornos de conducción.

– Ghanta Sai Krishna, Kundrapu Supriya, Jai Vardhan and Mallikharjuna Rao K. (2022). **Vision Transformers and YoloV5 based Driver Drowsiness Detection Framework** (Krishna et al., 2022).

Esta investigación tiene como objetivo principal desarrollar un algoritmo eficaz de detección de somnolencia en conductores para prevenir accidentes de tráfico. Abordando la seria amenaza que representa la somnolencia para la seguridad vial, el enfoque se centra en el uso de técnicas de visión computacional y arquitecturas de aprendizaje profundo, especialmente un nuevo marco basado en transformadores de visión y arquitecturas YoloV5. La metodología implica la utilización de arquitecturas YoloV5 preentrenadas y la introducción de transformadores de visión para la clasificación binaria de imágenes. El modelo alcanza elevadas precisiones del 96,2 % y 97,4 % en las fases de entrenamiento y validación, respectivamente, con el conjunto de datos UTA-RLDD. Además, se evalúa con éxito en condiciones prácticas, logrando una precisión del 95,5 % en un conjunto de datos personalizado de 39 participantes en diversas circunstancias lumínicas. Estos resultados respaldan la eficacia del marco propuesto y sugieren su aplicabilidad en sistemas de

transporte inteligentes, concluyendo que supera limitaciones previas y ofrece una contribución significativa para abordar la problemática de la somnolencia en la conducción.

– Salma Anber, Wafaa Alsaggaf and Wafaa Shalash. (2022). **A Hybrid Driver Fatigue and Distraction Detection Model Using AlexNet Based on Facial Features** (Anber et al., 2022).

El estudio aborda el creciente problema de accidentes de tráfico relacionados con la fatiga en conductores en las ciudades modernas, proponiendo y comparando dos modelos basados en AlexNet CNN para detectar comportamientos de somnolencia. El objetivo general es prevenir accidentes al diagnosticar y detectar a tiempo signos de fatiga, centrándose en la posición de la cabeza y movimientos de la boca como indicadores clave. La metodología involucra dos enfoques: aprendizaje por transferencia, con ajuste fino de AlexNet, y extracción de características mediante entrenamiento de capas superiores, reducción mediante NMF y clasificación con SVM. Los resultados revelan que el modelo de aprendizaje por transferencia alcanza una precisión del 95,7 %, mientras que el modelo basado en SVM con extracción de características logra una precisión aún mayor del 99,65 %. En conclusión, ambos modelos demuestran eficacia en la detección de comportamientos de somnolencia, resaltando la importancia de la detección temprana para prevenir accidentes de tráfico.

– Maryam Hashemi, Alireza Mirrashid, Aliasghar Beheshti Shirazi. (2020). **Driver Safety Development: Real-Time Driver Drowsiness Detection System Based on Convolutional Neural Network** (Hashemi et al., 2020).

El artículo aborda el desafío de la seguridad del conductor en carretera al presentar un sistema innovador de detección de la somnolencia del conductor. Utilizando redes neuronales convolucionales (CNN), el sistema se enfoca en alcanzar alta precisión y rapidez en la detección en tiempo real del estado de somnolencia del conductor. Tres redes

neuronales se emplean para clasificar el estado ocular, incluyendo una red neuronal totalmente diseñada (FD-NN) y dos que utilizan aprendizaje por transferencia en VGG16 y VGG19 con capas adicionales diseñadas (TL-VGG). Ante la carencia de conjuntos de datos precisos sobre los ojos para la detección del cierre ocular, los autores proponen un nuevo conjunto de datos integral. Los resultados experimentales destacan la eficacia del sistema al lograr alta precisión y baja complejidad computacional en la estimación del cierre de ojos, subrayando la capacidad del marco para la detección efectiva de la somnolencia del conductor en situaciones de tiempo real y consolidando su relevancia en la seguridad vial.

– Mohit Dua, Shakshi, Ritu Singla, Saumya Raj and Arti Jangra. (2020). **Deep CNN models-based ensemble approach to driver drowsiness detection** (Dua et al., 2021).

El artículo se enfoca en proponer un sistema innovador de detección de la somnolencia del conductor mediante el uso de arquitecturas de aprendizaje profundo como AlexNet, VGG-FaceNet, FlowImageNet y ResNet. Este sistema analiza vídeos RGB de conductores considerando gestos de la mano, expresiones faciales, comportamiento y movimientos de la cabeza para identificar la somnolencia. Cada modelo tiene un propósito específico, como AlexNet para cambios de entorno, VGG-FaceNet para características faciales, FlowImageNet para comportamiento y ResNet para gestos de las manos. La clasificación de las características en cuatro clases (no somnolencia, somnolencia con parpadeo, bostezo y cabeceo), se somete a un algoritmo de ensamblaje con un clasificador SoftMax, logrando una precisión del 85 %. Esta metodología efectiva destaca la importancia de considerar diversas señales y el éxito en identificar comportamientos relacionados con la somnolencia del conductor, ofreciendo una solución prometedora para la detección de esta condición en tiempo real durante la conducción.

2.2. Base teórica

2.2.1. Estudio de la somnolencia

Según la Real Academia Española (RAE, 2019).

"la somnolencia es la sensación de pesadez y torpeza de los sentidos motivadas por el sueño".

La somnolencia también se puede describir como un trastorno de pérdida de la agudeza sensorial, un estado entre la vigilia y el sueño profundo, que también puede causar una alteración de la conciencia.

2.2.1.1. Principales factores

La somnolencia puede ser causada por varios factores, entre ellos: conducción prolongada, aburrimiento, consumo de alcohol y/o drogas, enfermedades y trastornos del sueño (narcolepsia, etc.), consumos de fármacos que inducen la somnolencia, etc. (Flores Calero, 2009).

Según la Fundación-CEA (2022), los factores que causan la somnolencia son: la privación del sueño, el sueño fragmentado, cambios en el horario del sueño, las sustancias con efectos sedantes, los trastornos del sueño, en este ultimo tenemos el insomnio, hipersomnia, narcolepsia, trastornos respiratorios, trastornos del ritmo cardiaco, parasomninas, etc.

2.2.1.2. Características visuales de la somnolencia

Una persona somnolienta tiene ciertos rasgos faciales que se pueden reconocer en la cara, por lo que la cabeza, los ojos y la boca brindan suficiente información visual para determinar cuándo un conductor está somnoliento.

De acuerdo con la Fundación-CEA (2022), entre las principales características visuales de

somnolencia se tiene: distracción, movimiento de la cabeza (cabeceo), expresiones faciales, actividad espontánea oculomotora, aumento en la frecuencia del parpadeo, ojos cerrados y bostezo.

Todos estas características nos brindan una información valiosa, ya que es mas fácil determinar la somnolencia en una persona visualmente, por lo que son estudiadas y utilizadas por investigadores usando procesamiento digital de imágenes y técnicas de inteligencia artificial.

2.2.2. Sistema avanzado de asistencia a la conducción (ADAS)

Los ADAS (Advanced Driver Assistance Systems, por sus siglas en inglés) según European Road Safety Observatory, se definen como sistemas de seguridad inteligentes basados en vehículos que mejoran la seguridad vial en términos de prevención de colisiones, mitigación de colisiones y prevención de colisiones graves y sus consecuencias. De hecho, ADAS se puede definir como un sistema integrado en vehículos o basado en infraestructura que facilita múltiples fases de una emergencia. Por ejemplo, el ajuste inteligente de la velocidad y los sistemas de frenado avanzados pueden prevenir o reducir la gravedad de las colisiones (E.R.S.O., 2021).

De esta forma, ADAS se convierte en una alternativa para mejorar la seguridad y es el primer paso hacia la automatización completa de este proceso. En ellos se pueden distinguir dos componentes principales, se puede hacer una distinción donde el primero se integra con información relacionada con el exterior del vehículo y el segundo con su interior, generalmente enfocándose en la posición del conductor. Esta clasificación se denomina en la literatura ADAS como monitoreo externo e interno (Jiménez et al., 2016).

Mediante Zhao (2015), Intel proporciona una clasificación para las aplicaciones ADAS que se encuentran en varios modelos de vehículos en el mercado. Entre los mecanismos empleados para mejorar la seguridad de los coches se encuentran: Control Automático de Crucero (ACC), Alerta de Salida del Carril (LDW), Detección de Punto Ciego (BSD), Freno

Automático de Emergencia (AEB), Sistema de Navegación (NS), Reconocimiento de Señales de Tránsito (TSR), Monitoreo del Estado del Conductor (DSM), etc.

Este ultimo es objeto de investigación de la presente tesis, ya que con el DSM se puede detectar la somnolencia del conductor dentro de la cabina del vehículo.

2.2.3. Sistema de monitoreo del conductor (DSM)

El sistema avanzado de asistencia a la conducción (ADAS) está estrechamente relacionado con el sistema de advertencia de cambio de carril. El sistema de advertencia de cambio de carril está diseñado para evitar desviarse en cualquier situación, mientras que el sistema de monitoreo del conductor (DSM) está diseñado específicamente para reconocer signos de somnolencia del conductor.

El sistema descrito por Nieto et al. (2015) va un paso más allá al monitorear los ojos y la cara del conductor en busca de signos de somnolencia. Si el sistema detecta que el conductor tiene problemas para mantenerse despierto, puede tomar medidas correctivas emitiendo una alarma mientras conduce para evitar salidas de carril, despistes, choques, atropellos y mas, gracias a una cámara puesta en el tablero del vehículo.

2.2.3.1. Sistemas comerciales DSM

En la actualidad, existen varias empresas vinculadas al sector automotriz, ya sea indirectos - directos o a través de sus empresas fabricantes de componentes donde están desarrollando estos sistemas DSM, a continuación se describirán algunos de estos:

- STONKAM - DMS31.

El sistema de monitoreo del conductor STONKAM 1080P DMS31 (STONKAM, 2022) ayuda a monitorear y alertar al conductor cuando detecta somnolencia o distracción, lo que aumenta la seguridad durante la conducción. Equipada con una cámara NIR de 940nm FHD (1920x1080P @ 25fps o 30fps).

Figura 2.1: sistema de vigilancia del estado de conductor - STONKAM.

Fuente: STONKAM (2022).

- Seeing Machines Latin America - GUARDIAN.

Utilizando algoritmos de visión por computadora, inteligencia artificial y aprendizaje automático continuo, GUARDIAN detecta la somnolencia y el microsueño, así como las distracciones en tiempo real y emite las alertas correspondientes. Los algoritmos de GUARDIAN hacen seguimiento de la cara y la mirada, miden continuamente la posición de la cabeza del conductor, la velocidad de parpadeo, el cierre de los ojos y el movimiento de la pupila (Machines, 2020).

Figura 2.2: Sistema GUARDIAN - Seeing Machines Latin America.

Fuente: Machines (2020).

- CareDrive - MR688.

Haonai desarrolló el sistema de monitoreo de fatiga del conductor MR688 utilizando tecnología patentada de detección de pupila para detectar la somnolencia y distracción del conductor y alertar al conductor y al centro de gestión de flotas. El sistema usa un sensor de imagen en el automóvil para capturar una imagen infrarroja del conductor y usa un procesador de señal digital Pentium II de alta velocidad para analizar y detectar si se han vuelto distraídos debido a la somnolencia o a una distracción (Drive, 2016).

Figura 2.3: Monitor de fatiga del conductor MR688 - CareDrive.

Fuente: Drive (2016).

Así mismo, muchas compañías automotrices ya tienen integrados estos sistemas ADAS/DSM en sus respectivos vehículos, los cuales van mas para Europa y Asia, esto debido a su reglamento de disposición de uso de estos sistemas, a continuación se resume en la siguiente Tabla 2.1 las principales compañías automotrices que hacen uso del sistema DSM (Khan and Lee, 2019).

Tabla 2.1: Tecnología DSM en compañías de autos.

Compañía	Tecnología	Tipo	Sensor Primario	Función en cabina
Audi	AI:ME(concept)	DSM	Cámara NIR	Seguimiento de ojos/mirada
BMW	i Interaction EASE	DSM	Cámara NIR	Seguimiento de ojos/mirada y posición de movimiento de la cabeza
GM	Cadillac Cruise System	DSM	Cámara NIR	Seguimiento de ojos/mirada y posición de movimiento de la cabeza
Hyundai	Driver State Warning	DSM	Cámara NIR	Seguimiento de ojos/mirada y reconocimineto facial
Ford	Co-Pilot360	DMS	Cámara NIR	Seguimiento de ojos/mirada y posición de movimiento de la cabeza
Mazda	i-ACTIVSENSE	DSM	Cámara NIR	Seguimiento de ojos/mirada y posición de movimiento de la cabeza
Nissan	ProPILOT	DSM	Cámara NIR	Seguimiento de ojos/mirada y posición de movimiento de la cabeza
Subaru	EyeSight Driver Assist	DSM	Cámara NIR	Seguimiento de ojos/mirada, posición de movimiento de la cabeza y reconocimineto facial
Toyota	Lexus Safety System	DSM	Radar/cámara	Seguimiento de ojos/mirada y posición de movimiento de la cabeza

Fuente: recopilado y traducido de Khan and Lee (2019).

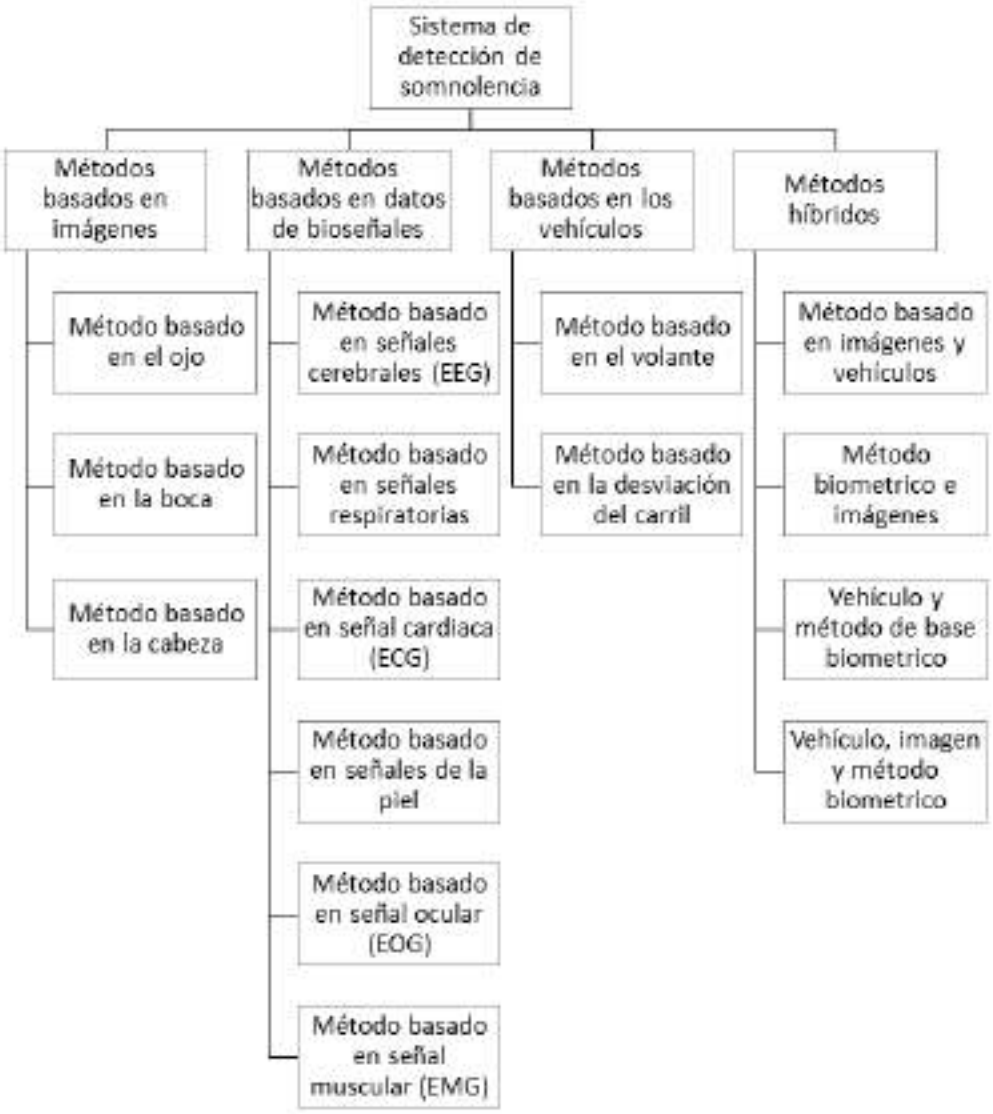
2.2.3.2. Limitaciones de los sistemas comerciales DSM

En el mercado actual, se encuentran disponibles diversos sistemas comerciales de Detección de Somnolencia y Monitoreo (DSM). Algunos de estos sistemas están integrados de fábrica en vehículos de varias marcas automotrices, mientras que otros se pueden adquirir directamente del fabricante. Sin embargo, es importante destacar que los sistemas de fábrica suelen estar disponibles principalmente en los mercados europeo y asiático y en vehículos con transmisión automática o eléctricos. Por otro lado, en América Latina, la mayoría de los vehículos disponibles son de transmisión mecánica y suelen ofrecer funciones de Freno Automático de Emergencia (AEB) como parte de sus características de seguridad. Además, las compañías que fabrican estos sistemas DSM generalmente los venden en cantidades y a empresas de flotas de vehículos, como autobuses interprovinciales o empresas mineras, lo que puede dificultar su adquisición para el público en general. Esto puede generar un acceso limitado para las personas individuales que desean adquirir este tipo de sistema.

2.2.4. Métodos de detección de somnolencia

Un estudio hecho por Albadawi (2022), indica que para determinar las diferentes etapas de la somnolencia, los investigadores estudiaron las reacciones del conductor y los hábitos de conducción del vehículo. Donde se muestra las cuatro medidas más utilizadas de detección de somnolencia. El gráfico de la Figura 2.4 muestra todas las medidas utilizadas actualmente para clasificar la somnolencia del conductor. Dos de estas medidas se observaron en los conductores: imagen y datos biométricos. La tercera medida se toma del propio automóvil, conocida como medidas basadas en el vehículo. La cuarta medida considerada es una medida mixta que combina al menos dos de los objetivos anteriores.

Figura 2.4: Medidas de detección de la somnolencia del conductor.



Fuente: Extraído y traducido de Albadawi (2022).

2.2.4.1. Métodos basados en imágenes

Algunos signos de somnolencia son visibles y pueden ser registrados por cámaras o sensores visuales. Estos incluyen las expresiones faciales y los movimientos del conductor, especialmente los movimientos de la cabeza, estado de los ojos y boca. En la literatura, estas señales se denominan objetivos visuales o basados en imágenes (Ramzan et al., 2019). También es importante señalar que las mediciones basadas en imágenes son una

subcategoría de las mediciones físicas o de comportamiento. Las mediciones físicas y de comportamiento se refieren a los movimientos corporales registrados en vídeo o utilizando sensores de movimiento como giroscopios y acelerómetros (Leng et al., 2015). Estas medidas basadas en imágenes se pueden dividir en tres métodos según se observen los movimientos de la boca, la cabeza o los ojos.

Los métodos basados en el procesamiento de imágenes son muy fiables y su principal característica es que no son invasivos y por tanto no provocan daños ni molestias al conductor.

En la presente tesis, se hará uso de este método, por el cual se profundizara mas en el tema, dando mas conceptos relacionados a este, el cual se hablara en un tema posterior.

2.2.4.2. Métodos basados en datos de bioseñales

En este método se utilizan bioseñales para detectar la somnolencia del conductor, incluidas las señales de actividad cerebral, frecuencia cardíaca, respiración, pulso y temperatura corporal. Estas señales biológicas, también conocidas como mediciones fisiológicas, son más precisas y confiables para detectar la somnolencia (Sikander and Anwar, 2018). Esta precisión está garantizada por su capacidad para capturar los primeros cambios biológicos que pueden ocurrir durante la somnolencia, alertando así al conductor antes de que aparezcan signos físicos de somnolencia.

2.2.4.3. Métodos basados en los vehículos

El método se basa en el seguimiento y análisis de patrones de conducción. Cada conductor tiene un modo de conducción único. Por lo tanto, el patrón de conducción de un conductor somnoliento se puede distinguir fácilmente del patrón de conducción de un conductor despierto. Las dos medidas relacionadas con el vehículo más utilizadas para identificar la fatiga del conductor son el ángulo del volante (SWA) y la salida del carril (Sikander and Anwar, 2018).

2.2.4.4. Métodos híbridos

El sistema híbrido utiliza una combinación de medidas vehiculares, biológicas y basadas en imágenes para extraer características de somnolencia con el objetivo de crear un sistema más robusto, preciso y confiable.

2.2.5. Métodos basados en el análisis de expresiones faciales mediante imágenes

Entre los métodos mas comunes para detectar la somnolencia en el análisis de expresiones faciales están:

1. Relación de aspecto ocular (EAR, Eye Aspect Ratio por sus siglas en inglés).
2. Relación de aspecto bucal (MAR, Mouth Aspect Ratio por sus siglas en inglés).
3. Estimación de la postura de la cabeza (HPE, Head Pose Estimation por sus siglas en inglés).

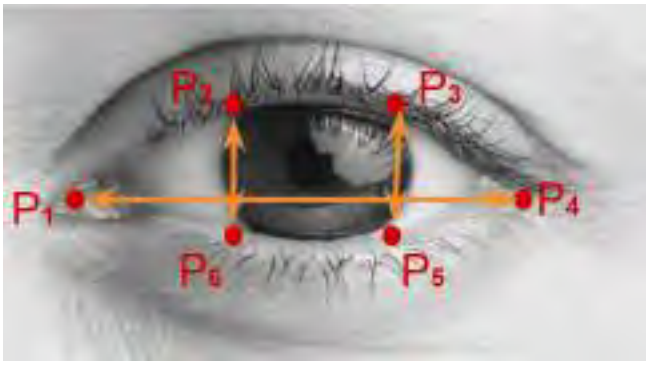
Donde, el primer método es correspondiente al método basado en el ojo, el segundo método es correspondiente al método basado en la boca y el tercer método es correspondiente al método basado en la cabeza.

2.2.5.1. Relación de aspecto ocular (EAR)

Para crear el sistema de detección de somnolencia del conductor, se necesita determinar si los ojos permanecen cerrados durante un intervalo de tiempo continuo. Donde Cech and Soukupova (2016), proponen un método de detección de parpadeo en tiempo real usando puntos de referencia faciales obteniendo un vector que contiene 6 posiciones de esos puntos de referencia para cada ojo (Figura 2.5). A partir de estos puntos, se puede determinar la relación de aspecto del ojo EAR entre la altura y el ancho del ojo, donde cada ojo posee

su propia relación EAR (Ecuación 2.1 y 2.2) respectivamente, y el valor final de EAR es el promedio de los valores de ambos ojos (Ecuación 2.3).

Figura 2.5: Puntos faciales para EAR.



$$EAR_i = \frac{||P_{2i} - P_{6i}|| + ||P_{3i} - P_{5i}||}{2||P_{1i} - P_{4i}||} \tag{2.1}$$

$$EAR_d = \frac{||P_{2d} - P_{6d}|| + ||P_{3d} - P_{5d}||}{2||P_{1d} - P_{4d}||} \tag{2.2}$$

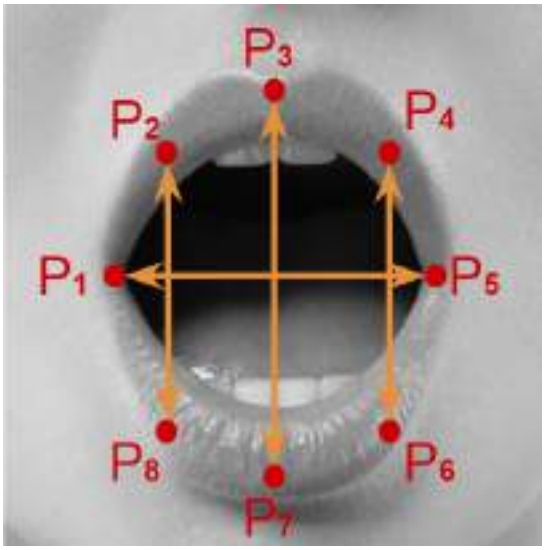
$$EAR = \frac{EAR_i + EAR_d}{2} \tag{2.3}$$

Donde EAR_i corresponde al valor del ojo izquierdo y EAR_d corresponde al valor del ojo derecho. Siendo los valores mínimos y máximos de 0.05 a 0.40 respectivamente, estos valores dependen de la altura de los ojos de cada persona.

2.2.5.2. Relación de aspecto bucal (MAR)

Para determinar la detección de bostezos, se puede utilizar un método muy similar al conteo de EAR identificando 6 puntos referentes a la boca. Sikander and Anwar (2018) plantea otro método, en el cual usa 8 puntos de referencia de la boca (Figura 2.6), donde se puede determinar la relación de aspecto bucal MAR (Ecuación 2.4) entre la altura y el ancho de la boca.

Figura 2.6: Puntos faciales para MAR.



$$MAR = \frac{||P_2 - P_8|| + ||P_3 - P_7|| + ||P_4 - P_6||}{3||P_5 - P_1||} \tag{2.4}$$

2.2.6. Visión computacional

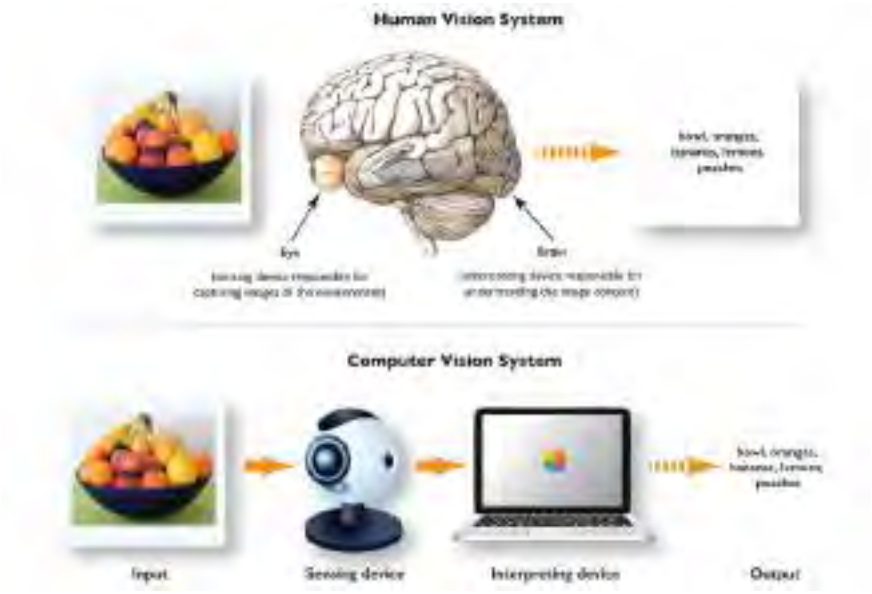
La inteligencia artificial en la actualidad cuenta con diversas especialidades que pueden ser de gran utilidad en el día a día de las personas, una de ellas es la visión computacional que se utiliza en muchos trabajos para detectar la somnolencia de los conductores.

La visión computacional (Computer Vision) ha surgido como un intento de dotar a las máquinas de sistemas de visión para automatizar el proceso de reconocimiento visual a través del procesamiento digital de imágenes (Martinsanz and de la Cruz García, 2007). Este es un tema complejo que incluye otras ciencias como matemáticas, física, ingeniería eléctrica, electrónica, robótica e informática. La visión computacional se puede definir como el proceso de obtención, caracterización e interpretación de información a partir de imágenes capturadas en un mundo tridimensional a partir de imágenes bidimensionales (Fu et al., 1988).

En otras palabras, la visión computacional es un campo de la inteligencia artificial que entrena a las computadoras para interpretar y comprender el mundo visual empleando un

conjunto de métodos o técnicas que trata sobre la obtención, el procesamiento y entendimiento de imágenes y vídeos, en lo que se trata de enseñar a las máquinas como pueden entender y automatizar tareas en el cual nosotros como humanos empleamos nuestro sistema visual.

Figura 2.7: similitud entre sistema de visión humana y sistema de visión computacional.



Fuente: Manning (2019).

2.2.6.1. Ventajas de la visión computacional

La visión computacional se distingue por su escena estructurada basada en frecuencia, medición proporcional, precisión y repetibilidad. Las principales ventajas de la visión artificial para la sociedad son: ahorro de tiempo; reducir los costos de producción; mejorar la productividad y la calidad del producto; reducir la fuerza laboral de prueba e inspección; reducir el número de productos no válidos; mejorar el uso de la máquina y así sucesivamente (COGNEX, 2016).

2.2.6.2. Aplicaciones de la visión computacional

Según Shenzhen Canrill Technologies, el sistema avanzado de visión computacional se utiliza a menudo para la medición, la inspección (como la detección de presencia, la detección de productos defectuosos, las estadísticas digitales, la detección de defectos), el posicionamiento y la detección, que incluyen la lectura de códigos y el reconocimiento de colores (SCTC, 2023).

2.2.6.3. Componentes de la visión computacional

Un sistema de visión artificial incluye componentes como iluminación, lentes, sensores de imagen, procesamiento de imágenes y comunicación de datos. La luz ilumina el área inspeccionada, resaltando las características y haciéndolas claramente visibles en la cámara. La lente toma una imagen y la presenta como luz al sensor. Los sensores de las cámaras de visión artificial convierten estas luces en imágenes digitales y las envían a un procesador para su análisis (SCTC, 2023).

2.2.7. Visión computacional en el espectro electromagnético

De acuerdo a la recopilación de Flores Calero (2009), un elemento esencial de un sistema de este tipo es el sistema de visión del propio vehículo, ya que proporciona información sobre la cabina en la que se encuentra el conductor. Otra característica principal que los caracteriza es su alta confiabilidad, así son no invasivos y no intrusivos y son relativamente económicos dado que su costo disminuirá a medida que avance la tecnología. Los sistemas propuestos en diferentes trabajos utilizando este enfoque se pueden clasificar de varias maneras. Las aplicaciones de visión computacional que se pueden desarrollar son variadas. Para Gonzalez and Woods (2008), pueden enfocar diferentes tipos de imágenes según de qué fuente de energía provengan. El más importante es el espectro electromagnético, que incluye rayos gamma, rayos X, ultravioleta (UV), luz visible, infrarrojo (IR), microondas y ondas de radio;

cada una de estas bandas proporciona un tipo diferente de información a la imagen y se utiliza para un propósito diferente. En este caso concreto, se hará según el tipo de iluminación, pues teniendo en cuenta la complejidad del problema, las condiciones de conducción (diurna y nocturna). Son los siguientes:

1. Métodos en el espectro visible (RGB).
2. Métodos en el infrarrojo cercano (NIR).

Figura 2.8: Rango de espectro electromagnético (UV - IR).

Fuente: Systems (2021).

En el primer caso, se suele utilizar una simple cámara junto con varios métodos clásicos de procesamiento de imágenes (información del color de la piel, correlación, etc.) para determinar la posición de los ojos en la imagen, usado para la conducción diurna.

En cambio, en el segundo caso, para la conducción nocturna se utiliza un complejo sistema de visión basado en iluminación infrarroja.

2.2.7.1. Métodos en el espectro visible (RGB)

Este método utiliza el espectro electromagnético en el rango de 380 nm a 780 nm (espectro visible) para el procesamiento. El sistema propuesto por Horng et al. (2004) emplea la detección del color de la piel para identificar el rostro, utilizando información del color del ojo para determinar si está cerrado o abierto, utilizando el espacio de color HSI. En contraste, Tian and Qin (2005) emplea el espacio de color YCbCr y construye una distribución normal bivariante para los componentes Cb y Cr. Después de localizar los ojos, calcula un índice de somnolencia basado en la distancia entre los párpados para determinar si los ojos están cerrados o abiertos.

Se puede ver en la siguiente Figura 2.9 un ejemplo de la región de interés (ROI) de los ojos presentado en 3 caso, los cuales se describen a continuación:

Los 3 casos habituales del conductor acerca de la vista, donde (A) es visión normal se observa en la Figura 2.9, (B) visión con lentes y (C) visión con lentes de sol u oscuros, como se puede ver en (C) al usar esos lentes y ser captados por una cámara RGB, se pierden la visibilidad de los ojos, lo cual es una limitante en este método.

Figura 2.9: Toma de ojos en el espectro visible.

2.2.7.2. Métodos en el infrarrojo cercano (NIR)

Algunas aplicaciones de la visión computacional requieren soluciones más allá del espectro visible debido a las características de emisión del objeto o la aplicación que se está evaluando. Por lo tanto, existen cámaras capaces de recibir señales en el espectro infrarrojo desde el rango visible hasta el rango de microondas (Alava, 2010).

Este método utiliza visión infrarroja y estereoscópica. Así, la respuesta de la pupila a esta luz determina la posición del ojo. En el rango de 700 nm y 900 nm, la pupila tiene una intensidad luminosa característica que ayuda a determinar su posición. Entre los sensores utilizados en cámaras infrarrojas (Källhammer, 2006). Se definen tres categorías: infrarrojo cercano (NIR), infrarrojo medio (MIR) e infrarrojo lejano (FAR-infrared, FIR). Donde, los sistemas NIR tienen un mejor rendimiento de detección.

- Infrarrojo cercano (NIR) - 780 nm a 2500 nm (12800 cm-1 - 4000 cm-1).
- Infrarrojo medio (MIR) - 2,5 µm a 25 µm (4000 cm-1 - 400 cm-1).
- Infrarrojo lejano (FIR) - 25 µm a 400 µm (400 cm-1 - 25 cm-1) .

La luz del infrarrojo cercano (NIR) cubre el rango de longitud de onda del espectro electromagnético de aproximadamente 780 nm a 2500 nm, que está más allá del rango de percepción visual humana. Las aplicaciones de detección basadas en luz infrarroja cercana proporcionan a las máquinas información sobre objetos de interés en el entorno físico (Systems, 2021).

Con base en Fatima et al. (2020); P Vikranth Reddy (2022); Park et al. (2019), en sus investigaciones indican que una cámara compuesta por un filtro NIR de 850 nm, la pupila refleja el 90 % de la luz infrarroja incidente, mientras que con un filtro de 940 nm refleja únicamente el 40 % de la luz infrarroja incidente sobre la retina. Lo cual en sus respectivas investigaciones hacen uso de una cámara NIR a 850 nm para una mejor visualización de los ojos.

Se puede observar en la Figura 2.10 la diferencia con una cámara NIR a 850 nm en comparación con la Figura 2.9.

Figura 2.10: Toma de ojos en el infrarrojo cercano a 850 nm.

Como se observa en la Figura 2.10, en (C) se puede apreciar los ojos al usar lentes de sol u oscuras, así mismo resalta la pupila del ojo en los 3 casos presentados a comparación del método en el espectro visible.

Por otro lado, El American Conference of Governmental Industrial Hygienists (ACGIH) anuncia en The Threshold Limit Values (TLVs) and Biological Exposure Indices (BEIs) que el límite de intensidad de radiación que se puede transmitir sin dañar la córnea y el cristalino en una sola longitud de onda por encima de 700 nm a 3000 nm en radiación LED no debe exceder los $10mW/cm^2$ de exposición durante más de 1000 segundos. Por lo tanto, es necesario utilizar un elemento de radiación de infrarrojo cercano con energía de radiación limitada.

2.2.7.2.1. Aplicaciones NIR por longitud de onda A continuación se detalla en la siguiente Tabla 2.2 las aplicaciones usadas de acuerdo a la longitud de onda en el infrarrojo cercano (NIR).

Tabla 2.2: Aplicaciones NIR por longitud de onda.

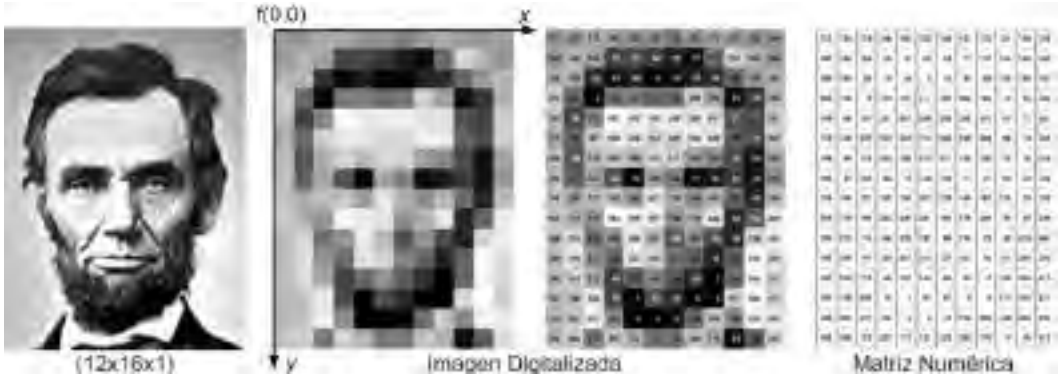
Longitud de onda NIR	Aplicación
780 nm	Registro visual
850 nm	Visión nocturna, Cámaras de seguridad
905 nm	Lidar a bordo de corto alcance (ADAS, vehículos autónomos)
940 nm	Reconocimiento facial y de gestos
1040 - 1060 nm	Topografía terrestre, sistema Lidar
1550 nm	Lidar a bordo de largo alcance (ADAS, vehículos autónomos)

Fuente: Extraído y traducido de Systems (2021).

2.2.8. Imagen digital

Una imagen se puede definir matemáticamente como una función bidimensional $f(x, y)$, donde x e y son coordenadas espaciales (en un plano) y f en cualquier par de coordenadas es la intensidad o el color en escala de grises de la imagen en esa coordenada. En los casos en que las coordenadas x e y y los valores de intensidad de la función f son tanto discretos como finitos, hablamos de una imagen digital. Una imagen digital (Figura 2.11) se compone de un número finito de elementos como una matriz rectangular, cada uno con un lugar y valor específicos. Estos elementos son llamados pixels. Está compuesta de diferentes tonos de gris. Generalmente las imágenes digitales tienen valores de intensidad entre 0 y 255, por lo que se necesitan 256 valores de intensidad (8 bits) para representar una imagen. Los tonos de gris más oscuros tienen valores más bajos (0), mientras que los tonos de gris más claros tienen valores de 255. La representación matemática de la imagen digital se ve en la Ecuación 2.5.

Figura 2.11: Representación de una imagen digital.



Fuente: Wevers and Smits (2020).

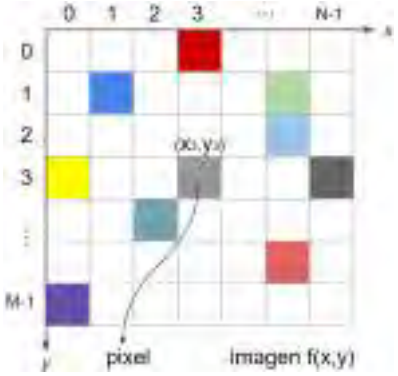
$$f(x,y) = \begin{pmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & \ddots & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{pmatrix} \tag{2.5}$$

En otras palabras, lo que para los humanos es proximidad visual, para las máquinas es proximidad numérica.

2.2.8.1. Píxel

Es la unidad homogénea más pequeña de color que forma parte de una imagen digital. Cada píxel tiene una posición determinada y un valor numérico dentro de la imagen digital. Los dispositivos de captura de imagen toman las imágenes mediante una cuadricula.

Figura 2.12: Píxel de una imagen digital.



2.2.9. Procesamiento digital de imágenes (PDI)

El procesamiento digital de imágenes es un grupo de técnicas que transforma imágenes digitales para mejorar la visibilidad de ciertas características de los objetos en la imagen, para un análisis posterior o simplemente para mejorar la visualización de la imagen. Al crear imágenes digitales, la intrusión de ruido o la degradación de la imagen son comunes, por lo que se deben considerar técnicas para mejorar la imagen o de acuerdo a la aplicación que se le da.

Para este caso especifico, se tomara en cuenta las siguientes técnicas de PDI:

1. Conversión a escala de grises.
2. Escalamiento.
3. Región de interés (ROI).

2.2.9.1. Conversión a escala de grises

El primer método, y más intuitivo, es convertir la imagen a escala de grises sacando el promedio aritmético de los valores RGB de una imagen digital.

$$G(x,y) = \frac{R(x,y) + G(x,y) + B(x,y)}{3}$$

El segundo método es calculando la variable luma. Los valores de los pesos para cada color recomendados varían según el tipo de pantalla. El Y'_{601} es el más conocido y se hizo originalmente para SDTV. Otros pesos para calcular la variable luma existen para HDTV (Y709), UHDTV HDR (Y2020) y de Adobe (Y240).

Los software de programación usan el Y'_{601}. Para eso se utiliza la siguiente ecuación:

$$Y'_{601}(x,y) = 0{,}299R(x,y) + 0{,}587G(x,y) + 0{,}114B(x,y) \tag{2.6}$$

La Ecuación 2.6 muestra que la intensidad de píxel de una imagen en escala de grises en la ubicación (x,y) es igual a la suma de las 3 intensidades de los componentes, cada una multiplicada por el peso de píxel de la imagen correspondiente para la transformación (x,y).

2.2.9.2. Escalamiento

Escalamiento se define como ajustar un tamaño ya sea más grande (aumento) o más pequeño (reducción) manteniendo sus proporciones originales. Esto significa que la tasa de expansión o contracción es constante en toda la imagen.

Si se tiene una imagen $f(x, y)$, donde x e y son sus dimensiones, para el escalamiento se multiplica por una constante llamados factores de escala "S_x" y "S_y". Si:

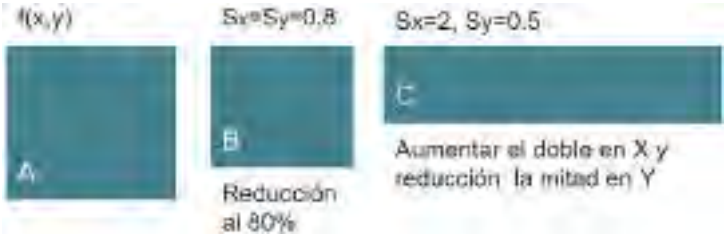
$$x' = S_x.x$$

$$y' = S_y.y$$

Se tiene $f(x', y')$, donde según el valor del factor de escala S se tiene:

- Si S mayor que 1, entonces aumento de tamaño.
- Si S igual que 1, entonces no cambia de tamaño.
- Si S menor que 1, entonces disminución de tamaño.

Figura 2.13: Escalamiento de una imagen.



En la Figura 2.13, se observa que A es reducido en un 80 % siendo "$S_x = S_y = 0{,}8$" obteniendo B, por otro lado el tamaño X de A es aumentado al doble y el tamaño Y de A es reducido a la mitad obteniendo la imagen C.

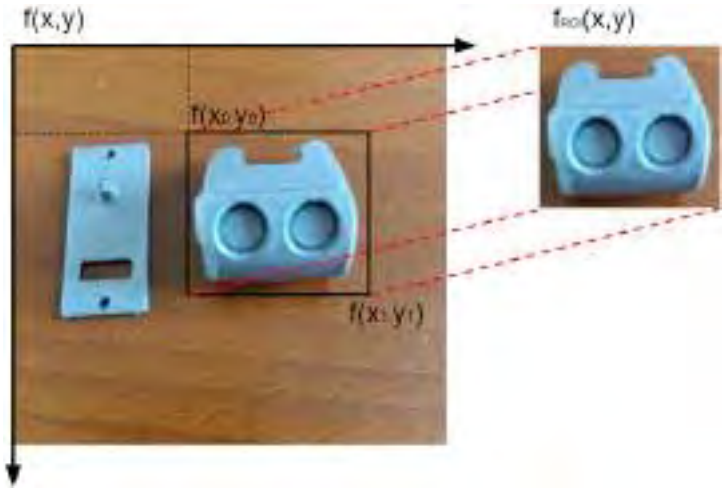
2.2.9.3. Región de interés (ROI)

La región de interés es una segmentación de extracción de características, en el cual se selecciona una área rectangular determinada de la imagen. ROI sirve para evaluar un segmento dentro de la imagen digital, haciendo su análisis respectivo de dicha área. La selección de ROI puede ser automática o manual, donde su selección viene dada por:

$$f_{ROI}(x, y) = f(x, y)[x_0 : x_1, y_0 : y_1] \tag{2.7}$$

Donde la Ecuación 2.7 nos indica que, la imagen ROI es la imagen original recortada desde un punto x_0 inicial hasta un punto x_1 final del tamaño de la imagen original en el eje x y desde un punto y_0 inicial hasta un punto y_1 final del tamaño de la imagen original en el eje y.

Figura 2.14: ROI de una imagen.



2.2.10. Detección de rostros y puntos faciales

El término "detección de rostros" se refiere a un mecanismo informático mediante el cual es posible determinar si una imagen digital dada contiene un rostro humano y, si corresponde, dar su ubicación y tamaño (Modi and Macwan, 2014).

La detección de objetos es una tarea bien estudiada y aplicada en visión computacional, por lo que no es de extrañar que haya tanto trabajo sobre este tema. Para detectar un objeto específico, uno busca sus características visuales en una imagen, pero cuando se trata de encontrar o detectar un rostro humano, la dificultad aumenta porque el rostro humano tiene una amplia gama de poses con infinitos movimientos y detalles visuales para cambiar su apariencia; en el caso de la localización facial mediante el procesamiento de imágenes y la visión computacional, cualquier cosa que ayude visualmente a nuestro cerebro a distinguir a una persona de otra agrega una capa adicional de complejidad (Dalal and Triggs, 2005).

De acuerdo a los antecedentes recopilados en el punto 2.1, se pone en lista aquellos métodos actuales que se relacionan directamente con los retos de la detección de rostros, a continuación su listado:

1. Haar cascade.
2. Dlib.
3. MediaPipe.

Donde el funcionamiento de estos algoritmos generan un rectángulo ROI que localiza la posición del rostro a través de las coordenadas de la esquina superior izquierda (x0, y0) y sus dimensiones de ancho y alto (w,h).

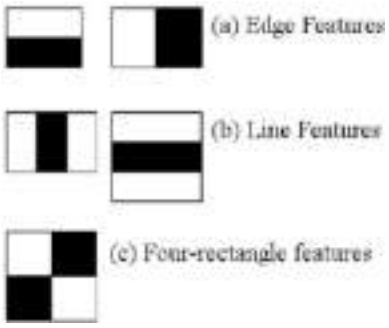
Los tres métodos son previamente entrenados, lo que quiere decir que hacen uso de redes neuronales convolucionales para tener una mejor precisión en la detección de rostros.

2.2.10.1. Haar cascade

También llamado algoritmo de Viola Jones por sus respectivos autores Paul Viola y Michael Jones, donde en Viola and Jones (2001), los autores proponen un método para la detección de objetos que presenta un bajo coste computacional y permite que sea empleado en tiempo real.

El algoritmo puede encontrar caras y objetos de forma fácil y rápida a partir de imágenes en escala de grises. Esto se hace mediante el uso de rectángulos de características y la adición y sustracción de píxeles entre las áreas rectangulares puede generar una característica de Haar para la detección. Los valores resultantes se utilizan en el algoritmo de aprendizaje automático de Adaboost para seleccionar las características correctas en función del hecho de que el área de los ojos es más oscura que el área de la nariz y las mejillas.

Figura 2.15: Detectores Haar.



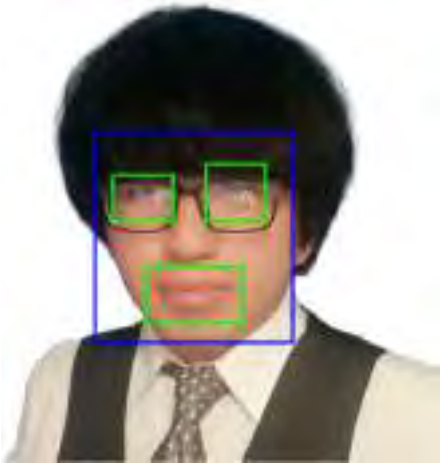
Fuente: Viola and Jones (2001).

Inicialmente, el algoritmo necesita muchas imágenes positivas (imágenes de rostros, ojos, bocas) e imágenes negativas (imágenes sin rostros) para entrenar al clasificador. Actualmente se tiene modelos pre-entrenados por diferentes autores que dieron sus aportes a distintas aplicaciones. Para este caso se necesita la detección de rostro, ojos y boca.

2.2.10.1.1. Haar cascades pre-entrenados Dentro del repositorio de los Haar Cascades pre-entrenados, se tiene 17 detectores, de los cuales para la presente investigación son necesarios los siguientes:

- haarcascade-frontalcatface.xml
- haarcascade-eye.xml
- haarcascade-smile.xml

Figura 2.16: Detección de características con Haar cascade.



2.2.10.2. Dlib

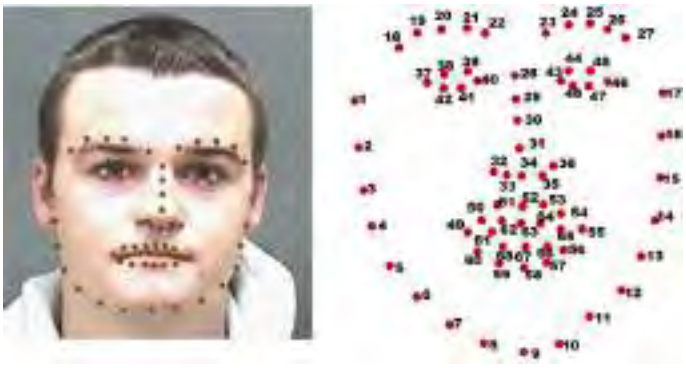
DLIB es una biblioteca de software universal de código abierto escrita en lenguaje C++. Su objetivo es proporcionar un mejor entorno para desarrollar software de aprendizaje automático (Machine Learning), visión computacional y análisis de datos en C++, ya que intenta llenar algunos de los vacíos en el soporte de herramientas que otras bibliotecas carecen (King, 2009). La biblioteca se publica bajo la licencia de software Boost, que permite la integración con software comercial y de código abierto, y tiene una documentación extensa y precisa. No requiere bibliotecas adicionales, instalación o configuración para su uso. A menudo se ha utilizado en entornos Windows, Linux y MacOS X, pero debería funcionar con cualquier compilador compatible con ISO C. Se usa ampliamente en la industria y el mundo académico, incluida la robótica, los dispositivos integrados, los teléfonos móviles y los entornos informáticos de alto rendimiento a gran escala.

2.2.10.2.1. Dlib face detection: Dentro de Dlib esta el face detection, el cual es el detector de rostros basado en 2 algoritmos, el primero esta en descriptores HOG con Linear SV, mientras que el segundo esta con redes neuronales convolucionales (CNN), esta hecho para detectar rostros en tiempo real con gran precisión.

2.2.10.2.2. Dlib face landmark: Sirve para detectar puntos de referencia faciales en una imagen. Los puntos de referencia faciales se utilizan para localizar y representar regiones destacadas de la cara, como: ojos, cejas, nariz, boca y mandíbula. Estos puntos de referencia faciales se han aplicado con éxito a la alineación de la cara, la estimación de la postura de la cabeza, el intercambio de caras, la detección de parpadeo y mucho más.

El detector de referencias faciales pre-entrenado dentro de la biblioteca dlib se utiliza para estimar la ubicación de 68 coordenadas (x, y) que se asignan a estructuras faciales en la cara.

Figura 2.17: Detección de características con Dlib face landmark.



Fuente: Elmahmudi and Ugail (2021).

2.2.10.3. MediaPipe

Mediapipe es una librería que proporciona varias soluciones de aprendizaje automático de última generación para diversos problemas de visión artificial. Se destaca porque es fácil de integrar, ofrece soluciones rápidas y livianas y puede ejecutarse en cualquier sistema operativo, incluidos Android e IoT (Lugaresi et al., 2019).

Entre las muchas y variadas funciones que ofrece la biblioteca en relación con este proyecto, destacan los modelos de reconocimiento facial y, más ambiciosamente, los modelos de generación de máscaras faciales.

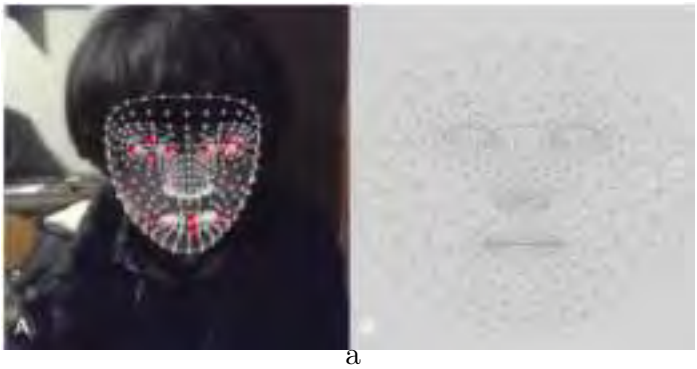
2.2.10.3.1. MediaPipe face detector El detector de rostros Mediapipe es un modelo conocido por su capacidad de respuesta y precisión. Se basa en BlazeFace, que se basa en el funcionamiento de Single Shot Multibox Detector, que consiste en un detector de rostros ligero y preciso diseñado para conseguir resultados satisfactorios en las tarjetas gráficas de los smartphones actuales y sistemas embebidos.

2.2.10.3.2. Mediapipe face mesh Mediapipe Face Mesh incluye una solución para estimar con precisión 468 puntos faciales 3D en tiempo real utilizando una sola cámara sin sensor de profundidad.

El modelo da respuesta en un tiempo de ejecución muy corto, por lo que es muy valioso poder realizar segmentaciones complejas de áreas de interés.

El módulo de malla facial almacena los puntos de la cara en una lista y accede a ellos mediante un índice, por lo que la selección de puntos de la cara consiste en un solo acceso a la lista generada por la biblioteca establecida.

Figura 2.18: Detección de características con mediapipe face mesh.



a

2.2.11. Deep learning y redes neuronales profundas

El aprendizaje profundo en español o deep learning es un campo popular del aprendizaje automático que tiene como objetivo aprender abstracciones de información de alto nivel utilizando arquitecturas jerárquicas (LeCun et al., 2015). El éxito del aprendizaje profundo se debe a la disponibilidad de hardware, es decir para unidades centrales de procesamiento

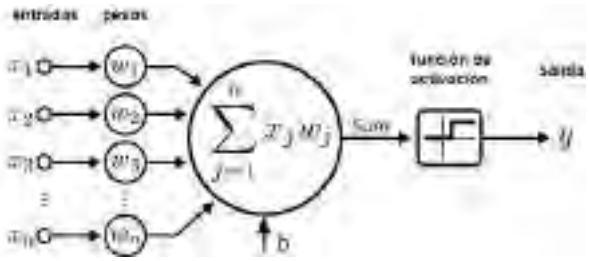
(CPU), unidades gráficas (GPU), disco duro, etc.

El Deep Learning tiene diversas aplicaciones, especialmente en medicina a través de diagnósticos por imágenes médicas y mercados financieros a través de modelos predictivos, pero otras industrias poco a poco están aprovechando esta tecnología y desarrollando sistemas de aprendizaje profundo para la detección de fraudes, auditoría de datos, detección de anomalías, etc. Donde estos sistemas están siendo cada vez mas populares ya que se utiliza en sistemas como: traductores inteligentes, procesamiento de lenguaje natural, reconocimiento de voz, reconocimiento facial, Vision computacional, etc.

Dentro del Deep Learning están las redes neuronales profundas. Una red neuronal profunda (DNN) es una red neuronal artificial (ANN) con muchas capas ocultas entre las capas de entrada y salida. Al igual que las redes neuronales superficiales, las DNN pueden modelar relaciones no lineales complejas (Fuente, 2019). El objetivo principal de las redes neuronales es tomar un conjunto de entradas, realizar cálculos cada vez más complejos y generar salidas para resolver problemas del mundo real, como la clasificación.

Además, el aprendizaje profundo se ha aplicado a muchas tareas desafiantes en visión computacional y se han logrado mejoras significativas en los últimos años (Huang et al., 2014).

Figura 2.19: Representación de una ANN.



Fuente: Ghorbel (2021).

La ecuación de la ANN viene dada por:

$$y = f(\sum_{n=1}^{n} x_n w_n + b) \tag{2.8}$$

Donde:

- y es la salida de la neurona.
- f es la función de activación.
- x_n es la matriz de entradas.
- w_n es la matriz de pesos.
- b es el umbral o bias de la activación de la neurona.

2.2.11.1. Métricas de evaluación

Cuando los problemas de clasificación se resuelven utilizando DNN, se obtiene una clasificación binaria mínima con cuatro combinaciones posibles de clases reales y esperadas.

- Verdadero Positivo (TP): Asignaciones a la clase positiva correctas. Si el resultado de la clasificación es un estado de somnolencia y corresponde al estado indicado por la señal de referencia.
- Falso Positivo (FP): Asignaciones a la clase positiva incorrectas. Si el resultado de la clasificación es un estado de despierto y corresponde al estado indicado por la señal de referencia.
- Verdadero Negativo (TN): Asignaciones a la clase negativa correctas. Si el resultado de la clasificación es somnolencia y la señal de referencia indica que está despierto.
- Falso Negativo (FN): Asignaciones a la clase negativa incorrectas. Si el resultado de la clasificación es despierto y la señal de referencia indica que está en estado de somnolencia.

Estas combinaciones se resumen en la siguiente Tabla 2.3 llamada Matriz de Confusión.

Tabla 2.3: Matriz de confusión.

	Valor real positivo (1)	**Valor real negativo (0)**
Predicción positiva (1)	Verdadero Positivo (TP)	Falso Positivo (FP)
Predicción negativa (0)	Falso Negativo (FN)	Verdadero Negativo (TN)

Las siguientes métricas se pueden extraer de la matriz de confusión, que brindan información sobre la calidad de la clasificación a evaluar.

- Precision: Representa la proporción correcta de asignaciones positivas entre todas las asignaciones positivas.

$$Precision = \frac{TP}{TP + FP} \tag{2.9}$$

- Recall: Representa la proporción correcta de asignaciones positivas entre las observaciones positivas reales.

$$Recall = \frac{TP}{TP + FN} \tag{2.10}$$

- F1-score: Representa la media armónica de recall y precisión. Esto permite evaluar la eficacia de estas dos medidas.

$$F1 - score = \frac{2 * precision * recall}{precision + recall} \tag{2.11}$$

- Accuracy: Representa el porcentaje de éxito del sistema para todas las observaciones en el conjunto de datos.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.12}$$

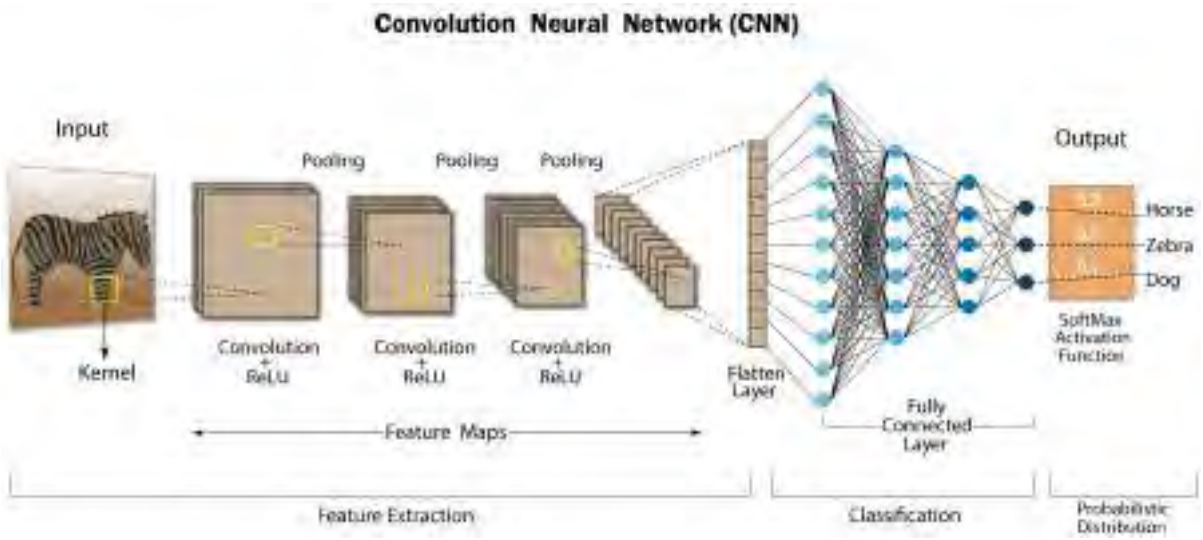
Estas cuatro métricas de evaluación servirán mas adelante para ver el desempeño del sistema en el entrenamiento de los datos mediante las redes neuronales convolucionales.

2.2.12. Redes neuronales convolucionales (CNN)

Una red neuronal convolucional (CNN) es un tipo especializado de red neuronal que reduce eficientemente el número de parámetros en comparación con red neuronal profunda, preservando la calidad del modelo. Las CNN tienen diversas aplicaciones en el procesamiento de imágenes y texto, destacándose en áreas como reconocimiento de imágenes, clasificación de imágenes, detección de objetos y reconocimiento facial (Elhamraoui, 2020).

En la Figura 2.20, se destacan las tres etapas de una CNN. En la primera etapa, Extracción de Características (Feature Extraction), se generan mapas de características (feature maps) mediante la convolución de las imágenes de entrada con diversos kernels y el proceso de pooling. Estos mapas se aplanan en una matriz $mx1$, que sirve como entrada para la siguiente etapa de Clasificación. En esta etapa, se emplea una red neuronal profunda (DNN) para aprender las características de las imágenes mediante capas conectadas (fully connected layer). La última etapa, Distribución Probabilística, utiliza una función de activación Softmax para clasificar el problema y determinar el porcentaje de distribución de las clases.

Figura 2.20: Representación de una CNN.



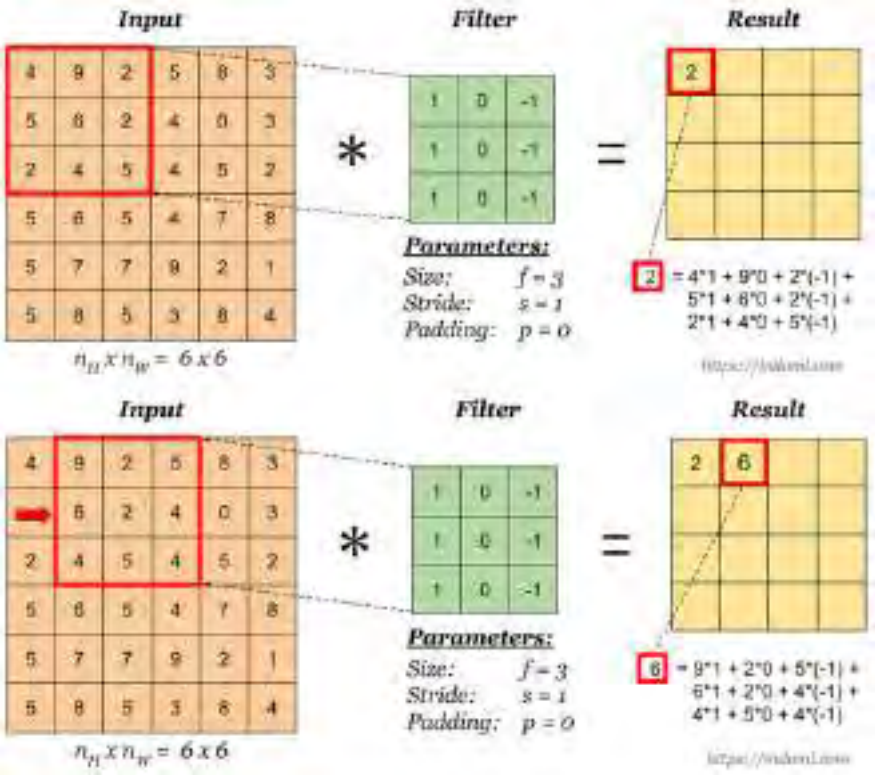
Fuente: Swapna (2020).

Dentro de la arquitectura de la CNN se tiene varias capas, las cuales son de extracción de las características de la imagen de entrada, estas se detallan a continuación:

2.2.12.1. Capa de convolución

La capa de convolución realiza cálculos para extraer información útil de los datos de imagen. Cada capa de convolución consta de filtros de entrenamiento, que son matrices 2D aplicadas a una imagen en color (matriz 3D de píxeles). Estos filtros determinan la presencia de características mientras se mueven a través del campo receptivo de la imagen, calculando productos escalares entre los pesos del filtro y los píxeles de entrada. El proceso, llamado convolución, produce mapas de activación o características que contienen información sobre las funciones extraídas de la imagen mediante un filtro específico (ProjectPro, 2023).

Figura 2.21: Proceso de convolución.



Fuente: ProjectPro (2023).

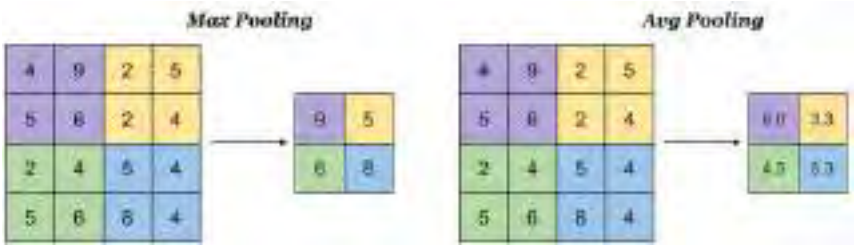
2.2.12.2. Capa de agrupación (pooling)

La capa de agrupación reduce la dimensionalidad de la imagen de entrada aplicando filtros y sumando los píxeles de entrada, en contraste con la capa de convolución que utiliza matrices de pesos. Aunque esta operación conlleva cierta pérdida de datos, las capas de agrupación ofrecen beneficios como la reducción de la complejidad de la función, disminuyendo la propensión al sobreajuste (overfitting), acelerando el cálculo y mejorando la eficiencia de la CNN. Se distinguen dos tipos de capas de agrupación (ProjectPro, 2023):

1. **Max pooling:** el píxel de entrada con el valor máximo se guarda en la matriz de salida.

2. **Average pooling:** el promedio de píxeles de entrada se guarda en la matriz de salida.

Figura 2.22: Proceso de pooling.



Fuente: ProjectPro (2023).

2.2.12.3. Capa totalmente conectada

En una capa completamente conectada, cada nodo está conectado a todos los nodos anteriores. Los mapas de características de salida de las dos capas restantes son vectores tridimensionales. Se puede aplanarlos para crear un vector unidimensional que se alimenta a esta capa completamente conectada para la tarea de clasificación final. Esta capa utiliza la función de activación softmax para predecir las probabilidades finales de la clase a clasificar (ProjectPro, 2023).

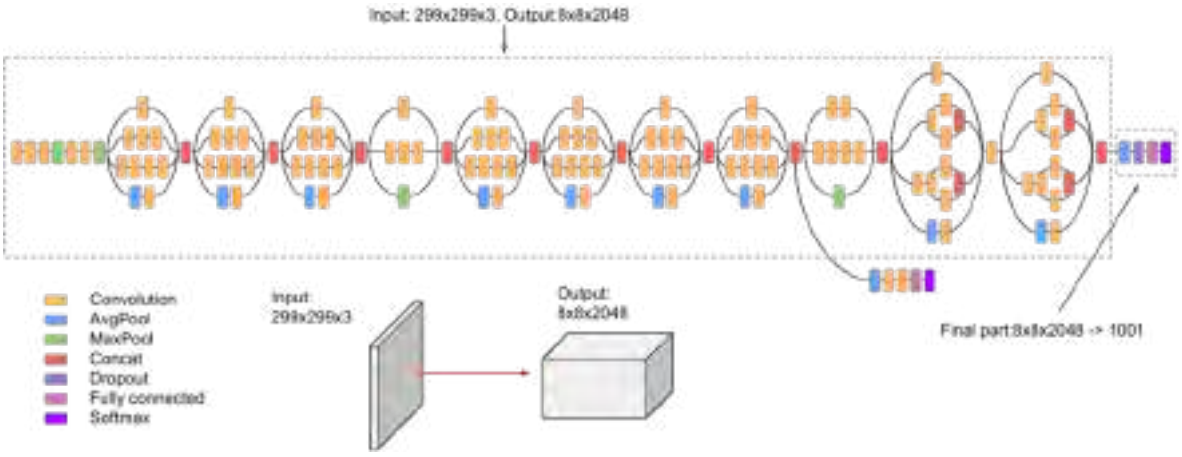
2.2.13. Arquitecturas de CNNs

Actualmente hay diferentes arquitecturas de CNN para la aplicación requerida. En este trabajo de investigación se usara 3 arquitecturas que dan buenos resultados para la clasificación binaria de la somnolencia. De acuerdo con lo recopilado en el punto 2.1 Antecedentes y Anber et al. (2022); Dua et al. (2021); Hashemi et al. (2020); Krishna et al. (2022); P Vikranth Reddy (2022), se eligieron las siguientes arquitecturas a tomar en cuenta para esta investigación:

2.2.13.1. InceptionV3:

InceptionV3 (Szegedy et al., 2016) es un modelo de reconocimiento de imágenes que ha logrado más del 78,1 % de precisión en el conjunto de datos de ImageNet. El modelo es la culminación de muchas ideas desarrolladas por varios investigadores a lo largo de los años. Se basa en el artículo original: Szegedy et al. Reestructuración de la arquitectura de inicio de visión artificial. El modelo InceptionV3 se lanzó en el año 2015, tiene un total de 42 capas y una tasa de error más baja que sus predecesores. (Google, 2023).

Figura 2.23: Arquitectura de InceptionV3.

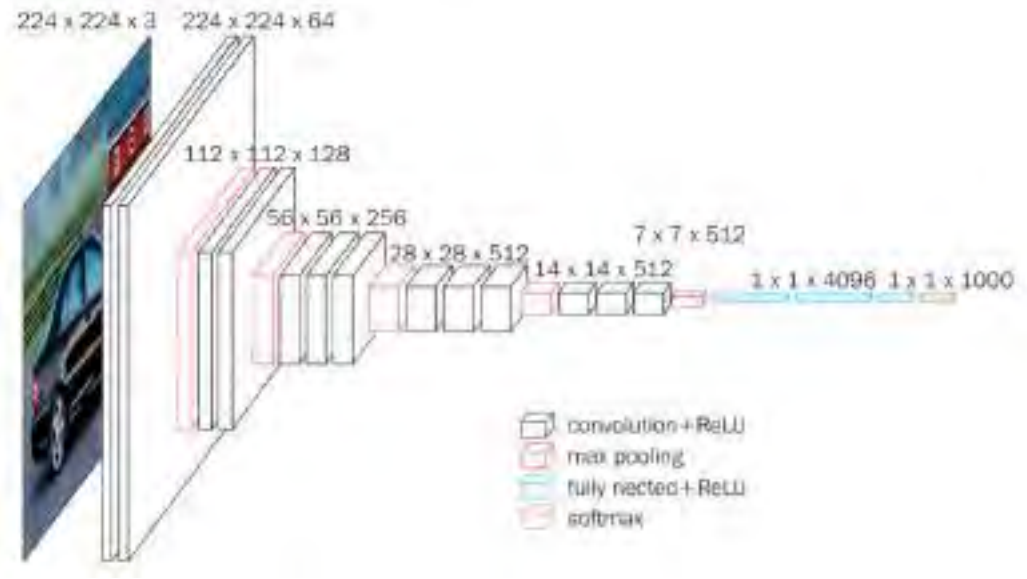


Fuente: Google (2023).

2.2.13.2. VGG-16:

En esta red, se adquieren imágenes RGB de 224x224 píxeles, pero las capas de tipo VGG usan escuelas para recibir 3x3 muy pequeñas y recibir el tamaño mínimo posible, también hay un filtro de tipo 1x1 que funciona como una conversión lineal de la entrada, y luego continúa la unidad ReLu. El VGG está completamente conectado a tres capas, las dos primeras capas tienen 4096 canales para 1000 canales, terceras clases y una clase para cada capa. La capa oculta VGG se reduce (Simonyan and Zisserman, 2014).

Figura 2.24: Arquitectura de VGG-16.



Fuente: Simonyan and Zisserman (2014).

2.2.13.3. ResNet50:

ResNet-50 es una red neuronal convolucional profunda de 50 capas. ResNet, abreviatura de Residual Networks, es una red neuronal clásica que se utiliza como elemento básico para muchas tareas de visión artificial. El avance clave de ResNet fue que permitió entrenar redes neuronales muy profundas con más de 150 capas. Es una red neuronal innovadora presentada por primera vez por Kaiming He, Xiangyu Zhang, Shaoqing Ren y Jian Sun en (He et al., 2016).

Figura 2.25: Arquitectura de ResNet50.

Fuente: He et al. (2016).

2.2.14. Transfer learning

El aprendizaje por transferencia o transfer learning es una técnica de aprendizaje automático en la que se entrena y desarrolla un modelo para una tarea y luego se reutiliza en otra tarea relacionada. Esto se refiere a una situación en la que lo que se aprende en un entorno se utiliza para mejorar la optimización en otro entorno (Gao and Mosalam, 2018). El aprendizaje de transferencia generalmente se usa cuando hay un nuevo conjunto de datos que es más pequeño que el original utilizado para entrenar el modelo previamente entrenado (Larsen-Freeman, 2013).

2.2.15. Base de datos

Para el caso de detectar la somnolencia actualmente existen varios repositorios que incluyen estos datos, como vídeos e imágenes. Según los antecedentes anteriormente vistos, se toma en cuenta 3 dataset para la detección de la somnolencia:

2.2.15.1. Driver drowsiness detection dataset

Este dataset es brindado por el "Computer Vision Lab de National Tsing Hua University (NTHU)", en el cual se recopilo de 36 personas de diferentes etnias, con y sin usar anteojos/gafas de sol en una variedad de escenarios de conducción simulados, que incluyen conducción normal, bostezos, parpadeo lento, quedarse dormido, reírse a carcajadas , etc., en condiciones de iluminación diurna y nocturna. Para la recopilación de

datos usaron iluminación infrarroja activa (IR) para adquirir videos IR en una resolución de vídeo es 640x480 a 30 fps en formato AVI de unas 9 horas y media de duración (Weng et al., 2017).

Figura 2.26: Imagen referencial del dataset NTHU.

Fuente: Weng et al. (2017).

2.2.15.2. UTA real-life drowsiness dataset

Esta base de datos es proporcionada por la "University of Texas at Arlington (UTA)", donde el conjunto de datos consta de alrededor de 30 horas de videos RGB de 60 participantes con un total de 180 vídeos y un peso de 111,3 Gigabytes. Los vídeos se tomaron desde ángulos aproximadamente diferentes en diferentes entornos y fondos de la vida real utilizando su teléfono celular o cámara web a 30 fps (Ghoddoosian et al., 2019).

Figura 2.27: Imagen referencial del dataset UTA.

Fuente: Ghoddoosian et al. (2019).

2.2.15.3. Night-time yawning-microsleep-eyeblink-driver distraction

Este conjunto de datos es por parte de la empresa ESDA Laboratory, que nos ofrece 130 vídeos disponibles que muestran a los conductores en automóviles reales, moviéndose en condiciones nocturnas, capturados en Patras, Grecia. Los vídeos han sido capturados por una cámara montada en el tablero del automóvil. Los pilotos participantes son: 11 varones y 10 mujeres con diferentes características (color de pelo, barba, gafas, etc). Los vídeos se dividen en 2 categorías: Yawning y Microsleep. Este dataset viene en 2 resoluciones, HDTV720 (1280px720p) y FHD (1920px1080p) ambos a 25 fps (Petrellis et al., 2021).

Figura 2.28: Imagen referencial del dataset NITYMED.

Fuente: Petrellis et al. (2021).

2.2.16. Cámara RGB-NIR

El sensor primario y principal para el presente proyecto de investigación es la cámara, ya que con este se obtendrá las imágenes faciales del conductor para luego hacer su respectivo procesamiento y análisis, y así poder determinar el estado de somnolencia; para eso se pone a conocer 3 cámaras web que ayudaran en el proyecto:

2.2.16.1. ELP-USB3MP01H

Cámara web por la empresa ELP, tiene la funcion WDR (del inglés Wide Dynamic Range) a 3Mp imcorporado con visión nocturna automática a 850nm o 940nm (ELP, 2023a).

Figura 2.29: Cámara Web ELP-USB3MP01H.

Fuente: ELP (2023a).

2.2.16.2. ELP-USBFHD05MT-KL36IR

Esta cámara web también por la empresa ELP ofrece, una resolución de 2Mp con el sensor CMOS OV2710 con 10 piezas led IR para visión nocturna a 850nm (ELP, 2023b).

Figura 2.30: Cámara Web ELP-USBFHD05MT-KL36IR.

Fuente: ELP (2023b).

2.2.16.3. ELP-USBFHD08S-KV100IR

Cámara web también por la empresa ELP, usa una lente de alta velocidad sin distorsión de 2MP con sensor CMOS OV4689 y luz infrarroja con LED IR a 850nm (ELP, 2023c).

Figura 2.31: Cámara Web ELP-USBFHD08S-KV100IR.

Fuente: ELP (2023c).

2.2.17. Sistema embebido

Un sistema embebido (E.S.) es un sistema de computación diseñado para realizar ciertas funciones en tiempo real y diseñado para satisfacer necesidades específicas. la mayoría de sus componentes están en la misma placa base (Aguirre and Giraldo, 2014).

Hoy en día, los sistemas basados en visión computacional se han convertido en una herramienta imprescindible para los sistemas de seguridad privada, ya que son herramientas económicas con la ventaja de una alta precisión. Estos sistemas deben implementarse en hardware cada vez más pequeños para una mayor portabilidad, velocidad y bajo costo; por lo tanto, estos algoritmos se están migrando a plataformas integradas cada vez más rápidas que permiten que estos sistemas se ejecuten en tiempo real (Pérez et al., 2018).

Dicho eso, para este proyecto de investigación se recopilo 2 sistemas embebidos que cumplen los requisitos del costo computacional, los cuales son:

1. Raspberry Pi

2. NVIDIA Jetson Nano

2.2.17.1. Raspberry Pi

Raspberry Pi es el nombre de una gama de computadoras de placa única creada por la Fundación Raspberry Pi (Raspberry, 2020), una organización benéfica británica dedicada a educar a las personas sobre las computadoras y hacer que la educación informática sea más accesible. Lanzada en 2012, la Raspberry Pi ha pasado desde entonces por varias iteraciones y variantes. En todo el mundo, las personas usan Raspberry Pi para aprender habilidades de programación, crear proyectos de hardware, crear domótica, implementar clústeres de Kubernetes y Edge Computing, e incluso usarlo en aplicaciones industriales. La Raspberry Pi es una computadora muy económica que ejecuta Linux, pero también tiene un conjunto de pines GPIO (entrada/salida de uso general) para que pueda controlar los componentes electrónicos para la computación física y explorar el Internet de las cosas (IoT).

El modelo de Raspberry Pi que cumple con los mínimos requisitos computacionales es el Raspberry Pi 4 model B.

Figura 2.32: Raspberry Pi 4 modelo B.

Fuente: Raspberry (2020).

2.2.17.2. NVIDIA Jetson Nano

NVIDIA Jetson es la plataforma Edge AI (inteligencia artificial procesada localmente) líder en el mundo. Su cómputo de bajo consumo y alto rendimiento para el aprendizaje profundo y la visión computacional lo convierte en una plataforma ideal para proyectos de

cómputo intensivo. NVIDIA Jetson Nano (Nvidia, 2022) es una computadora pequeña pero poderosa para aplicaciones de inteligencia artificial integradas con una GPU Maxwell integrada de 128 núcleos, una CPU ARM A57 de cuatro núcleos de 64 bits, 4 GB de memoria LPDDR4 y soporte de E/S de alta velocidad que permite el uso de múltiples A-nets. para redes neuronales que ejecutan aplicaciones como clasificación de imágenes, detección de objetos, segmentación y procesamiento de voz en paralelo.

Figura 2.33: NVIDIA Jetson Nano.

Fuente: Nvidia (2022).

2.2.18. Herramientas computacionales

2.2.18.1. Python:

Python (2023). Es un lenguaje de programación de propósito general, interpretado y de alto nivel. Su uso ha crecido en los últimos años y es uno de los mejores lenguajes de programación utilizados en el desarrollo de software en la actualidad. Python está disponible en varias plataformas y sistemas operativos, entre los que podemos destacar los más populares como Windows, Mac OS X y Linux. Con Python, se puede desarrollar software para aplicaciones científicas, comunicación en red, aplicaciones de escritorio con

interfaz gráfica de usuario (GUI), crear juegos, aplicaciones web, aplicaciones de inteligencia artificial, ciencia de datos y muchas más. Empresas y organizaciones como Light & Magic, Walt Disney, NASA, Google, Yahoo!, Red Hat y Nokia utilizan el lenguaje ampliamente para desarrollar sus productos y servicios. Esto muestra que Python se puede usar en diferentes tipos de industrias, independientemente de su tipo de negocio. Una de sus principales características es un lenguaje potente, flexible y con una sintaxis clara y concisa. Además, la compilación no toma mucho tiempo a medida que se traduce, lo que lo convierte en un lenguaje muy eficiente. Python es de código abierto y cualquiera puede contribuir a su desarrollo y distribución (Fernandez, 2013).

2.2.18.2. OpenCV:

Open Source Computer Vision Library (OpenCV, 2023) es una biblioteca de software de aprendizaje automático y visión artificial de código abierto. OpenCV se creó para proporcionar una infraestructura común para aplicaciones de visión artificial y para acelerar la visión artificial en productos comerciales. La biblioteca contiene más de 2500 algoritmos optimizados, incluido un conjunto de algoritmos de visión computacional y aprendizaje automático clásicos y de última generación. Estos algoritmos se pueden utilizar para el reconocimiento y detección de rostros, detección de objetos, clasificación de actividad humana en vídeos, seguimiento de movimiento de cámara, seguimiento de objetos en movimiento, extracción de modelos 3D de objetos, generación de nubes de puntos 3D a partir de cámaras estéreo, combinación de imágenes para alta resolución, búsqueda de imágenes similares en una base de datos, eliminación de los ojos rojos en las imágenes con flash, seguimiento de movimientos de los ojos, etc.

2.2.18.3. Tensorflow:

TensorFlow (2023). Es un sistema de aprendizaje automático que funciona en entornos heterogéneos y a gran escala. Su modelo computacional se basa en diagramas de flujo de

datos de estado mutable. Los nodos de gráficos se pueden asignar a diferentes máquinas en un clúster y a CPU, GPU y otros dispositivos en cada máquina. TensorFlow es compatible con una serie de aplicaciones, pero se centra especialmente en el entrenamiento y la inferencia de redes neuronales profundas. Sirve como plataforma para la investigación e implementación de sistemas de aprendizaje automático en muchos campos, como el reconocimiento de voz, la visión artificial, la robótica, la recuperación de información y el procesamiento del lenguaje natural (Abadi, 2016).

2.2.18.4. ONNX:

Open Neural Network Exchange (ONNX, 2019) es un formato abierto diseñado para representar modelos de aprendizaje automático. ONNX define un conjunto común de operadores (los componentes básicos de los modelos de aprendizaje automático y aprendizaje profundo) y un formato de archivo común para que los desarrolladores de IA puedan usar los modelos con diferentes frameworks (Tensorflow, Pytorchs, etc), herramientas, tiempos de ejecución y compiladores.

Capítulo 3

Planteamiento del sistema

En esta sección, se determinan los requisitos de desarrollo del sistema, su lógica y subsistemas que lo conforman. Donde se establece que es necesario monitorear, detectar y alertar la presencia de somnolencia en el conductor.

3.1. Requerimientos del sistema

La principal función del sistema detector de somnolencia en el presente trabajo de investigación es monitorear, detectar y alertar en tiempo real la presencia de somnolencia en el conductor para poder evitar los siniestros viales. El sistema en cuestión, se divide en 2 partes:

- **Diseño:** Corresponde al proceso de creación del software.
- **Implementación:** Corresponde al proceso de creación del hardware.

Los requerimientos del sistema consta de 3 partes: requerimientos generales, requerimientos de diseño (software) y requerimientos de implementación (hardware). Donde, para realizar el diseño e implementación del sistema se debe cumplir con dichos requerimientos que se muestran a continuación.

3.1.1. Requerimientos generales

- Precisión de detección: El sistema debe tener un alto grado de precisión en la detección de somnolencia, minimizando los falsos positivos y falsos negativos.
- Tiempo de respuesta rápido: El sistema debe ser capaz de detectar la somnolencia en tiempo real y dar avisos oportunos para evitar que se produzcan situaciones de peligro.
- Adaptabilidad: El sistema debe poder adaptarse a diferentes usuarios, teniendo en cuenta las diferencias individuales en los patrones de sueño y la somnolencia.
- Portabilidad y comodidad: El sistema debe ser portátil, ligero y cómodo para su uso en diferentes tipos de automóviles. Donde, el sistema no debe ser intrusivo ni causar molestias al usuario durante su uso prolongado.
- Robustez: El sistema debe ser robusto a los cambios de las condiciones de iluminación en la cabina del conductor.
- Escalable y replicable: El sistema debe tener la capacidad de actualización para incorporar mejoras, correcciones y nuevas funcionalidades a lo largo del tiempo. También el sistema debe tener la facilidad de ser replicable en distintos tipos de entornos y situaciones de acuerdo a las necesidades del usuario (conductor).

3.1.2. Requerimientos de diseño (software)

- Sistema operativo: El software debe ser compatible con el sistema operativo específico en el que se ejecutará, como Windows o Linux.
- Interfaz de usuario: Una interfaz de usuario intuitiva y fácil de usar para que los conductores interactúen con el sistema, vean alertas y tomen medidas.
- Adquisición y procesamiento de datos: Software capaz de recopilar datos en tiempo real de los sensores y procesarlos mediante algoritmos de detección de somnolencia.

- Algoritmos de detección: El software debe incluir algoritmos avanzados de detección de somnolencia que analicen y procesen datos provenientes de sensores para identificar patrones asociados con la somnolencia, como movimientos oculares, parpadeo, y bostezo.

3.1.3. Requerimientos de implementación (hardware)

- Sensores de detección: Cámaras para rastrear los ojos y la cabeza, sensores de movimiento en el volante, sensores biométricos para medir la frecuencia cardíaca, etc.
- Unidad de procesamiento: Un microprocesador o una unidad de procesamiento dedicada para analizar los datos en tiempo real y ejecutar algoritmos de detección.
- Pantalla o interfaz de visualización: Pantalla en el tablero para mostrar alertas visuales y mensajes interactivos al conductor.
- Sistema de notificación: Altavoces o bocinas para generar alertas sonoras.
- Fuente de energía: El sistema requerirá una fuente de energía para operar, que podría ser la batería del vehículo u otra fuente de alimentación.

3.2. Lógica del sistema

La lógica del sistema es una metodología híbrida que se muestra en la Figura 3.1, el cual se propone para la detección de somnolencia del conductor en tiempo real. El sistema propuesto tiene 6 fases para para su funcionamiento, los cuales se describen de forma general a continuación.

En la fase 1 se hace la adquisición de imagen del sistema que son los datos de entrada, para eso se captura las imágenes con una cámara de alta resolución y alta frecuencia de adquisición e iluminación infrarroja (NIR), donde sin importar la presencia o ausencia de luz,

la iluminación infrarroja de la cámara estará siempre activa para la constante visualización de la cara del conductor enfocando los ojos y la boca.

Luego, en la fase 2, para detectar los puntos del rostro se hace mediante Facial landmark detection de MediaPipe Face Mesh, el cual entrega 468 puntos faciales, de esos puntos interesa solo 8 puntos referentes a la boca y 4 puntos a los extremos de los ojos, esto para poder hacer la fase 3.

Luego de la detección de los puntos faciales, en la fase 3 se hace la selección de las Regiones de Interés (ROI), este se divide en 2. El primer ROI corresponde a la zona de los ojos, entre las cejas y mas arriba de la punta de la nariz, cuyos puntos que nos da Mediapipe Face Mesh son: 63, 117, 293 y 346. El segundo ROI corresponde a la zona de la boca, donde solo se hace uso de 8 puntos que son: 11, 16, 61, 73, 180, 291, 303 y 404.

En la fase 4 se hace la extracción de los ROIs establecidos por esos puntos. Donde, para el ROI de los ojos, se propone una técnica de mejoramiento del área que lo rodea, esto con el propósito que cuando el conductor realice movimientos laterales y de inclinación de la cabeza, no se pierda la información del área de los ojos. Mientras que, para la boca, solo se extrae la posición de los 8 puntos para poder realizar el cálculo de la apertura bucal mediante distancias de esos puntos. En la fase 5 se pone el método a usar para cada ROI. Donde del primer ROI que es la zona de ojos, se hace la clasificación mediante Redes Neuronales Convolucionales (CNN), aplicando las arquitecturas CNN basadas en InceptionV3, VGG16 y ResNet50V2 mediante Transfer Learning, así también se propone dos arquitecturas CNNs llamadas “Drowsiness Detection-AI (DD-AI)” y “Drowsiness Detection-AI-G (DD-AI-G)”, las 5 CNN hacen una clasificación binaria de los ojos (abiertos y cerrados). El segundo ROI que corresponde a los puntos de la boca sirve para detectar los bostezos, esta detección se hace mediante el Mouth Aspect Ratio (MAR), donde mediante un umbral se determina si el conductor presenta bostezo o no. En esta fase 5 se ve la combinación híbrida del método propuesto, por CNN y MAR para los ojos y boca respectivamente.

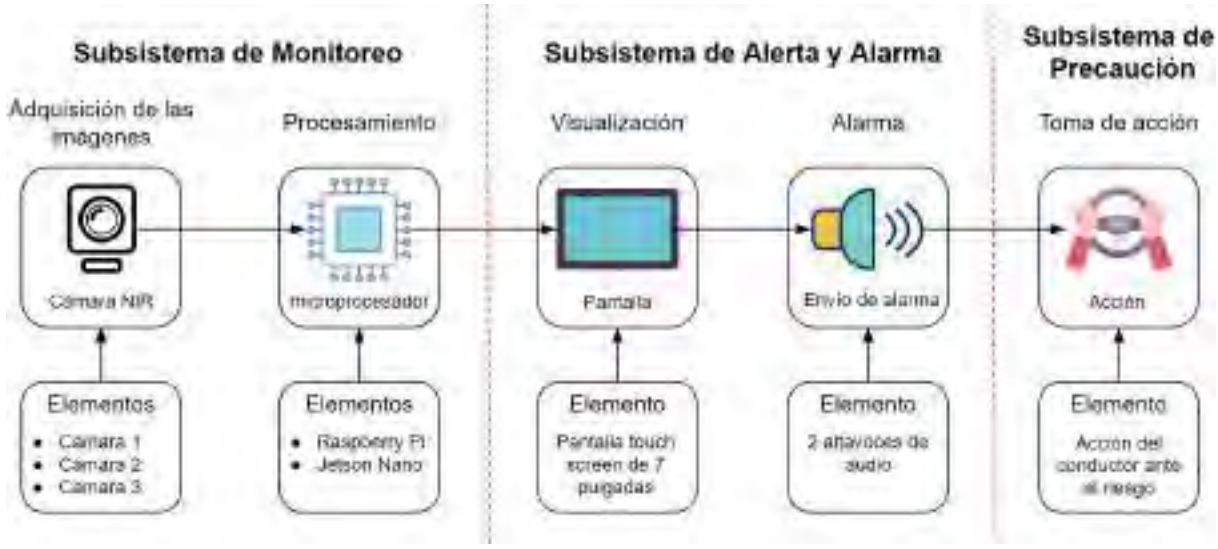
Finalmente, mediante los resultados obtenidos de la clasificación de la fase 5, se activa una alarma sonora en la fase 6, el cual avisa al conductor si presenta síntomas de somnolencia mediante el estado de los ojos y/o la boca. Adicionalmente se puede ver el estado del conductor en tiempo real mediante una pantalla de 7 pulgadas, esto es importante al viajar con un acompañante que puede observar constantemente al conductor.

Figura 3.1: Lógica general del sistema - propuesta híbrida para la detección de somnolencia.

3.3. Subsistemas del sistema detector de somnolencia

El sistema detector de somnolencia, lo conforman tres subsistemas como se puede observar en la Figura 3.2, los cuales se describen a continuación:

Figura 3.2: Subsistemas que conforman del sistema detector de somnolencia.



3.3.1. Subsistema de monitoreo

Este subsistema es, la parte del sistema encargado de recopilar y analizar datos en tiempo real relacionados con el conductor y su comportamiento para detectar signos de somnolencia o fatiga. Este subsistema utiliza tecnología específica para supervisar constantemente diversas señales físicas y de comportamiento del conductor. El objetivo principal del subsistema de monitoreo es identificar patrones y cambios que puedan indicar que el conductor está en riesgo de quedarse dormido mientras está al volante.

Las características clave del subsistema de monitoreo en un sistema detector de somnolencia pueden incluir:

- Cámaras y sensores de visión: Utilizando cámaras y sensores de visión, el sistema puede rastrear la posición de los ojos, detectar parpadeos, evaluar la dirección de la mirada y observar la posición y movimiento de la cabeza del conductor.

- Algoritmos de análisis de datos: Estos algoritmos procesan los datos recopilados de las cámaras y aplican modelos de inteligencia artificial para identificar posibles señales de somnolencia. Estos incluyen la detección de patrones de parpadeo y análisis de bostezo.

- unidad de procesamiento dedicada: El microprocesador desempeña un papel fundamental en el subsistema de monitoreo al procesar y analizar los datos en tiempo real para identificar signos de somnolencia y generar respuestas que ayuden a mantener la seguridad en la carretera.

3.3.2. Subsistema de alerta y alarma

El subsistema de alerta y alarma en un sistema detector de somnolencia es la parte encargada de notificar al conductor cuando se detectan signos de somnolencia o fatiga. Su objetivo principal es alertar al conductor para que tome medidas inmediatas y evite

situaciones peligrosas en la carretera. Este subsistema utiliza diversos métodos para comunicarse con el conductor y captar su atención de manera efectiva.

Algunas características dentro de este subsistema son:

- Notificaciones en la pantalla: Si el vehículo está equipado con una pantalla en el tablero, se pueden mostrar mensajes visuales que indiquen la detección de somnolencia y la necesidad de descansar.
- Alertas sonoras: Las alertas audibles, como tonos, bocinas o mensajes de voz pregrabados, pueden ser emitidas para advertir al conductor sobre su estado de somnolencia. Los tonos pueden aumentar en intensidad si la somnolencia persiste.

3.3.3. Subsistema de precaución

Este subsistema se refiere a la parte integral que interviene en la interacción con el conductor para evitar o reducir los efectos de la fatiga y la somnolencia durante la conducción. Este subsistema está diseñado para influir en las decisiones y acciones del conductor de manera activa y proactiva. En lugar de simplemente advertir sobre la somnolencia, el subsistema de precaución impulsa al conductor a tomar medidas inmediatas para mantenerse alerta y seguro en la carretera.

Capítulo 4

Diseño del sistema detector de somnolencia

El diseño de un sistema detector de somnolencia es la fase inicial en la creación de una solución tecnológica para identificar signos de somnolencia en las personas. Involucra la selección de sensores, algoritmos y la estructura del sistema, junto con la definición de criterios de detección y respuestas.

4.1. Elección de elementos para el diseño del sistema

La elección de los elementos pueden ser categorizados en función de su utilidad, variando según la tarea que desempeñen, y pueden clasificarse en:

- Adquisición.
- Base de datos.
- Características faciales.

El elemento de adquisición hace referencia al sensor primario del sistema detector de somnolencia que es la cámara, y el elemento de base datos hace referencia a los datos a usar para crear los elementos del dataset.

4.1.1. Elección de la cámara

Como se mencionó en la sección 2.2.16, se optó por tres cámaras de la empresa ELP, las cuales cumplen con el límite de intensidad de radiación LED de $10mW/cm^2$ a una distancia de 50 cm a 70 cm, como se discutió previamente en la sección 2.2.7.2. A continuación, se proporcionan detalles de las características de estas tres cámaras para facilitar su evaluación y selección, que se presenta en la Tabla 4.1.

Tabla 4.1: Comparación de las tres cámaras RGB-NIR.

	Cámara 1	**Cámara 2**	**Cámara 3**
Sensor	1/3 AR0331	CMOS OV2710	OV4689
Resolución máxima	2048x1536p	1920x1080p	1920x1080p
Formato de imagen	MJPEG	MJPEG	MJPEG
Velocidad de fps	2048x1536p@15fps 1600x1200p@15fps 1920x1080p@30fps 1280x1024p@30fps 1280x720p@30fps 800x600p@30fps 640x480p@30fps	1920x1080p@30fps 1280x1024p@30fps 1280x720p@60fps 1024x768p@30fps 800x600p@60fps 640x480p@120fps 352x288p@120fps 320x240p@120fps	1920x1080p@60fps 1280x720p@120fps 640x360p@260fps
Iluminación	0.05lux	0.05lux	0.1lux
IR LED Board	850nm o 940nm	850nm o 940nm	850nm o 940nm
AEC, AEB, AGC	Soporta	Soporta	Soporta
Tensión	DC 5V	DC 5V	DC 5V
Corriente	140mA - 220mA	120mA - 220mA	150mA - 185mA
Precio	S/. 373.63	S/. 246.61	S/. 343.98

Al comparar las 3 cámaras, considerando las características proporcionadas y para un rendimiento más avanzado, se elige la cámara 2 ELP-USBFHD05MT-KL36IR por su combinación excepcional de velocidad de cuadros adecuada y una iluminación integrada con láser infrarrojo. Las distintas resoluciones de píxeles de la ELP-KL36IR permite capturar detalles con claridad, lo que resulta fundamental para la detección precisa de pequeños cambios en los patrones faciales asociados con la somnolencia. También las distintas velocidades de cuadros (fps) garantiza una captura fluida y en tiempo real, permitiendo una evaluación continua y precisa de las condiciones del conductor.

La ELP-KL36IR se destaca significativamente en términos de relación calidad-precio en comparación con las otras dos cámaras. Además de esto, ofrece variadas resoluciones de píxeles y velocidades de cuadros que le confieren versatilidad. Una característica especialmente destacable es su integración de un láser infrarrojo a 850nm, el cual proporciona una fuente de luz coherente y precisa para la detección infrarroja. Esta característica es de suma importancia para garantizar mediciones fiables y coherentes, especialmente en condiciones de baja iluminación y en presencia de obstrucciones como lentes de sol. La presencia del láser infrarrojo potencia la capacidad de la cámara para detectar los ojos del conductor, independientemente de las condiciones lumínicas y la existencia de objetos en los ojos, permitiendo una identificación temprana y precisa de los signos de somnolencia.

En resumen, la ELP-KL36IR se destaca por su calidad-precio, combinación de resoluciones de alta calidad, velocidades de cuadros y la incorporación de láser infrarrojo para iluminación. Estas características trabajan en conjunto para proporcionar un rendimiento óptimo en un sistema de detección de somnolencia, lo que la convierte en una elección sólida y confiable para esta aplicación crítica de seguridad vial. Donde en el Anexo I se puede ver el datasheet completo de la cámara ELP-KL36IR.

4.1.2. Elección de la base de datos

Para diseñar de manera eficiente el sistema de detección de somnolencia, resulta imperativo disponer de una base de datos particularmente adaptada a la aplicación en cuestión. Dicha base de datos debe comprender registros de individuos, específicamente conductores, que muestren señales de somnolencia, enfocándose especialmente en el estado de sus ojos y boca. La Tabla 4.2 detalla las características predominantes de las tres bases de datos mencionadas en la sección 2.2.15.

Tabla 4.2: Comparación de las bases de datos.

	NTHU	**UTA**	**NITYMED**
Cantidad	90 videos	180 videos	130 videos
Resolución	640x480p	1920x1080p	1920x1080p, 1280x720p
Tipo	IR	RGB	RGB
Formato	AVI	MOV	MP4
Velocidad de fps	15 y 30fps	30fps	25fps
Anotaciones	somnolientos y no somnolientos	estado de alerta, baja vigilancia y somnolencia	Bostezos y Microsueños
Tipo de conducción	Simulado	Simulado	Real
Condición de luz	Constante	Constante	Variable
Disponibilidad	Mediante solicitud y acuerdo de uso	Pública y de acceso gratuito	Pública y de acceso gratuito

Viendo la comparación de las tres bases de datos en la Tabla anterior, se hizo la elección de la base de datos NITYMED como la preferida para la detección de somnolencia, se respalda por una serie de factores clave que la hacen destacar en comparación con las otras opciones.

Esta base de datos se distingue por su equilibrio entre la cantidad, la resolución y los dos tipos de anotaciones, lo que la convierte en una herramienta valiosa para desarrollar y evaluar algoritmos de detección de somnolencia de manera efectiva.

Su tamaño adecuado de 130 vídeos de conductores con bostezo y somnolientos permite obtener una cantidad significativa de datos de muestra, lo que garantiza la robustez y confiabilidad de los resultados obtenidos. Además, las resoluciones de 1920x1080 y 1280x720 píxeles proporciona detalles suficientes para capturar los rasgos faciales relevantes, que son esenciales para identificar los signos tempranos de somnolencia.

Lo que realmente destaca es la presencia de un entorno de conducción real, donde se ve la variación lumínica en los rostros de los conductores en diferentes movimientos de la cabeza. Esta información es esencial para el entrenamiento y la validación de algoritmos de detección, ya que permite relacionar de manera precisa los patrones visuales con los estados de somnolencia.

Por último, su disponibilidad pública y gratuita fomenta la colaboración y la investigación en el campo de la detección de somnolencia, lo que es fundamental para avanzar en la seguridad vial. En resumen, la base de datos NITYMED sobresale por su equilibrio entre cantidad, tipos de anotaciones, disponibilidad y un entorno de conducción real, lo que la convierte en la elección preferida para desarrollar soluciones efectivas de detección de somnolencia para la presente tesis.

4.1.3. Elección del método de extracción de características faciales

Con el propósito de detectar la somnolencia en los conductores, resulta fundamental identificar las características faciales que permitan determinar el estado de sus ojos y boca. En este sentido, se emplean tres métodos para la extracción de estas características, tal como se detalla en la sección 2.2.10. En la Tabla 4.3 se muestra una comparación de las características principales de los tres métodos.

Tabla 4.3: Comparación de los métodos de extracción de características faciales.

	Haar cascade	**Dlib**	**Mediapipe**
Puntos faciales	Detecta características faciales básicas, como ojos y boca.	Detecta 68 puntos faciales detallados.	Detecta 478 puntos faciales detallados.
Compatibilidad	Altamente compatible con sistemas embebidos y microprocesadores de recursos limitados	Requiere recursos moderados a altos. Puede ser viable en sistemas embebidos más potentes.	Requiere recursos significativos, adecuado para sistemas embebidos con mayor capacidad de procesamiento.
Robustez	Menos robusto en condiciones de iluminación variable y ángulos extremos.	Altamente robusto, capaz de manejar variaciones de iluminación y ángulos.	Muy robusto en el seguimiento en tiempo real de características faciales en entornos cambiantes.

De acuerdo a las características principales mostradas anteriormente y basándose en estudios como: Kao and Chan (2022); LearnOpenCV (2022). Se hace la elección de MediaPipe como la mejor opción para la extracción de características faciales en la detección de somnolencia, demostrando su alta robustez y capacidad para capturar detalles faciales en tiempo real. Aunque puede requerir más recursos que Haar Cascade y dlib, MediaPipe sobresale en situaciones cambiantes y puede detectar una gran cantidad de puntos faciales detallados, lo que es esencial para identificar signos sutiles de somnolencia.

Su enfoque en el seguimiento en tiempo real lo hace ideal para detectar cambios dinámicos en los ojos y boca, lo que es crucial en aplicaciones de detección de somnolencia. Si bien su compatibilidad con sistemas embebidos de mayor capacidad de procesamiento puede ser un factor limitante, su alta robustez y capacidad de captura detallada justifican su elección como la mejor herramienta para esta aplicación específica.

4.2. Método del diseño del sistema de detección de somnolencia

El diseño de un sistema de detección de somnolencia es esencial para abordar uno de los desafíos más críticos en la seguridad vial: la somnolencia del conductor. Este enfoque tiene como objetivo desarrollar una solución efectiva que utilice tecnologías de procesamiento de imágenes y análisis facial para identificar de manera temprana los indicios de somnolencia en tiempo real. A lo largo de una secuencia de fases interconectadas, que abarcan desde la recopilación y preparación de datos hasta la implementación en condiciones reales, el proceso de diseño se orienta hacia la garantía de precisión y confiabilidad en la detección. Este esfuerzo contribuye significativamente a prevenir accidentes y a elevar la seguridad en las carreteras.

Este capítulo presenta los métodos empleados en diseño de la lógica propuesta del sistema (Fig. 3.1). Se enfocará particularmente en la fase 5, ya que en esta fase se desarrolla el enfoque híbrido propuesto, el cual combina las capacidades de las redes neuronales convolucionales (CNN) y la relación de aspecto de la boca (MAR). Para este propósito, se llevará a cabo el entrenamiento de cinco arquitecturas distintas de CNN en un ordenador con una CPU AMD Ryzen 7 5800H, una GPU NVIDIA GeForce RTX 3060 de 6GB y 16GB de RAM. Además, se analizarán los resultados preliminares obtenidos de estas cinco arquitecturas CNN, lo que permitirá determinar cuál de ellas es la más adecuada para la implementación definitiva del sistema. A continuación se describen todas las fases de la lógica del sistema.

4.2.1. Fase 1: Adquisición de imágenes

La obtención de las imágenes para el sistema se lleva a cabo a través de la cámara NIR seleccionada, la cual está instalada en el tablero del automóvil, ubicada detrás del volante y enfocada hacia el conductor. Esta cámara captura el vídeo en tiempo real del conductor en diversas condiciones lumínicas, abarcando desde el día hasta el atardecer y la noche. El enfoque se centra en un único rostro, específicamente el del conductor, que constituye el elemento primordial a analizar en este contexto (Figura 4.1).

Figura 4.1: Adquisición de imágenes.

Cuyo pseudocódigo para la lectura de la cámara se muestra en Algorithm 1 y el código en Python se muestra en Anexo 2.

Algorithm 1 Lectura de una cámara en Python usando OpenCV

Require: OpenCV
1: **import** cv2
2: captura ← $cv2.VideoCapture(0)$ ▷ Abre la cámara
3: **while verdadero do**
4: ret, imagen ← $captura.read()$ ▷ Lee un cuadro de video
5: **Mostrar la imagen en una ventana** ▷ Utiliza funciones de OpenCV
6: **if** presionar la tecla "q" **then**
7: **break** ▷ Termina el ciclo si se presiona la tecla "q"
8: **end if**
9: **end while** ▷ Libera la cámara y cierra las ventanas

4.2.2. Fase 2: Detección de puntos faciales

Se utiliza la técnica de detección de landmarks faciales mediante MediaPipe Face Mesh para identificar 468 puntos en la cara. Estos puntos se superponen en cada cuadro del video

de entrada (Figura 4.2). Se seleccionan los puntos pertinentes para localizar con precisión los ojos y la boca.

Figura 4.2: Detección de puntos faciales con MediaPipe.

(a) Imagen de entrada. (b) Mask. (c) Puntos faciales.

A continuación se muestra el pseudocódigo para la detección de puntos faciales en Algorithm 2 y el código en Python se muestra en Anexo 3.

Algorithm 2 Uso de MediaPipe Face Mesh

Require: Iniciar cámara

```
Iniciar la captura de imágenes
while La cámara esté encendida do
    Capturar un fotograma de la cámara
    Procesar el fotograma con MediaPipe para detecciones
    if Se ha realizado una detección then
        Almacenar la imagen capturada
        Mostrar la imagen en pantalla
    end if
end while
```

4.2.3. Fase 3: Selección de ROI

Después de obtener los 468 puntos faciales, se realiza una selección de cuatro puntos clave para la zona de los ojos: 63, 117, 293 y 346, y ocho puntos para los labios: 11, 16, 61, 73, 180, 291, 303 y 404. Estos puntos se extraen de la máscara proporcionada por MediaPipe Face Mesh. A partir de estos puntos, se define la región de interés (ROI) para los ojos, mientras que para los labios se captura únicamente la posición en los ejes "x" e "y". La Figura 4.3 ilustra la implementación del método propuesto para mejorar el área de la ROI de los ojos.

Los puntos mencionados previamente se colocan en el rostro como se observa en la Figura 4.3a. Sin embargo, al unir los cuatro puntos que delimitan el área de la región de los ojos, se advierte una asimetría en la forma de la región, lo que podría dificultar la extracción precisa del ROI, como se visualiza en la Figura 4.3b.

Para resolver este inconveniente, se propone una técnica de corrección mejorada basada en la comparación de distancias, cuya base se toma de estudios anteriores, como de: Ahmed and Laskar (2022); Florez et al. (2023); Kumari and KR (2021); Liu et al. (2017); Pandey and Muppalaneni (2022). Inicialmente, se toman las coordenadas "x" e "y" de los puntos extremos superiores derechos de cada ojo (puntos 63 y 336 para los ojos derecho e izquierdo, respectivamente). Posteriormente, se calculan las distancias entre los puntos extremos y superiores de cada ojo. Por ejemplo, para el ojo derecho, se obtiene la distancia desde el punto 63 hasta el punto 117 (d1) y desde el punto 63 hasta el punto 107 (d2), añadiendo un nuevo punto en la posición 107. Similarmente, para el ojo izquierdo, se calculan las distancias desde el punto 336 al punto 293 (d3) y desde el punto 293 al punto 346 (d4). Una vez que se tienen estos valores, se procede a comparar las posiciones de los puntos extremos superiores derechos de ambos ojos, en sus respectivas componentes, como se ilustra en la Figura 4.3c.

Para calcular la distancia entre cada punto asignado de los ojos, se hace un código que define una función llamada "Distance" que calcula la distancia euclidiana entre dos puntos en un plano bidimensional. Los puntos se representan como pares de coordenadas (x, y).

La función toma dos argumentos, p1 y p2, que representan los dos puntos entre los cuales se desea calcular la distancia. Cada punto se descompone en sus componentes x e y.

La fórmula matemática utilizada para calcular la distancia euclidiana entre dos puntos (x1, y1) y (x2, y2) es la siguiente:

$$dist = \sqrt{(x2 - x1)^2 + (y2 - y1)^2} \tag{4.1}$$

En el código, se utiliza la función "math.hypot" para calcular la hipotenusa de un triángulo rectángulo con los lados $x2 - x1$ y $y2 - y1$, lo que es equivalente a la fórmula de la distancia

euclidiana. Finalmente, el valor de la distancia calculada se devuelve como resultado de la función. Se muestra el pseudocódigo en Algorithm 3.

Algorithm 3 Calculo de la distancia

```
1: function DISTANCE(p1, p2)          ▷ Se define la función Distance para 2 puntos
2:     x1, y1 ← p1                    ▷ el punto p1 tiene las coordenadas x1 y y1
3:     x2, y2 ← p2                    ▷ el punto p2 tiene las coordenadas x2 y y2
4:     dist ← math.hypot(x2 − x1, y2 − y1)   ▷ se calcula la distancia euclidiana
5:     return dist                    ▷ devuelve el valor calculado
6: end function
```

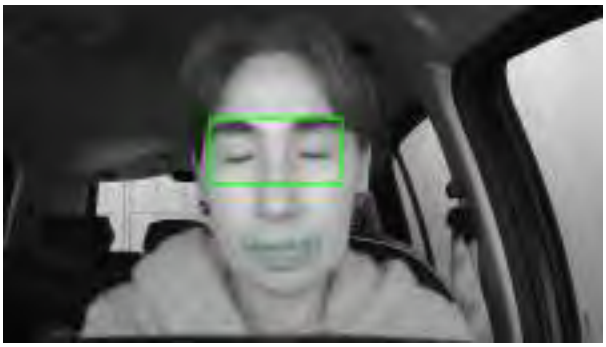
Finalmente, se dibuja un rectángulo corregido que abarca el área de la región de los ojos. Esta corrección asegura que, incluso en distintas posiciones de la cabeza del conductor, como movimientos rotacionales y de inclinación, la información crucial del ROI de los ojos se mantenga intacta y disponible para su posterior clasificación. La Figura 4.3d muestra el ROI corregido mediante el método propuesto.

Figura 4.3: Corrección de ROI de la zona de los ojos.

(a) Puntos de las dos regiones de interés.

(b) Región de interés ocular asimétrica.

(c) Método de corrección de la región de interés.



(d) ROI del ojo corregido y ROI de la boca.

En el Algoritm 4, se presenta el pseudocódigo como contribución al a la corrección de la región de interés de los ojos. El código completo se puede encontrar en el Anexo 4.

Algorithm 4 : Correccion de ROI

Require: Points: $[63, 117, 293, 346, 107, 336]$ ▷ puntos de la región ocular

Ensure: $Corrected\_ROI$

$x_a, y_a = P_{336}[x], P_{336}[y]$ ▷ componentes "x" e "y" de los puntos superiores derechos

$x_b, y_b = P_{63}[x], P_{63}[y]$

$d_1 = int(distance(P_{63}, P_{117}))$ ▷ distancias de los puntos extremos y superiores

$d_2 = int(distance(P_{63}, P_{107}))$

$d_3 = int(distance(P_{336}, P_{293}))$

$d_4 = int(distance(P_{293}, P_{346}))$

if $x_a > x_b$ **then** ▷ comparación de distancias en componente "x"

    $start\_x, end\_x = x_b, (x_a + d_3)$

else

    $start\_x, end\_x = x_a, (x_b + d_2)$

end if

if $y_a > y_b$ **then** ▷ comparación de distancias en componente "y"

    $start\_y, end\_y = y_b, (y_a + d_4)$

else

    $start\_y, end\_y = y_a, (y_b + d_1)$

end if

if $(end\_x - start\_x) > 10 \& (end\_y - start\_y) < 400$ **then**

    $start\_x, start\_y = start\_x - 10, start\_y - 10$

    $end\_x, end\_y = end\_x + 10, end\_y + 10$

    $Corrected\_ROI = [start\_y : end\_y, start\_x : end\_x]$ ▷ ROI corregido

end if

A continuación, se explica paso a paso lo que hace el algoritmo:

1. Requisito y resultado esperado: El algoritmo recibe como entrada un conjunto de puntos representando las coordenadas de varios puntos relevantes en la región de los ojos (se denotan como P_{63}, P_{117}, P_{293}, P_{346}, P_{107} y P_{336}). La salida esperada es una "Corrected ROI" (Región de Interés Corregida).

2. Cálculo de componentes x e y de los puntos extremos: Se calculan las componentes "x" e "y" de los puntos extremos superiores derechos (P_{336} y P_{63}) y se almacenan en x_a, y_a, x_b y y_b.

3. Cálculo de distancias: Se calculan las distancias entre los puntos extremos mencionados anteriormente y se almacenan en d_1, d_2, d_3 y d_4.

4. Comparación de distancias en componentes x: Se verifica cuál de las distancias calculadas en el paso anterior es mayor. Si la distancia asociada con P_{336} es mayor, se ajustan las coordenadas de inicio y final en la dirección "x" usando x_b y x_a, junto con d_3. Si la distancia asociada con P_{63} es mayor, se hace el ajuste con x_a y x_b, junto con d_2.

5. Comparación de distancias en componentes y: Similar al paso anterior, aquí se compara cuál de las distancias (d_1 y d_4) es mayor. Se ajustan las coordenadas de inicio y final en la dirección "y" utilizando y_b y y_a, junto con las distancias correspondientes.

6. Creación de la ROI corregida: Si la diferencia entre las coordenadas finales y las coordenadas iniciales en la dirección "x" es mayor que 10, y la diferencia en la dirección "y" es menor que 400, entonces se realiza una corrección en las coordenadas de inicio y final. Luego se crea una "Corrected ROI" seleccionando la subimagen en la región corregida.

   Los valores de 400 y 10 fueron determinados a través de pruebas realizadas con movimientos de rotación de la cabeza hacia la derecha e izquierda, así como movimientos de extensión y flexión a ángulos más amplios. Estos valores se ajustaron

empíricamente para asegurar que la región de interés se adapte de manera adecuada a distintos movimientos y posturas de la cabeza.

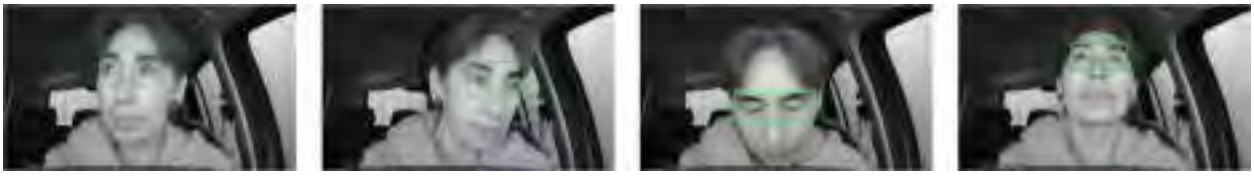
La aplicación del ROI corregido se puede ver en la Figura 4.4, donde se muestra distintos escenarios comunes en la postura de la cabeza del conductor.

Figura 4.4: ROIs oculares corregidos en diferentes posturas de la cabeza del conductor.

(a) Frames originales.

(b) Área inicial de ROI.

(c) Comparación de los ROI corregidos y los ROI iniciales.



(d) ROI ocular corregido.

La figura anterior muestra la posición de la cabeza a la derecha (primera columna). Posición de la cabeza hacia la izquierda (segunda columna). Posición de la cabeza hacia abajo (tercera columna) y posición de la cabeza hacia arriba (cuarta columna).

4.2.4. Fase 4: Extracción de ROI

En esta etapa, se procede a la extracción de las regiones de interés (ROIs) correspondientes a los ojos y la boca, siguiendo el método de corrección previamente mencionado para el ROI del área de los ojos. Como resultado, se obtienen dos ROIs: uno para los ojos y otro para la boca. El primer ROI, destinado a los ojos, abarca dos clases distintas: "Ojos abiertos" y "Ojos cerrados" (Despierto y No despierto), dependiendo del estado del conductor. Estos ROIs se utilizarán como entradas para la red neuronal convolucional (CNN), la cual se encargará de la clasificación para determinar si el conductor está experimentando somnolencia. Mientras tanto, el ROI de la boca mantiene únicamente los 8 puntos relevantes para determinar si el conductor está bostezando o no (boca cerrada o boca abierta). La Figura 4.5 ilustra los ROIs correspondientes a cada región, donde se aprecia que de la boca se extraen los 8 puntos, mientras que de los ojos se obtiene una imagen completa.

Figura 4.5: ROIs de los ojos y la boca del estado del conductor.

(a) ROI de ojos abiertos y puntos en boca cerrada.

(b) ROI de ojos abiertos y puntos en boca abierta.

4.2.5. Fase 5: Método aplicado

Después de haber extraído las regiones de interés correspondientes a los ojos y la boca, es necesario aplicar los métodos de evaluación respectivos para determinar el estado de somnolencia del conductor en cada caso. Estos métodos incluyen:

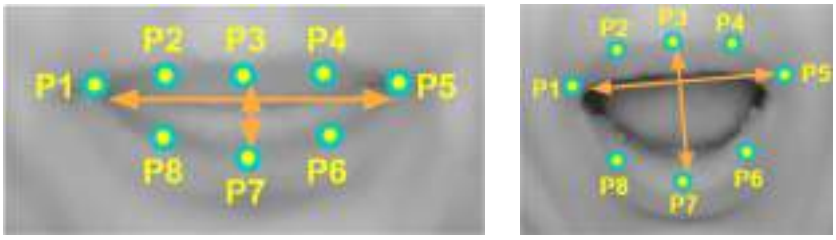
- Relación de aspecto bucal (MAR) para la detección de bostezos.
- Redes Neuronales Convolucionales para determinar el estado de los ojos.

La combinación de estos dos métodos aplicados para la detección de somnolencia conforma el modelo híbrido, cuyos detalles se exponen a continuación:

4.2.5.1. Método por relación de aspecto bucal (MAR)

La relación de aspecto bucal es un parámetro geométrico esencial para evaluar la apertura y cierre de la boca en imágenes o vídeos. En el contexto de la detección de bostezos, esta relación se determina utilizando la distancia euclidiana para comparar las dimensiones horizontales y verticales de la boca del conductor. Un aumento significativo en la relación de aspecto bucal indica una apertura más amplia de la boca, lo que puede señalar un posible bostezo. Esta métrica cuantitativa brinda una forma precisa de identificar el estado de la boca y, en consecuencia, la probabilidad de bostezo, contribuyendo así a la detección temprana de indicios de somnolencia en el conductor. Cuyos puntos se pueden ver en la Figura 4.6.

Figura 4.6: Extracción de los puntos de la boca.



(a) Puntos boca cerrada. (b) Puntos boca abierta.

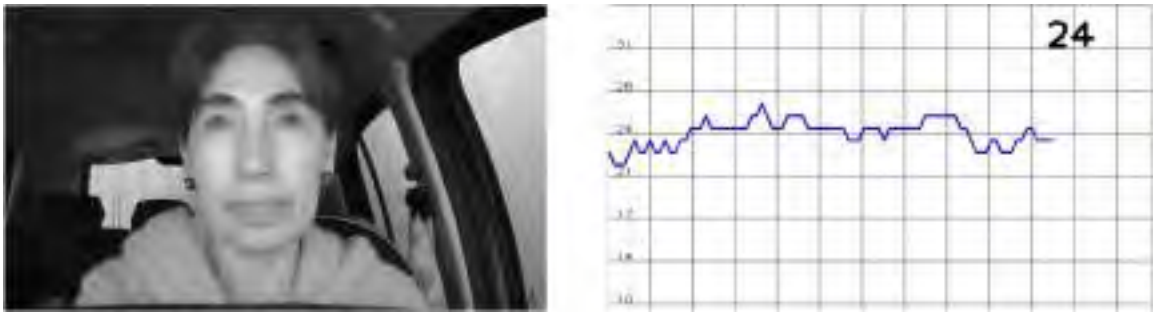
Basándose en la Ecuación 2.4, se ha implementado el código correspondiente para detectar la apertura de la boca, cuyo ejemplo se presenta el pseudocódigo Algorithm 5.

Algorithm 5 Cálculo del Índice de Relación de Apertura de Boca (MAR)

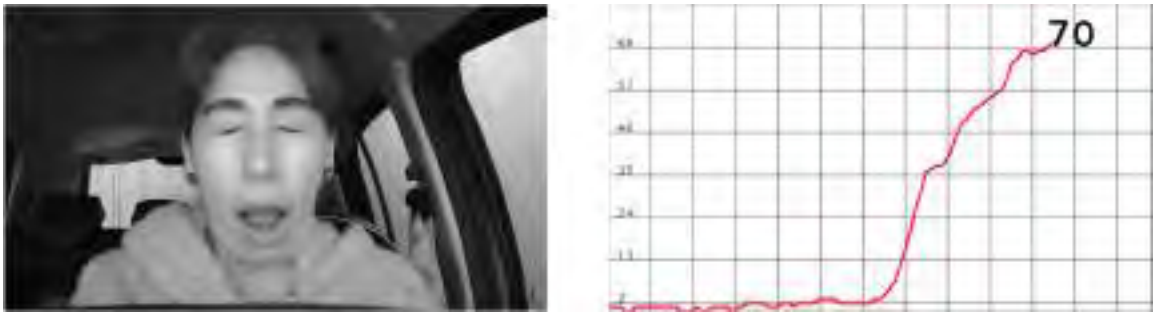
Definir los índices de puntos de la boca: $mouth = [61, 73, 11, 303, 291, 404, 16, 180]$
function MOUTH_ASPECT_RATIO($p1, p2, p3, p4, p5, p6, p7, p8$)
    $d_A \leftarrow \text{Distance}(p2, p8)$
    $d_B \leftarrow \text{Distance}(p3, p7)$
    $d_C \leftarrow \text{Distance}(p4, p6)$
    $d_D \leftarrow \text{Distance}(p5, p1)$
    return $(d_A + d_B + d_C)/(3 \times d_D)$
end function
Obtener los puntos correspondientes a los índices:
$p1 \leftarrow puntos[mouth[0]]$
$p2 \leftarrow puntos[mouth[1]]$
$p3 \leftarrow puntos[mouth[2]]$
$p4 \leftarrow puntos[mouth[3]]$
$p5 \leftarrow puntos[mouth[4]]$
$p6 \leftarrow puntos[mouth[5]]$
$p7 \leftarrow puntos[mouth[6]]$
$p8 \leftarrow puntos[mouth[7]]$
Calcular el índice de relación de apertura de boca (MAR):
$MAR \leftarrow \text{int}(mouth\_aspect\_ratio(p1, p2, p3, p4, p5, p6, p7, p8) \times 100)$
Imprimir el índice de relación de apertura de boca (MAR)

Una vez que el código se encuentra en ejecución, se procede a obtener el valor del índice de MAR. Este valor varía en un rango aproximado de 0 a 100. Cuando la boca está cerrada, el valor oscila entre 0 y 30, mientras que durante un bostezo del conductor, este valor se encuentra entre 50 y 90. Estas observaciones pueden apreciarse de manera gráfica en la Figura 4.7.

Figura 4.7: Índices de bostezo.



(a) Índice bostezo boca cerrada.



(b) Índice bostezo boca abierta.

Luego de analizar los valores del índice, se realizaron diversas pruebas para establecer un umbral de detección para el bostezo, siendo este fijado en 55. Durante este proceso, se observó que durante el habla normal del conductor, el índice MAR no supera el umbral de 55, lo que garantiza que el sistema de detección de somnolencia no arroje falsos positivos. Por lo tanto, se considera que los valores por debajo del umbral no corresponden a bostezos, mientras que aquellos iguales o mayores al umbral son identificados como bostezos.

Es importante destacar que, hasta el momento, la detección se centra exclusivamente en bostezos mediante el umbral establecido. Sin embargo, de acuerdo con Rajkumarsingh and Totah (2021), se establece que la duración típica de un bostezo oscila entre 5 y 15 segundos. Para lograr una diferenciación efectiva entre el bostezo y los movimientos normales de la boca, es necesario implementar un umbral temporal de ese rango.

En este contexto particular, se introduce un umbral temporal de 5 segundos para la detección de bostezo. En el escenario en el que la apertura de la boca del conductor supera

el umbral de 55 pero se mantiene por debajo del umbral temporal de 5 segundos, el sistema no considerará esta instancia como un bostezo. En contraste, si la apertura de la boca del conductor supera el umbral de 55 y se mantiene durante un periodo superior a los 5 segundos establecidos, el sistema lo identificará como un bostezo.

A continuación, se muestra el pseudocódigo para la detección de bostezo en Algorithm 6 y el algoritmo completo en Python se muestra en Anexo 5. Posteriormente se explica el pseudocódigo mostrado.

Algorithm 6 Detección de bostezo

```
if MAR ≥ 55 then
    contador_mouth += 1
end if
if MAR < 55 then
    contador_mouth −= 1
end if
if contador_mouth ≥ 5 then
    contador_mouth = 5
    print("BOSTEZO")
end if

if (contador_mouth > 0 and contador_mouth < 5) then
    print("Normal")
end if

if contador_mouth < 0 then
    contador_mouth = 0
end if
```

1. Medición de la apertura de la boca: Basado en el pseudocódigo del Anexo 5, se calculó el índice MAR, que es la medida de la apertura de la boca del conductor.

2. Contador de bostezos: El código utiliza una variable llamada contador_mouth para llevar un registro de cuánto tiempo la boca ha estado abierta en una secuencia continua. Inicialmente, esta variable se establece en 0.

3. Condiciones de detección:

    - Si el valor de MAR es igual o mayor a 55, se interpreta que la boca está abierta y se incrementa el contador contador_mouth en 1.
    - Si el valor de MAR es menor que 55, se considera que la boca se ha cerrado o no está suficientemente abierta, por lo que se reduce el contador contador_mouth en 1.

4. Determinación de bostezo:

    - Si el contador contador_mouth alcanza o supera el valor de 5, se establece en 5 para evitar que siga incrementando. Esto sirve como umbral de duración para considerar un bostezo genuino. Luego, se imprime "BOSTEZO" para indicar que el conductor está bostezando.
    - Si el contador contador_mouth es mayor que 0 pero menor que 5, se asume que la boca está en proceso de abrirse pero aún no ha llegado a la duración de un bostezo. En este caso, se imprime "Normal".
    - Si el contador contador_mouth es negativo (lo cual no debería ocurrir si el contador se maneja correctamente), se ajusta a 0 para evitar valores negativos.

4.2.5.2. Método por redes neuronales convolucionales (CNN)

En la detección de somnolencia mediante el estado de los ojos, se hace mediante las CNN que desempeñan un papel clave al permitir la creación de sistemas automatizados que monitorean los ojos del conductor y determinan su nivel de somnolencia en función de la apertura y movimiento de los ojos. El proceso general implica los siguientes pasos:

- Captura de datos: Se adquieren imágenes o vídeos de los ojos del conductor en tiempo real utilizando cámaras u otros dispositivos de captura. Estos datos visuales capturan la forma, posición y movimiento de los ojos.

- Preprocesamiento: Las imágenes capturadas pueden estar en diferentes tamaños y resoluciones, por lo que es esencial normalizarlas y ajustarlas a un formato estándar antes de ingresarlas a la red neuronal. Esto puede incluir operaciones como la redimensión y normalización de píxeles.

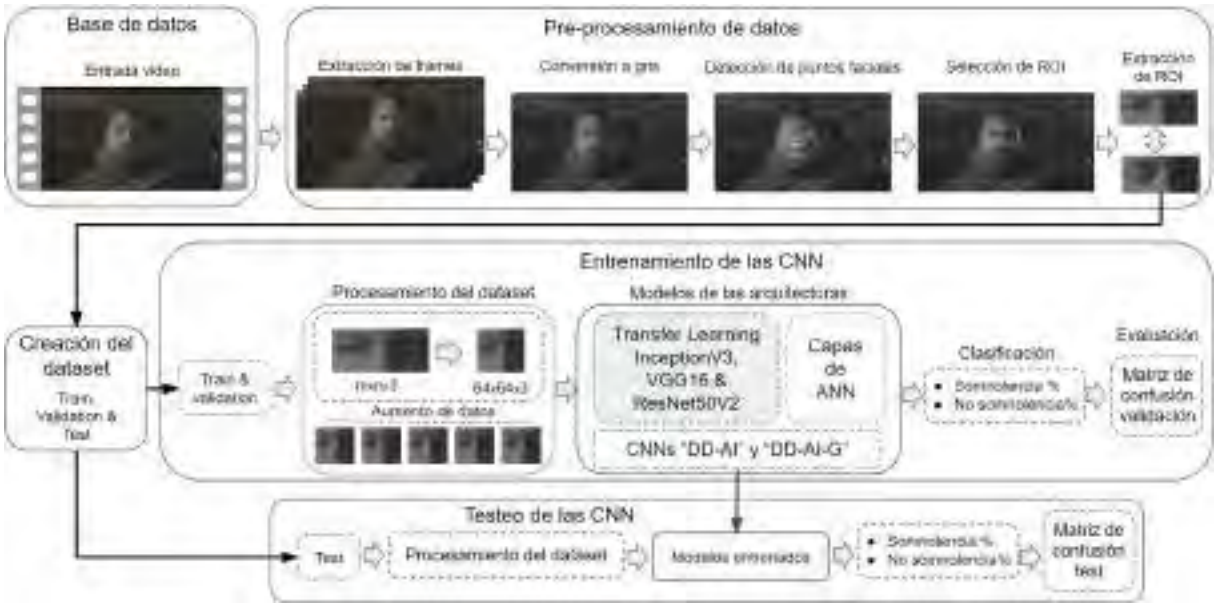
- Detección de características: Las CNN aprenden automáticamente características relevantes de las imágenes mediante capas de convolución y pooling. En el contexto de la somnolencia, estas características pueden incluir la posición de los párpados, la apertura de los ojos y otros detalles relevantes.

- Entrenamiento de la red: Se requiere un conjunto de datos etiquetado que contenga ejemplos de diferentes estados de los ojos (abiertos, cerrados, entrecerrados, etc.). Mediante el entrenamiento supervisado, la CNN ajusta sus pesos y parámetros internos para aprender a reconocer y distinguir entre estos diferentes estados.

- Clasificación: Una vez entrenada, la CNN se utiliza para clasificar nuevas imágenes de los ojos y determinar si el conductor está somnoliento o no. La red puede asignar una probabilidad a cada clase de somnolencia y tomar decisiones basadas en umbrales predefinidos.

Como se mencionó previamente en la sección 4.1.2, se ha seleccionado la base de datos NITYMED para llevar a cabo el proceso de detección de somnolencia mediante el entrenamiento de redes neuronales convolucionales. En la Figura 4.8 se ilustra el diagrama que representa el proceso de detección de somnolencia en función del estado de los ojos, en el cual se incorpora el entrenamiento de las CNN.

Figura 4.8: Procedimiento de entrenamiento mediante CNN.



Partiendo de la Figura previamente presentada, ya se tiene la base de datos NITYMED, compuesta por vídeos que tienen una duración que oscila entre 30 y 120 segundos. Durante la fase de preprocesamiento de datos, se han abordado los pasos esenciales: "Detección de puntos faciales", "Selección de ROI" y "Extracción de ROI", los cuales han sido descritos en detalle anteriormente. A continuación, se describirán los pasos adicionales necesarios para realizar el preprocesamiento del conjunto de vídeos de la base de datos, que incluyen la "Extracción de frames" y la "Conversión a escala de grises".

4.2.5.2.1. Extracción de frames: De la base de datos NITYMED, se ha focalizado el análisis en la carpeta "Full" (1080p). Dentro de esta, se encuentra la subcarpeta "microsleep", la cual contiene dos subdivisiones: "female" con 11 vídeos y "male" con 12 vídeos. De esta gama, se ha elegido 6 vídeos específicos (2 de "female" y 4 de "male") con los siguientes nombres: P1043106_na, P1043129_na, P1042751_na, P1042767_na, P1042792_na y P1043075_na.

La selección de estos 6 vídeos se basa en la notable presencia de variaciones en las condiciones de iluminación en la región de los ojos durante la conducción. Adicionalmente,

la mayoría de estos vídeos exhiben sujetos con gafas, un escenario propicio para aprovechar la cámara NIR (Near-Infrared) para la detección del estado de los ojos mediante material cristalino.

Luego de esta etapa, el siguiente paso consiste en la extracción de los frames de los 6 vídeos seleccionados. Como bien sabemos, un vídeo es una secuencia de imágenes a lo largo del tiempo. Por lo tanto, podemos representar un vídeo en términos de imágenes consecutivas como la Ecuación 4.2:

$$f(n) = f_1, f_2, f_3, ..., f_n \tag{4.2}$$

Donde $f(n)$ actúa como un contador de imágenes según los 25 fps de los vídeos y la duración de cada uno de ellos, se hace uso de la Ecuación 4.3:

$$f(n) = k * A_v \tag{4.3}$$

Donde k es un valor constante de 25, tomado de los fps; mientras que A_v es el tiempo de duración de los vídeos, que oscila entre 30 a 120 segundos. Un ejemplo de la extracción de frames, tomando como los valores de A_v de 30s y 120s se da continuación:

$$f(n_1) = 25 * 30 = 750 \ frames$$
$$f(n_2) = 25 * 120 = 3000 \ frames$$

Se muestra el pseudocódigo usado para la extracción de frames en Algorithm 7 y el código en Python se muestra en Anexo 6.

4.2.5.2.2. Conversión a gris: Después de capturar los fotogramas de los vídeos, es esencial convertir estas imágenes de RGB a escala de grises (Figura 4.9). Esto se hace para que coincidan con las imágenes de la cámara NIR, que son imágenes infrarrojas con tres canales de información. La conversión se aplica a todos los fotogramas en la carpeta de los frames. El código completo se encuentra en el Anexo 7.

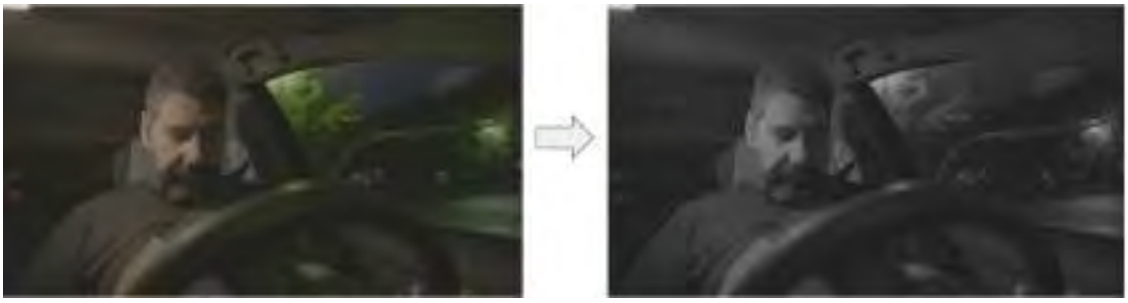
Algorithm 7 Extracción de Frames de un Vídeo

Require: Ruta del video: `video_path`
Ensure: Frames guardados en la ruta: `frames_path`

```
capture ← cv2.VideoCapture(video_path)
cont ← 0
frames_path ← "C:/Users/ruta para guardar los frames/"
fps ← capture.get(cv2.CAP_PROP_FPS)
frames ← capture.get(cv2.CAP_PROP_FRAME_COUNT)
while (capture.isOpened()) do
    ret, frame ← capture.read()
    if (ret == True) then
        cv2.imwrite(frames_path + ''nombre_img.jpg'' cont, frame)
        cont ← cont + 1
        if (cv2.waitKey(1) == ord(''s'')) then
            break
        end if
    else
        break
    end if
end while
```

Figura 4.9: Conversión a escala de grises.

(a) Conversión 1.



(b) Conversión 2.

Posteriormente a la conversión de las imágenes a escala de grises, se procede a seguir tres pasos adicionales para el preprocesamiento, "Detección de puntos faciales" que involucra la identificación de puntos clave en la cara, lo que se ha discutido previamente en la sección 4.2.2, luego por "Selección de ROI" donde se elige la región específica de la imagen donde se centrará el análisis. Para lograr esto, se puede aplicar el código descrito en la sección 4.2.3 y finalmente se hace la "Extracción de ROI" que involucra la extracción de la Región de Interés (ROI) que se ha definido previamente en la sección 4.2.4. Esto se realiza para aislar y enfocar en la parte relevante de la imagen. El código completo que hace los 3 pasos necesarios para guardar las imágenes de los ROI extraídos se ve en Anexo 8.

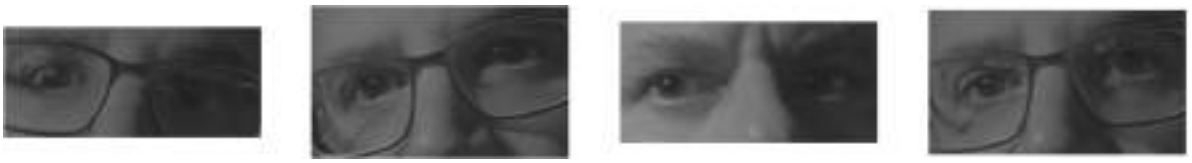
4.2.5.2.3. Generación del dataset: Luego de extraer las regiones de interés (ROI) de la zona de los ojos en los 6 vídeos seleccionados, se creó un conjunto de datos para entrenar las redes neuronales convolucionales. Este conjunto de datos incluye un total de 6,800 imágenes, distribuidas en dos categorías: "ojos abiertos" y "ojos cerrados". Cada categoría consta de 3,400 imágenes. Para garantizar un entrenamiento equilibrado y una clasificación binaria precisa del estado de los ojos, los datos se dividen de la siguiente manera: 70 % (4760 imágenes) para entrenamiento, 15 % (1020 imágenes) para validación y 15 % (1020 imágenes) para pruebas. La distribución equilibrada del conjunto de datos se muestra en la Tabla 4.4, donde las clases "ojos abiertos" y "ojos cerrados" tienen la misma cantidad de imágenes, lo que es esencial para evitar un sesgo en el entrenamiento del modelo.

Tabla 4.4: Distribución del dataset.

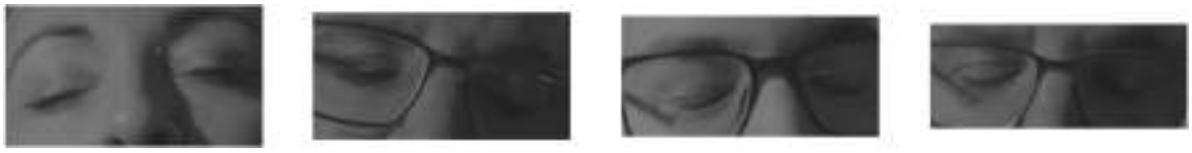
Archivos de:	Clases:	
	ojos abiertos	ojos cerrados
entrenamiento	2380	2380
validación	510	510
testeo	510	510

Además, en la Figura 4.10 se presentan ejemplos aleatorios de muestras del conjunto de datos. Estas muestras proporcionan una representación visual de las imágenes en cada clase, lo que permite una comprensión más clara de la variabilidad de los datos y cómo se relacionan con el estado de los ojos.

Figura 4.10: Ejemplos aleatorios de muestras del conjunto de datos.



(a) Clase ojos abiertos



(b) Clase ojos cerrados

4.2.5.2.4. Entrenamiento de las CNNs:

1. Procesamiento del dataset:

Como se puede observar en la Figura 4.10, las imágenes muestran dimensiones diversas, mientras que las redes CNN requieren tamaños de entrada estándar. En este contexto, se llevo a cabo el procesamiento utilizando imágenes de 64x64x3 píxeles para las arquitecturas VGG16, ResNet50V2 y DD-AI. Para la arquitectura InceptionV3, se optó por dimensiones de 75x75x3 píxeles. Mientras que para la arquitectura DD-AI-G se optó por dimensiones de 64x64x1. Esta elección busca favorecer un procesamiento eficiente en sistemas embebidos, garantizando un rendimiento óptimo.

Después de la conversión de tamaños de las imágenes, se procede a realizar la normalización de cada una de ellas. Este proceso implica dividir el valor de cada píxel en la imagen por el valor máximo de píxel. Generalmente, las imágenes en formato RGB contienen 8 bits por píxel, lo que implica que el valor máximo que puede alcanzar un píxel

es 256. Sin embargo, al trabajar con el lenguaje de programación Python, la indexación comienza desde cero, por lo que el valor máximo real de un píxel es 255.

Para lograr la normalización de las imágenes, se realiza una operación de división, donde cada valor de píxel se divide entre el valor máximo posible, que es 255. Esta acción transforma los valores de píxeles que originalmente estaban en el rango de 0 a 255, a un nuevo rango de valores que va de 0 a 1. Esta normalización es crucial para optimizar los recursos computacionales y también para garantizar que los datos se encuentren en una escala coherente, lo cual puede mejorar el rendimiento de las redes neuronales durante el proceso de entrenamiento y validación.

Otro paso crucial en el proceso es la implementación del data augmentation (aumento de datos), que se realiza con el propósito de prevenir el overfitting durante el entrenamiento de las redes CNNs. En este contexto, se generan cinco imágenes artificiales para cada imagen individual del conjunto de entrenamiento. Para lograr esto, se emplearon los siguientes parámetros en el proceso de data augmentation: rotation_range = 20, horizontal_flip = True y fill_mode = nearest. La rotación con un ángulo máximo de 20 grados, la posibilidad de volteo horizontal y el modo de relleno “nearest” se aplican a las imágenes con el fin de aumentar la variabilidad de los datos de entrenamiento, lo cual puede mejorar la generalización del modelo y reducir el riesgo de que este se adapte excesivamente a los datos de entrenamiento.

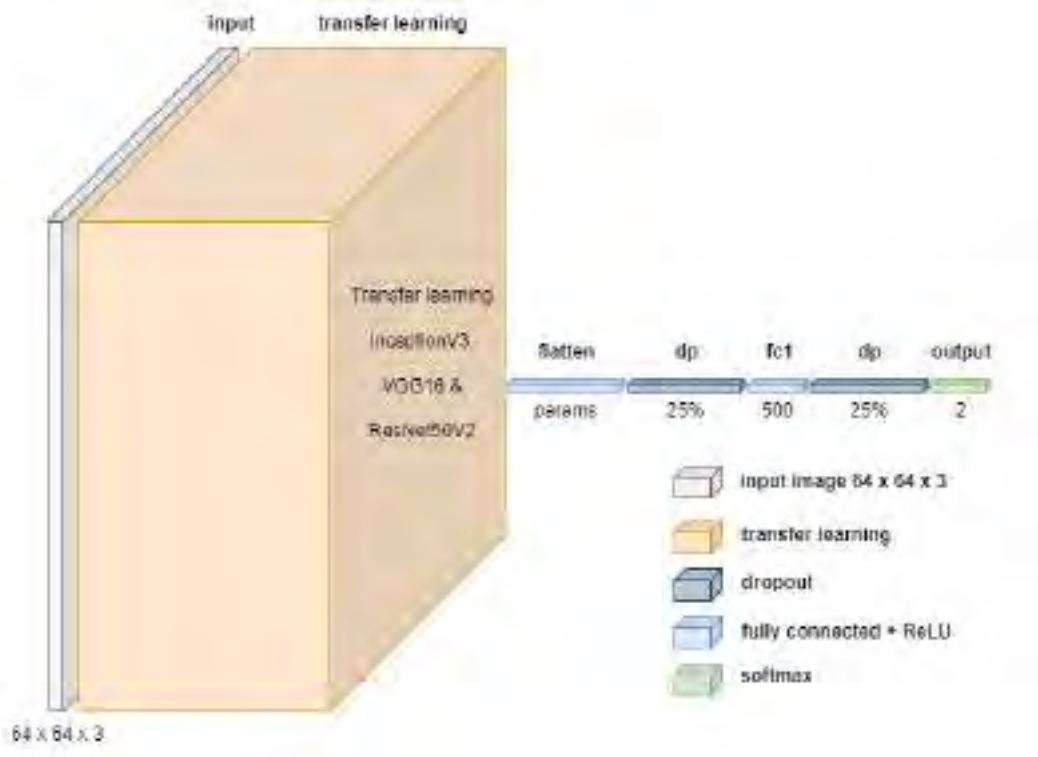
2. Creación de las arquitecturas CNNs:

Luego de completar el procesamiento de las imágenes, se procede a la creación de las arquitecturas de las redes CNN. Para las tres primeras redes CNN, se emplea la técnica de transfer learning, en la cual se retienen los pesos preentrenados de las capas convolucionales y se diseñan únicamente las estructuras de las redes neuronales totalmente conectadas que se encargan de la clasificación binaria.

La configuración de estas arquitecturas implica una secuencia de capas clave. Inicialmente, se aplica una capa flatten para transformar los mapas de características en un vector unidimensional. A continuación, se introduce una capa dropout con una tasa del 25 % para prevenir el overfitting. Posteriormente, se implementa una capa dense con 500 neuronas y función de activación ReLU, seguida por otra capa dropout del 25 %. Finalmente, se incluye la capa de salida con dos neuronas y activación SoftMax, que permite la clasificación binaria.

Se puede observar el diseño propuesto a través de la técnica de transfer learning para las redes InceptionV3, VGG16 y ResNet50V2 en la Figura 4.11. Esta metodología de diseño facilita la creación de modelos robustos y efectivos, aprovechando los patrones aprendidos previamente por estas arquitecturas pre-entrenadas en tareas similares.

Figura 4.11: Arquitectura propuesta con transfer learning.



Se presenta la arquitectura DD-AI desarrollada en la Figura 4.12 para llevar a cabo la tarea de detección de somnolencia mediante una clasificación binaria. La estructura de la red propuesta se compone de varias capas, diseñadas de manera específica para capturar características relevantes en el contexto de la detección de somnolencia.

La arquitectura se inicia con la capa de entrada, que recibe las imágenes procesadas. A continuación, se incorporan tres capas de convolución, cada una compuesta por 50 kernels y activación ReLU. Adicionalmente, se implementan dos capas de maxpooling con un tamaño de ventana de 2x2 para reducir la dimensionalidad y conservar las características más importantes.

Entre la última capa de convolución y la capa flatten, se introduce una capa de dropout con una tasa de retención del 80 %. Esta capa es fundamental para evitar el overfitting y promover una mayor generalización del modelo. En la etapa de clasificación, se replica una arquitectura similar, incluyendo una capa densa con 500 neuronas y función de activación ReLU, seguida por dos capas de dropout con una tasa de retención del 25 % cada una. Finalmente, la capa de salida se compone de dos neuronas con activación SoftMax, permitiendo la clasificación binaria.

Asimismo, en la Figura 4.13 se ilustra la arquitectura DD-AI-G, la cual comparte la misma estructura que la arquitectura DD-AI. La distinción fundamental radica en que DD-AI-G recibe imágenes con un tamaño de 64x64x1, lo que se traduce en imágenes de una sola capa de profundidad.

Las arquitecturas propuestas se han diseñado cuidadosamente para capturar patrones significativos relacionados con la somnolencia y ofrecer una capacidad de clasificación precisa en esta tarea específica.

Figura 4.12: Arquitectura propuesta DD-AI.

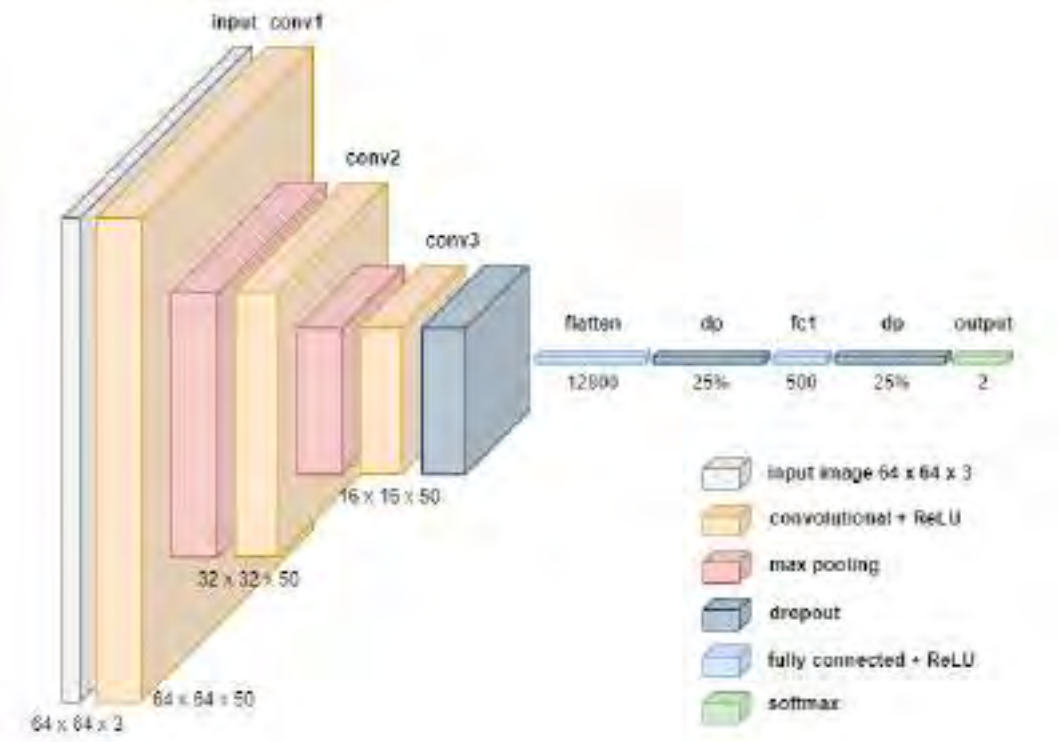
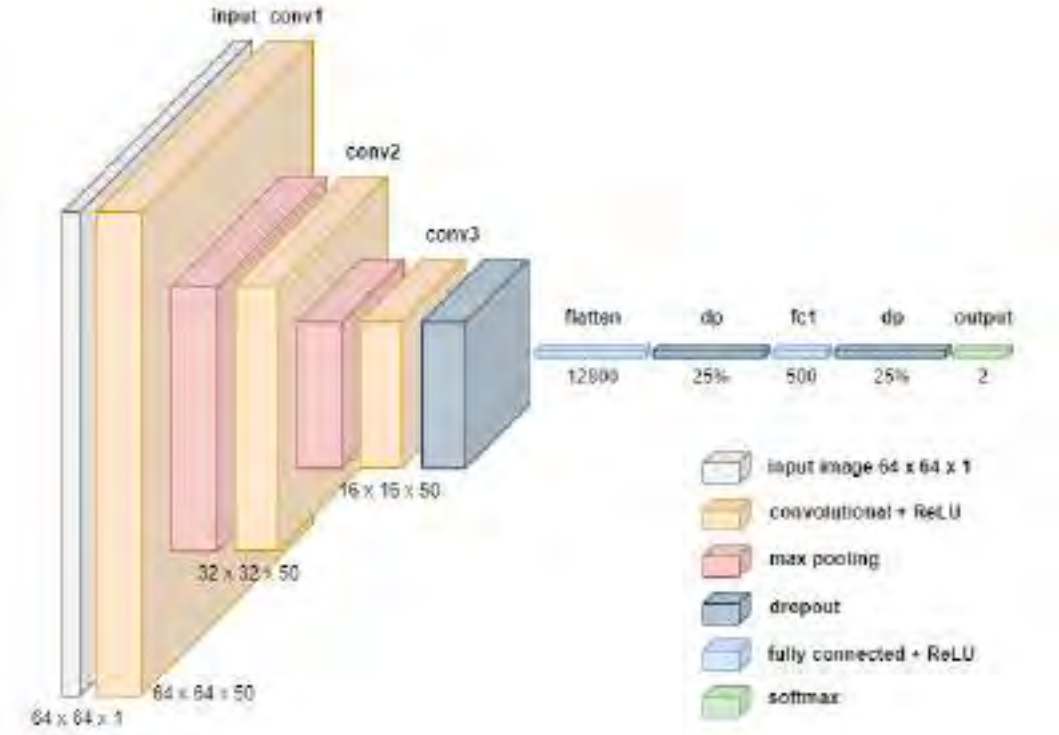


Figura 4.13: Arquitectura propuesta DD-AI-G.



3. Proceso de entrenamiento en las arquitecturas CNNs:

En esta etapa del proceso, se procede a llevar a cabo el entrenamiento de las cinco arquitecturas de redes neuronales convolucionales (CNN). Se realizaron un total de 10 experimentos de entrenamiento para cada una de las arquitecturas de CNN, sumando así un total de 50 experimentos en consideración del tamaño de entrada de las imágenes.

Esta decisión se fundamenta en la variabilidad inherente a la inicialización de los pesos y otros factores en una red CNN. Al realizar múltiples experimentos de entrenamiento con la misma arquitectura, se busca obtener una visión más precisa del rendimiento promedio de la red y su capacidad para generalizar en diferentes conjuntos de datos.

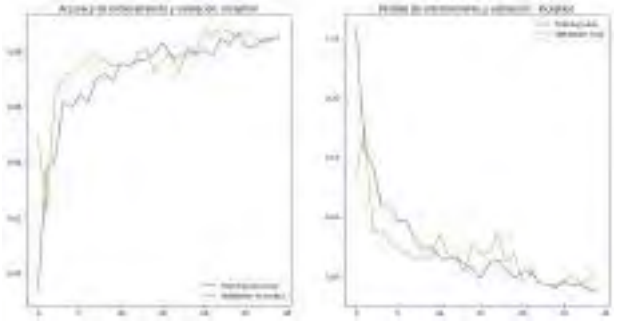
Para realizar el entrenamiento, se utilizo los siguientes hyper-parámetros mostrados en la Tabla 4.5.

Tabla 4.5: Parámetros de entrenamiento.

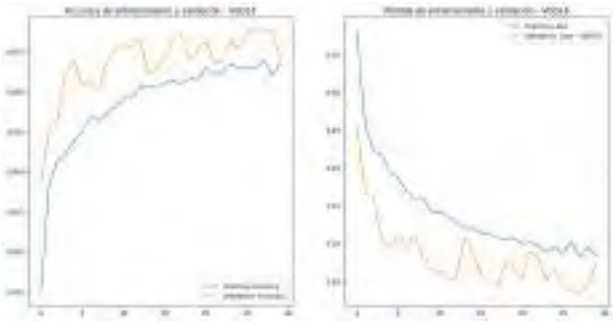
Hyper-parametros	Valor
Optimizer	ADAM
B1	0.001
B2	0.9
Learning rate	0.999
Epochs	30
Batch size	32
Numero de experimentos	10 por cada CNN

4.2.5.2.5. Resultados de los entrenamientos: Durante el entrenamiento, el objetivo es ajustar los pesos de la red neuronal para lograr alta exactitud (accuracy) y baja pérdida (loss). Esto implica que el modelo haga predicciones precisas y coherentes. De los 10 entrenamientos de cada CNN, se eligió arbitrariamente los gráficos de entrenamiento resultantes mostrados en la Figura 4.14.

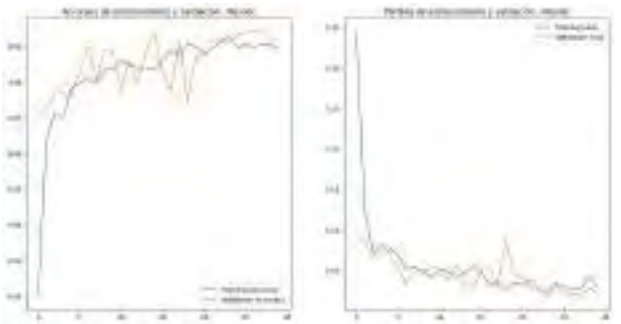
Figura 4.14: Resultados Accuracy y Loss de las CNNs.



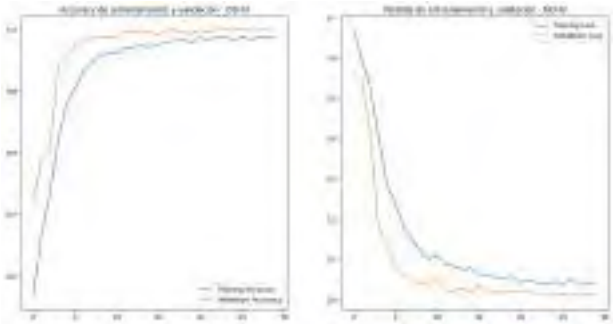
(a) Accuracy y loss - InceptionV3



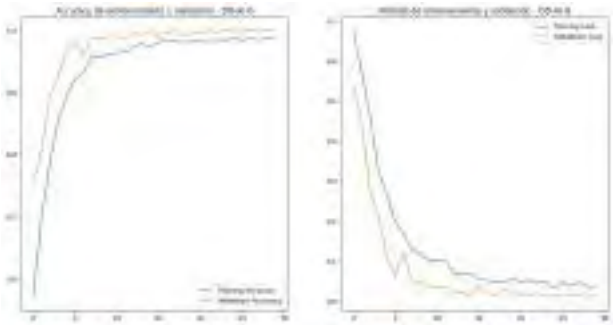
(b) Accuracy y loss - VGG16



(c) Accuracy y loss - ResNet50V2



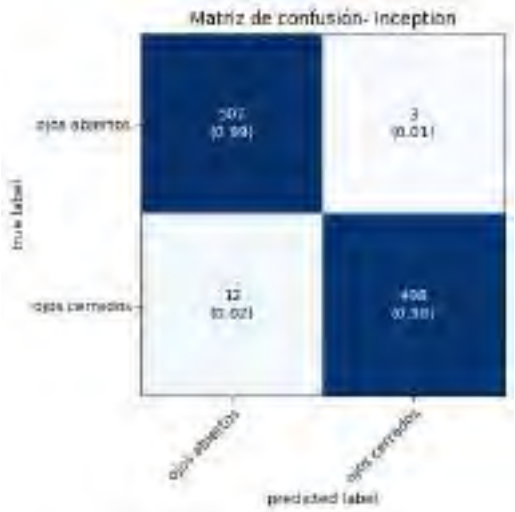
(d) Accuracy y loss - DD-AI



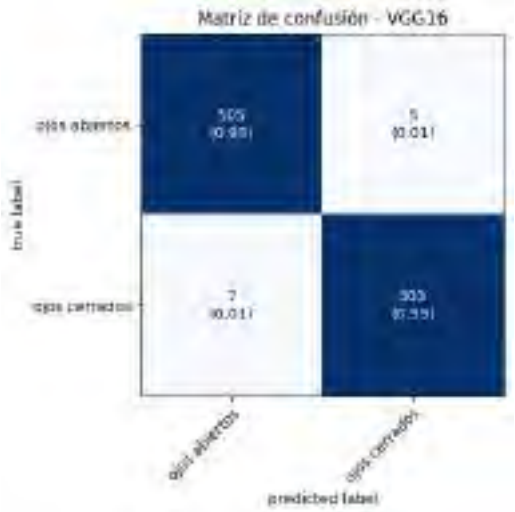
(e) Accuracy y loss - DD-AI-G

Seguidamente, se obtiene las matrices de confusión de cada CNN. en la Figura 4.15 se observan 5 matrices de confusión que fueron elegidos arbitrariamente.

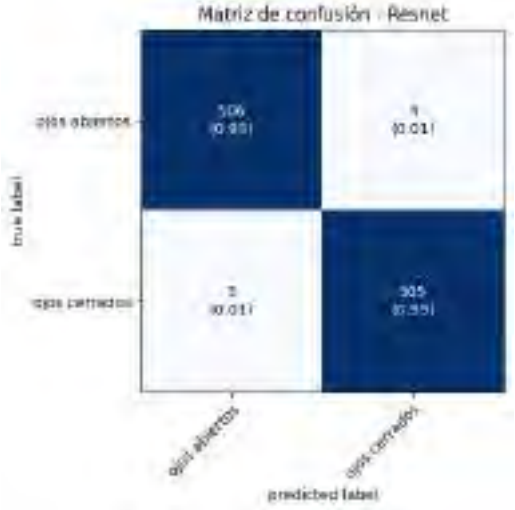
Figura 4.15: Matriz de confusión de las CNNs en validación.



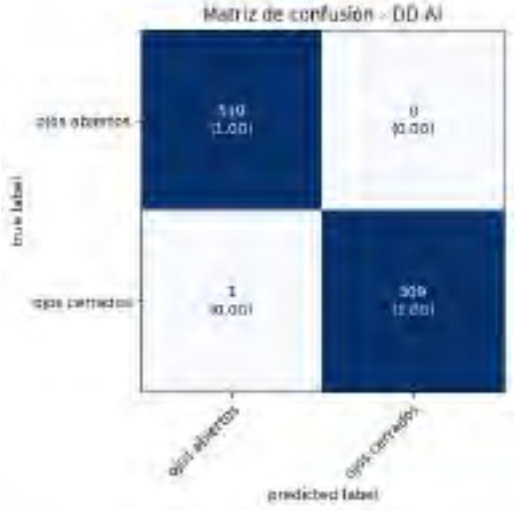
(a) Matriz de confusión - InceptionV3



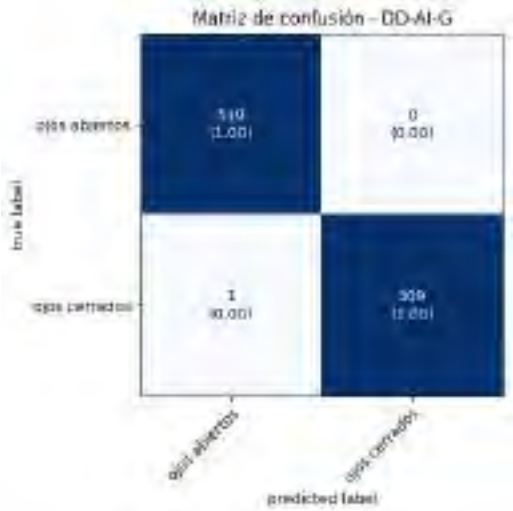
(b) Matriz de confusión - VGG16



(c) Matriz de confusión - ResNet50V2



(d) Matriz de confusión - DD-AI



(e) Matriz de confusión - DD-AI-G

La matriz de confusión se convierte en una herramienta crítica para evaluar la efectividad de los modelos que han sido entrenados. En esta instancia, es de vital importancia minimizar los casos de falsos negativos en la categoría "ojos cerrados", donde la condición real es que los ojos estén cerrados, pero el modelo predijo incorrectamente que están abiertos. Esta discrepancia podría tener consecuencias graves, ya que podría dar lugar a situaciones peligrosas si el sistema no logra alertar al conductor sobre la presencia de somnolencia debido a una mala clasificación del estado ocular.

La Figura 4.15a presenta un ejemplo donde de las 510 imágenes que pertenecen a la clase "ojos cerrados" según el modelo InceptionV3, 12 imágenes fueron incorrectamente etiquetadas como "ojos abiertos". En la Figura 4.15b, se identifica que 7 imágenes de la misma clase fueron erróneamente clasificadas como "ojos abiertos" en el modelo VGG16. Mientras tanto, la Figura 4.15c muestra que 5 imágenes enfrentaron un fallo de clasificación similar en el modelo ResNet50V2. Por otro lado, en los modelos DD-AI y DD-AI-G, las Figuras y demuestran una mejoría en este aspecto, con solo 1 imagen clasificada incorrectamente como "ojos abiertos" de las 510 imágenes en la categoría "ojos cerrados".

La interpretación de estos resultados y la corrección de los falsos negativos en la categoría crítica son elementos esenciales para garantizar la precisión y la seguridad del sistema de detección de somnolencia.

Como se vio en la sección 2.2.11.1, la matriz de confusión da 4 métricas de evaluación: Precision, Recall, F1-score y Accuracy. Estas métricas sirven para determinar el desempeño de los modelos entrenados. Usando las fórmulas 2.9, 2.10, 2.11 y 2.12 correspondientes a Precision, Recall, F1-score y Accuracy, se muestra en la Tabla 4.6 las 4 métricas de evaluación con su desvanecían estándar correspondientes al conjunto de validación de los 10 entrenamientos para cada modelos de CNN.

Tras analizar los resultados preliminares, es evidente que las redes propuestas DD-AI y DD-AI-G destacan con una exactitud (accuracy) superior, alcanzando un $99.84\,\% \pm 0.1\,\%$ y $99.87\,\% \pm 0\,\%$ respectivamente. Esto supera de manera significativa a las arquitecturas CNN

de ResNet50V2, InceptionV3 y VGG16, que obtuvieron un 99.39 %±0.2 %, 98.70 %±0.2 % y 98.57 %±0.2 % respectivamente.

En este sentido, la prioridad es reducir los falsos negativos en la clase de "ojos cerrados". Para lograrlo, la métrica de Recall debe ser elevada, y en este aspecto, las redes DD-AI y DD-AI-G destacan con un Recall del 99.80 %, sin desviación estándar. Estos valores sobrepasan el rendimiento de las tres redes previamente mencionadas, reafirmando la eficacia de las propuestas en la detección precisa de la condición de "ojos cerrados".

Tabla 4.6: Métricas de evaluación en los datos de validación.

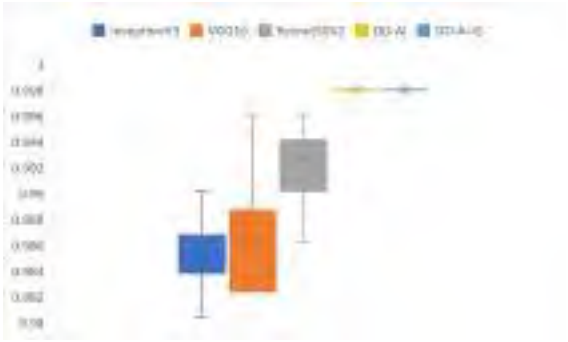
	Clase	Precision	Recall	F1-score	Accuracy
InceptionV3	ojos abiertos	0.9852±0.003	0.9888±0.003	0.9870±0.002	0.9870±0.002
	ojos cerrados	0.9888±0.003	0.9851±0.003	0.9869±0.002	
VGG16	ojos abiertos	0.9865±0.004	0.9849±0.004	0.9857±0.002	0.9857±0.002
	ojos cerrados	0.9849±0.004	0.9865±0.004	0.9857±0.002	
ResNet50V2	ojos abiertos	0.9926±0.003	0.9953±0.003	0.9939±0.002	0.9939±0.002
	ojos cerrados	0.9953±0.003	0.9926±0.003	0.9939±0.002	
DD-AI	ojos abiertos	0.9980±0.000	0.9988±0.001	0.9984±0.001	0.9984±0.001
	ojos cerrados	0.9988±0.001	0.9980±0.000	0.9984±0.001	
DD-AI-G	ojos abiertos	0.9980±0.000	0.9994±0.001	0.9987±0.000	0.9987±0.000
	ojos cerrados	0.9994±0.001	0.9980±0.000	0.9987±0.000	

El boxplot comparativo para la evaluación del rendimiento de las 5 CNNs en la fase de validación del entrenamiento se presenta en la Figura 4.16, donde se pueden observar las variaciones de rendimiento de la Tabla 4.6 en las métricas de precisión y recall de la clase ojos cerrados de cada una de las tres redes en las 10 muestras de entrenamiento. En la Figura 4.16a, la CNN propuesta DD-AI-G presentó un mejor rendimiento en la validación de datos con una mediana del 99,9 % de las 10 ejecuciones de entrenamiento. Del mismo modo, en la

Figura 4.16b, las CNNs propuestas DD-AI y DD-AI-G tuvieron el mejor rendimiento en la clase "ojos cerrados" con una mediana del 99.9 % y 100 % respectivamente, minimizando así los falsos negativos de la caja Somnolencia.

Figura 4.16: Boxplot de dos métricas de evaluación para la validación.

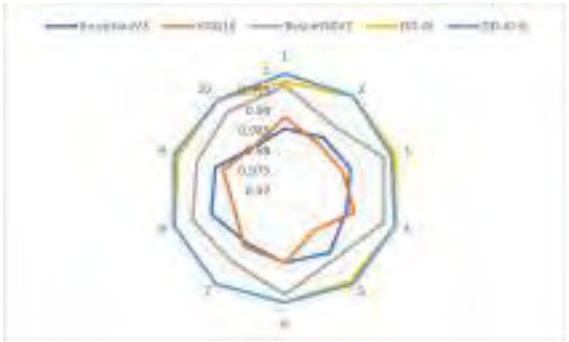
(a) Boxplot de accuracy.



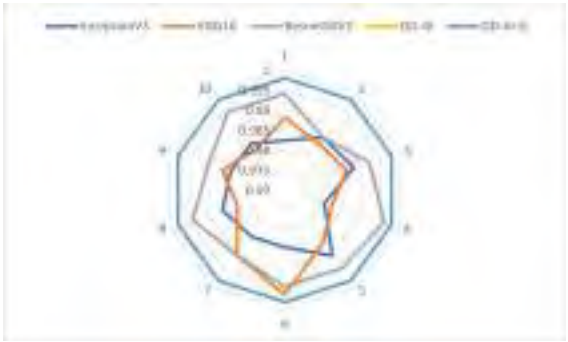
(b) Boxplot de recall.

La figura 4.17 muestra el comportamiento radial de las cinco CNNs, para las dos métricas de los 10 entrenamientos realizados. En la Figura 4.17a, la CNN propuesta DD-AI-G tuvo un comportamiento constante en los 10 entrenamientos. Por otro lado, la red propuesta DD-AI tuvo un comportamiento poco inferior en comparación con la red DD-AI-G; por otro lado la red ResNet50V2 presenta un mejor comportamiento en comparación con las redes InceptionV3 y VGG16. Mientras tanto, en la Figura 4.17b, las CNNs propuestas presenta un mejor rendimiento en comparación con las otras tres redes. Obteniendo en todos los experimentos un 99,8 % de recall en la clase "ojos cerrados".

Figura 4.17: Comportamiento radial de dos métricas para la validación.



(a) Comportamiento radial de accuracy.



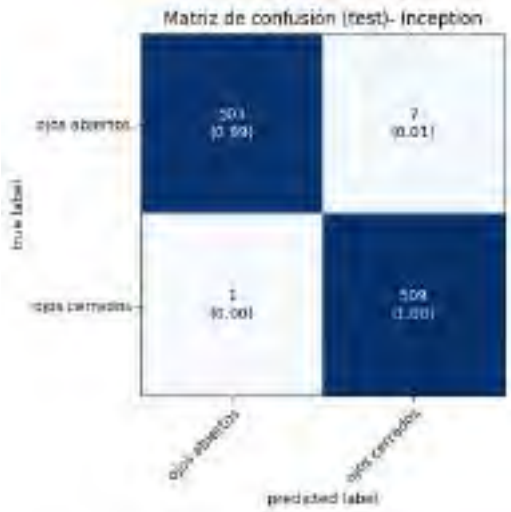
(b) Comportamiento radial de recall.

4.2.5.2.6. Prueba de las CNNs entrenadas: Después de completar el proceso de entrenamiento de las redes neuronales convolucionales, se procedió a someterlas a pruebas utilizando los conjuntos de datos de testeo. Durante esta fase, las imágenes del conjunto de testeo se procesaron sin la aplicación de técnicas de aumento de datos (data augmentation). Se optó por utilizar un tamaño de lote de 1 para analizar individualmente cada imagen, realizando un total de 10 pruebas con cada red previamente entrenada.

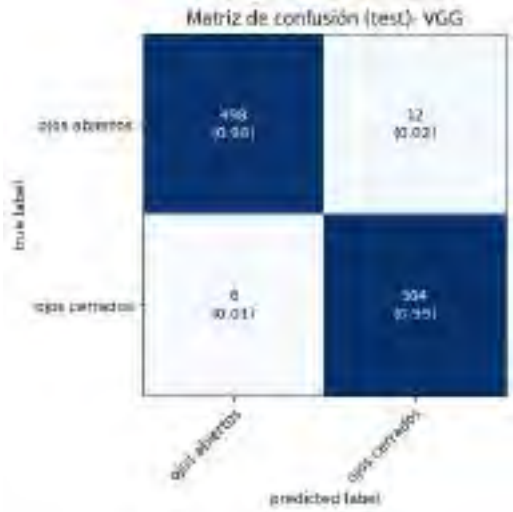
Es crucial resaltar que en esta fase, los modelos evaluaron imágenes nuevas que no habían sido utilizadas durante el proceso de entrenamiento, lo que permitió una evaluación más precisa y objetiva de su desempeño.

Un ejemplo de las matrices de confusión (elegidos arbitrariamente) obtenidas durante estas pruebas para cada CNN en el conjunto de datos de prueba se presenta en la Figura 4.18.

Figura 4.18: Matriz de confusión de las CNNs en testeo.

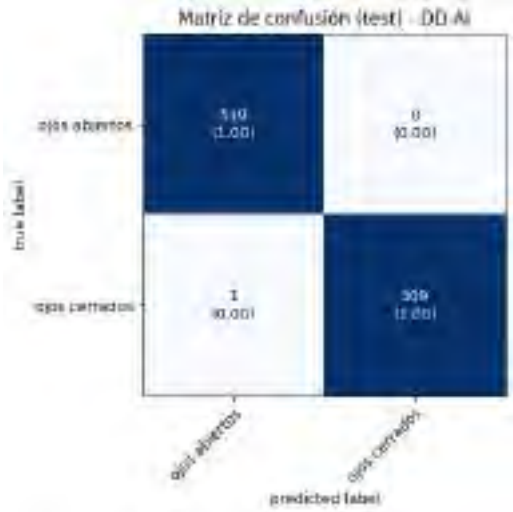


(a) Matriz de confusión - InceptionV3

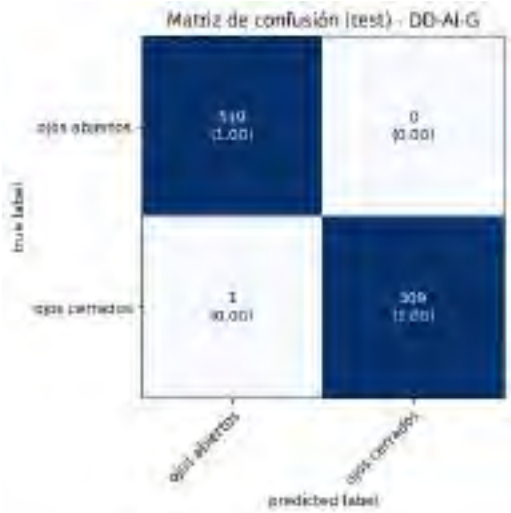


(b) Matriz de confusión - VGG16

(c) Matriz de confusión - ResNet50V2



(d) Matriz de confusión - DD-AI



(e) Matriz de confusión - DD-AI-G

Empleando las mismas ecuaciones para determinar las métricas de evaluación a partir de las matrices de confusión, se derivaron los siguientes resultados en la evaluación de las redes CNN, que se presentan de manera detallada en la Tabla 4.7. Los resultados revelan que las propuestas redes CNN, DD-AI y DD-AI-G, sobrepasan con un 99.88 %±0.1 % y 99.91 % respectivamente, a las otras tres redes, InceptionV3 con 98.95 %±0.2 %, VGG16 con 98.33 %±0.3 % y ResNet50V2 con 99.49 %±0.3 %. En la métrica Recall de la clase "ojos cerrados", la red CNN DD-AI-G alcanzó un 99.92 %±0.1 %, superando a la red DD-AI que obtuvo 99.88 %±0.1 %, así mismo superando a las otras tres redes.

Tabla 4.7: Métricas de evaluación en los datos de testeo.

	Clase	Precision	Recall	F1-score	Accuracy
InceptionV3	ojos abiertos	0.9895±0.006	0.9896±0.006	0.9895±0.002	0.9895±0.002
	ojos cerrados	0.9897±0.005	0.9894±0.006	0.9895±0.002	
VGG16	ojos abiertos	0.9868±0.001	0.9798±0.006	0.9833±0.003	0.9833±0.003
	ojos cerrados	0.9800±0.006	0.9869±0.001	0.9834±0.003	
ResNet50V2	ojos abiertos	0.9967±0.003	0.9931±0.003	0.9949±0.003	0.9949±0.003
	ojos cerrados	0.9932±0.003	0.9967±0.003	0.9949±0.003	
DD-AI	ojos abiertos	0.9988±0.001	0.9988±0.001	0.9988±0.001	0.9988±0.001
	ojos cerrados	0.9988±0.001	0.9988±0.001	0.9988±0.001	
DD-AI-G	ojos abiertos	0.9992±0.001	0.9990±0.001	0.9991±0.001	0.9991±0.000
	ojos cerrados	0.9990±0.001	0.9992±0.001	0.9991±0.001	

La Figura 4.19 muestra la exactitud (accuracy) y sensibilidad (recall) de la clase "ojos cerrados" resultante de las 10 pruebas realizadas en cada modelo CNN. En detalle, en la Figura 4.19a se evidencia que la red propuesta DD-AI-G logró el rendimiento más sobresaliente, con una mediana de 99.9 % de exactitud, seguida por la red CNN DD-AI con un 99.8 %. Por otro lado, en la Figura 4.19b, se destaca que la red CNN DD-AI-G alcanzó

una mediana de 100 % de sensibilidad, superando ligeramente a la red CNN DD-AI que obtuvo una mediana de 99.8 %.

Figura 4.19: Boxplot de dos métricas de evaluación para el testeo.

(a) Boxplot de accuracy

(b) Boxplot de recall

El comportamiento radial de los 10 experimentos de prueba se muestran en la Figura 4.20b, mostrando las dos métricas de evaluación de exactitud y sensibilidad de la clase "ojos cerrados". Donde, en la Figura 4.20a, la red CNN propuesta DD-AI-G obtuvo mejores resultados al igual que la red DD-AI. Mientras que en la Figura 4.20b, se observó que las redes CNN DD-AI y DD-AI-G tuvieron un rendimiento similar en los resultados de los 10 experimentos. Si embargo, el rendimiento de la red basada en ResNet50V2 mejoró significativamente en comparación con los resultados de entrenamiento presentados en la Figura 4.17b.

Figura 4.20: Comportamiento radial de dos métricas para el testeo.

(a) Comportamiento radial de accuracy.

(b) Comportamiento radial de de recall.

4.2.5.2.7. Resultados visuales de las CNNs: Para efectuar una comparación entre las arquitecturas CNN en relación a su capacidad de detección y clasificación del estado de los ojos, se tomaron en consideración los resultados más destacados obtenidos durante las pruebas para cada una de las CNN. Específicamente, se consideraron aquellos resultados que presentaron los mayores valores de exactitud (accuracy) y sensibilidad (recall) para la clase "ojos cerrados", tal como se observa en la Figura 4.20.

Para la arquitectura basada en InceptionV3, se eligió la séptima prueba, que arrojó un 99.8 % de exactitud y un 99.22 % de sensibilidad. En cuanto a la CNN basada en VGG16, se optó por la primera prueba con un 98.73 % de exactitud y un 98.82 % de sensibilidad. La arquitectura basada en ResNet50V2 presentó su mejor desempeño en la tercera prueba, alcanzando un 99.8 % de exactitud y un 100 % de sensibilidad.

En relación a las CNN propuestas, la arquitectura DD-AI demostró su máximo rendimiento en la novena prueba, logrando un 100 % de exactitud y sensibilidad. Finalmente, la CNN propuesta DD-AI-G obtuvo su mejor desempeño en la sexta prueba, alcanzando un 100 % de exactitud y sensibilidad.

Para una mejor comprensión del comportamiento, se utilizó el método gradient-weighted class activation mapping (Grad-CAM) (Selvaraju et al., 2017, 2016). Usando Grad-CAM, es posible visualizar las regiones que son importantes para la clasificación; este método busca identificar partes de la imagen que guían a la CNN a tomar la decisión final para la determinación de la clase. El método consiste en generar un mapa de calor que representa las regiones con mayor relevancia para la clasificación de la imagen de entrada recibida.

Se muestra cinco ejemplos de diferentes escenarios para la visualización de los mapas de calor con Grad-CAM en la Figura 4.21. En los escenarios 1 y 2, el conductor se encuentra en un estado normal, es decir, con los ojos abiertos; en el escenario 3, el conductor se encuentra en un estado de vigilia, que es la transición a la somnolencia; en los escenarios 4 y 5, el conductor presenta los ojos cerrados. En cada mapa de calor, el color rojo representa las regiones de mayor importancia para la predicción de cada una de las cinco CNN entrenadas,

y el color azul representa las regiones de menor importancia. Los cinco ejemplos se probaron para cada CNN, generando ROIs I-1 a I-5 correspondientes a la CNN basada en InceptionV3, V-1 a V-5 correspondientes a la CNN basada en VGG16, R-1 a R-5 correspondientes a la CNN basada en ResNet50V2, D-1 a D-5 correspondiente a la CNN propuesta DD-AI y DG-1 a DG-2 correspondiente a la CNN propuesta DD-AI-G.

En las ROI I-1 a I-5, observamos que los mapas de calor se centran en el borde ojo derecho e izquierdo (I-1), en el ojo derecho y pequeña parte del ojo izquierdo (I-2 a I-4) y la parte inferior del ojo derecho y ojo izquierdo (I-5). Mientras que los mapas de calor de los ROIs V-1 a V-5 se centran en V-1 y V-2 en el ojo derecho y parte superior de ojo izquierdo, V-3, V- y V-5 en la parte superior de la ceja del ojo derecho y mejilla izquierda. En las ROIs R-1 a R-5, se observa que en R-1 a R-5, el modelo se enfoca en la parte inferior, lo que corresponde a las mejillas y nariz. Mientras que, en las ROI de las CNN propuestas DD-AI y DD-AI-G, en ambos casos se muestra que los modelos se enfocan en pequeñas partes de ambos ojos (D-1, D-2 y DG-1, DG-2), en gran parte del ojo izquierdo (D-3 a D-5 y DG-3 a DG-5) .Debajo de cada ROI, se muestra la visualización Grad-CAM con su respectivo porcentaje de predicción para ojos abiertos y ojos cerrados.

Por lo tanto, las arquitecturas CNN propuestas, DD-AI y DD-AI-G, muestran un enfoque destacado en la clasificación precisa del estado de los ojos, como se evidencia en los ejemplos y escenarios evaluados.

Figura 4.21: Visualización de las CNNs con Grad-CAM.

(a) Extracción de ROI de los ojos en 5 escenarios.

(b) InceptionV3 Grad-CAM.

(c) VGG16 Grad-CAM.

(d) ResNet50V2 Grad-CAM.

(e) DD-AI Grad-CAM.

(f) DD-AI-G Grad-CAM.

4.2.5.2.8. Identificación de somnolencia visual: Para la identificación de somnolencia visual en conductores, con los modelos entrenados y probados, se evalúa la probabilidad de que el ROI de los ojos pertenezca a la clase de "ojos cerrados" superior al 95 % y posteriormente se utiliza la propuesta presentada por Kwon et al. (2013), que consiste en contar el tiempo de un parpadeo normal de 100 a 300 ms; cuando los ojos permanecen cerrados durante más de 300 ms del tiempo contado, se considera un estado de somnolencia visual.

A continuación, se muestra el pseudocódigo para la detección de somnolencia visual en Algorithm 8 y el código completo en Python se muestra en Anexo 9. Posteriormente se explica el pseudocódigo mostrado.

Algorithm 8 Detección de somnolencia visual

```
if predict_class = ojos_cerrados and prob ≥ 95 % then
    contador_somno += 1
end if
if predict_class = ojos_abiertos and prob > 95 % then
    contador_somno −= 1
end if
if contador_somno ≥ 300ms then
    contador_somno = 300
    print("Somnolencia Visual")
end if
if (contador_somno > 0 and contador_somno < 300ms) then
    print("Normal")
end if
if contador_somno < 0 then
    contador_somno = 0
end if
```

1. Detección de Ojos Cerrados:

    - Si el valor predicho por el modelo es "ojos cerrados" y la probabilidad asociada con esta predicción es mayor o igual al 95 %, se considera que los ojos están cerrados con alta confianza.

- Si esta condición es verdadera, se incrementa el contador contador_somno. Esto indica que los ojos han sido detectados como cerrados durante un período de tiempo.

2. Detección de Ojos Abiertos:

   - Si el valor predicho por el modelo es "ojos abiertos" y la probabilidad asociada con esta predicción es mayor al 80 %, se considera que los ojos están abiertos.
   - Si esta condición es verdadera, se decrementa el contador contador_somno.

3. Detección de Somnolencia Prolongada:

   - Si el contador contador_somno alcanza o supera el valor de 300 milisegundos, se interpreta como un posible período prolongado de somnolencia visual.
   - El contador se establece en 300 y se imprime el mensaje "Somnolencia Visual". Esto se considera una alerta al usuario sobre el estado de somnolencia.

4. Detección de Somnolencia en Progreso (menos de 300 ms):

   - Si el contador "contador_somno" es mayor que cero pero menor que 300 milisegundos, se considera que el estado de somnolencia aún está en progreso, pero no ha alcanzado un nivel prolongado.
   - En este caso, es un estado "Normal". Esto podría indicar que la persona todavía está en un estado consciente, pero ha experimentado somnolencia en algún grado.

5. Restricción del Contador a Valor Positivo:

   - Si el contador "contador_somno" se vuelve negativo (lo cual no debería ocurrir), se restablece a cero.

4.2.5.2.9. Resultados generales de las CNNs: Los resultados de las pruebas de las CNNs para detectar somnolencia basada en el estado de los ojos se presentan en la Tabla 4.8. Esta tabla detalla el tiempo de entrenamiento, el tamaño del archivo del modelo, el número de parámetros entrenados y el tiempo de respuesta para cada CNN. Estos valores proporcionan información sobre la eficiencia computacional, el almacenamiento necesario, la complejidad del modelo y la velocidad de clasificación de cada CNN en relación con la imagen de entrada (ROI). En resumen, la Tabla 4.8 ofrece una visión completa y comparativa de los resultados y características clave de las CNN evaluadas en esta investigación.

Tabla 4.8: Resultados generales de las CNNs.

	Tiempo de entrenamiento	Tamaño de archivo (KB)	Parámetros	Tiempo de respuesta (ms)
InceptionV3	4.3min ± 2s	98,055	22,828,286	51.01
VGG16	3.1min ± 3s	69,586	15,740,190	39.00
ResNet50V2	3.4min ± 3s	140,594	27,662,302	60.02
DD-AI	3.1min ± 1s	75,618	6,448,002	33.00
DD-AI-G	2.8min ± 6s	75,608	4,134,782	30.00

Igualmente, se registraron los resultados presentados en la Tabla 4.9, los cuales detallan el consumo de memoria RAM al ejecutar cada CNN. Asimismo, se capturó el consumo de memoria gráfica (GPU) de cada CNN durante la clasificación en tiempo real de la imagen de entrada (ROI). Estos datos resultarán especialmente valiosos para la futura implementación en un sistema embebido, contribuyendo a una toma de decisiones informada.

Tabla 4.9: Resultados de consumo de memoria RAM y GPU de las CNNs.

	Consumo de RAM (GB)	Consumo de GPU (GB)
InceptionV3	3.1	4.4/6
VGG16	3	4.4/6
ResNet50V2	3.1	4.4/6
DD-AI	3	4.4/6
DD-AI-G	3	4.4/6

4.2.6. Fase 6: Activación de alarma

Al tener la Fase 5 (sección 4.2.5) con los dos métodos MAR y CNN obtenidos previamente, se procede a realizar la activación de la alarma, que, primeramente es una alarma visual, ya que en esta etapa aun no se controla ningún actuador.

La activación de la alarma visual viene dada por los valores establecidos de MAR con un valor de 55 como umbral y CNN con valores mayores a 95 % de predicción de la clase "ojos cerrados" y 300ms de cierre de los ojos.

4.3. Diagrama de flujo del diseño del sistema

El diagrama de flujo integral para el diseño del sistema detector de somnolencia se observa en la Figura 4.22. Donde, se ofrece una explicación detallada del funcionamiento:

1. Inicio del proceso:
    - El proceso comienza con la activación de la cámara para la captura de vídeo.

2. Procesamiento de vídeo:

   - La fase de procesamiento de vídeo se divide en dos bloques en paralelo, cada uno dirigido a la detección de aspectos específicos.
   - Para ambos bloques, se inicia con la detección de puntos faciales clave, especialmente enfocándose en los ojos y la boca.
   - La Región de Interés (ROI) se selecciona alrededor de los ojos y la boca.

   a) Bloque 1: Detección de somnolencia (ojos cerrados):

      - Los datos relevantes se extraen de la ROI de los ojos y se redimensionan según sea necesario.
      - Se utiliza un modelo de Red Neuronal Convolucional (CNN) para inferir si los ojos están cerrados.
      - Se verifica si la probabilidad de ojos cerrados supera el 95 %.
      - Si es negativo, el proceso vuelve a la detección de puntos faciales; si es afirmativo, se inicia un contador (cont_somno).
      - Se verifica si el conteo supera los 300 milisegundos, activando una alarma visual 1 y notificando la detección de somnolencia visual.

   b) Bloque 2: Detección de bostezos (boca abierta):

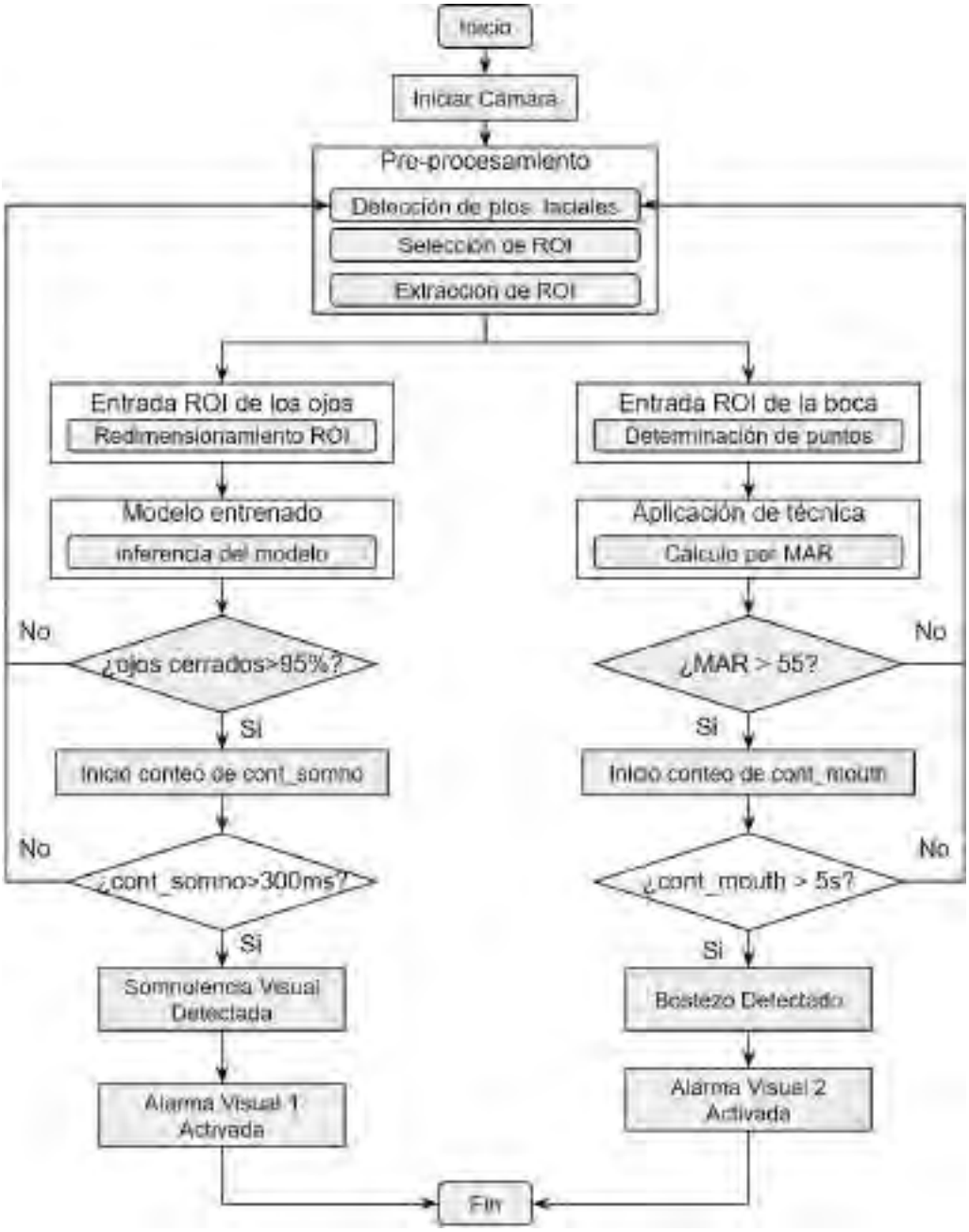
      - Puntos clave en la boca se determinan para calcular la relación de apertura de la boca (MAR).
      - Se evalúa si la MAR supera el umbral 55.
      - En caso negativo, el proceso regresa a la detección de puntos de la boca; en caso afirmativo, se inicia otro contador (cont_mouth).
      - Se verifica si el conteo supera los 5 segundos, activando una alarma visual 2 y notificando la detección de un bostezo.

3. Fin del proceso:

   - El proceso concluye tras ejecutar ambos bloques en paralelo.

Esta arquitectura busca identificar signos de somnolencia, mediante el estado de ojos cerrados y bostezos, con la combinación de técnicas de procesamiento facial, generando alertas visuales pertinentes en respuesta a la detección de estos eventos.

Figura 4.22: Diagrama de flujo del diseño del sistema.



4.3.1. Subdiagramas del diagrama principal

En el diagrama principal que describe el funcionamiento del diseño del sistema, se identifican cinco bloques que albergan subsistemas. A continuación, se proporciona una descripción de cada uno de ellos.

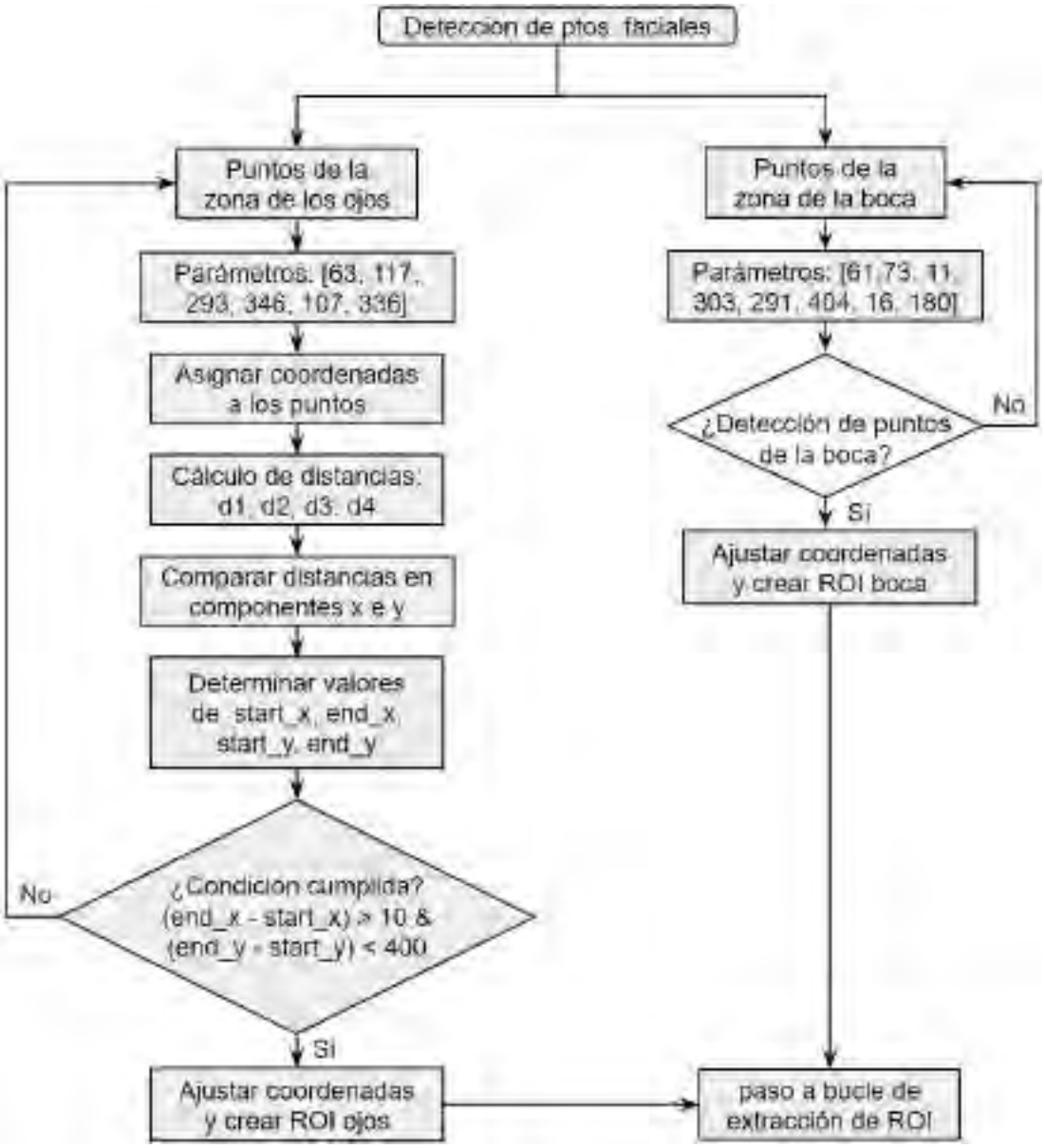
1. Bloque de pre-procesamiento: Este bloque comprende los tres pasos fundamentales para el pre-procesamiento: detección de puntos faciales, selección de ROI y extracción de ROI. A continuación, se presentan los diagramas de flujo correspondientes a cada uno de ellos.

    - Detección de puntos faciales: El diagrama de flujo representa la detección de puntos faciales mediante MediaPipe Face Mesh, el cual se observa en la Figura 4.23. Luego de ser detectado los puntos faciales, se pasa al siguiente paso.

Figura 4.23: Diagrama de flujo de detección de puntos faciales.

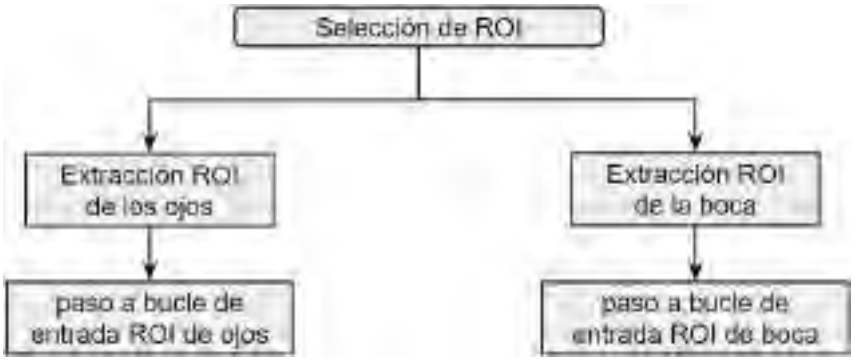
- Selección de ROI: Esta etapa se subdivide en dos diagramas de flujo: uno al ROI de los ojos y otro al ROI de la boca, como se ilustra en la Figura 4.24. Una vez completada la creación de ambos ROIs, se avanza al último paso de este bloque.

Figura 4.24: Diagrama de flujo de selección de ROI.



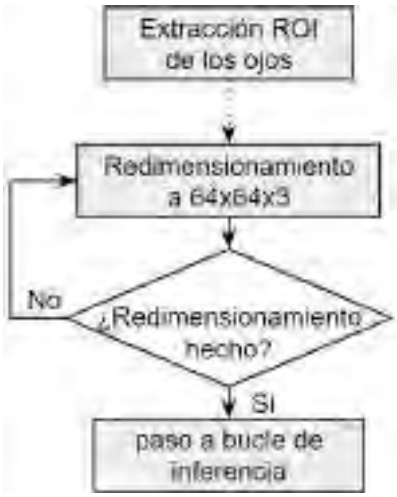
- Extracción de ROI: En este paso solo se realiza la extracción de los ROIs correspondientes a la zona de los ojos en imagen y a la boca en puntos, el cual se divide en los 2 ROIs como se ilustra en la Figura 4.25.

Figura 4.25: Diagrama de flujo de extracción de ROI.



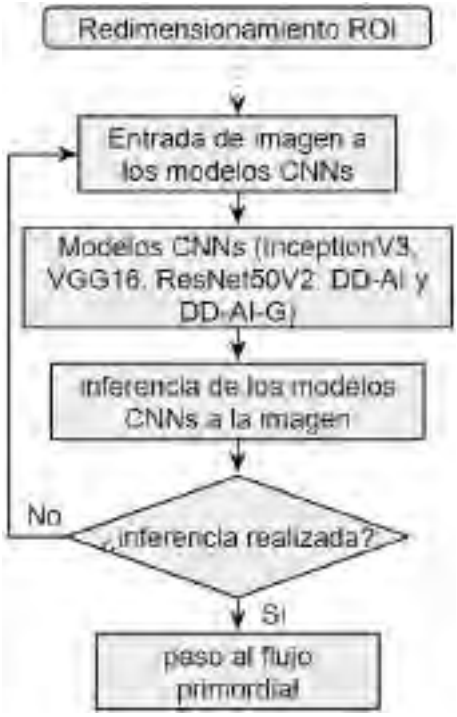
2. Bloque de entrada ROI de los ojos: Este bloque solo consta del redimensionamiento de la imagen de ROI de los ojos, como se observa en la Figura 4.26. El cual sirve como entrada para el siguiente bloque.

Figura 4.26: Diagrama de flujo de redimensionamiento de ROI.



3. Bloque de modelo entrenado: En este bloque, se ejecuta la inferencia utilizando los 5 modelos de CNN en la imagen de entrada redimensionada, tal como se muestra en el diagrama de flujo de la Figura 4.27. Una vez completada la inferencia para clasificar el estado de los ojos, se procede al flujo principal, que constituye la condición fundamental para determinar la presencia de somnolencia visual.

Figura 4.27: Diagrama de flujo de inferencia del modelo.

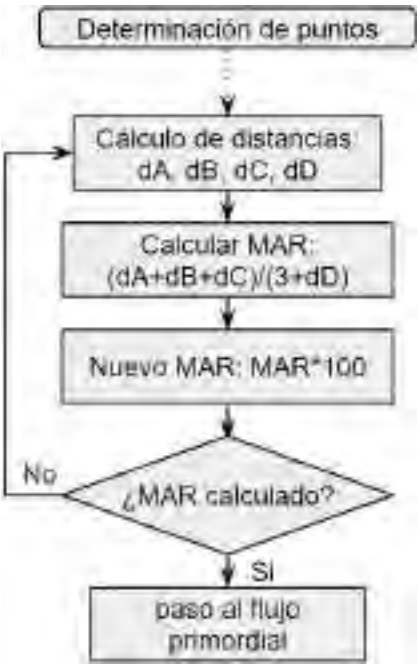


4. Bloque de entrada ROI de la boca: En este bloque, se lleva a cabo la determinación de los 8 puntos asociados a la boca, tal como se muestra en la Figura 4.28. Este proceso implica la ubicación precisa de dichos puntos durante el movimiento de la boca. Una vez que se han establecido los 8 puntos, se avanza al bloque siguiente.

Figura 4.28: Diagrama de flujo de determinación de puntos.

5. Bloque de aplicación de técnica: En este bloque, se implementa la técnica mediante el cálculo de MAR, utilizando la ecuación 2.4, la cual se encuentra vinculada a la Figura 4.6. Una vez que se ha determinado el valor de MAR mediante la condición de la boca, detallada en la Figura 4.29, se procede al flujo principal, que constituye la condición principal para identificar el bostezo.

Figura 4.29: Diagrama de flujo de calculo por MAR.

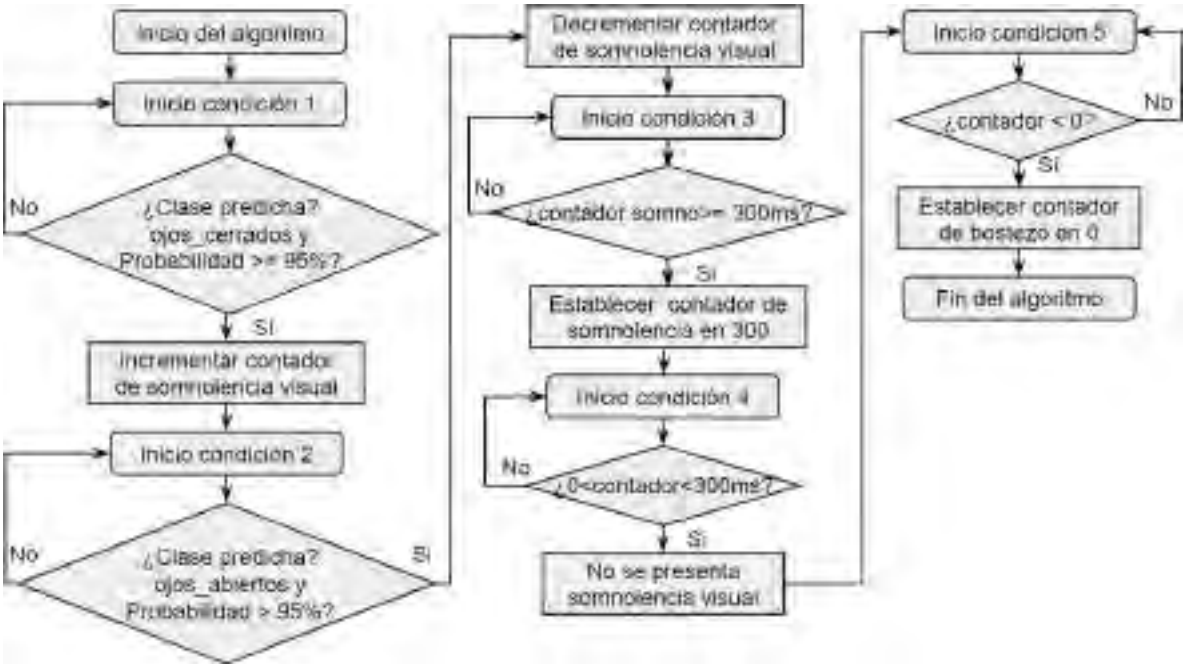


4.4. Diagramas de flujo complementarios

Como se ilustra en la Figura 4.22, solo se tienen en cuenta las condiciones principales para determinar la somnolencia del conductor a través del estado de los ojos y la boca. En esta sección, se presenta el diagrama de flujo que detalla las condiciones para ambos estados (ojos y boca), los cuales se ponen a continuación.

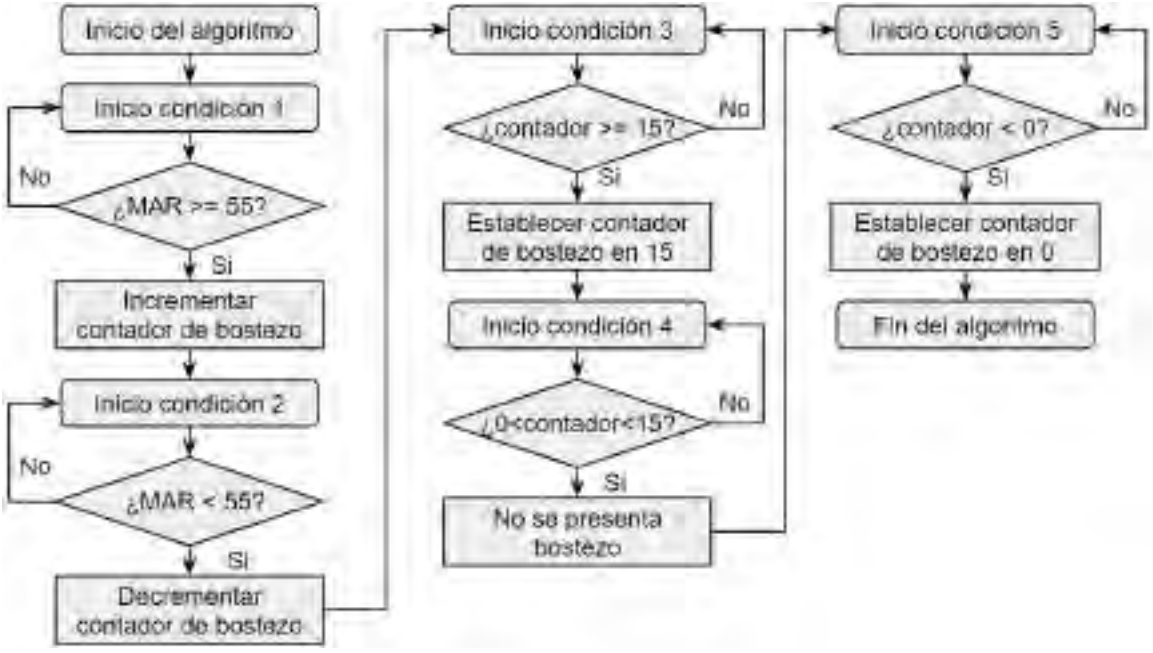
1. Diagrama de flujo de la somnolencia visual: El diagrama de flujo representado en la Figura 4.30, se deriva del pseudocódigo proporcionado en Algorithm 8.

Figura 4.30: Diagrama de flujo de la somnolencia visual.



2. Diagrama de flujo del bostezo: El diagrama de flujo representado en la Figura 4.31, se deriva del pseudocódigo proporcionado en Algorithm 6. Donde el valor de 15 es establecido de acuerdo a los fps (30 a 40 fps) que se tiene en el entorno del computador.

Figura 4.31: Diagrama de flujo del bostezo.



Capítulo 5

Implementación del sistema detector de somnolencia

La implementación es la fase en la que el diseño cobra vida a través de la construcción y ensamblaje real de los componentes. En esta etapa, se traducen las especificaciones del diseño en código de programación y configuración de hardware. Se conectan y configuran los sensores para capturar datos en tiempo real, y se desarrolla la interfaz de usuario que permitirá al usuario interactuar con el sistema. La implementación también implica la realización de pruebas exhaustivas para validar el funcionamiento del sistema en diferentes situaciones, ajustando y refinando el software y el hardware según sea necesario. Finalmente, se ensamblan todos los componentes, se ajustan y se ponen en funcionamiento, permitiendo que el sistema realice la detección de somnolencia y genere alertas de acuerdo con el diseño previamente establecido.

5.1. Elección de elementos para la implementación del sistema

La elección de los elementos pueden ser categorizados en función de su utilidad, variando según la tarea que desempeñen, y pueden clasificarse en:

- Unidad de procesamiento.
- Visualización en interfaz.
- Suministro de energía.

El elemento de unidad de procesamiento hace referencia al sistema embebido donde se hace el procesamiento de la etapa de diseño, el elemento visualización en interfaz hace referencia a la pantalla donde se visualiza el estado del conductor y el elemento de suministro de energía hace referencia a la fuente de alimentación que suministra la potencia necesaria para el funcionamiento del sistema dentro de la unidad vehicular.

5.1.1. Elección del sistema embebido

Como se vio en la sección 2.2.17, se eligió dos sistemas embebidos que cumplen el costo computacional para tareas sobre procesamiento digital de imágenes, visión computacional e inteligencia artificial.

A continuación en la Tabla 5.1 se detalla sus características correspondientes de cada uno de los dos sistemas embebidos para su respectiva evaluación y elección de uno de ellos.

Tabla 5.1: Comparación de los dos sistemas embebidos.

	Jetson Nano 4GB	**Raspberry Pi 4B 4GB**
CPU	Quad-Core ARM Cortex-A57 MPCore	Broadcom BCM2711 Cortex-A72
CPU Frecuencia	64bit Soc a 1.43GHz	64bit 1.5GHz Quad-core
GPU	128-core NVIDIA Maxwell GPU	Broadcom VideoCore VI a 500MHz
AI Perfomance	473 GFLOPS	200 GFLOPS
RAM	4GB 64-bit (LPDDR4 25.65GB/s)	4GB LPDDR4
Wifi	No incluido	2.4G/5G 802.11.b/g/n/ac
Bluetooh	No incluido	Bluetooh 5.0, BLE
Internet	Gigabit Ethernet (RJ45)	Gigabit Ethernet (RJ45)
POE powered	Si	Si
Potencia nominal	5W - 10W	Max 6.7W
Display	HDMI | DisplayPort	Micro HDMI*2 support 4k60
Cámara	CSI/USB	CSI/USB
IO	40 Pin	40 Pin
USB	4*USB 3.0	2*USB 3.0 2*USB 2.0
Precio	S/. 1270.00	S/. 520.00

Dentro de las características principales de los sistemas embebidos, podemos encontrar:

1. **Potencia de procesamiento:**

   - Jetson Nano: El Jetson Nano está equipado con una GPU NVIDIA Maxwell con 128 núcleos CUDA, lo que proporciona un rendimiento excepcional en tareas de

procesamiento de imágenes y aprendizaje profundo.

- Raspberry Pi: El Raspberry Pi 4 cuenta con una GPU VideoCore VI, que es adecuada para aplicaciones de propósito general, pero no es tan potente como la GPU del Jetson Nano.

2. **Memoria:**

   - Jetson Nano: Tiene 4GB de memoria RAM, lo que permite la ejecución eficiente de modelos de redes neuronales grandes.
   - Raspberry Pi: Al igual que el Jetson Nano, cuenta con 4GB de RAM, lo que es suficiente para muchas aplicaciones de aprendizaje profundo.

3. **Consumo de energía:**

   - Jetson Nano: Tiene un mayor consumo de energía en comparación con el Raspberry Pi, lo que podría ser un factor a considerar si se necesita una solución más eficiente en términos de energía. El jetson nano presenta un consumo mínimo de 2.5 vatios y máximo a 10 vatios.
   - Raspberry Pi 4B: Es conocido por su eficiencia energética y puede funcionar con una variedad de fuentes de alimentación. El raspberry pi 4B presenta un consumo mínimo de 1.5 vatios y máximo a 5 vatios.

4. **Soporte de librerías y frameworks:**

   - Jetson Nano: NVIDIA ofrece un amplio soporte para librerías de aprendizaje profundo, incluyendo TensorFlow, PyTorch y cuDNN, lo que facilita el desarrollo de aplicaciones de visión computacional.
   - Raspberry Pi: Aunque también es compatible con bibliotecas de aprendizaje profundo, el soporte de hardware específico no es tan completo como el de Jetson Nano.

5. **Rendimiento en redes neuronales convolucionales:**

    - Jetson Nano: Debido a su GPU de NVIDIA, el Jetson Nano es más eficiente en la ejecución de modelos de redes neuronales convolucionales complejos y grandes.
    - Raspberry Pi: Puede ejecutar modelos de redes neuronales convolucionales, pero su rendimiento no es tan alto como el del Jetson Nano.

Viendo las características antes mencionadas, y en especial los requerimientos de computo obtenidos en la Tabla 4.9 del diseño del sistema y el uso de redes neuronales convolucionales, el Jetson Nano 4GB es la mejor opción debido a su potencia de procesamiento significativamente superior. Su GPU NVIDIA proporciona un rendimiento más adecuado para tareas de visión computacional y aprendizaje profundo. Aunque es un poco más costoso y consume más energía que el Raspberry Pi, su capacidad de ejecución de modelos de redes neuronales convolucionales más complejos lo convierte en la elección más sólida para aplicaciones de detección de somnolencia que requieren un alto rendimiento y precisión. El datasheet del NVIDIA Jetson Nano se puede ver en Anexo 10.

5.1.2. Elección de la pantalla

En el mercado existen varias opciones de pantallas para la implementación del sistema. Donde, para el sistema detector de somnolencia utilizando el Jetson Nano, la "Waveshare 7-inch HDMI LCD (C)" es una excelente elección de pantalla táctil de 7 pulgadas. A continuación, se proporciona algunas razones para esta elección:

- **Tamaño ideal:** Con 7 pulgadas, ofrece un tamaño de pantalla cómodo para mostrar información a los conductores sin ocupar demasiado espacio en el vehículo.
- **Resolución adecuada:** Tiene una resolución de 1024x600 píxeles, lo que proporciona una calidad de imagen adecuada para la mayoría de las aplicaciones de detección de somnolencia.

- **Tecnología táctil capacitiva:** La pantalla táctil capacitiva permite una interacción precisa y sensible, lo que es importante para cualquier sistema de detección de somnolencia que requiera entrada del usuario.

- **Parlantes incorporados:** Viene con dos parlantes incorporados que proporcionan salida de audio directamente desde la pantalla. Esto es beneficioso para proporcionar alertas de sonido o información audible al conductor.

- **Compatibilidad con Jetson Nano:** Es compatible con el Jetson Nano, lo que facilita su integración en el sistema sin problemas. La conexión HDMI estándar hace que sea fácil conectarla al Jetson Nano.

Por lo tanto, la pantalla táctil Waveshare 7-inch HDMI LCD (C) es una excelente elección para un sistema detector de somnolencia con el Jetson Nano debido a su tamaño adecuado, calidad de imagen, tecnología táctil capacitiva y compatibilidad con el Jetson Nano. Esta pantalla proporcionará una interfaz efectiva con el conductor y se puede integrar fácilmente al sistema de detección de somnolencia. El datasheet de la pantalla se muestra en Anexo 11.

5.1.3. Elección de la fuente de alimentación

Para alimentar el Jetson Nano desde la batería de un automóvil, se necesita una fuente de alimentación que sea capaz de convertir la tensión de la batería del automóvil (generalmente 12V) a la tensión requerida por el Jetson Nano (generalmente 5V). Una fuente de alimentación adecuada para esta tarea es un "convertidor DC-DC step-down" o regulador de voltaje. A continuación, se hace la elección del Convertidor DC-DC Step-Down (Buck) LM2596.

El LM2596 es un convertidor de voltaje muy utilizado y ampliamente disponible que puede convertir la tensión de la batería de un automóvil (12V) en los 5V requeridos por el Jetson Nano. Aquí hay algunas razones para considerar este convertidor:

- **Eficiencia:** El LM2596 es conocido por su eficiencia en la conversión de voltaje, lo que significa que no desperdiciará mucha energía en forma de calor.
- **Ajustable:** Se puede ajustar la tensión de salida para garantizar que suministre exactamente 5V al Jetson Nano.
- **Protección:** Algunos modelos vienen con protección contra sobrecorriente y sobretensión para proteger al Jetson Nano.

El datasheet del Convertidor DC-DC Step-Down LM2596 se muestra en el Anexo 12.

5.2. Método de la implementación del sistema de detección de somnolencia

El método de implementación de un sistema de detección de somnolencia es un proceso que utiliza tecnología y algoritmos para identificar signos de fatiga o somnolencia en un conductor mientras opera un vehículo. Este método implica la adquisición de datos de un conductor, normalmente a través de una cámara que monitorea los movimientos faciales y oculares, seguido por un análisis en tiempo real de estos datos para determinar si el conductor muestra señales de estar quedándose dormido o desatento al volante.

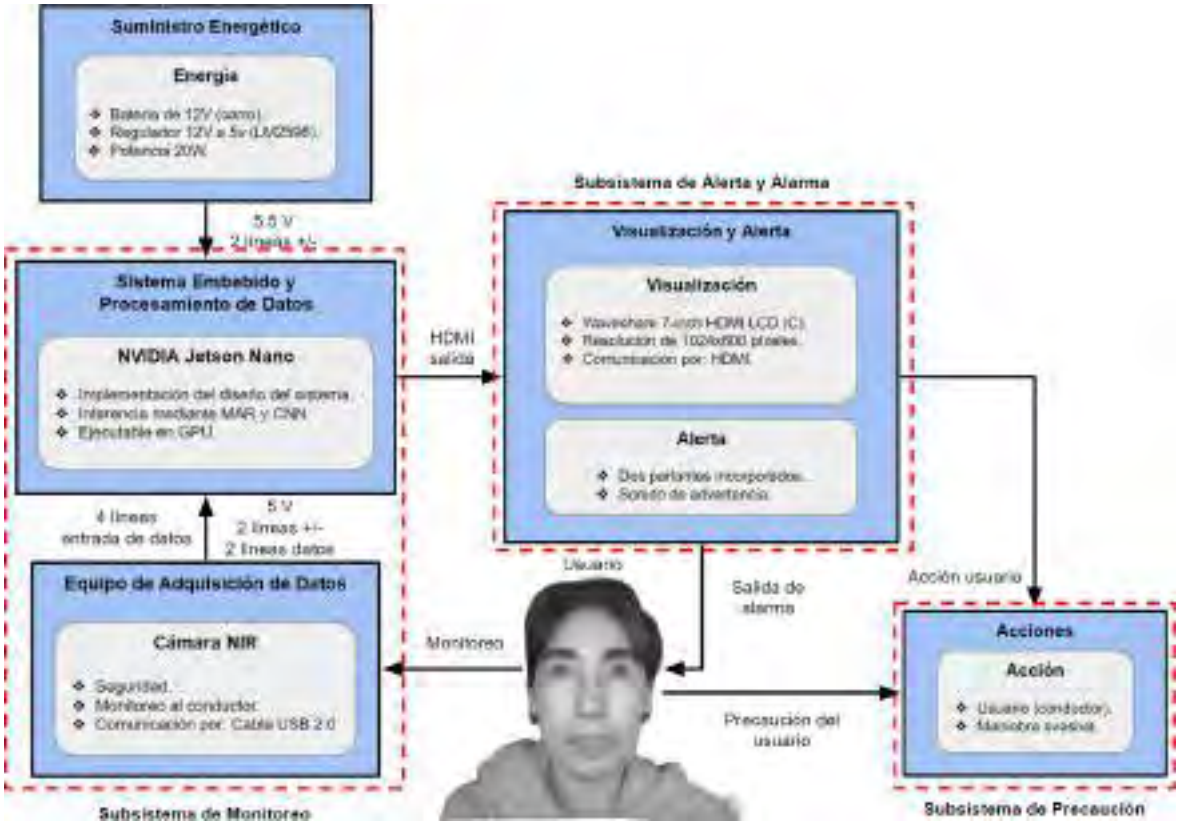
Como se vio en la sección 4.2 el diseño en software del sistema detector de somnolencia mediante Mouth Aspect Ratio (MAR) y Redes Neuronales Convolucionales (CNN), correspondiente a los pasos de adquisición de datos, detección de rostro, ojos y boca, análisis de somnolencia, establecimiento de umbrales, detección de somnolencia y alarma visual.

Este diseño es una propuesta híbrida que se tiene que implementar dentro del sistema embebido elegido que es el NVIDIA Jetson Nano, el cual primero se tiene que configurar de acuerdo a los requisitos de computo, instalando las bibliotecas y frameworks requeridos para su correcto funcionamiento, luego se tiene que hacer la conexión de los diversos componentes

como la cámara NIR, el suministro energético y la pantalla con los parlantes correspondiente a la visualización y alerta del estado del conductor, para finalmente, el conductor tomar las acciones necesarias.

El diagrama de funcionamiento del sistema de detección de somnolencia se presenta en la Figura 5.1. Como se ilustró previamente en la Figura 3.2, que describe los subsistemas del sistema de detección de somnolencia, se profundizara en el sistema embebido y el procesamiento de datos. En esta sección, se detallará la configuración inicial del hardware, seguida de la conexión de los dispositivos de adquisición de datos, ambos componentes pertenecientes al subsistema de monitoreo. A continuación, se llevará a cabo la conexión del suministro de energía al hardware y se establecerá la vinculación con la pantalla y los altavoces que conforman el subsistema de alerta y alarma. Finalmente, se abordará la interacción del usuario, es decir, el conductor, que forma parte del subsistema de precaución.

Figura 5.1: Esquema del sistema detector de somnolencia.



5.2.1. Subsistema de monitoreo

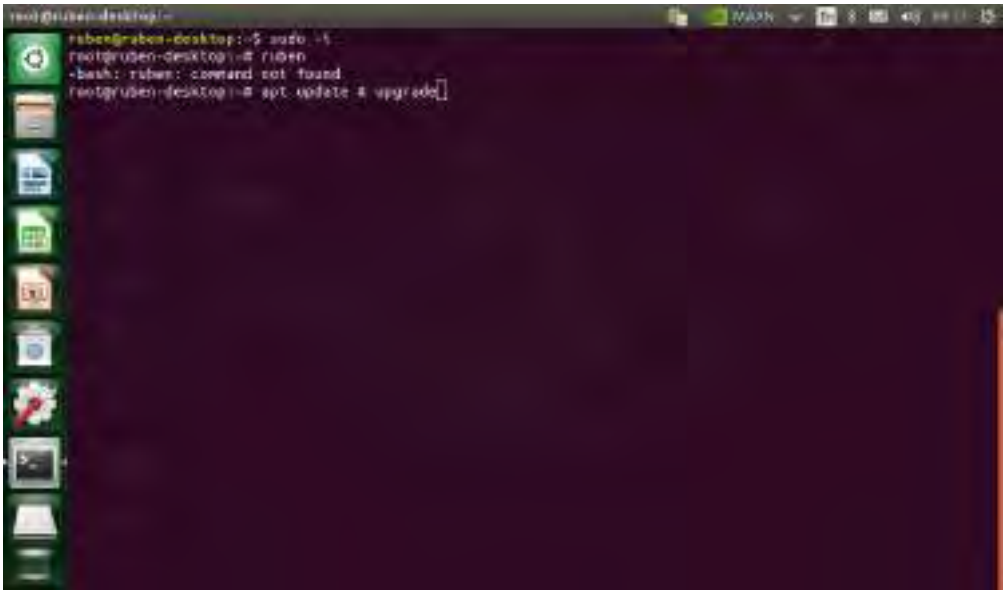
El subsistema de monitoreo esta conformado por dos componentes, sistema embebido y procesamiento de datos, y equipo de adquisición de datos. En esta sección se verá la configuración del NVIDIA Jetson Nano y finalmente la conexión con la cámara NIR.

5.2.1.1. Sistema embebido y procesamiento de datos

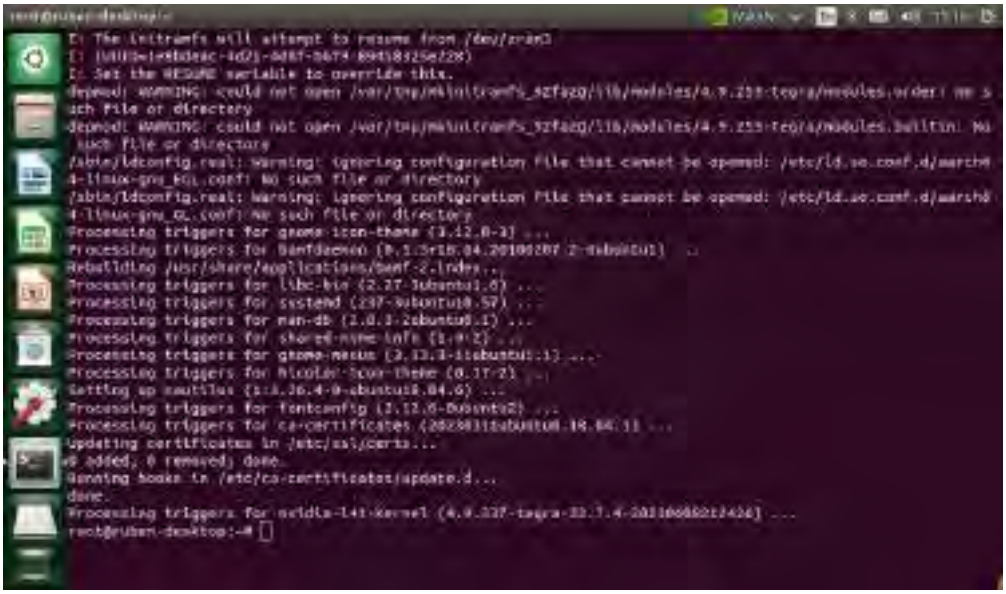
En primer lugar, es esencial realizar la configuración inicial del hardware NVIDIA Jetson Nano, que incluye la instalación del sistema operativo y la configuración de las bibliotecas y frameworks de trabajo necesarios. Posteriormente, se procederá con la importante tarea de adaptar los modelos de redes neuronales convolucionales (CNN) previamente entrenados para que sean compatibles y eficientes en el hardware.

5.2.1.1.1. Configuración del hardware: La instalación del sistema operativo comienza con la descarga del archivo correspondiente. En este caso, se emplea la versión Jetpack 4.6.4, que utiliza el sistema operativo Ubuntu 18.04 LTS de 64 bits. Para llevar a cabo este proceso de instalación de manera efectiva, es importante contar con un nivel intermedio de conocimiento en el sistema operativo GNU/Linux. Esto se debe a que el procedimiento implica la ejecución de comandos en una terminal para instalar los paquetes necesarios y configurar el sistema adecuadamente. Para realizar la instalación del sistema operativo, se requiere un programa que pueda grabar la imagen en una tarjeta de memoria SD. En este caso, se utilizó Balena Etcher, y se empleó una tarjeta de memoria de 64 GB. Se recomienda el uso de una tarjeta de memoria de 32 GB o superior, dado que la imagen del sistema operativo tiene un tamaño de 14.4 GB. Después de grabar la imagen, se inicia el sistema operativo en el Jetson Nano desde la tarjeta SD. Se recomienda crear un usuario "root" y establecer una contraseña. Luego de iniciar el S.O., se debe actualizar el sistema mediante los comandos "sudo apt update" y "sudo apt upgrade". Después de completar la actualización, se reinicia el hardware. Su ejecución puede observarse en la Figura 5.2.

Figura 5.2: Actualización del sistema operativo - sistema embebido.



(a) Comandos de actualización.

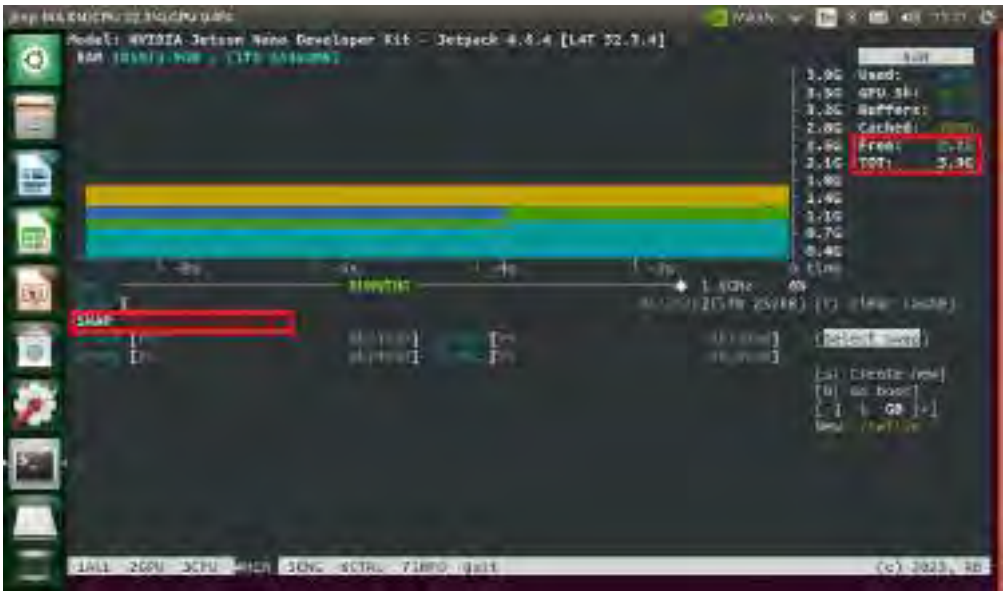


(b) Respuesta de actualización.

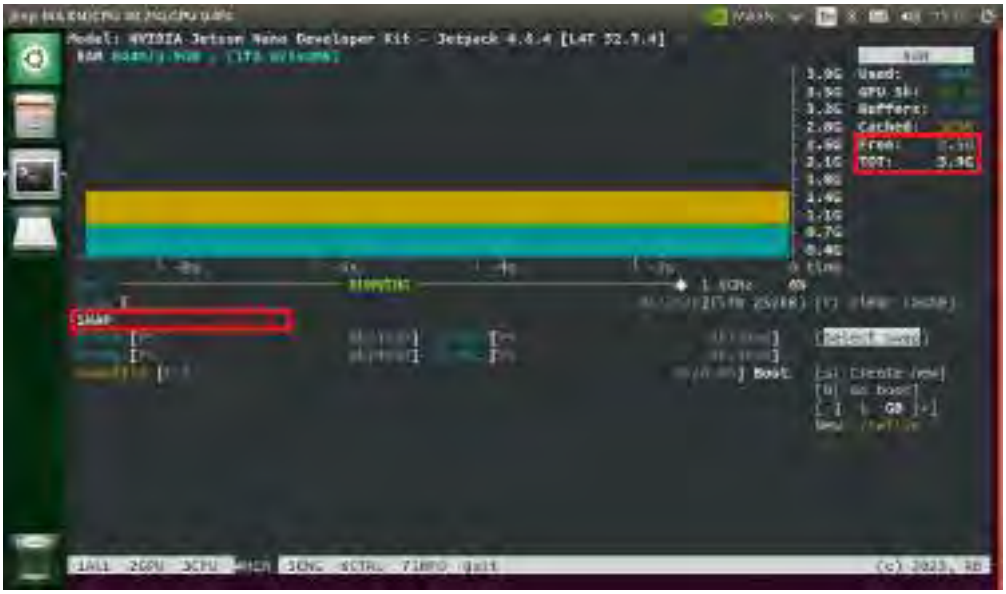
Seguidamente, es necesario instalar la memoria swap, que se refiere a una parte del espacio de almacenamiento, generalmente en una tarjeta microSD o un dispositivo de almacenamiento adicional, que se utiliza como una extensión de la memoria RAM física del sistema cuando esta última se agota. La memoria swap permite a un sistema, como el Jetson Nano, utilizar el espacio en disco como una especie de "memoria virtual" para

almacenar datos y programas cuando la RAM física se está utilizando en su totalidad. Los comando necesarios para la instalación se encuentra en el artículo original de JetsonHacks, solo es necesario clonar el github, luego entrar a la carpeta “installSwapfile” y ejecutar el archivo “./installSwapfile.sh”, la instalacion es de 6 GB de memoria swap. La muestra de la memoria swap se muestra en la Figura 5.3.

Figura 5.3: Instalación de memoria swap - sistema embebido.



(a) Sin memoria swap.



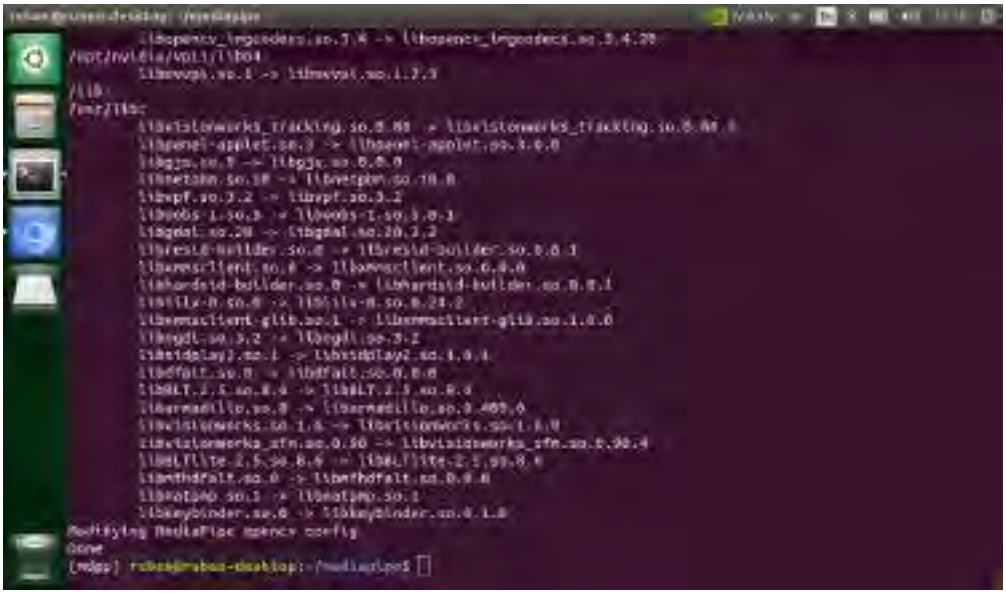
(b) Con memoria swap.

Después de completar los pasos iniciales de configuración del sistema operativo en el Jetson Nano, es aconsejable crear un entorno virtual. Esto habilita la instalación de paquetes específicos destinados a un proyecto en particular. En otras palabras, un entorno virtual permite mantener un conjunto aislado de paquetes y bibliotecas para un proyecto específico, independiente de la instalación principal de Python en el sistema. Esta práctica es fundamental para evitar conflictos entre dependencias y asegurar la consistencia del entorno de desarrollo.

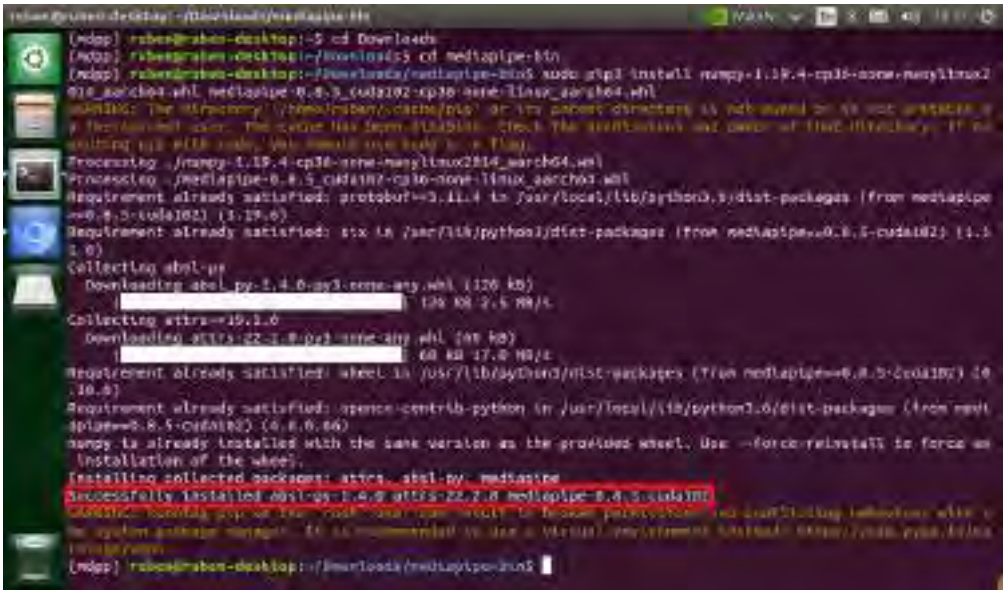
Después de crear el entorno virtual, el siguiente paso implica la instalación de las bibliotecas y frameworks necesarios. En este contexto, es esencial instalar dos bibliotecas fundamentales: MediaPipe y OpenCV. Estas bibliotecas son esenciales para garantizar el funcionamiento adecuado y eficiente del diseño del sistema durante su implementación.

1. **Instalación de MediaPipe:** Al momento de instalar MediaPipe en el Jetson Nano, se debe compilar desde el código fuente, ya que la instalacion normal en GNU/Linux no funciona, esto por la arquitectura del Jetson Nano. MediaPipe hace uso de Tensorflow, por lo que, en la instalación se procede a instalar Tensorflow. Al seguir el tutorial de github de Melvinsajith, se instala correctamente MediaPipe, en este tutorial se omite la primera parte del paso 4, ya que se hizo la previa instalación de la memoria swap. Al ejecutar el archivo "./setup_opencv.sh" tiene que salir como en la Figura 5.4a sin obtener ningún error. En el paso 5, al instalar opencv contrib, se tiene que hacer con la versión 4.6.0.66, esta versión es compatible con cuda. Finalmente, se instala mediapipe-0.8.5-cuda102 (Figura 5.4b), lo cual se ejecuta en la GPU del Jetson Nano.

Figura 5.4: Instalación de MediaPipe - sistema embebido.



(a) Salida de ejecución.

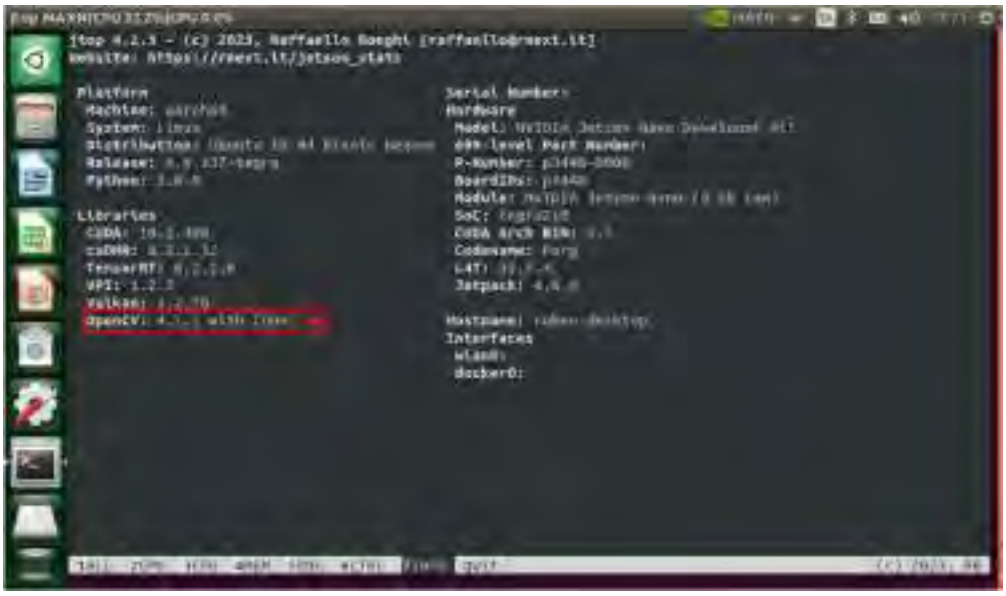


(b) MediaPipe instalado.

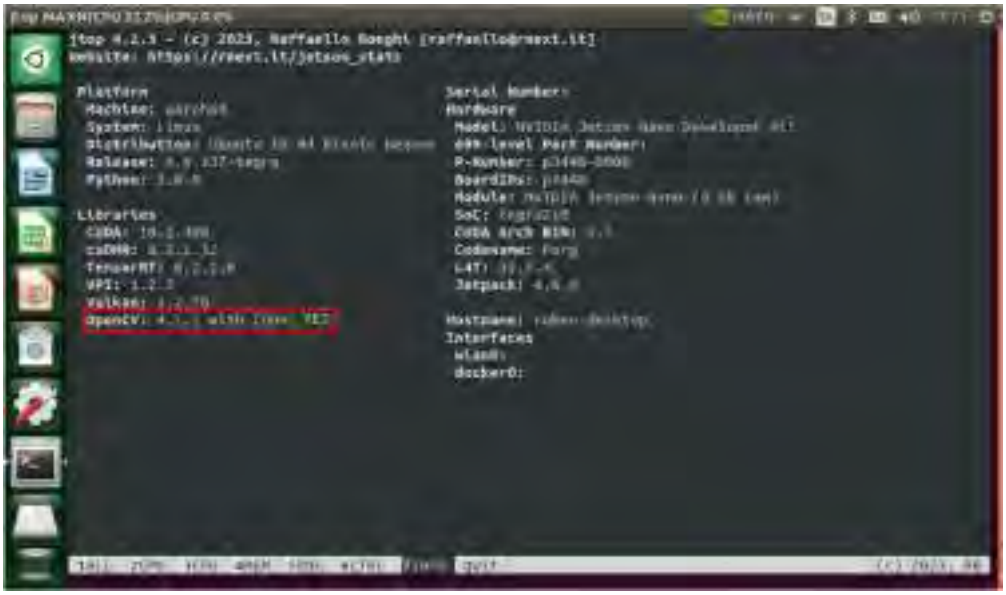
2. **Instalación de OpenCV:** OpenCV ya se encuentra preinstalado en Jetpack 4.6.4, pero su ejecución predeterminada se realiza en la CPU. Sin embargo, para aprovechar al máximo el rendimiento de la GPU en el Jetson Nano, es recomendable reinstalar OpenCV con soporte para CUDA. En este caso particular, se optó por la versión 4.1.1 de OpenCV-cuda.

Siguiendo el tutorial proporcionado por Q-engineering, desde la sección “Dependencies” hasta “Checking”, la instalación se realiza de manera exitosa. Es importante mencionar que este proceso de instalación puede llevar de una hora y media a dos horas, por lo que se requiere paciencia durante este periodo. La Figura 5.5a muestra OpenCV sin cuda y la Figura 5.5b muestra OpenCV con cuda.

Figura 5.5: Instalación de OpenCV - sistema embebido.



(a) OpenCV sin cuda



(b) OpenCV con cuda

Con esta configuración, el Jetson Nano está preparado para llevar a cabo la detección de somnolencia en tiempo real en los conductores utilizando el indicador de la relación de aspecto de la boca (Mouth Aspect Ratio, MAR). Sin embargo, hasta este punto, solo se ha instalado TensorFlow, lo que permite la ejecución de modelos de redes neuronales convolucionales (CNN). No obstante, se presenta un desafío: el GPU del Jetson Nano no admite el formato de archivo ".h5". Como solución, se requiere convertir estos modelos a un formato compatible con la GPU del Jetson Nano. En esta investigación, se ha optado por el formato ".onnx" debido a su amplia compatibilidad con sistemas embebidos, lo que resulta en una mejora significativa en el rendimiento de la inferencia de los modelos CNN.

5.2.1.1.2. Compatibilidad de modelos CNNs entrenados: Para lograr un rendimiento óptimo en el Jetson Nano, es fundamental adaptar los modelos de redes neuronales convolucionales (CNN) previamente entrenados a un formato que sea compatible con la GPU del Jetson Nano. Esto permitirá aprovechar al máximo el potencial de la GPU mediante TensorRT, acelerando significativamente el proceso de inferencia para la detección de la somnolencia visual sin notables demoras en la respuesta.

El código de conversión de un modelo desde el formato ".h5" al formato ".onnx" se detalla en el Anexo 13. Es importante destacar que el modelo en formato ".onnx" exhibe diferencias notables en comparación con el modelo ".h5", en términos de tamaño del archivo y tiempo de respuesta, como se ilustra en la Tabla 5.2. Estas diferencias se deben a la orientación de ".onnx" hacia la optimización de la inferencia, lo que resulta en la compresión de los pesos del modelo original.

Tabla 5.2: Comparación de los modelos h5 y onnx.

	h5		onnx	
	Tamaño de archivo (KB)	Tiempo de respuesta (ms)	Tamaño de archivo (KB)	Tiempo de respuesta (ms)
InceptionV3	98,055	51.01	89,091	57.00
VGG16	69,586	39.00	61,494	43.08
ResNet50V2	140,594	60.02	108,012	65.00
DD-AI	75,618	33.00	25,191	39.00
DD-AI-G	75,608	30.00	25,187	34.00

Con la configuración de hardware completa y los modelos CNN adaptados para funcionar eficientemente en la GPU del Jetson Nano, el sistema está completamente preparado para llevar a cabo la detección de somnolencia. Este proceso se realiza mediante redes neuronales convolucionales (CNN) que analizan el estado de los ojos y la relación de aspecto de la boca (Mouth Aspect Ratio, MAR), permitiendo una detección precisa y en tiempo real.

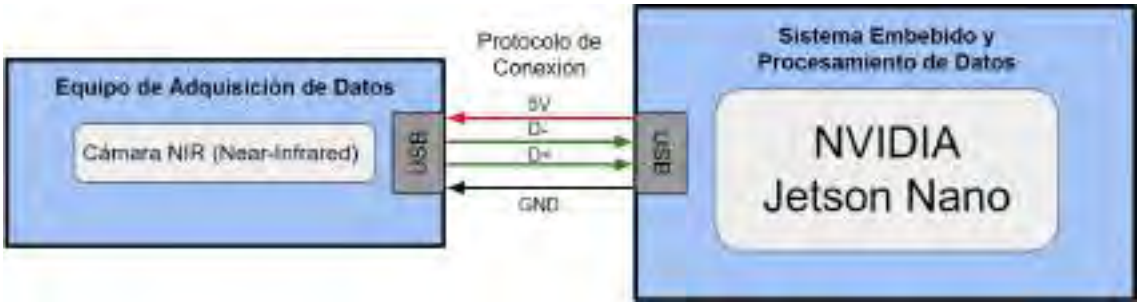
5.2.1.2. Equipo de adquisición de datos

Como se vio en la sección 4.1.1, se eligió a la cámara NIR ELP-USBFHD05MT-KL36IR, el cual puede capturar de manera efectiva las características faciales del conductor, como los ojos y la boca, en condiciones de poca luz sin depender del uso de cristales como los lentes o gafas de sol. Esto permite el seguimiento y análisis de los indicadores de somnolencia, como el cierre de los ojos y el bostezo, dando seguridad al conductor en el monitoreo continuo.

La conexión al sistema embebido (Jetson Nano) se realiza a través de un cable USB 2.0 que desempeña un doble papel: suministrar energía eléctrica desde el Jetson Nano hacia la cámara y transmitir los datos de vídeo desde la cámara NIR hacia el Jetson Nano. Este cable utiliza dos pines para la transmisión de energía y otros dos para la transmisión de datos, asegurando una comunicación eficiente entre los componentes del sistema.

A continuación, en la Figura 5.6 se observa el esquema correspondiente al subsistema de monitoreo.

Figura 5.6: Esquema del subsistema de monitoreo.



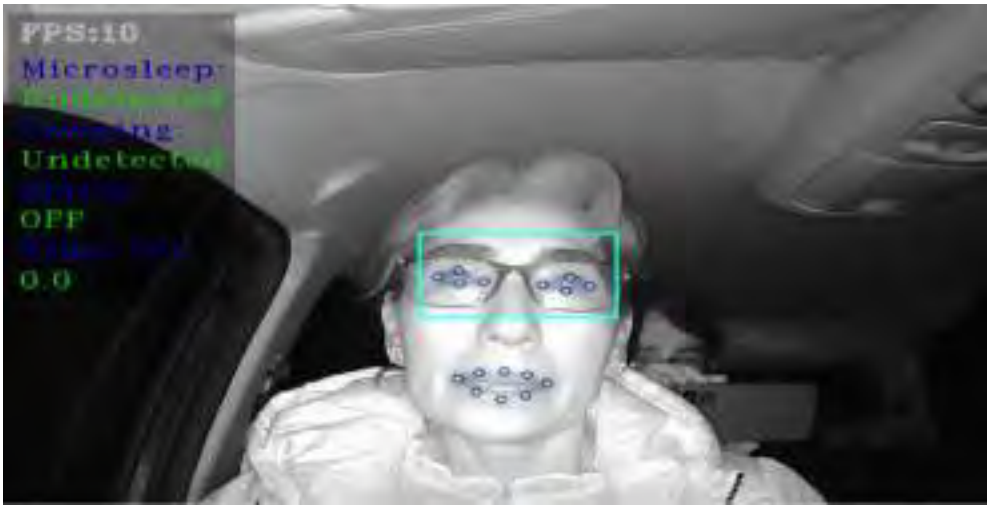
5.2.2. Subsistema de alerta y alarma

El subsistema de alerta y alarma está compuesto por la pantalla "Waveshare 7-inch HDMI LCD (C)", que incluye dos altavoces incorporados en la parte trasera. En esta sección, se presenta la interfaz gráfica (GUI) diseñada para la visualización, así como el control de los altavoces utilizados para emitir la alerta.

5.2.2.1. Visualización

Con el objetivo de mejorar el control del monitoreo del conductor, se ha desarrollado una interfaz gráfica (GUI) que proporciona una visualización detallada del estado del conductor. Esta interfaz permite observar la detección de micro-sueño (somnolencia visual), el estado de la boca (bostezo), la activación de alarmas y el tiempo de seguimiento del estado de los ojos. Además, se incluye la representación visual de la región de interés (ROI) que se enfoca en la zona de los ojos, así como los puntos de referencia de esta región. Asimismo, se muestran los ocho puntos de referencia relacionados con la boca. La Figura 5.7 proporciona una vista general de la GUI.

Figura 5.7: Interfaz gráfica para la visualización del estado del conductor.



5.2.2.2. Alerta

La alerta del sistema se activa a través de los dos parlantes cuando se detectan signos de somnolencia en el conductor. Su objetivo principal es despertar al conductor o mantenerlo alerta para prevenir situaciones peligrosas, como micro-sueños, bostezos o somnolencia al volante.

Esta alerta se manifiesta en forma de un sonido fuerte que advierte al conductor sobre su estado de somnolencia. La duración de la alerta puede variar según la gravedad de los signos de somnolencia detectados, como el cierre prolongado de los ojos (superior a 300 ms) o la apertura de la boca en forma de bostezo (superior a 5 segundos).

El tipo de control de la alerta es ON/OFF en lazo cerrado. Al tener una retroalimentación activa que monitorea continuamente la respuesta del conductor después de activar la alarma. Si el conductor responde adecuadamente (por ejemplo, abre los ojos o deja de bostezar), el sistema puede tomar esta retroalimentación en tiempo real y ajustar la alarma según sea necesario. El sistema responde activamente a la retroalimentación del conductor para ajustar la alarma de acuerdo con las necesidades y el estado actual del conductor, lo que mejora la eficacia del sistema en la prevención de somnolencia al volante.

El diagrama de bloques del control de la alerta se muestra en la Figura 5.8. Donde la explicación de cada bloque se detalla a continuación:

1. **Entrada de vídeo de la cámara:** Esta es la señal de vídeo capturada por la cámara que supervisa al conductor en tiempo real.

2. **Detección de somnolencia:** Este bloque detecta signos de somnolencia en el conductor basándose en la entrada de vídeo de la cámara.

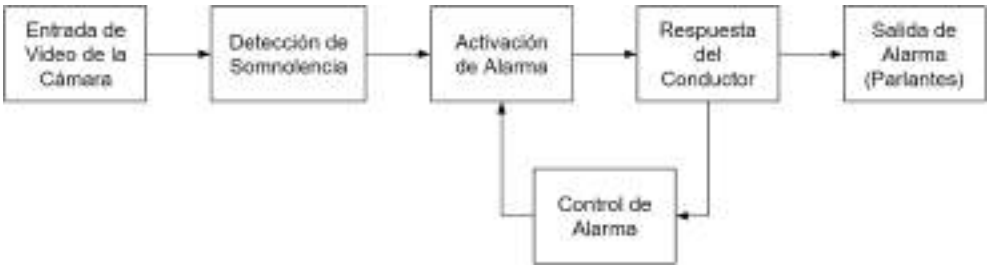
3. **Activación de alarma:** Cuando se detectan signos de somnolencia que superan los umbrales predefinidos, se activa la alarma para alertar al conductor. La alarma es una respuesta del sistema a la detección de somnolencia.

4. **Respuesta del conductor:** La respuesta del conductor se monitorea continuamente. Si el conductor responde adecuadamente a la alarma (abriendo los ojos o parando el bostezo), se envía una señal de respuesta al sistema.

5. **Control de alarma:** El bloque de control de alarma ajusta la alarma en función de la respuesta del conductor. Si el conductor responde correctamente, la alarma se apaga. Esto representa el lazo cerrado, ya que la respuesta del conductor influye en la acción del sistema (control de la alarma) para mantener la seguridad y minimizar las distracciones innecesarias.

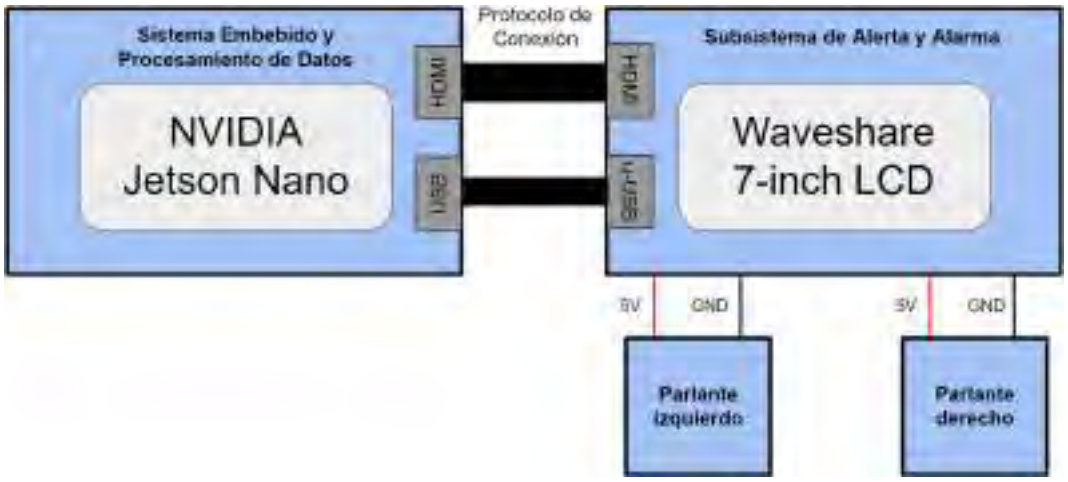
6. **Salida de alarma:** La alarma activada se transmite a través de los dos parlantes incorporados en la parte trasera de la pantalla.

Figura 5.8: Diagrama de bloques del control de la alerta.



A continuación, en la Figura 5.9 se presenta el esquema correspondiente al subsistema de alerta y alarma. Donde, la conexión de la pantalla con el sistema embebido se hace mediante dos cables, HDMI y micro-USB. Siendo el cable micro-USB para la alimentación de la pantalla, además, la conexión con los dos parlantes es mediante dos cables, que es la alimentación a cada parlante.

Figura 5.9: Esquema del subsistema de alerta y alarma.



5.2.3. Subsistema de precaución

El subsistema de precaución dentro un sistema detector de somnolencia es una parte integral del sistema diseñada para tomar medidas preventivas y asegurar la seguridad durante la conducción. Su propósito principal es alertar al conductor cuando se detectan signos de somnolencia, ayudar a mantener al conductor despierto y consciente, en última instancia, prevenir accidentes relacionados con la somnolencia al volante. El subsistema de precaución utiliza una variedad de métodos, como alertas visuales y sonoras, para interactuar con el conductor y garantizar una respuesta adecuada ante los signos de fatiga. Este subsistema se centra en la acción que toma el conductor en respuesta a la somnolencia, en contraposición a la respuesta del conductor, que puede manifestarse en acciones como abrir los ojos o detener un bostezo.

En un nivel más específico, el subsistema de precaución de un sistema detector de somnolencia puede incluir:

1. **Alertas sonoras y visuales:** Utiliza parlantes y pantalla para emitir alertas audibles y visuales cuando se detectan signos de somnolencia en el conductor. Estas alertas incluyen sonidos fuertes, que advierten al conductor.

2. **Interfaz de usuario:** Proporciona una interfaz de usuario para configurar las preferencias del conductor y ajustar la sensibilidad de las alertas. Esto puede incluir botones o controles en el tablero o una aplicación en un dispositivo móvil.

3. **Respuesta del conductor:** Supervisa la respuesta del conductor a las alertas emitidas, observando cómo el conductor reacciona ante los signos de somnolencia. Esto puede manifestarse en acciones como maniobras evasivas o la reducción de la velocidad del vehículo, medidas que el conductor toma en respuesta a las alertas para mantenerse alerta y seguro en la carretera.

En conjunto, el subsistema de precaución tiene como objetivo proporcionar una estrategia efectiva para mantener al conductor alerta y prevenir situaciones peligrosas en la carretera. La retroalimentación constante del conductor y la capacidad de adaptación hacen que este subsistema sea fundamental para la seguridad en la conducción.

5.2.4. Bloque de suministro energético

El suministro energético se refiere al sistema que proporciona la energía eléctrica necesaria para alimentar y hacer funcionar de manera confiable todos los componentes del sistema detector de somnolencia que incluye un Jetson Nano, una pantalla de 7 pulgadas y una cámara NIR. Este suministro energético es fundamental para garantizar que el sistema opere de manera continua y efectiva durante su implementación en un vehículo.

El consumo energético puede variar según varios factores, como la configuración específica del hardware, la intensidad del uso y la eficiencia energética de los componentes individuales.

Sin embargo, se realiza un cálculo promedio del consumo energético de todo el sistema detector de somnolencia, teniendo en cuenta la potencia máxima de cada componente, con el objetivo de realizar una estimación para un período de 6 horas. Este lapso de tiempo se considera recomendable para la conducción. La Tabla 5.3 muestra el consumo energético de los componentes.

Tabla 5.3: Cálculo de consumo de potencia de los componentes del sistema.

Componente	Voltaje	Corriente	Potencia	Horas	Consumo
NVIDIA Jetson Nano	5 V	2 A	10 W	6 h	60 W-h
Cámara NIR	5 V	0.22 A	1.1 W	6 h	6.6 W-h
Pantalla LCD 7-inch	5 V	1.6 A	8 W	6 h	48 W-h
Potencia total (W) y Consumo total (W-h)			19.1 W		114.6 W-h

Por lo tanto, se espera que el sistema consuma aproximadamente 114.6 vatios-hora de energía en un período de 6 horas con los valores estimados para cada componente.

Ahora, se tiene una batería de automóvil típica con una capacidad de 60 amperios-hora (Ah). Para calcular si el consumo es viable durante 6 horas, primero se debe convertir los vatios-hora a amperios-hora, ya que la capacidad de la batería se mide en amperios-hora (Ah).

El consumo total calculado anteriormente es de 114.6 vatios-hora. Para convertir esto a amperios-hora, se usa la ecuación 5.1:

$$Amperios - hora(Ah) = Vatios - hora(Wh)/Voltios(V) \tag{5.1}$$

La mayoría de las baterías de automóvil funcionan a 12 voltios (V), por lo que al aplicar la ecuación 5.1, se obtiene:

$$Amperios - hora(Ah) = 114{,}6 vatios - hora/12V = 9{,}55 Ah$$

Entonces, el consumo durante 6 horas sería de aproximadamente 9.55 Ah. En general, si la capacidad de la batería es sustancialmente mayor que el consumo estimado y la batería está en buen estado, debería ser capaz de alimentar el sistema durante 6 horas sin problemas.

Anteriormente, se eligió la fuente de alimentación DC-DC Step-Down (Buck) LM2596 con corriente de salida máximo de 4 A. Para determinar si la fuente de alimentación satisface el consumo total estimado del sistema, primero se debe comparar la capacidad de salida de la fuente con el consumo total calculado.

El consumo total calculado anteriormente es de 114.6 vatios-hora durante 6 horas, lo que equivale a un promedio de aproximadamente 19.1 vatios (W).

Ahora, para evaluar si la fuente de alimentación satisface este consumo, se considera la potencia (P) en relación con la corriente (I) y el voltaje (V) utilizando la siguiente ecuación 5.2:

$$P = V * I \tag{5.2}$$

Donde:

- P es la potencia en vatios (W).
- V es el voltaje en voltios (V).
- I es la corriente en amperios (A).

Si la fuente proporciona 5V a 4A, se calcula su potencia:

$$P_1 = 5V * 4A = 20W$$

La fuente de alimentación tiene una capacidad de 20 vatios (W). Dado que la capacidad de la fuente de alimentación (20 W) es ligeramente mayor que el consumo promedio (19.1 W). Por lo tanto, esta fuente es adecuada para alimentar el sistema durante 6 horas.

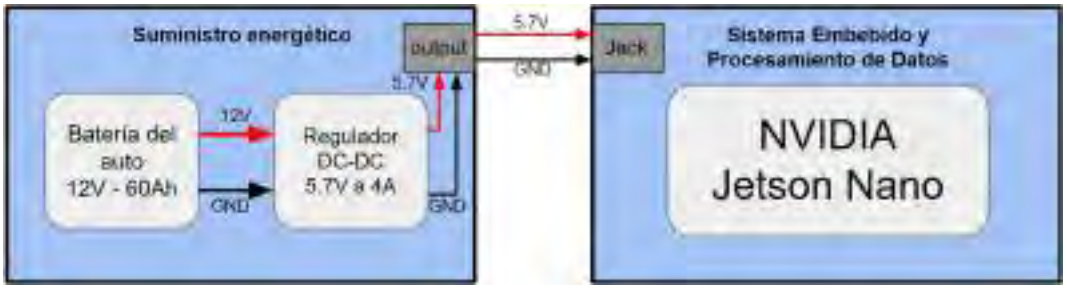
Dado que, en muchos casos se tiene una caída de tensión, el regulador puede llegar a 4.7 V, lo cual puede incurrir a bajar la capacidad de la fuente de alimentación y no ser adecuado para el consumo que requiere el sistema. En estos casos, es recomendable subir el voltaje del regulador a 5.7 V para compensar la caída de tensión producida por el consumo de los diferentes componentes, siendo así, se hace un nuevo calculo mediante la ecuación 5.2:

$$P_2 = 5{,}7V * 4A = 22{,}8W$$

Con la capacidad mejorada de la fuente de alimentación, que ahora es de 22.8 vatios (W), se asegura un funcionamiento óptimo del sistema detector de somnolencia, el cual requiere en promedio 19.1 vatios (W) de potencia máxima.

A continuación, en la Figura 5.10 se observa el esquema correspondiente al bloque de suministro energético.

Figura 5.10: Esquema del bloque de suministro energético.



5.3. Construcción del sistema de detección de somnolencia

La construcción del sistema detector de somnolencia es un proceso que implica la integración de software (sección 4.2) y hardware (sección 5.2) para garantizar la seguridad durante la conducción. Este sistema esta diseñado para identificar signos de somnolencia en el conductor y tomar medidas preventivas, como activar alertas e intervenir en el estado del

conductor, para evitar accidentes relacionados con la somnolencia al volante.

El proceso de construcción del sistema detector de somnolencia sigue los pasos detallados en la sección 5.2. Una vez que se han adquirido los componentes necesarios, se procede con el ensamblaje del sistema, que se describe a continuación:

1. **Adquisición de los componentes:** La mayoría de los componentes utilizados en la construcción del sistema detector de somnolencia fueron importados de un país extranjero, específicamente de China. Estos componentes incluyen el NVIDIA Jetson Nano (kit B01), la cámara NIR ELP-USBFHD05MT-KL36IR, la pantalla táctil Waveshare 7-inch HDMI LCD (C), el protector Jetson Nano (DeskPi) y el convertidor DC-DC Step-Down (Buck) LM2596 4A. Por otro lado, algunos componentes menores, como el cargador de carro, cables, conectores, entre otros, fueron adquiridos localmente. La Figura 5.11 ilustra todos estos componentes que se utilizaron en la construcción del sistema detector de somnolencia.

Figura 5.11: Componentes del sistema detector de somnolencia.

2. **Asignación de puertos para los componentes:** Para llevar a cabo el ensamblaje de los componentes, es esencial identificar los puertos del Jetson Nano que se utilizarán para las conexiones, como se explicó anteriormente. En el Jetson Nano, se

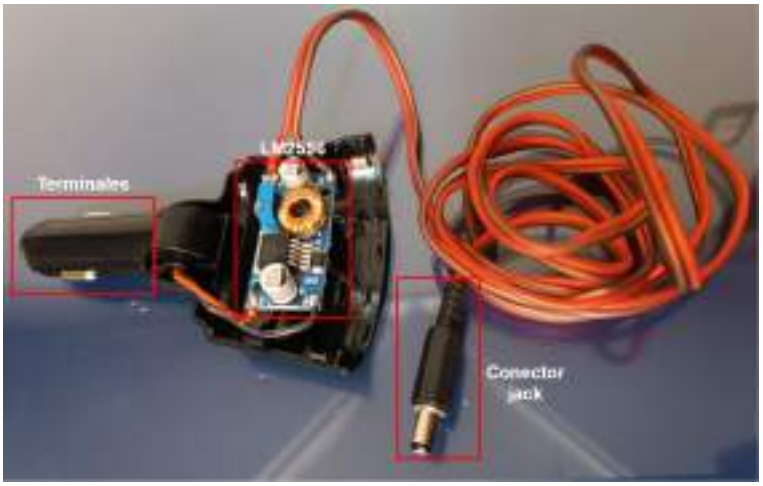
conectan tres componentes claves: la cámara NIR (Cámara), la pantalla LCD (HDMI y Pantalla), y la fuente de alimentación DC (Fuente). Los puertos específicos utilizados para estas conexiones se detallan claramente en la Figura 5.12. Esto asegura una correcta integración de los componentes en el sistema.

Figura 5.12: Puertos asignados del Jetson Nano.

3. **Adaptación del suministro energético:** Con el objetivo de garantizar que el sistema detector de somnolencia no dependa de una batería portátil y pueda operar de manera continua, se realiza una adaptación mediante el uso de un cargador de automóvil.

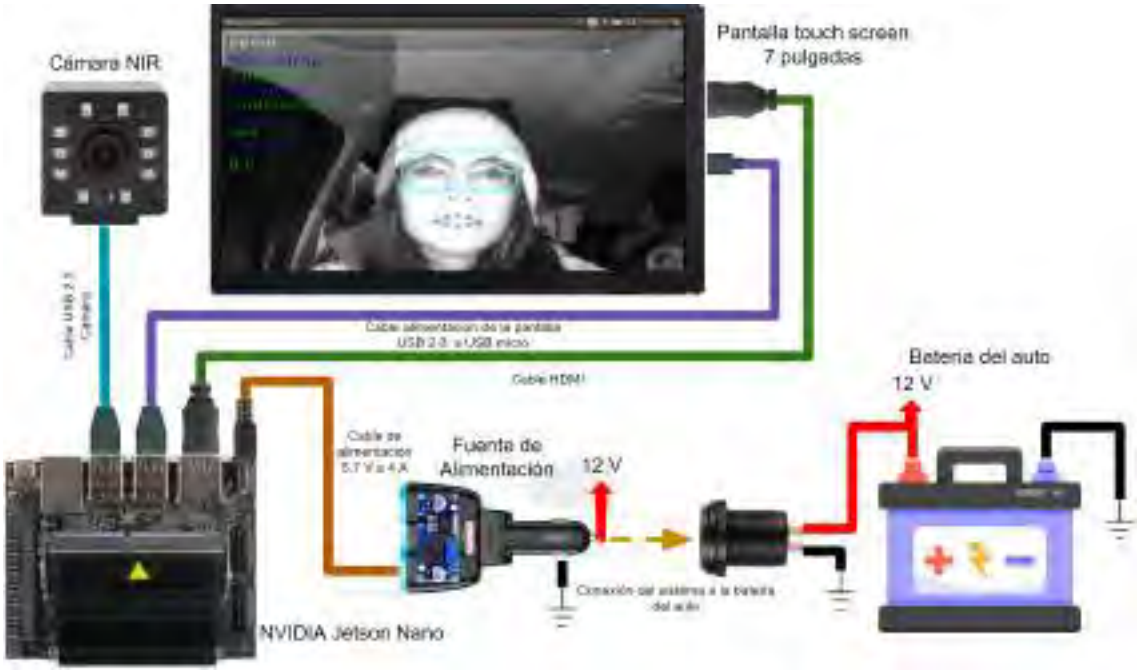
   Los terminales del cargador se conectan al conector de 12V del vehículo, y desde allí se establece una conexión con las entradas del regulador LM2596. Posteriormente, la salida del regulador se adapta a través de un conector jack mediante un cable de dos líneas. Esta adaptación del suministro energético se ilustra en la Figura 5.13.

Figura 5.13: Fuente adaptada para el sistema.



4. **Diagrama de conexión eléctrica del sistema detector de somnolencia:** Para realizar el ensamblaje del sistema, es necesario proporcionar el diagrama de conexión eléctrica del sistema. El cual ayuda a comprender cómo estos elementos interactúan entre sí mediante conexiones eléctricas, proporcionando una guía clara para la implementación y el mantenimiento del sistema. El cual se puede ver en la Figura 5.14.

Figura 5.14: Diagrama de conexiones eléctricas del sistema.



5. **Ensamblaje del sistema detector de somnolencia:** Para concluir el proceso de ensamblaje del sistema detector de somnolencia, se procede a unir todos los componentes de manera integrada. Esto implica montar la pantalla sobre el protector del Jetson Nano y conectar los cables de los diferentes componentes a los puertos asignados del Jetson Nano. Para ofrecer una visualización más detallada del sistema ensamblado, se presenta la vista frontal en la Figura 5.15a, mientras que en la Figura 5.15b se proporciona una vista posterior que muestra cómo se configuran los componentes en el conjunto completo.

Figura 5.15: Ensamblaje del sistema detector de somnolencia.

(a) Vista frontal

(b) Vista posterior

Capítulo 6

Pruebas y resultados

Las pruebas iniciales del sistema se realizaron con los componentes Nvidia Jetson Nano, la pantalla táctil LCD y la cámara NIR. En esta fase, se priorizó verificar el funcionamiento correcto de cada componente por separado.

Luego, se realizaron dos tipos de pruebas. En primer lugar, se llevaron a cabo pruebas en condiciones simuladas de somnolencia, utilizando la detección del estado de los ojos y la boca. Estas pruebas se realizaron inicialmente en un entorno de computadora para verificar el funcionamiento del sistema y luego se replicaron en el sistema embebido.

Posteriormente, después de evaluar el desempeño del sistema en condiciones simuladas, se procedió a probar en un entorno de conducción real. Esto implicó la instalación del sistema dentro de un vehículo en funcionamiento para evaluar su capacidad de detectar y prevenir la somnolencia del conductor en un contexto de conducción realista.

6.1. Pruebas de los componentes

Las pruebas de los componentes se realizaron en Nvidia Jetson Nano (sistema embebido), la pantalla táctil LCD y la cámara NIR. A continuación se expone las pruebas de cada componente.

6.1.1. Prueba del sistema embebido y pantalla LCD

Al adquirir el kit NVIDIA Jetson Nano, se incluye una fuente de alimentación de 5.3V a 4A, que se utiliza para realizar las pruebas en el sistema embebido. Al encender el sistema con esta fuente de alimentación, se puede observar el funcionamiento tanto del Jetson Nano como de la pantalla LCD de 7 pulgadas.

6.1.2. Prueba de la cámara NIR

Para evaluar el rendimiento de la cámara NIR, se utilizó el mismo código que se presenta en el Anexo 2. A continuación, se procedió a seleccionar la resolución y la velocidad de cuadros por segundo (fps) adecuadas para su uso en el sistema. Como se detalló previamente, la cámara tiene la función de enfocar el rostro del conductor y proporcionar la entrada de datos en forma de vídeo. Dado que se aplican redes neuronales convolucionales (CNN) para la detección, las imágenes se redimensionan a un tamaño de 64x64 píxeles.

En este contexto, se optó por seleccionar una resolución de 800x600p a 30fps, tal como se muestra en la Tabla 4.1. Esta elección se basa en el hecho de que el conjunto de datos NITYMED utilizado para entrenar las CNN se capturó a una velocidad de 25 fps. Se muestra el código en el Anexo 14 de la elección de resolución y fps.

6.2. Pruebas y resultados de los modelos CNNs en condición simulada

En esta sección, se presentan los resultados de las pruebas realizadas utilizando diferentes modelos de redes neuronales convolucionales (CNN). Se seleccionaron modelos específicos de cada arquitectura de CNN, incluyendo el séptimo modelo de InceptionV3, el primer modelo de VGG16 y el tercer modelo de ResNet50V2. De las CNN propuestas, se seleccionaron el noveno modelo de DD-AI y el sexto modelo de DD-AI-G. Estas selecciones se basaron

en criterios de exactitud (accuracy) y sensibilidad (recall), como se detalló en la sección 4.2.5.2.7.

Las pruebas se llevaron a cabo en tres escenarios distintos. En el primer escenario, se evaluó el sistema cuando el usuario no tenía ningún objeto en los ojos. En el segundo escenario, se probó cuando el usuario lleva puesto gafas normales, y en el tercer escenario, se evaluó el sistema cuando el usuario lleva puesto gafas de sol o lentes oscuros. Estos tres escenarios se aplicaron tanto en las pruebas realizadas en el computador como en el sistema embebido.

6.2.1. Pruebas en el entorno de la computadora

Las pruebas se llevaron a cabo en la misma computadora donde se diseñó el sistema, utilizando el código que se detalla en los Anexos 5 y 9. Durante estas pruebas, se evaluó la capacidad del sistema para detectar la somnolencia a través del análisis del estado de los ojos y la boca en los tres escenarios mencionados anteriormente. Estas pruebas se completaron en 33 segundos para los tres escenarios con luminosidad ambiental promedio de 184 lux, evaluando 1000 imágenes de cada escenario, con los cinco modelos de las CNNs para cada escenario. En la Figura 6.1 se puede observar un ejemplo de la detección de somnolencia visual y bostezo en lo tres escenarios correspondiente al modelo DD-AI-G.

Figura 6.1: Pruebas en el entorno de la computadora.

(a) Sin objeto en los ojos

(b) Con lentes normales

(c) Con gafas de sol (lunas negras)

6.2.2. Resultados en el entorno de la computadora

En el entorno de la computadora, los resultados de las pruebas demuestran que la red CNN propuesta DD-AI-G lidera con el porcentaje de aciertos en un escenario simulado de somnolencia, alcanzando un 96.3 %. Le sigue de cerca la red DD-AI con un 94.8 % de aciertos. ResNet50V2 logra un 91.4 % de aciertos, mientras que VGG16 y InceptionV3 obtienen 91.1 % y 88.6 % de aciertos respectivamente. Se midió el porcentaje de aciertos en la detección de somnolencia visual (ojos cerrados). Los escenarios se definieron como sigue: Escenario 1, "Sin objeto en los ojos"; Escenario 2, "Con lentes normales"; y Escenario 3, "Con gafas de sol (lunas negras)". Cada escenario constó de 1000 imágenes, de las cuales, respecto a imágenes de la somnolencia visual (ojos cerrados) se obtuvieron 642 correspondientes al escenario 1, 723 al escenario 2 y 548 al escenario 3. La evaluación se realizó de manera manual, comparando los resultados de los modelos con la detección humana de somnolencia visual (Tabla 6.1).

Tabla 6.1: Resultado de acierto en entorno de la computadora.

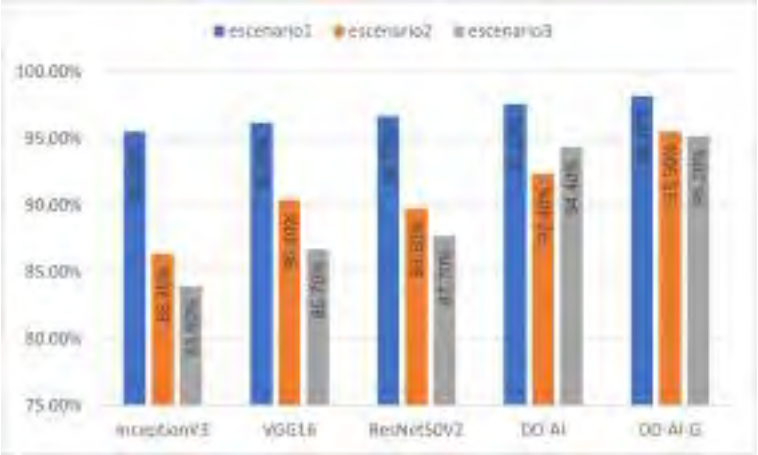
Escenarios	Frames usados	InceptionV3	VGG16	ResNet50V2	DD-AI	DD-AI-G
		Acierto %	Acierto %	Acierto %	Acierto %	Acierto %
Escenario 1	642	95.5	96.2	96.7	97.6	98.2
Escenario 2	723	86.4	90.4	89.8	92.4	95.5
Escenario 3	548	83.9	86.7	87.7	94.4	95.2
Promedio de acierto (%)		88.6	91.1	91.4	94.8	96.3

Gráficamente, en la Figura 6.2, se presenta de manera gráfica el porcentaje de aciertos de las cinco redes en los tres escenarios evaluados. En el Escenario 1, la red propuesta DD-AI-G obtuvo 98.2 % de aciertos en las 642 imágenes relacionadas con la somnolencia visual. Falló en solo 29 imágenes, representando un 1.8 % de error. La red DD-AI le siguió de cerca con un 97.6 % de aciertos. Seguido de ResNet50V2, VGG16 e InceptionV3 con 96.7 %, 96.2 % y 95.5 % de aciertos respectivamente.

En el Escenario 2, nuevamente DD-AI-G demostró un alto rendimiento, alcanzando un 95.5 % de aciertos de las 723 imágenes evaluadas. La red DD-AI obtuvo un 92.4 % de aciertos. En este escenario, VGG16 alcanzó un 90.4 % de aciertos, seguido de ResNet50V2 e InceptionV3 con 89.8 % y 86.4 % respectivamente.

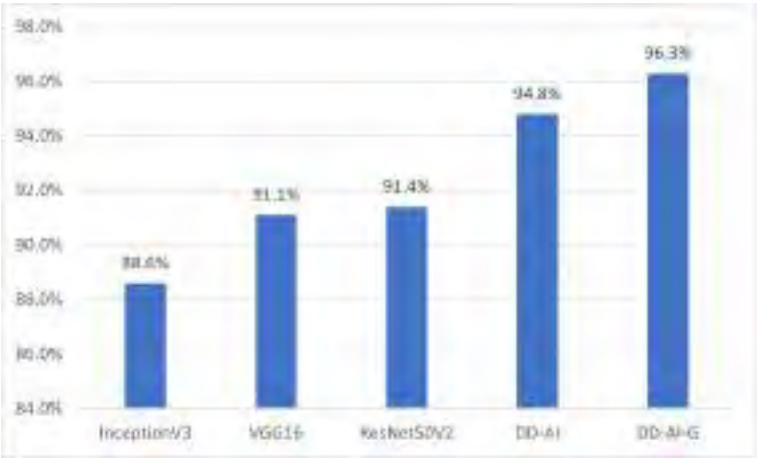
En cuanto al Escenario 3, DD-AI-G presenta un 95.2 % de aciertos de las 548 imágenes de somnolencia visual. La red DD-AI le sigue con un 94.4 % de aciertos. ResNet50V2 mostró un 87.7 % de aciertos, mientras que VGG16 y InceptionV3 obtuvieron un 86.7 % y 83.9 % respectivamente.

Figura 6.2: Porcentaje de acierto de las cinco CNN en los tres escenarios - computadora.



Gráficamente, la Figura 6.3 muestra los porcentajes acertados promedios alcanzados por las cinco redes CNN en los tres escenarios en el entorno de la computadora.

Figura 6.3: Porcentaje promedio de las cinco CNN en los tres escenarios - computadora.



6.2.3. Pruebas en el sistema embebido

Asimismo, se llevaron a cabo pruebas en el sistema embebido en un entorno simulado de somnolencia. En las pruebas iniciales, se observó que el sistema no detectaba adecuadamente los casos de somnolencia visual y bostezo en el tiempo especificado debido a discrepancias en los fps del sistema embebido. Mientras que en el computador se mantenían entre 30 y 40 fps,como se observa en la Figura 6.1; en el sistema embebido, los fps varían entre 8 y 14 fps.

Para abordar este problema, se realizó una corrección basada en los valores de fps en lugar de usar un valor de tiempo constante para cada caso de somnolencia. Se establecieron relaciones entre los umbrales de tiempo y los fps proporcionados por el sistema embebido, que se detallan a continuación:

1. **Para somnolencia visual:** Como se mencionó previamente, para detectar la somnolencia visual, se requiere que los ojos estén cerrados durante más de 300 ms. Por lo tanto, se estableció una relación entre este umbral de tiempo y la tasa de cuadros por segundo (fps), que se expresa de la siguiente manera:

$$Umbral_{visual} = 0{,}3 * fps_{embebido}$$

2. **Para bostezo:** Del mismo modo, para detectar el bostezo somnoliento, se requiere que el evento dure más de 5 segundos. Por lo tanto, se estableció una relación entre este umbral de tiempo y la tasa de cuadros por segundo (fps) de la siguiente manera:

$$Umbral_{bostezo} = 5 * fps_{embebido}$$

Después de realizar las correcciones necesarias, se llevaron a cabo las pruebas en el sistema embebido NVIDIA Jetson Nano. Igualmente, para cada escenario se trabajo con 33 segundos, obteniendo 1000 imágenes por cada escenario. El código empleado en el sistema embebido se encuentra en Anexo 15.

6.2.4. Resultados en el sistema embebido

Los resultados en el sistema embebido, son de acuerdo a las 1000 imágenes de cada escenario, donde, respecto a imágenes de la somnolencia visual (ojos cerrados) se obtuvieron 584 correspondientes al escenario 1, 612 al escenario 2 y 592 al escenario 3. Los resultados de

estas pruebas se presentan en la Tabla 6.2, la cual muestra el desempeño de las cinco redes CNN en un entorno simulado de somnolencia dentro del sistema embebido.

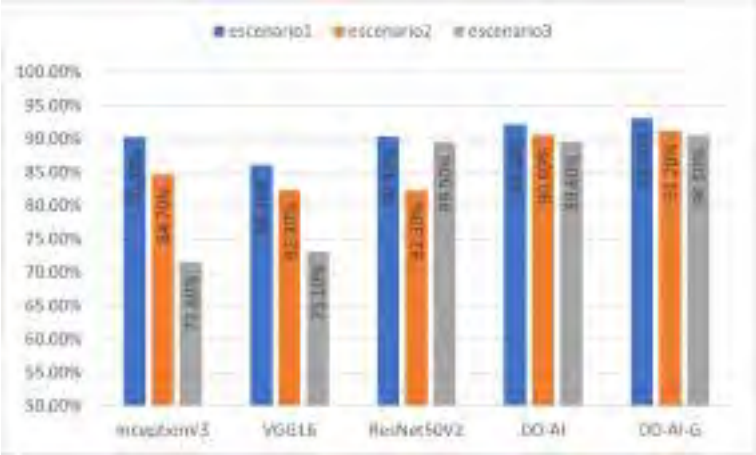
De la Tabla 6.2, la red propuesta DD-AI-G, presenta el mayor porcentaje de acierto con 91.6 %, seguido de la otra red propuesta DD-AI con 90.8 %, ResNet50V2 obtuvo un 87.4 % de acierto, InceptionV3 un 82.2 % y finalmente VGG16 un 80.5 %. Por lo tanto, en estos primeros resultados, la red propuesta DD-AI tiene el mejor desempeño para la detección de somnolencia en los tres escenario en términos generales.

Tabla 6.2: Resultado de acierto en el sistema embebido.

Escenarios	Frames usados	InceptionV3	VGG16	ResNet50V2	DD-AI	DD-AI-G
		Acierto %	Acierto %	Acierto %	Acierto %	Acierto %
Escenario 1	584	90.3	86.1	90.4	92.2	93.1
Escenario 2	612	84.7	82.3	82.3	90.6	91.2
Escenario 3	592	71.6	73.1	89.5	89.6	90.5
Promedio de acierto (%)		82.2	80.5	87.4	90.8	91.6

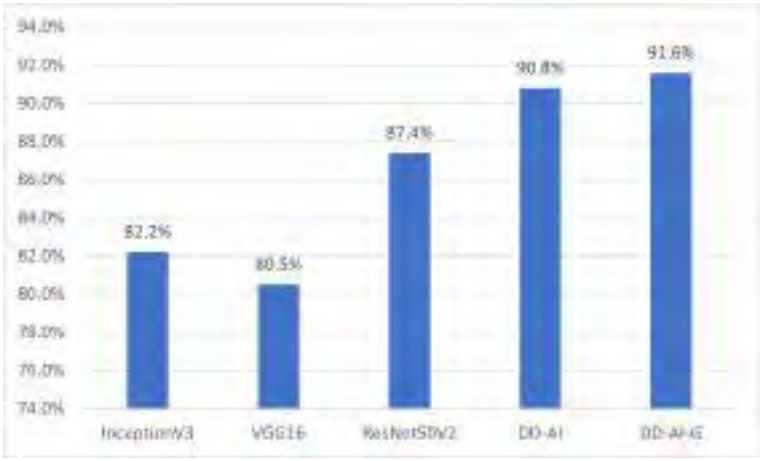
Se presenta de manera gráfica el porcentaje de aciertos de las cinco redes en los tres escenarios evaluados en la Figura 6.4. En el Escenario 1, la red propuesta DD-AI-G obtuvo 93.1 % de aciertos en las 584 imágenes relacionadas con la somnolencia visual. La red DD-AI le siguió de cerca con un 92.2 % de aciertos. Seguido de ResNet50V2 con 90.4 % y muy de cerca con 90.3 % la red InceptionV3, finalmente VGG16 con 86.1 % . En el Escenario 2, nuevamente DD-AI-G demostró un alto rendimiento, alcanzando un 91.2 % de aciertos de las 612 imágenes evaluadas. La red DD-AI obtuvo un 90.6 % de aciertos. En este escenario, InceptionV3 alcanzó un 84.7 % de aciertos, seguido de ResNet50V2 y VGG16 con 82.3 % de aciertos ambos respectivamente. En cuanto al Escenario 3, DD-AI-G presenta un 90.5 % de aciertos de las 592 imágenes de somnolencia visual. La red DD-AI le sigue con un 89.6 % de aciertos y ResNet50V2 con un 89.5 % de aciertos, mientras que VGG16 y InceptionV3 obtuvieron un 73.1 % y 71.6 % respectivamente.

Figura 6.4: Porcentaje de acierto de las cinco CNN en los tres escenarios - jetson nano.



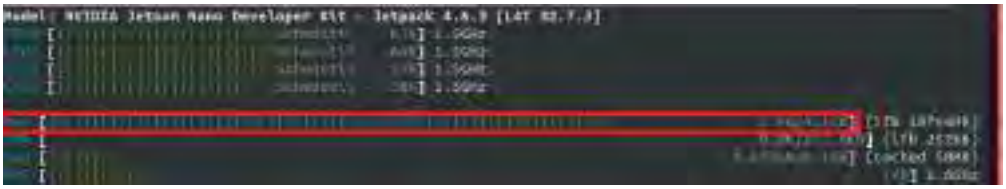
Gráficamente, la Figura 6.5 muestra los porcentajes acertados promedios alcanzados por las cinco redes CNN en los tres escenarios en el entorno del sistema embebido.

Figura 6.5: Porcentaje promedio de las cinco CNN en los tres escenarios - jetson nano.

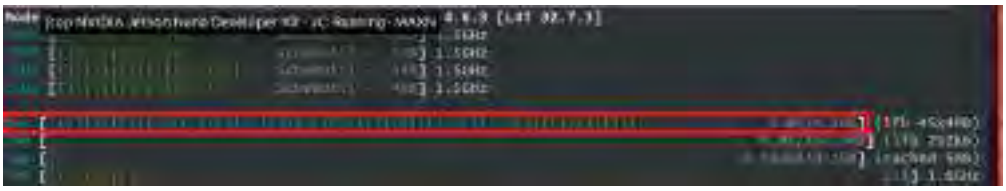


En el sistema embebido, se evaluó el consumo de memoria RAM y GPU de las cinco redes CNN. Las pruebas se realizaron ejecutando solo el código en Python, sin otros programas abiertos para garantizar mediciones precisas. Los resultados indican que la red basada en InceptionV3 consume alrededor de 2.9 GB de memoria RAM, la red VGG16 utiliza cerca de 3 GB, y la red ResNet50V2 requiere 3.1 GB. Por otro lado, las redes propuestas DD-AI y DD-AI-G consumen aproximadamente 2.8 GB cada una. Esto se puede observar en la Figura 6.6.

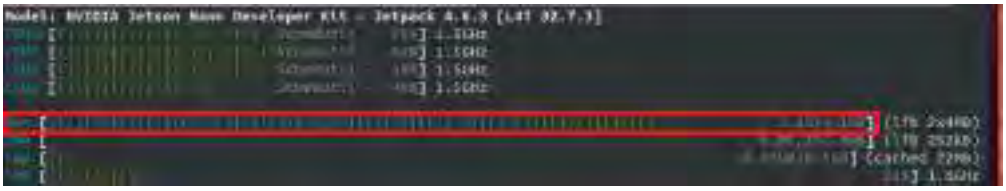
Figura 6.6: Consumo de memoria RAM de las cinco CNNs.



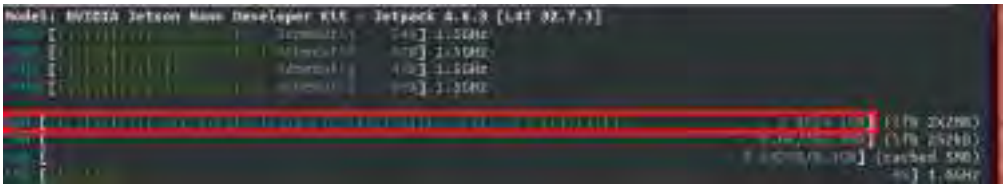
(a) Red InceptionV3



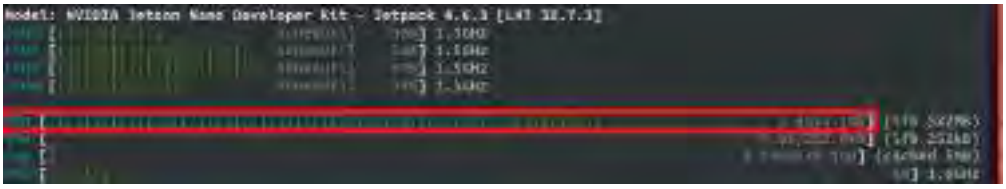
(b) Red VGG16



(c) Red ResNet50V2



(d) Red DD-AI



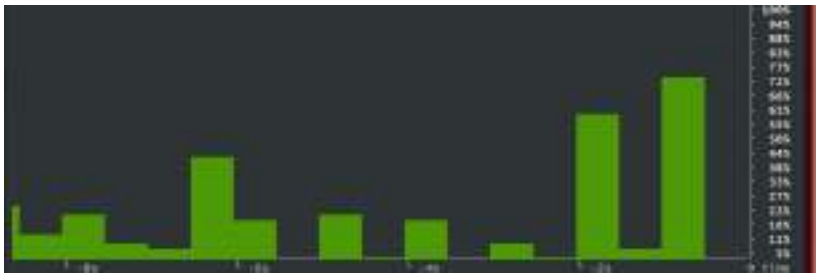
(e) Red DD-AI-G

El consumo de la GPU de cada red CNN se muestra en la Figura 6.7. Donde, la red basada en InceptionV3 en promedio hace uso de 70 % de la GPU (Figura 6.7a), la red basada en VGG16 consume en promedio el 55 % de la GPU (Figura 6.7b) y la red basada en ResNet50V2 hace uso en promedio un 60 % de la GPU (Figura 6.7c).

Las redes propuestas DD-AI y DD-AI-G, en promedio hacen uso del 25 % de la GPU (Figura 6.7d y 6.7e). Logrando optimizar el consumo de la GPU del sistema embebido mediante estas dos propuestas de arquitectura.

Figura 6.7: Consumo de GPU de las cinco CNNs.

(a) Red InceptionV3



(b) Red VGG16

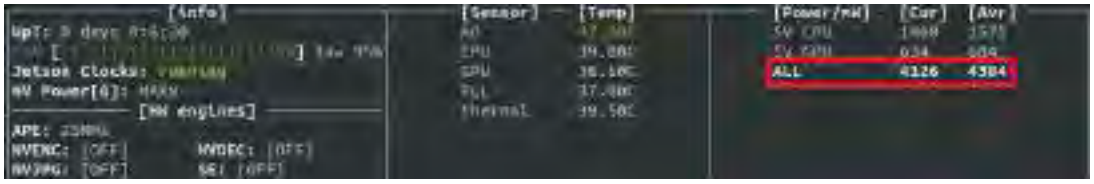
(c) Red ResNet50V2

(d) Red DD-AI

(e) Red DD-AI-G

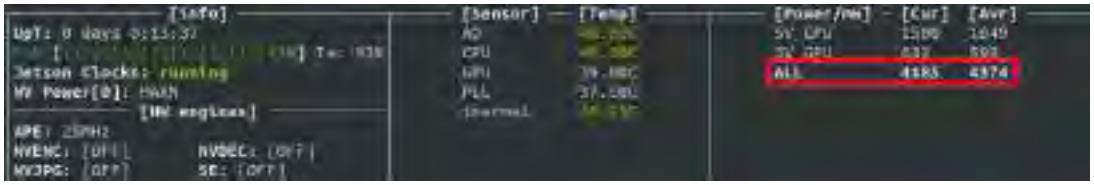
Se analizó el consumo máximo de vatios de las 5 redes CNN con todos los componentes en funcionamiento. Con la red InceptionV3 presenta un consumo promedio de 4.384 W (Figura 6.8a), con VGG16 consume en promedio 4.42 W (Figura 6.8b), con ResNet50V2 tiene un consumo promedio de 4.374 W (Figura 6.8c), con la red propuesta DD-AI consume en promedio 4.362 W (Figura 6.8d) y con la red propuesta DD-AI-G tiene un consumo promedio de 4.267 W (Figura 6.8e). Todas las evaluaciones se realizaron con el sistema funcionando a más del 90 % de capacidad.

Figura 6.8: Consumo de potencia del sistema con las cinco CNNs.



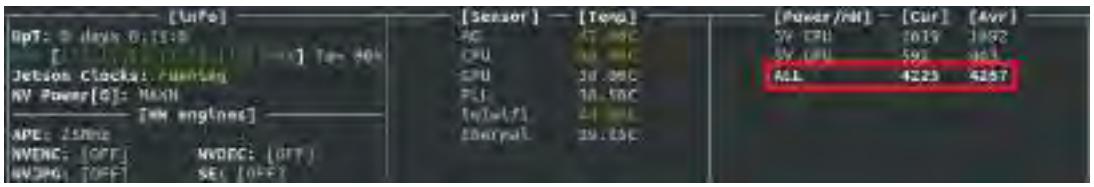
(a) Red InceptionV3

(b) Red VGG16



(c) Red ResNet50V2

(d) Red DD-AI



(e) Red DD-AI-G

6.3. Elección de la red CNN

Teniendo todos los resultados en un ambiente simulado de somnolencia, se procedió a seleccionar la red CNN que servirá para las pruebas en un ambiente de conducción real.

Según los resultados vistos en la sección 6.2.2, la red que mejor desempeño tiene, es la red propuesta DD-AI-G en el entorno de la computadora. Mientras que, en los resultados presentados en la sección 6.2.4, también la red propuesta DD-AI-G tiene el mejor desempeño en el sistema embebido (Jetson Nano), esto considerando la optimización de la arquitectura propuesta DD-AI-G en el consumo de la memoria RAM, GPU y potencia a comparación de las redes basadas en transfer learning. Por lo tanto, la red a usar para las pruebas en un ambiente de conducción real es la "DD-AI-G".

6.4. Pruebas del sistema en un entorno real

Para llevar a cabo las pruebas en un entorno de conducción real, fue necesario instalar el sistema detector de somnolencia en el interior del vehículo.

Posteriormente, las pruebas se llevaron a cabo con dos personas durante 5 días en tres momentos diferentes: a las 7 de la mañana (diurno), a las 3 de la tarde (tarde) y a las 8 de la noche (nocturno), según las estadísticas de la PNP (2020). Estos horarios fueron seleccionados debido a que son momentos en los que suelen ocurrir más accidentes relacionados con la somnolencia.

6.4.1. Instalación del sistema en el vehículo

Durante la instalación del sistema en el vehículo, se realizó una prueba inicial para verificar el funcionamiento de la fuente de alimentación que se conecta a la salida de 12V del vehículo. Se confirmó que la fuente proporcionaba un voltaje de 5.7V, como se muestra en la Figura 6.9.

Figura 6.9: Comprobación del suministro energético.

Luego, se procedió a montar el sistema en el tablero del vehículo. La cámara fue posicionada detrás del volante para tener una vista óptima del rostro del conductor. El sistema embebido y la pantalla se colocaron a la altura del auto-radio, como se muestra en la Figura 6.10. La instalación de los componentes en el tablero se realizó utilizando cinta de doble contacto transparente para evitar dañar el vehículo.

Figura 6.10: Sistema instalado en el vehículo.

6.4.2. Pruebas de funcionamiento del sistema en el vehículo

Como se mencionó previamente, las pruebas se llevaron a cabo en tres momentos diferentes en el día, la ruta utilizada para las pruebas se muestra en la Figura 6.11. La ruta se inició cerca del Hotel de la Derrama Magisterial, transitando detrás del Penal de Quencoro, llegando hasta la Universidad Andina del Cusco. Posteriormente, se realizó el retorno por la Avenida de la Cultura, entrando por el Mercado Vinocanchon, se pasó por la Plaza de San Jerónimo y la ruta concluyó en el domicilio del investigador.

Figura 6.11: Ruta de las pruebas.

6.4.2.1. Pruebas diurnas

Las pruebas diurnas se llevaron a cabo de 7:00 a.m. a 7:30 a.m. en la ruta previamente mencionada, ambas personas usaron gafas normales durante todas las pruebas. En la Figura 6.12 se puede apreciar la condición normal, la detección de bostezo y la detección de somnolencia visual.

Figura 6.12: Pruebas diurnas del sistema.

(a) Condición normal.

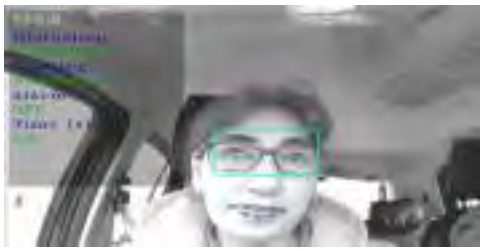
(b) Detección de bostezo.

(c) Detección de somnolencia visual.

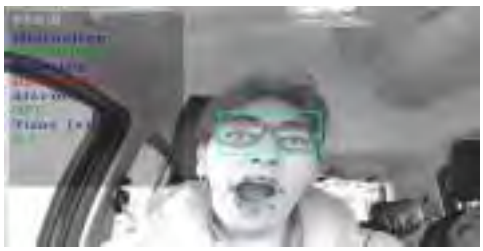
6.4.2.2. Pruebas en la tarde

Las pruebas en la tarde se llevaron a cabo de 2:00 p.m. a 3:30 p.m. en la ruta previamente mencionada, ambas personas usaron gafas normales y una de ellas sin objetos en los ojos durante todas las pruebas. En la Figura 6.12 se puede apreciar la condición normal, la detección de bostezo y la detección de somnolencia visual.

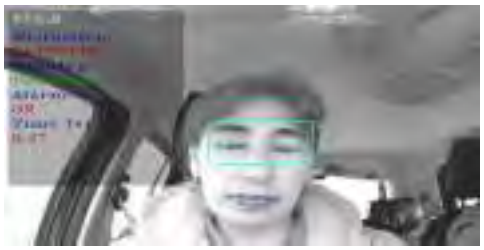
Figura 6.13: Pruebas en la tarde del sistema.



(a) Condición normal.



(b) Detección de bostezo.

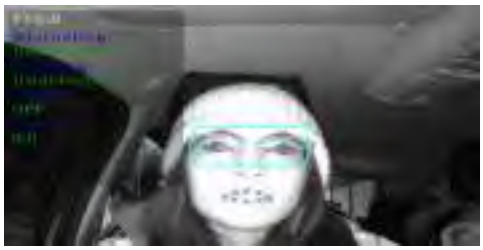


(c) Detección de somnolencia visual.

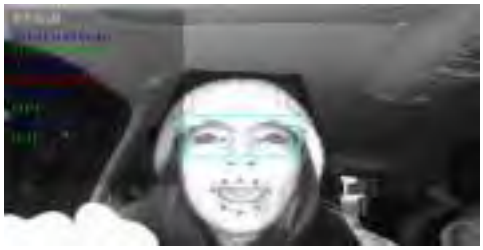
6.4.2.3. Pruebas nocturnas

Las pruebas nocturnas se llevaron a cabo de 8:00 p.m. a 8:30 p.m. en la ruta mencionada, ambas personas usaron gafas normales durante todas las pruebas. En la Figura 6.14 se puede apreciar la conducción normal, la detección de bostezo y la detección de somnolencia visual.

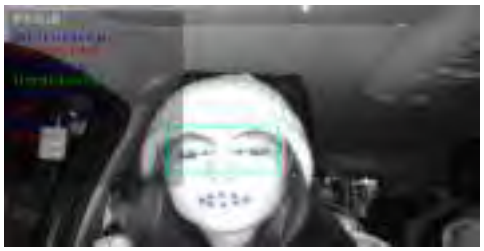
Figura 6.14: Pruebas nocturnas del sistema.



(a) Condición normal.



(b) Detección de bostezo.



(c) Detección de somnolencia visual.

6.5. Detección de fallos del sistema

Después de llevar a cabo las pruebas del sistema en el vehículo, se identificaron casos de falsos positivos (5.36 %) y falsos negativos (8.56 %), especialmente en las pruebas realizadas por la tarde. Estos errores se deben a varios factores, como los cambios en la iluminación y las rápidas variaciones en la posición de la cabeza del conductor. Como se explicó anteriormente, es crucial minimizar los falsos negativos para mejorar la precisión en la detección de la somnolencia. Asimismo, se busca reducir los falsos positivos para evitar que el sistema genere alertas innecesarias. En la Figura 6.15 se observa los falsos positivos y falsos negativos.

Figura 6.15: Casos de falsos positivos y falsos negativos.

(a) Falso positivo. (b) Falso negativo.

6.6. Resultados de las pruebas del sistema en un entorno real

Los resultados de las pruebas del sistema en un entorno real de conducción se obtuvieron a partir de los fotogramas (imágenes) extraídos de los vídeos grabados durante la conducción de las dos voluntarias. El tiempo promedio de manejo en la ruta fue de 15 minutos. De cada vídeo se extrajeron aproximadamente 9000 fotogramas, de los cuales se evaluaron 3000 fotogramas en tres condiciones diferentes: sin objetos en los ojos, con lentes normales y con gafas de sol. A continuación, se presenta los resultados de cada voluntaria en los tres horarios diferentes durante los 5 días de pruebas. Los resultados están de acuerdo a Tabla 2.3 matriz de confusión. Para medir el desempeño general del sistema, se usaron las ecuaciones 2.12 y 2.10 de exactitud (accuracy) y sensibilidad (recall) respectivamente.

6.6.1. Resultados del sistema en voluntaria 1 (señora conductora)

Los resultados obtenidos en los tres horarios a lo largo de los cinco días se presentan en las siguientes Tablas: Tabla 6.3 (horario diurno), Tabla 6.4 (horario tarde) y Tabla 6.5 (horario nocturno). Donde, el sistema tuvo el mejor rendimiento en el horario nocturno con 93.7 % de exactitud, seguido del horario diurno con 92.9 % de exactitud y en el horario de la tarde obtuvo una exactitud de 90.1 %.

Tabla 6.3: Resultado de las pruebas diurnas - voluntaria 1.

Día	Número de frames	Con somnolencia visual		Sin somnolencia visual		Accuracy	Recall
		TP	FN	TN	FP		
1	3000	452	103	2313	132	92.2 %	81.4 %
2	3000	684	95	2068	156	91.7 %	88.1 %
3	3000	384	21	2269	326	88.4 %	94.8 %
4	3000	573	14	2299	114	95.7 %	97.6 %
5	3000	786	25	2105	84	96.4 %	96.9 %
Promedio						92.9 %	91.8 %

Tabla 6.4: Resultado de las pruebas en la tarde - voluntaria 1.

Día	Número de frames	Con somnolencia visual		Sin somnolencia visual		Accuracy	Recall
		TP	FN	TN	FP		
1	3000	752	106	1908	234	88.7 %	87.6 %
2	3000	524	94	2230	152	91.8 %	84.8 %
3	3000	648	156	1938	258	86.2 %	80.6 %
4	3000	852	103	1870	175	90.7 %	89.2 %
5	3000	697	87	2100	116	93.2 %	88.9 %
Promedio						90.1 %	86.2 %

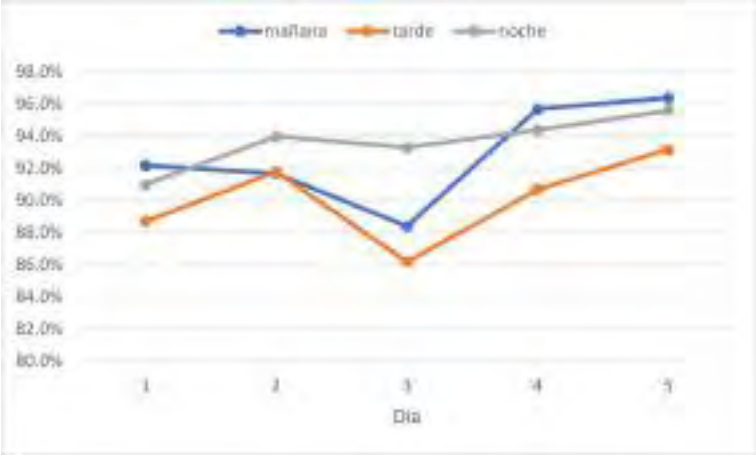
Tabla 6.5: Resultado de las pruebas nocturnas - voluntaria 1.

Día	Número de frames	Con somnolencia visual		Sin somnolencia visual		Accuracy	Recall
		TP	FN	TN	FP		
1	3000	1156	156	1573	115	91.0 %	88.1 %
2	3000	1084	92	1737	87	94.0 %	92.2 %
3	3000	956	108	1843	93	93.3 %	89.8 %
4	3000	914	59	1919	108	94.4 %	93.9 %
5	3000	1261	81	1606	52	95.6 %	94.0 %
Promedio						93.7 %	91.6 %

La tendencia de exactitud (accuracy) de las pruebas durante los cinco días se muestra en la Figura 6.16. Se puede observar que el sistema presenta un mejor rendimiento por la noche, ya que la cámara NIR funciona de manera más efectiva en ese horario.

En segundo lugar, se encuentra el horario de la mañana, donde el sol no parece influir significativamente en el rostro del conductor. Por último, en el horario de la tarde, se observa un rendimiento inferior del sistema debido a que los rayos del sol inciden directamente en el rostro del conductor, especialmente si este lleva gafas, lo que provoca reflejos que en ocasiones ocultan ambos ojos, generando así falsas alarma (falsos positivos) y alarmas no detectadas (falsos negativos) durante ese horario.

Figura 6.16: Tendencia de Accuracy del sistema - voluntaria 1.



6.6.2. Resultados del sistema en voluntaria 2 (señorita conductora)

De igual menra, los resultados obtenidos en los tres horarios a lo largo de los cinco días se presentan en las siguientes Tablas: Tabla 6.6 (horario diurno), Tabla 6.7 (horario tarde) y Tabla 6.8 (horario nocturno). Donde, el sistema tuvo el mejor rendimiento en el horario nocturno con 92.2 % de exactitud, seguido del horario diurno con 91.6 % de exactitud y en el horario de la tarde obtuvo una exactitud de 88.6 %.

Tabla 6.6: Resultado de las pruebas diurnas - voluntaria 2.

Día	Número de frames	Con somnolencia visual		Sin somnolencia visual		Accuracy	Recall
		TP	FN	TN	FP		
1	3000	523	98	2263	116	92.9 %	84.2 %
2	3000	486	156	2179	179	88.8 %	75.7 %
3	3000	465	108	2171	256	87.9 %	81.2 %
4	3000	635	56	2213	96	94.9 %	91.9 %
5	3000	671	73	2140	116	93.7 %	90.2 %
Promedio						91.6 %	84.6 %

Tabla 6.7: Resultado de las pruebas en la tarde - voluntaria 2.

Día	Número de frames	Con somnolencia visual		Sin somnolencia visual		Accuracy	Recall
		TP	FN	TN	FP		
1	3000	667	232	1756	345	80.8 %	74.2 %
2	3000	549	131	2105	215	88.5 %	80.7 %
3	3000	756	143	1880	221	87.9 %	84.1 %
4	3000	718	96	2063	123	92.7 %	88.2 %
5	3000	627	107	2172	94	93.3 %	85.4 %
Promedio						88.6 %	82.5 %

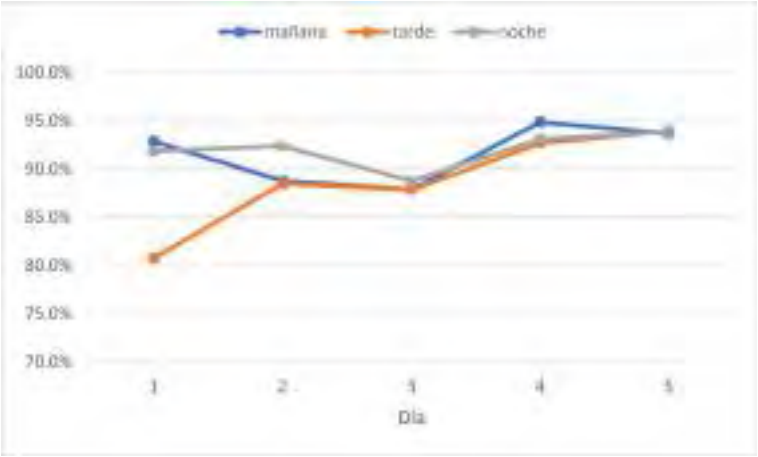
Tabla 6.8: Resultado de las pruebas nocturnas - voluntaria 2.

Día	Número de frames	Con somnolencia visual		Sin somnolencia visual		Accuracy	Recall
		TP	FN	TN	FP		
1	3000	927	146	1831	96	91.9 %	86.4 %
2	3000	987	113	1786	114	92.4 %	89.7 %
3	3000	1238	152	1427	183	88.8 %	89.1 %
4	3000	1158	98	1636	108	93.1 %	92.2 %
5	3000	1086	87	1730	97	93.9 %	92.6 %
Promedio						92.0 %	90.0 %

La tendencia de exactitud (accuracy) de las pruebas durante los cinco días se muestra en la Figura 6.17. En este caso, se observa que el sistema tiene un comportamiento diferente en los primeros tres días en comparación con los dos últimos días, donde la tendencia del accuracy es similar en los tres horarios. Según las estadísticas obtenidas, el sistema presenta un mejor rendimiento por la noche, en algunos casos se presentaba una mayor luminosidad

en el rostro de la voluntaria, causada por la cercanía hacia la cámara NIR. En segundo lugar, se encuentra el horario de la mañana, en el cual el sistema no presenta dificultades significativas para detectar la somnolencia mediante el estado de los ojos y la boca. Por último, en el horario de la tarde, se observa un rendimiento inferior del sistema, ya que la excesiva luz que incide en el rostro de la voluntaria dificulta la detección y genera un mayor número de falsas alarmas (falsos positivos).

Figura 6.17: Tendencia de Accuracy del sistema - voluntaria 2.



6.7. Resultados generales del sistema

Para determinar el resultado general del sistema, se calculó un promedio de exactitud (accuracy) y sensibilidad (recall) correspondiente a los tres horarios para cada voluntaria. Luego, se obtuvo una exactitud y sensibilidad general tomando en cuenta ambos promedios. En la detección de bostezo mediante mouth aspect ratio o aspecto radial bucal (MAR), se logró un 100 % de exactitud (accuracy), ya que esta detección se basa en puntos de la boca, lo que permite que el sistema detecte la apertura en los tres horarios sin dificultad.

Los promedios de exactitud (accuracy) y sensibilidad (recall) para cada voluntaria se observa en las Tablas 6.9 y 6.10

Tabla 6.9: Promedio Accuracy y Recall - voluntaria 1.

	Accuracy	Recall
Promedio 1	92.23 %	86.86 %

Tabla 6.10: Promedio Accuracy y Recall - voluntaria 2.

	Accuracy	Recall
Promedio 2	90.73 %	85.70 %

El promedio general del sistema detector de somnolencia se presenta en la Tabla 6.11. Donde se obtuvo un 91.48 % de exactitud (accuracy) promedio y 86.28 % de sensibilidad (recall) promedio.

Tabla 6.11: Promedio General del sistema en Accuracy y Recall

	Accuracy	Recall
Promedio General	91.48 %	86.28 %

Discusión

Este análisis aborda la relevancia y eficacia de diferentes enfoques para la detección de somnolencia en conductores, evaluando tanto la investigación principal como tres investigaciones adicionales. Cada estudio contribuye de manera significativa a la comprensión y solución de esta problemática crítica.

En la presente investigación, la implementación realizada en un hardware avanzado, como NVIDIA Jetson Nano, junto con una cámara de infrarrojo cercano (NIR), demuestra un enfoque técnico y tecnológico de vanguardia para prevenir siniestros viales por somnolencia del conductor. La combinación de múltiples CNN y arquitecturas propuestas, especialmente la red DD-AI-G, evidencia una alta exactitud (accuracy) del 91.48 % y una sólida tasa de sensibilidad (recall) del 86.28 % en entornos de conducción reales. La elección del hardware resalta la importancia de recursos de procesamiento para sistemas avanzados de detección.

En la tesis de Alba Neppas (2020), se centra en la accesibilidad y simplicidad al utilizar hardware más básico, como Raspberry Pi 3B+ y una cámara RGB. Aunque logra una precisión aceptable del 88.25 % en ambientes controlados y 87 % en situaciones reales diurnas, la limitación de operar solo en modo diurno sugiere la necesidad de mejoras para enfrentar condiciones nocturnas.

El estudio de Custodio Effio and Tarrillo Nuntón (2021), destaca el uso de tecnologías como Python, Dlib, Haar Cascade y la sugerencia de implementación en un dispositivo portátil utilizando NVIDIA Jetson Nano. Aunque alcanza una eficacia del 97.78 % en condiciones diurnas, las limitaciones en condiciones nocturnas resaltan la importancia de recursos de procesamiento robustos para un rendimiento continuo.

En el trabajo hecho por Montiel et al. (2019), se centra en el desarrollo de un algoritmo de detección de somnolencia utilizando descriptores HOG y clasificadores en cascada. La investigación destaca una precisión del 90.65 % en la detección facial y utiliza medidas como EAR y MAR. Además, eligieron Raspberry Pi 3B para implementar el sistema, logrando un tiempo de procesamiento de 0.179s o 179ms.

El estudio de Anber et al. (2022), se centra en prevenir accidentes al diagnosticar y detectar a tiempo signos de fatiga. Utiliza dos enfoques: aprendizaje por transferencia con ajuste fino de AlexNet y extracción de características mediante NMF y clasificación con SVM. Ambos modelos alcanzan altas precisiones (95.7 % y 99.65 %).

En la tesis de Arroyo Rodriguez (2021), se centra en rasgos específicos de somnolencia, como bostezo, pestañeo y cabeceo, utilizando Haar Cascade para la detección inicial. Dando como resultado de detección de bostezo del 96 % y pestañeo del 98 % mostrando una eficacia considerable en la identificación de comportamientos somnolientos, con un tiempo de procesamiento de 0.3s implementado en Raspberry Pi 3 B+.

La investigación realizada por Krishna et al. (2022), se orienta hacia arquitecturas modernas y técnicas de aprendizaje profundo, específicamente el uso de YoloV5. Con resultados impresionantes, alcanzando un 95.5 % de precisión en condiciones prácticas, sugiere una aplicabilidad prometedora en sistemas de transporte inteligentes. El enfoque en transformadores de visión y aprendizaje profundo respalda la eficacia del marco propuesto.

Por lo que, cada enfoque presenta sus propias fortalezas y limitaciones. La elección entre ellos dependerá de la disponibilidad de recursos, la complejidad deseada y las condiciones específicas de implementación. La presente investigación destaca por su rendimiento avanzado, pero la accesibilidad de la primera investigación y el enfoque moderno de la tercera investigación sugieren opciones viables según las necesidades y restricciones del entorno de aplicación. En general, estos estudios colectivos enriquecen el panorama de soluciones para la detección de somnolencia, proporcionando diversas perspectivas y contribuciones valiosas al campo.

Conclusiones

1. Se ha desarrollado y puesto en funcionamiento un sistema detector de somnolencia aprovechando las capacidades de la visión computacional y las redes neuronales convolucionales para detectar signos de somnolencia en tiempo real. La implementación en un sistema embebido, demostró un rendimiento eficaz con promedio general del sistema del 91.48 % y 86.28 % en exactitud (accuracy) y sensibilidad (recall) respectivamente y eficiente con 34 ms de respuesta, consumo promedio de 4.267 W y uso de memorias RAM y GPU de 2.8 GB y 25 % respectivamente. Logrando optimizar los recursos del sistema embebido.

2. Se implementaron diferentes arquitecturas que utilizan redes neuronales convolucionales para la detección de somnolencia. Estas arquitecturas incluyeron tres modelos entrenados mediante transfer learning, específicamente InceptionV3, VGG16 y ResNet50V2, además se propone dos arquitecturas denominadas DD-AI y DD-AI-G.

3. Se logró validar de manera efectiva los resultados obtenidos para evaluar el rendimiento de los algoritmos CNN implementados en la detección de somnolencia. Los resultados de esta validación se basaron en la matriz de confusión durante el proceso de prueba con un conjunto de datos, lo que arrojó un alto nivel de exactitud (accuracy) para cada una de las redes CNN evaluadas. En particular, se obtuvieron tasas de exactitud del 98.95 % para la red basada en InceptionV3, del 98.33 % para la red basada en VGG16, del 99.49 % para la red basada en ResNet50V2, del 99.88 %

para la primera red propuesta DD-AI y del 99.91 % para la segunda red propuesta DD-AI-G. Estos resultados respaldan la efectividad y confiabilidad del sistema en la detección de somnolencia.

4. Se llevó a cabo un análisis de plataformas de hardware para la implementación del sistema embebido en la unidad vehicular. Durante este proceso, se evaluaron dos opciones principales de hardware, el NVIDIA Jetson Nano y el Rasberry Pi, ambos con 4GB de memoria. Estos dispositivos fueron evaluados en términos de potencia de procesamiento, capacidad de memoria, consumo de energía, compatibilidad con bibliotecas y frameworks, así como su rendimiento en la ejecución de redes neuronales convolucionales (CNN). Se tomó la decisión de seleccionar el NVIDIA Jetson Nano como la plataforma para la implementación del sistema.

5. En las pruebas realizadas, se evaluó el funcionamiento del sistema detector de somnolencia en tiempo real en dos contextos diferentes: condiciones simuladas de somnolencia y un entorno real de conducción. Para ambas pruebas, se seleccionó la red propuesta DD-AI-G debido a su alto rendimiento en la detección de somnolencia visual. En el entorno de conducción real utilizando el sistema embebido, se lograron resultados prometedores con un nivel de exactitud (accuracy) general del 91.48 % y una sensibilidad (recall) del 86.28 %. A pesar de algunos falsos positivos y falsos negativos, estos resultados destacan la efectividad del sistema para detectar signos de somnolencia en condiciones no controladas, proporcionando una base sólida para futuras mejoras y adaptaciones del sistema.

Recomendaciones

- Dada la continua evolución del Deep Learning, se sugiere explorar y emplear diversas arquitecturas de redes neuronales convolucionales adicionales para una evaluación comparativa de su desempeño en relación con las utilizadas y propuestas en esta investigación. Esto permitirá una comprensión más completa de las capacidades y limitaciones de diferentes arquitecturas en la detección de somnolencia, lo que puede conducir a mejoras y avances en la precisión y eficiencia del sistema.

- Dada la constante evolución tecnológica, es aconsejable considerar la posibilidad de emplear otros sistemas embebidos que cumplan con los requisitos de cálculo necesarios para la inferencia de redes neuronales convolucionales. Esto permitiría evaluar su rendimiento en comparación con el sistema embebido utilizado en esta investigación, potencialmente reduciendo costos y optimizando el espacio en la instalación del vehículo. Esta exploración puede contribuir a una implementación más eficiente y rentable del sistema detector de somnolencia.

- Con el fin de reducir tanto las falsas alarmas (falsos positivos) como las alarmas no detectadas (falsos negativos), se sugiere considerar la utilización de una cámara NIR con tecnología WDR (Wide Dynamic Range) para compensar los problemas con la exposición a la luz y lograr un equilibrio en la detección. Además, se recomienda hacer uso de diferentes dataset para la somnolencia visual para entrenar y evaluar el sistema detector de somnolencia, incorporando diferentes escenarios y condiciones de iluminación que puedan encontrarse en situaciones de conducción reales. Esto podría

mejorar la robustez y la precisión del sistema al enfrentar una variedad de condiciones adversas.

- Como perspectivas futuras, adicionalmente se planea incorporar al sistema detector de somnolencia un actuador de vibración en el asiento del conductor, lo que contribuiría a mejorar aún más la seguridad en la conducción. Además, se tiene previsto integrar un módulo GPS que habilite la monitorización del conductor únicamente cuando el vehículo supere cierta velocidad, evitando así posibles molestias por las alarmas cuando el vehículo esté detenido. También se buscará mejorar el sistema al implementar la detección de distracciones basada en la postura de la cabeza del conductor, ampliando así la capacidad de alerta y prevención de situaciones de riesgo en la conducción. Estas expansiones y mejoras potenciales consolidarán aún más la eficacia y versatilidad del sistema en la detección y mitigación de riesgos asociados con la somnolencia y la distracción en la conducción.

Abreviaciones y acrónimos

- ADAS: Sistema avanzado de asistencia a la conducción (dvanced Driver Assistance Systems en inglés)
- DSM: Sistema de monitoreo del conductor (Driver Monitor System en inglés)
- NIR: Infrarrojo Cercano (Near-Infrared en inglés).
- MAR: Mouth Aspect Ratio (Relación de Aspecto de la Boca).
- ANN: Redes Neuronales Artificiales (Artificial Neural Networks en inglés).
- DNN: Redes Neuronales Profundas (Deep Neural Networks en inglés).
- CNN: Redes Neuronales Convolucionales (Convolutional Neural Networks en inglés).
- ROI: Región de Interés (Region of Interest en inglés).
- DD-AI: Drowsiness Detection with Artificial Intelligence.
- DD-AI-G: Drowsiness Detection Gray with Artificial Intelligence.
- FPS: Fotogramas por Segundo (Frames Per Second en inglés).
- GPU: Unidad de Procesamiento Gráfico (Graphics Processing Unit en inglés).

Referencias

Abadi, M. (2016). Tensorflow: learning functions at scale. In *Proceedings of the 21st ACM SIGPLAN International Conference on Functional Programming*, pages 1–1.

Aguirre, Á. G. and Giraldo, P. J. R. (2014). Sistema embebido de bajo costo para visión artificial. *Scientia et technica*, 19(2):163–173.

Ahmed, M. and Laskar, R. H. (2022). Eye center localization using gradient and intensity information under uncontrolled environment. *Multimedia Tools and Applications*, 81(5):7145–7168.

Alava, I. (2010). Cámaras infrarrojas para investigación y desarrollo. FLIR Systems.

Alba Neppas, J. M. (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. B.S. thesis.

Albadawi, Y. (2022). Driver drowsiness detection systems. *In Encyclopedia*, page 10.

Anber, S., Alsaggaf, W., and Shalash, W. (2022). A hybrid driver fatigue and distraction detection model using alexnet based on facial features. *Electronics*, 11(2):285.

Arroyo Rodriguez, D. J. L. (2021). Diseño e implementación de sistema de visión artificial para alerta y detección de somnolencia mediante aprendizaje profundo aplicable en conductores de vehículos.

CDC-Perú (2022). Informe de siniestralidad vial y las acciones para promover la seguridad vial. `https://goo.su/dLxz`.

Cech, J. and Soukupova, T. (2016). Real-time eye blink detection using facial landmarks. *Cent. Mach. Perception, Dep. Cybern. Fac. Electr. Eng. Czech Tech. Univ. Prague*, pages 1–8.

COGNEX (2016). Machine vision applications.

Custodio Effio, A. J. and Tarrillo Nuntón, L. (2021). Sistema de monitoreo para detección de somnolencia en choferes de rutas interprovinciales de empresas de transporte de chiclayo.

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. Ieee.

Drive, C. (2016). Monitor de fatiga del conductor mr688. `https://es.care-drive.com/product/driver-fatigue-monitor-mr688`.

Dua, M., Singla, R., Raj, S., and Jangra, A. (2021). Deep cnn models-based ensemble approach to driver drowsiness detection. *Neural Computing and Applications*, 33:3155–3168.

Elhamraoui, Z. (2020). Introduction to convolutional neural network. `https://medium.com/analytics-vidhya/introduction-to-convolutional-neural-network-6942c189a723`.

Elmahmudi, A. and Ugail, H. (2021). A framework for facial age progression and regression using exemplar face templates. *The Visual Computer*, 37.

ELP, W. U. (2023a). Elp 3mp wdr micro ar0331 sensor cctv usb webcam 10pcs ir leds night vision infrared usb camera with 3.6mm lens (support microphone). `https://goo.su/nyUd`.

ELP, W. U. (2023b). Elp camera night vision 2mega pixels 1920x1080p 30fps ov2710 cmos hd usb webcam led for video door phone. https://goo.su/PsmPhw.

ELP, W. U. (2023c). Lente de alta velocidad sin distorsión de 2mp, 1080p, 60fps, 720p, 120fps, cámara usb cmos ov4689, cámara web usb infrarroja con led ir para pc y portátil. https://es.aliexpress.com/item/1005004602608442.html?gatewayAdapt=glo2esp.

E.R.S.O. (2021). Advanced driver assistance systems. https://road-safety.transport.ec.europa.eu/system/files/2022-04/Road_Safety_Thematic_Report_ADAS_2021.pdf.

Fatima, B., Shahid, A. R., Ziauddin, S., Safi, A. A., and Ramzan, H. (2020). Driver fatigue detection using viola jones and principal component analysis. *Applied Artificial Intelligence*, 34(6):456–483.

Fernandez, A. (2013). *Python 3 al descubierto*. Alfaomega Grupo Editor.

Flores Calero, M. J. (2009). Sistema avanzado de asistencia a la conducción mediante visión por computador para la detección de la somnolencia.

Florez, R., Palomino-Quispe, F., Coaquira-Castillo, R. J., Herrera-Levano, J. C., Paixão, T., and Alvarez, A. B. (2023). A cnn-based approach for driver drowsiness detection by real-time eye state identification. *Applied Sciences*, 13(13).

Fu, K., González, R., and Lee, C. (1988). *Robótica: control, detección, visión e inteligencia*. McGraw-Hill.

Fuente, C. (2019). Redes neuronales profundas–tipos y caracterısticas. *urlhttps://www.codigofuente. org/redes-neuronales-profundas-tiposcaracteristicas*.

Fundación-CEA (2022). Estudio fundación cea: Estudio somnolencia en la conducción. https://www.fundacioncea.es/actualidad/estudios-fundacion/66-estudio-fundacion-cea-estudio-somnolencia-en-la-conduccion.

Gao, Y. and Mosalam, K. M. (2018). Deep transfer learning for image-based structural damage recognition. *Computer-Aided Civil and Infrastructure Engineering*, 33(9):748–768.

Ghoddoosian, R., Galib, M., and Athitsos, V. (2019). A realistic dataset and baseline temporal model for early drowsiness detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0.

Ghorbel, H. (7 de enero, 2021). Machine learning activation function in neural network. `https://hamdi-ghorbel78.medium.com/machine-learning-activation-function-in-neural-network-12caac615964`.

Gonzalez, R. C. and Woods, R. E. (2008). *Digital image processing*. Prentice Hall, Upper Saddle River, N.J.

Google, C. (2023). Advanced guide to inception v3. `https://cloud.google.com/tpu/docs/inception-v3-advanced?hl=es-419`.

Hashemi, M., Mirrashid, A., and Beheshti Shirazi, A. (2020). Driver safety development: Real-time driver drowsiness detection system based on convolutional neural network. *SN Computer Science*, 1:1–10.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Horng, W.-B., Chen, C.-Y., Chang, Y., and Fan, C.-H. (2004). Driver fatigue detection based on eye tracking and dynamic template matching. In *IEEE International Conference on Networking, Sensing and Control, 2004*, volume 1, pages 7–12. IEEE.

Huang, W., Qiao, Y., and Tang, X. (2014). Robust scene text detection with convolution neural network induced mser trees. In *Computer Vision–ECCV 2014: 13th European*

Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part IV 13, pages 497–511. Springer.

Jiménez, F., Naranjo, J. E., Anaya, J. J., García, F., Ponz, A., and Armingol, J. M. (2016). Advanced driver assistance system for road environments to improve safety and efficiency. *Transportation research procedia*, 14:2245–2254.

Källhammer, J.-E. (2006). Night vision: Requirements and possible roadmap for fir and nir systems. In *Photonics in the automobile II*, volume 6198, pages 131–141. SPIE.

Kao, I.-H. and Chan, C.-Y. (2022). Comparison of eye and face features on drowsiness analysis. *Sensors*, 22(17):6529.

Khan, M. Q. and Lee, S. (2019). A comprehensive survey of driving monitoring and assistance systems. *Sensors*, 19(11):2574.

King, D. E. (2009). Dlib-ml: A machine learning toolkit. *The Journal of Machine Learning Research*, 10:1755–1758.

Krishna, G. S., Supriya, K., Vardhan, J., et al. (2022). Vision transformers and yolov5 based driver drowsiness detection framework. *arXiv preprint arXiv:2209.01401*.

Kumari, P. and KR, S. (2021). An optimal feature enriched region of interest (roi) extraction for periocular biometric system. *Multimedia Tools and Applications*, 80(24):33573–33591.

Kwon, K.-A., Shipley, R. J., Edirisinghe, M., Ezra, D. G., Rose, G., Best, S. M., and Cameron, R. E. (2013). High-speed camera characterization of voluntary eye blinking kinematics. *Journal of the Royal Society Interface*, 10(85):20130227.

Larsen-Freeman, D. (2013). Transfer of learning transformed. *Language Learning*, 63:107–129.

LearnOpenCV (6 de setiembre, 2022). What is face detection? ultimate guide 2023 + model comparison. `https://learnopencv.com/what-is-face-detection-the-ultimate-guide/`.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.

Leng, L. B., Giin, L. B., and Chung, W.-Y. (2015). Wearable driver drowsiness detection system based on biomedical and motion sensors. In *2015 IEEE SENSORS*, pages 1–4. IEEE.

Liu, P., Guo, J.-M., Tseng, S.-H., Wong, K., Lee, J.-D., Yao, C.-C., and Zhu, D. (2017). Ocular recognition for blinking eyes. *IEEE Transactions on Image Processing*, 26(10):5070–5081.

Lugaresi, C., Tang, J., Nash, H., McClanahan, C., Uboweja, E., Hays, M., Zhang, F., Chang, C.-L., Yong, M. G., Lee, J., et al. (2019). Mediapipe: A framework for building perception pipelines. *arXiv preprint arXiv:1906.08172*.

Machines, S. (2020). Sistema guardian. `https://guardianlatam.com`.

Manning (2019). How does computer vision work?

Martinsanz, G. and de la Cruz García, J. (2007). *Visión por computador. Imágenes Digitales y Aplicaciones. 2a Edición.* Ra-Ma S.A. Editorial y Publicaciones.

MINSA (2017). Minsa: Choferes deben dormir seis horas por lo menos para evitar accidentes. `https://www.gob.pe/institucion/minsa/noticias/14013-minsa-choferes-deben-dormir-seis-horas-por-lo-menos-para-evitar-accidentes`.

Modi, M. and Macwan, F. (2014). Face detection approaches: a survey. *International Journal of Innovative Research in science, engineering and technology*, 3(4):11107–11116.

Montiel, C. V. A. et al. (2019). Sistema embebido de bajo costo para la asistencia al conductor mediante procesamiento de expresiones faciales.

Nieto, M., Otaegui, O., Vélez, G., Ortega, J. D., and Cortés, A. (2015). On creating vision-based advanced driver assistance systems. *IET intelligent transport systems*, 9(1):59–66.

Nvidia (2022). Nvidia jetson nano. `https://www.nvidia.com/es-la/autonomous-machines/embedded-systems/jetson-nano`.

OMS (2022). Traumatismos causados por el tránsito. `https://www.who.int/es/news-room/fact-sheets/detail/road-traffic-injuries`.

ONNX (2019). Open neural network exchange. `https://onnx.ai/index.html`.

ONSV-Perú (2022). Cdc perú reportó cerca de 12 mil lesionados por accidentes de tránsito durante la primera mitad del 2022. `https://www.onsv.gob.pe/post/informe-de-siniestralidad-vial-y-las-acciones-para-promover-la-seguridad-vial/`.

OpenCV (2023). Opencv about. `https://opencv.org/about`.

P Vikranth Reddy, Joshua D'Souza, S. R. S. B. P. B. (2022). A survey on driver safety systems using internet of things. volume 11.

Pandey, N. N. and Muppalaneni, N. B. (2022). A novel drowsiness detection model using composite features of head, eye, and facial expression. *Neural Computing and Applications*, 34(16):13883–13893.

Park, S. H., Yoon, H. S., and Park, K. R. (2019). Faster r-cnn and geometric transformation-based detection of driver's eyes using multiple near-infrared camera sensors. *Sensors*, 19(1):197.

Pérez, M., Nisso, G. A. C., Buitrago, F. V., et al. (2018). Sistema embebido de detección de movimiento mediante visión artificial. *Visión electrónica*, 12(1):97–101.

Petrellis, N., Zogas, S., Christakos, P., Mousouliotis, P., Keramidas, G., Voros, N., and Antonopoulos, C. (2021). Software acceleration of the deformable shape tracking

application: How to eliminate the eigen library overhead. In *2021 2nd European Symposium on Software Engineering*, pages 51–57.

PNP (2020). Anuario estadístico policial 2020. `https://www.policia.gob.pe/estadisticopnp/documentos/anuario-2020/anuario-estadistico-policial-2020.pdf`.

ProjectPro (2023). Introduction to convolutional neural networks architecture. `https://www.projectpro.io/article/introduction-to-convolutional-neural-networks-algorithm-architecture/560`.

Python (2001-2023). Python is powerful... and fast; plays well with others; runs everywhere; is friendly easy to learn; is open. `https://www.python.org/about`.

Rajkumarsingh, B. and Totah, D. (2021). Drowsiness detection using android application and mobile vision face api. *R&D Journal*, 37:26–34.

Ramzan, M., Khan, H. U., Awan, S. M., Ismail, A., Ilyas, M., and Mahmood, A. (2019). A survey on state-of-the-art drowsiness detection techniques. *IEEE Access*, 7:61904–61919.

Raspberry, F. P. (2020). About us. `https://www.raspberrypi.org/about`.

Rondón Condori, L. Á. and Paucara Núñez, F. J. (2013). Reconocimiento de somnolencia en conductores bajo condiciones simuladas.

SCTC (2023). Machine vision development trend. `https://www.canrilloptics.com/machine-vision-development-trend.html`.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626.

Selvaraju, R. R., Das, A., Vedantam, R., Cogswell, M., Parikh, D., and Batra, D. (2016). Grad-cam: Why did you say that? *arXiv preprint arXiv:1611.07450*.

Sikander, G. and Anwar, S. (2018). Driver fatigue detection systems: A review. *IEEE Transactions on Intelligent Transportation Systems*, 20(6):2339–2352.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

STONKAM (2022). 1080p driver monitor system driving status detection. `https://www.stonkam.com/products/Driver-Status-Detection-System-DMS31.html`.

Suarez Buitrago, E. A. (2022). Sistema de detección de somnolencia en conductores de automóviles empleando técnicas de procesamiento de imágenes y machine learning.

Swapna, K. E. (2020). Convolution neural network (cnn). `https://developersbreach.com/convolution-neural-network-deep-learning`.

Systems, R. V. (2021). Near-infrared sensing. `https://www.radiantvisionsystems.com/applications/light-measurement-applications/near-infrared-sensing`.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

TensorFlow (2023). Create production-grade machine learning models with tensorflow. `https://www.tensorflow.org`.

Tian, Z. and Qin, H. (2005). Real-time driver's eye state detection. In *IEEE International Conference on Vehicular Electronics and Safety, 2005.*, pages 285–289. IEEE.

UE (2019). Regulation (eu) 2019/2144 of the european parliament and of the council. `https://eur-lex.europa.eu/eli/reg/2019/2144/`.

Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, volume 1, pages I–I. Ieee.

Weng, C.-H., Lai, Y.-H., and Lai, S.-H. (2017). Driver drowsiness detection via a hierarchical temporal deep belief network. In *Computer Vision–ACCV 2016 Workshops: ACCV 2016 International Workshops, Taipei, Taiwan, November 20-24, 2016, Revised Selected Papers, Part III 13*, pages 117–133. Springer.

Wevers, M. and Smits, T. (2020). The visual digital turn: Using neural networks to study historical images. *Digital Scholarship in the Humanities*, 35(1):194–207.

Yauri Machaca, M. R. (2021). Sistema de detección de somnolencia del conductor de vehículo mediante el procesamiento de imágenes usando matlab.

Zhao, M. (2015). Advanced driver assistant system, threats, requirements, security solutions. *Intel Labs*, pages 2–3.

Anexos

Anexo I: Datasheet de la cámara ELP-USBFHD05MT-KL36IR

Model **ELP-USBFHD05MT-KL36IR** is 2.0megapixel 1920H(X1080(V) webcam Full HD USB2.0 cameras with 10pcs IR LED Board for night vision use standard 3.6mm lens ,box casing ,3m USB Cable

Note:

Features

*2MP USB Camera ,max resolution 1920x1080P HD USB2.0 Camera.

*Equipped with M12 Mount 3.6mm lens ,optional to change other M12 Mount lens.

*Can drive 30fps@1080P and 60fps@720p, 120fps at 480P. Support IR Cut

*CMOS OV2710 sensor for high quality image and low power consumption

*High speed USB 2.0 interface for high resolution PC camera interface.

*Night vision infrared camera module with 10pcs IR LED ,can work at day and night

*UVC for use in Linux, Windows XP, WIN CE, MAC, SP2 or above.

*Camera with UVC.

Specifications

	Model : ELP-USBFHD05MT-KL36IR		Model: ELP-USBFHD05MT-KL36IR
Resolution & frame rate	1920x1080 (Full HD) MJPEG@30fps; 352x288 CIF MJPEG@30fps 640x480 (VGA) MJPEG@120fps; 800x600 SVGA MJPEG@60fps 1024x768 XGA MJPEG@30fps; 1280x720 HD MJPEG@60fps 2304x1536 3MP MJPEG@30fps; 1920x1080 FHD MJPEG@60fps		
Sensor	CMOS OV2710	Picture Format	MJPEG/ YUV2(YUY2)
Sensor Size	1/2.7 inch	Min Illumination	0.05lux
Pixel Size	3um x 3μm	Lens Parameter	standard M12 Mount 3.6mm lens optional 2.5/2.8/6/8/12mm FOV(D) 70°/100degree
Image Area	5.856μm x 3.954μm	Cable	standard 3M / optional 5m,7m
Max. Resolution	FULL HD 1920(H)x1080(V)pixels	Board Size/ Weight	38*38mm/32mm*32mm optional / about 30g
Interface	USB 2.0 High-speed	Shutter Type	Rolling shutter/ Frame exposure
Support free driver	USB Video class(UVC)	Adjustable parameters	Brightness, Contrast, Hue, Sharpness, Gamma, White balance, Backlight Compen, Exposure
USB Protocol	USB2.0/1.1/1.0	Special Function	Lens correction, Defective pixel correction Black sun compensation
OTG protocol	USB2.0 OTG	AEC, AEB,AGC	Support
Power Supply	DC 5V	Working temperature	0~50 degree
Working Current	130mA~220mA	Storage temperature	-20~75 degree
Night vision	Support, need to equipped IR sensor 850nm or 940nm lens and IR LED Board	LED board power connector	Support 2P 2.54mm socket
Support OS	Windows XP/Vista/Win7/Win8/ Linux with UVC Mac OS X 10.4.8 or later Windows CE UVC Android 4.4 or above		

Anexo 2: Código - lectura de cámara

```
import cv2
# Abre la cámara
captura = cv2.VideoCapture(0)
while True:
    ret, imagen = captura.read() # Lee un cuadro de video
    cv2.imshow('Video', imagen) # Muestra la imagen en una ventana
    if cv2.waitKey(1) & 0xFF == ord('q'):
           break # Termina el ciclo si se presiona la tecla 'q'
# Libera la cámara y cierra las ventanas
captura.release()
cv2.destroyAllWindows()
```

Anexo 3: Código - detección de puntos faciales

```
import mediapipe as mp
import cv2
# Inicializar el módulo de MediaPipe Face Mesh
mp_face_mesh = mp.solutions.face_mesh
mp_drawing = mp.solutions.drawing_utils
# Iniciar la captura de video desde la cámara
cap = cv2.VideoCapture(0)
with mp_face_mesh.FaceMesh(
    static_image_mode=False,
    max_num_faces=1,
    min_detection_confidence=0.5) as face_mesh:
    while cap.isOpened():
        ret, frame = cap.read()
        if not ret:
            break
        # Convertir el marco a RGB para MediaPipe
        frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        # Procesar el marco con MediaPipe Face Mesh
        results = face_mesh.process(frame_rgb)

        if results.multi_face_landmarks:
            for face_landmarks in results.multi_face_landmarks:
                mp_drawing.draw_landmarks(
                    image=frame,
                    landmark_list=face_landmarks,
                    connections=mp_face_mesh.FACEMESH_TESSELATION,
                    landmark_drawing_spec=None,
                    connection_drawing_spec=mp_drawing.DrawingSpec(
                        color=(0, 255, 0), thickness=1, circle_radius=1),)
        cv2.imshow('MediaPipe Face Mesh', frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
cap.release()
cv2.destroyAllWindows()
```

Anexo 4: Código - corrección de ROI

```
# Importar las bibliotecas necesarias
import cv2
import math
import mediapipe as mp
# Función para calcular la distancia entre dos puntos en un plano
def Distance(p1, p2):
  x1, y1 = p1
  x2, y2 = p2
  dist = math.hypot(x2 - x1, y2 - y1)
  return dist
# Configuración de dibujo y captura de cámara
mp_drawing = mp.solutions.drawing_utils
mp_face_mesh = mp.solutions.face_mesh
drawing_spec = mp_drawing.DrawingSpec(color=(255, 0, 0), thickness=1,
    circle_radius=1)
cap = cv2.VideoCapture(0)
# Inicialización de variables para almacenar coordenadas y dimensiones de
    ojos
left_x, left_y, left_w, left_h = 0,0,0,0
right_x, right_y, right_w, right_h = 0,0,0,0
# Inicialización de detección de puntos faciales usando MediaPipe
with mp_face_mesh.FaceMesh(
    max_num_faces=1,
    min_detection_confidence=0.75,
    min_tracking_confidence=0.5) as face_mesh:
  while cap.isOpened():
    success, frame = cap.read()
    if not success:
      print("camara␣sin␣detectar")
      continue
    frame = cv2.flip(frame, 1)
    frame1 = frame.copy()
    roi = frame1.copy()
    frame_rgb = cv2.cvtColor(frame1, cv2.COLOR_BGR2RGB)
```

```
    results = face_mesh.process(frame_rgb)
    puntos = []
    if results.multi_face_landmarks:
      for face_landmarks in results.multi_face_landmarks:
        punto = []
        for id, lm in enumerate(face_landmarks.landmark):
          ih, iw, ic = frame1.shape
          x, y = int(lm.x * iw), int(lm.y * ih)
          punto.append([x, y])
        puntos.append(punto)
        if puntos:
          puntos = puntos[0]
          eyes = [159, 145, 33, 133, 386, 374, 362, 263]
          mouth = [61, 73, 11, 303, 291, 404, 16, 180]
          # Dibujar líneas que forman un cuadrilátero alrededor de los ojos
          cv2.line(frame1,(puntos[63]),(puntos[117]),(0,0,255),2)
          cv2.line(frame1,(puntos[117]),(puntos[346]),(0,0,255),2)
          cv2.line(frame1,(puntos[346]),(puntos[293]),(0,0,255),2)
          cv2.line(frame1,(puntos[293]),(puntos[63]),(0,0,255),2)
          # Cálculo de dimensiones y coordenadas para el rectángulo de los
              ojos
          ojoder1 = puntos[336]
          ojoder2 = puntos[293]
          ojoder3 = puntos[346]
          ojoizq1 = puntos[63]
          ojoizq2 = puntos[107]
          ojoizq3 = puntos[117]
          dedW = Distance(ojoder1, ojoder2)
          dedH = Distance(ojoder2, ojoder3)
          deiW = Distance(ojoizq1, ojoizq2)
          deiH = Distance(ojoizq1, ojoizq3)
          xed = puntos[336][0]
          yed = puntos[336][1]
          xei = puntos[63][0]
          yei = puntos[63][1]
          right_x, right_y, right_w, right_h = xed, yed, int(dedW), int(dedH)
          left_x, left_y, left_w, left_h = xei, yei, int(deiW), int(deiH)
          # Ajuste de coordenadas para el rectángulo que rodea los ojos
          if (left_x > right_x):
            start_x, end_x = right_x, (left_x + left_w)
          else:
            start_x, end_x = left_x, (right_x + right_w)
          if (left_y > right_y):
            start_y, end_y = right_y, (left_y + left_h)
          else:
```

```
77              start_y, end_y = left_y, (right_y + right_h)
78            # Ajuste de tamaño para el rectángulo si cumple con ciertas
                 condiciones
79            if ((end_x-start_x)>10 and (end_y-start_y)<400):
80              start_x, start_y, end_x, end_y = start_x-10, start_y-10, end_x
                   +10, end_y+10
81              cv2.rectangle(frame1, (start_x, start_y), (end_x, end_y), (0,
                   255, 0), 3)
82              face_roi = roi[start_y:end_y, start_x:end_x]
83      # Mostrar los marcos de detección y los recuadros de los ojos
84      cv2.imshow("Detector", frame1)
85      cv2.imshow("Ojos", face_roi)
86     # Finalizar el bucle si se presiona la tecla 'Esc'
87      if cv2.waitKey(1) & 0xFF == 27:
88        break
89  # Liberar la cámara y cerrar ventanas
90  cap.release()
91  cv2.destroyAllWindows()
```

Anexo 5: Código - MAR

```
import cv2
import math
import mediapipe as mp
# Función para calcular la distancia entre dos puntos
def Distance(p1, p2):
  x1, y1 = p1
  x2, y2 = p2
  dist = math.hypot(x2 - x1, y2 - y1)
  return dist
# Función para calcular el ratio de apertura de la boca
def mouth_aspect_ratio(p1, p2, p3, p4, p5, p6, p7, p8):
     d_A = Distance(p2, p8)
     d_B = Distance(p3, p7)
     d_C = Distance(p4, p6)
     d_D = Distance(p5, p1)
     return (d_A + d_B + d_C) / (3 * d_D)
# Configuración inicial de Mediapipe y OpenCV
mp_drawing = mp.solutions.drawing_utils
mp_face_mesh = mp.solutions.face_mesh
drawing_spec = mp_drawing.DrawingSpec(color=(255, 0, 0), thickness=1,
    circle_radius=1)
cap = cv2.VideoCapture(0)
cap.set(cv2.CAP_PROP_FPS, 30)
cap.set(cv2.CAP_PROP_FOURCC,cv2.VideoWriter_fourcc(*'MJPG'))
contador_mouth = 0
# Inicia el proceso de detección de puntos faciales
with mp_face_mesh.FaceMesh(
    max_num_faces=1,
    min_detection_confidence=0.75,
    min_tracking_confidence=0.5) as face_mesh:
  while cap.isOpened():
    success, frame = cap.read()
    if not success:
      print("Ignoring␣empty␣camera␣frame.")
```

```
34          continue
35      frame = cv2.flip(frame, 1)
36      frame1 = frame.copy()
37      roi = frame1.copy()
38      frame_rgb = cv2.cvtColor(frame1, cv2.COLOR_BGR2RGB)
39      results = face_mesh.process(frame_rgb)
40      puntos = []
41      # Procesa los puntos faciales si hay detección
42      if results.multi_face_landmarks:
43        for face_landmarks in results.multi_face_landmarks:
44          punto = []
45          for id, lm in enumerate(face_landmarks.landmark):
46            ih, iw, ic = frame1.shape
47            x, y = int(lm.x * iw), int(lm.y * ih)
48            punto.append([x, y])
49          puntos.append(punto)
50          if puntos:
51            puntos = puntos[0]
52            mouth = [61, 73, 11, 303, 291, 404, 16, 180]
53            # Dibuja puntos en la boca en la imagen
54            for ids in mouth:
55              cv2.circle(frame1, tuple(puntos[ids]), 4, (200, 0, 0), cv2.FILLED
                  )
56              cv2.circle(frame1, tuple(puntos[ids]), 2, (0, 255, 255), cv2.
                  FILLED)
57            p1 = puntos[mouth[0]]
58            p2 = puntos[mouth[1]]
59            p3 = puntos[mouth[2]]
60            p4 = puntos[mouth[3]]
61            p5 = puntos[mouth[4]]
62            p6 = puntos[mouth[5]]
63            p7 = puntos[mouth[6]]
64            p8 = puntos[mouth[7]]
65            # Calcula el ratio de apertura de la boca (MAR)
66            MAR = int(mouth_aspect_ratio(p1, p2, p3, p4, p5, p6, p7, p8)*100)
67            print(MAR)
68            # Evalúa si el conductor está bostezando
69            if MAR >= 55:
70              contador_mouth+=1
71            if MAR < 55:
72              contador_mouth-=1
73            if contador_mouth >= 15:
74              contador_mouth = 15
75              print("BOSTEZO")
76            elif (contador_mouth > 0 and contador_mouth < 15):
```

```
77              print("Normal")
78            elif contador_mouth < 0:
79              contador_mouth = 0
80      cv2.imshow('Detector', frame1)
81      # Termina el programa si se presiona la tecla ESC
82      if cv2.waitKey(1) & 0xFF == 27:
83        break
84  cap.release()
85  cv2.destroyAllWindows()
```

Anexo 6: Código - extracción de frames de un vídeo

```python
import cv2
# Abrir el archivo de video para captura
capture = cv2.VideoCapture(r'C:\ruta\_del\_video')
# Contador para numerar los frames extraídos
cont = 0
# Ruta donde se guardarán los frames extraídos
path = 'C:/Users/ruta/_para/_guardar/_los/_frames/'
# Obtener la velocidad de fotogramas (frames per second - fps) del video
fps = capture.get(cv2.CAP_PROP_FPS)
# Obtener el número total de frames en el video
frames = capture.get(cv2.CAP_PROP_FRAME_COUNT)
# Imprimir información sobre la velocidad de fotogramas y el número de
    frames
print('fps', fps)
print('frames', frames)
# Bucle para recorrer todos los frames en el video
while (capture.isOpened()):
    # Leer un frame del video
    ret, frame = capture.read()
    # Verificar si se leyó correctamente el frame
    if (ret == True):
        # Guardar el frame como una imagen en la ubicación especificada
        cv2.imwrite(path + 'nombre_img.jpg' % cont, frame)
        # Incrementar el contador de frames
        cont += 1
        # Verificar si se presiona la tecla 's'
        if (cv2.waitKey(1) == ord('s')):
            break
    else:
        # Si no se pudo leer un frame, salir del bucle
        break
# Liberar la captura de video
capture.release()
# Cerrar todas las ventanas de OpenCV
cv2.destroyAllWindows()
```

Anexo 7: Código - conversión a escala de grises

```python
import cv2
import os
# Ruta donde se encuentran las imagenes RGB originales
input_images_path = "C:/ruta/_de/_frames/_RGB"
# Lista de nombres de archivos en la ruta de entrada
files_names = os.listdir(input_images_path)
print(files_names)
# Ruta donde se guardaran las imagenes en escala de grises
output_images_path = "C:/ruta/_para/_guardar/_frames/_gris"
# Comprobar si el directorio de salida no existe y crearlo si es necesario
if not os.path.exists(output_images_path):
    os.makedirs(output_images_path)
    print("Directorio creado: ", output_images_path)
# Inicializar un contador para nombres de archivo
count = 0
# Iterar a traves de los nombres de archivos en la lista
for file_name in files_names:
    # Construir la ruta completa de la imagen
    image_path = input_images_path + "/" + file_name
    print(image_path)
    # Cargar la imagen desde la ruta
    image = cv2.imread(image_path)
    # Verificar si la imagen se cargo correctamente
    if image is None:
        continue # Saltar al siguiente archivo si la imagen es nula
    # Convertir la imagen a escala de grises
    image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    # Guardar la imagen en escala de grises en la ruta de salida con nombre
       incrementado
    cv2.imwrite(output_images_path + "/noombre_img.jpg" % count, image)
    # Incrementar el contador
    count += 1
```

Anexo 8: Código - guardar ROIs extraídos en carpeta

```
import os
import cv2
import math
import numpy as np
import mediapipe as mp
# Función para calcular la distancia entre dos puntos
def Distance(p1, p2):
  x1, y1 = p1
  x2, y2 = p2
  dist = math.hypot(x2 - x1, y2 - y1)
  return dist
# Función para extraer una región de interés (ROI) del rostro
def rostro(img):
    # ... (Definición de variables y configuración de mediapipe)
    # Procesa la imagen y detecta puntos faciales utilizando Mediapipe
    frame1 = img.copy()
    roi = frame1.copy()
    frame_rgb = cv2.cvtColor(frame1, cv2.COLOR_BGR2RGB)
    results = face_mesh.process(frame_rgb)
    puntos = []
    # Si hay resultados de detección facial, procesa los puntos faciales
    if results.multi_face_landmarks:
        # ... (Código para procesar y dibujar los puntos faciales)
        # Cálculos para determinar la posición y dimensiones de los ojos
        # ...
        # Determina el área de los ojos y extrae la región de interés (ROI)
           de la imagen
        if ((end_x-start_x)>10 and (end_y-start_y)<400):
            start_x, start_y, end_x, end_y = start_x-10, start_y-10, end_x
               +10, end_y+10
            face_roi = img[start_y:end_y, start_x:end_x]
        else:
```

```
            face_roi = np.zeros(img.shape, np.uint8)
    else:
        face_roi = np.zeros(img.shape, np.uint8)
    return face_roi
# Directorios de entrada y salida
input_images_path = r"C:\Users\ruta carpeta imagenes"
output_images_path = 'C:/Users/guardar_roi/'
# Lista de nombres de archivos en el directorio de entrada
files_names = os.listdir(input_images_path)
print(len(files_names))
# Crea el directorio de salida si no existe
if not os.path.exists(output_images_path):
    os.makedirs(output_images_path)
    print("Directorio creado: ", output_images_path)
# Procesamiento de imágenes
count = 0
for file_name in files_names:
    image_path = input_images_path + "/" + file_name
    print(image_path)
    image = cv2.imread(image_path)
    if image is None:
        continue
    output = rostro(image)
    cv2.imwrite(output_images_path + "nombre_img" + str(count) + ".jpg",
        output)
    count += 1
```

Anexo 9: Código - somnolencia visual

```
# Importar las bibliotecas necesarias
import cv2
import math
import mediapipe as mp
from tensorflow.keras.models import load_model
# Función para calcular la distancia entre dos puntos en un plano
def Distance(p1, p2):
  x1, y1 = p1
  x2, y2 = p2
  dist = math.hypot(x2 - x1, y2 - y1)
  return dist
# Configuración de dibujo y captura de cámara
mp_drawing = mp.solutions.drawing_utils
mp_face_mesh = mp.solutions.face_mesh
drawing_spec = mp_drawing.DrawingSpec(color=(255, 0, 0), thickness=1,
    circle_radius=1)
cap = cv2.VideoCapture(0)
# Variables para almacenar coordenadas y dimensiones de ojos
left_x, left_y, left_w, left_h = 0,0,0,0
right_x, right_y, right_w, right_h = 0,0,0,0
contador_somno = 0
# Inicialización del modelo entrenado
clases = {0:'ojos abiertos', 1:'ojos cerrados'}
model= load_model(r"D:\ruta_del_modelo.h5_entrenado")
# Inicialización de detección de puntos faciales usando MediaPipe
with mp_face_mesh.FaceMesh(
    max_num_faces=1,
    min_detection_confidence=0.75,
    min_tracking_confidence=0.5) as face_mesh:
  while cap.isOpened():
    success, frame = cap.read()
    if not success:
      print("camara sin detectar")
      continue
    frame = cv2.flip(frame, 1)
    frame1 = frame.copy()
    roi = frame1.copy()
```

```
    frame_rgb = cv2.cvtColor(frame1, cv2.COLOR_BGR2RGB)
    results = face_mesh.process(frame_rgb)
    puntos = []
    if results.multi_face_landmarks:
      for face_landmarks in results.multi_face_landmarks:
        punto = []
        for id, lm in enumerate(face_landmarks.landmark):
          ih, iw, ic = frame1.shape
          x, y = int(lm.x * iw), int(lm.y * ih)
          punto.append([x, y])
        puntos.append(punto)
        if puntos:
          puntos = puntos[0]
          eyes = [159, 145, 33, 133, 386, 374, 362, 263]
          mouth = [61, 73, 11, 303, 291, 404, 16, 180]
          # Dibujar líneas que forman un cuadrilátero alrededor de los ojos
          cv2.line(frame1,(puntos[63]),(puntos[117]),(0,0,255),2)
          cv2.line(frame1,(puntos[117]),(puntos[346]),(0,0,255),2)
          cv2.line(frame1,(puntos[346]),(puntos[293]),(0,0,255),2)
          cv2.line(frame1,(puntos[293]),(puntos[63]),(0,0,255),2)
          # Cálculo de dimensiones y coordenadas para el rectángulo de los
              ojos
          ojoder1 = puntos[336]
          ojoder2 = puntos[293]
          ojoder3 = puntos[346]
          ojoizq1 = puntos[63]
          ojoizq2 = puntos[107]
          ojoizq3 = puntos[117]
          dedW = Distance(ojoder1, ojoder2)
          dedH = Distance(ojoder2, ojoder3)
          deiW = Distance(ojoizq1, ojoizq2)
          deiH = Distance(ojoizq1, ojoizq3)
          xed = puntos[336][0]
          yed = puntos[336][1]
          xei = puntos[63][0]
          yei = puntos[63][1]
          right_x, right_y, right_w, right_h = xed, yed, int(dedW), int(dedH)
          left_x, left_y, left_w, left_h = xei, yei, int(deiW), int(deiH)
          # Ajuste de coordenadas para el rectángulo que rodea los ojos
          if (left_x > right_x):
            start_x, end_x = right_x, (left_x + left_w)
          else:
            start_x, end_x = left_x, (right_x + right_w)
          if (left_y > right_y):
            start_y, end_y = right_y, (left_y + left_h)
```

```
81            else:
82              start_y, end_y = left_y, (right_y + right_h)
83            # Ajuste de tamaño para el rectángulo si cumple con ciertas
                condiciones
84            if ((end_x-start_x)>10 and (end_y-start_y)<400):
85              start_x, start_y, end_x, end_y = start_x-10, start_y-10, end_x
                  +10, end_y+10
86              cv2.rectangle(frame1, (start_x, start_y), (end_x, end_y), (0,
                  255, 0), 3)
87              face_roi = roi[start_y:end_y, start_x:end_x]
88              # Ajuste de la imagen face_roi
89              face_roi = cv2.resize(face_roi, (64, 64))
90              img_rsp = face_roi.reshape(1,64,64,3)
91              img_rsp = (img_rsp/255.0).astype(np.float32)
92              # Prediccion del ROI en las dos clases
93              prediction = model.predict(img_rsp)
94              predicted_class = np.argmax(prediction, axis=None)
95              prediction_label = clases[predicted_class]
96              nmax = np.max(prediction,axis=None)
97              prob = round(nmax,4)*100
98              print(prediction_label, prob)
99              # Condicion para detectar somnolencia visual
100             if predicted_class == 1 and nmax >= 0.95:
101               contador_somno += 1
102             if predicted_class == 0 and nmax >= 0.8:
103               contador_somno -= 1
104             if contador_somno >= 7:
105               contador_somno = 7
106               print("Somnolencia␣Visual")
107             elif (contador_somno > 0 and contador_somno < 7):
108               print("Normal")
109             elif contador_somno < 0:
110               contador_somno = 0
111     # Mostrar los marcos de detección y los recuadros de los ojos
112     cv2.imshow("Detector", frame1)
113     cv2.imshow("Ojos", face_roi)
114    # Finalizar el bucle si se presiona la tecla 'Esc'
115     if cv2.waitKey(1) & 0xFF == 27:
116       break
117 # Liberar la cámara y cerrar ventanas
118 cap.release()
119 cv2.destroyAllWindows()
```

Anexo 10: Datasheet - NVIDIA Jetson Nano

NVIDIA

DATA SHEET

NVIDIA Jetson Nano System-on-Module

Maxwell GPU

128-core GPU | End-to-end lossless compression | Tile Caching | OpenGL® 4.6 | OpenGL ES 3.2 | Vulkan™ 1.1 | CUDA® | OpenGL ES Shader Performance (up to) 512 GFLOPS (FP16) | Maximum Operating Frequency: 921MHz

CPU

ARM® Cortex®-A57 MPCore (Quad-Core) Processor with NEON Technology | L1 Cache: 48KB L1 instruction cache (I-cache) per core, 32KB L1 data cache (D-cache) per core | L2 Unified Cache: 2MB | Maximum Operating Frequency: 1.43GHz

Audio

Industry standard High Definition Audio (HDA) controller provides a multi-channel audio path to the HDMI interface

Memory

Dual Channel | System MMU | Memory Type: 4x 16-bit LPDDR4 | Maximum Memory Bus Frequency: 1600MHz | Peak Bandwidth: 25.6 GB/s | Memory Capacity: 4GB

Storage

eMMC 5.1 Flash Storage | Bus Width: 8-bit | Maximum Bus Frequency: 200MHz (HS400) | Storage Capacity: 16GB

Boot Sources

eMMC and USB (recovery mode)

Networking

10/100/1000 BASE-T Ethernet | Media Access Controller (MAC)

Imaging

Dedicated RAW to YUV processing engines processing up to 1400Mpix/s (up to 24MP sensor) | MIPI CSI-2 D-up to 1.5Gb/s per lane) | Support for x4 and x2 configurations (up to four active cameras)

Operating Requirements

Temperature Range (T): -25 – 97°C | Module Power: 5 – 10W | Power Input: 5.0V

Display Controller

Two independent display controllers support DSI, HDMI, DP, eDP

MIPI-DSI (1.5Gbps/lane) Single x2 lane | Maximum Resolution: 1920x1200 at 60Hz (up to 240Hz)

HDMI 2.0a/b (up to 6Gbps) | DP 1.2a (HBR2) 5.4 Gbps) | eDP 1.4 (HBR2) 5.4Gbps) | Maximum Resolution (DP/eDP/HDMI): 3840 x 2160 at 60Hz (up to 240Hz)

Clocks

System clock: 38.4MHz | Reference clock: 32.768kHz | Dynamic clock scaling and clock source selection

Multi-Stream HD Video and JPEG

Video Decode

H.265 (Main, Main 10): 2160p 60fps | 1080p 240fps

H.264 (BP/MP/HP/Stereo SEI half-res): 2160p 60fps | 1080p 240fps

H.264 (MVC Stereo per view): 2160p 30fps | 1080p 120fps

VP9 (Profile 0, 8-bit): 2160p 60fps | 1080p 240fps

VP8: 2160p 60fps | 1080p 240fps

VC-1 (Simple, Main, Advanced): 1080p 120fps | 1080i 240fps

MPEG-2 (Main): 2160p 60fps | 1080p 240fps | 1080i 240fps

Video Encode

H.265: 2160p 30fps | 1080p 120fps

H.264 (BP/MP/HP): 2160p 30fps | 1080p 120fps

H.264 (MVC Stereo per view): 1440p 30fps | 1080p 60fps

VP9: 2160p 30fps | 1080p 120fps

JPEG (Decode and Encode): 600MP/s

Peripheral Interfaces

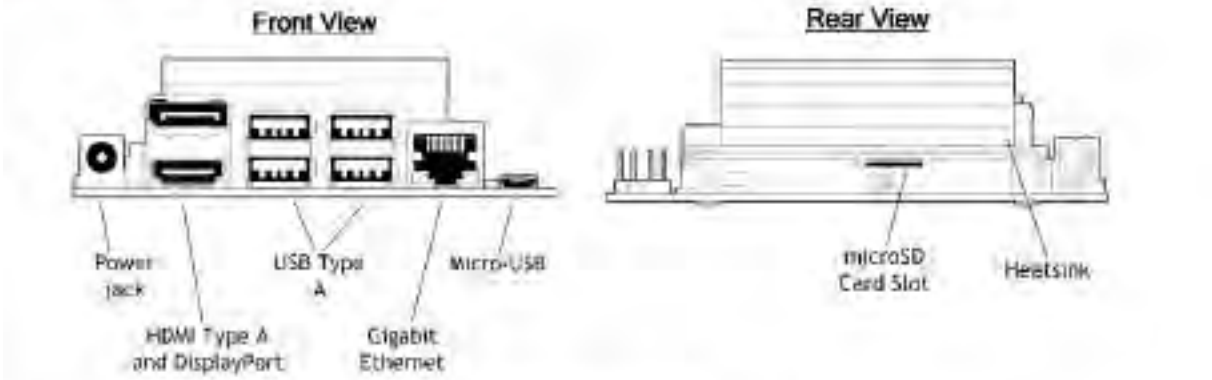
xHCI host controller with integrated PHY: 1 x USB 3.0, 3 x USB 2.0 | USB 3.0 device controller with integrated PHY | EHCI controller with embedded PHY for USB 2.0 | 4-lane PCIe, one x1/2/4 controller | single SD/MMC controller supporting SDIO 4.0, SD HOST 4.0 | 3 x UART | 2 x SPI | 4 x I2C | 2 x I2S; support I2S, RJM, LJM, PCM TDM (multi-slot mode) | GPIOs

Mechanical

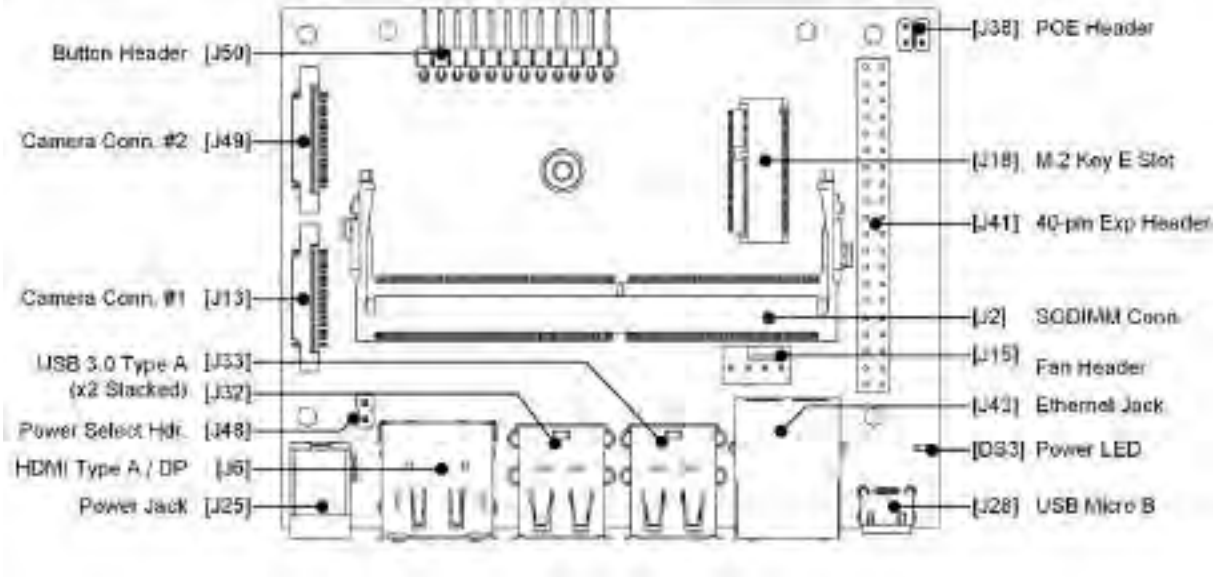
Module Size: 69.6mm x 45mm | PCB: 8L HDI | Connector: 260-pin SO-DIMM

DEVELOPER KIT INTERFACES

Developer kit module and carrier boards: front and rear views



Developer kit module and carrier board: rev B01 top view



Anexo 11: Datasheet - pantalla 7 inch

7 Inch HDMI compatible Touch Screen Display for Raspberry Pi 4B/3B Jetson Nano

Feature:

- Brand new and high quality
- IPS full angle display, the viewing Angle is large and dynamic picture quality is excellent
- Support other mainstream systems Raspbian, Kali linux, ubuntu, etc. when used with Raspberry Pi, single touch plug and play without driver
- Support mainstream development boards such as Raspberry Pi, Banana Pi, Bit Black, etc.
- Support win10/win8/win7 system, support five-point touch plug and play driver
- Suitable for Raspberry Pi, Jetson Nano, PC screen IPS wide-angle display

Specification:

- Screen Type: IPS screen
- Screen Size: 7.0 inch
- Resolution: 1024 x 600
- Backlight Adjustment: Independent Button Adjustment
- Touch Screen Type: Capacitive Touch Screen
- Touch IC: GT911
- Power: Micro USB (5V/2A)
- Video Input Interface: HDMI-compatible
- Audio Output Interface: 3.5mm Audio Interface + 2 Speakers (8Ω 2W)

Anexo 12: Datasheet - Buck LM2596

LM2596 SIMPLE SWITCHER® Power Converter 150-kHz 3-A Step-Down Voltage Regulator

1 Features

- New product available. LMR33630 36-V, 3-A, 400-kHz synchronous converter
- 3.3-V, 5-V, 12-V, and adjustable output versions
- Adjustable version output voltage range: 1.2-V to 37-V ±4% maximum over line and load conditions
- Available in TO-220 and TO-263 packages
- 3-A output load current
- Input voltage range up to 40 V
- Requires only four external components
- Excellent line and load regulation specifications
- 150-kHz Fixed-frequency internal oscillator
- TTL shutdown capability
- Low power standby mode, I_Q, typically 80 µA
- High efficiency
- Uses readily available standard inductors
- Thermal shutdown and current-limit protection
- Create a custom design using the LM2596 with the WEBENCH Power Designer

2 Applications

- Appliances
- Grid infrastructure
- EPOS
- Home theater

3 Description

The LM2596 series of regulators are monolithic integrated circuits that provide all the active functions for a step-down (buck) switching regulator, capable of driving a 3-A load with excellent line and load regulation. These devices are available in fixed output voltages of 3.3 V, 5 V, 12 V, and an adjustable output version.

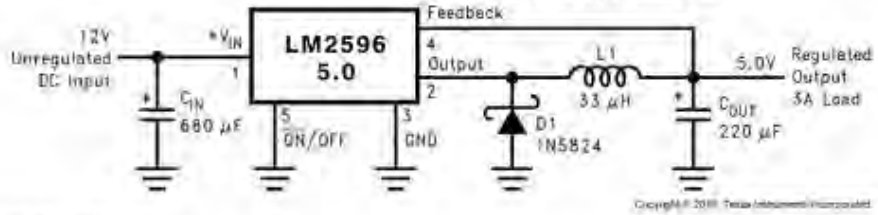
Requiring a minimum number of external components, these regulators are simple to use and include internal frequency compensation, and a fixed-frequency oscillator.

The LM2596 series operates at a switching frequency of 150 kHz, thus allowing smaller sized filter components than what would be required with lower frequency switching regulators. Available in a standard 5-pin TO-220 package with several different lead bend options, and a 5-pin TO-263 surface mount package.

The new product, LMR33630, offers reduced BOM cost, higher efficiency, and an 85% reduction in solution size among many other features. Start WEBENCH Design with the LMR33630.

PART NUMBER	PACKAGE(1)	BODY SIZE (NOM)
LM2596	TO-220 (5)	14.986 mm × 10.16 mm
	TO-263 (5)	10.10 mm × 8.89 mm

(1) For all available packages, see the orderable addendum at the end of the data sheet.



(Fixed Output Voltage Versions)

Typical Application

Anexo 13: Código - .h5 a .onnx

```
import onnx
import os
import tf2onnx
from tensorflow.keras.models import load_model
loaded_keras_model = load_model(r'C:\ruta\del\archivo\.h5')
onnx_model, _ = tf2onnx.convert.from_keras(loaded_keras_model)
onnx.save(onnx_model, r'C:\ruta\para\guardar\archivo\.onnx')
```

Anexo 14: Código - cámara fps y resolución

```python
import cv2
# Abrir la cámara con la resolución y velocidad de cuadros deseados
cap = cv2.VideoCapture(0) # 0 para la cámara predeterminada
# Configurar la resolución
cap.set(3, 800) # Ancho
cap.set(4, 600) # Alto
# Configurar la velocidad de cuadros
cap.set(cv2.CAP_PROP_FPS, 30) # 30 fps
cap.set(cv2.CAP_PROP_FOURCC,cv2.VideoWriter_fourcc(*'MJPG')) # tipo de
    imagen
while True:
    # Captura un cuadro desde la cámara
    ret, frame = cap.read()
    # Muestra el cuadro capturado
    cv2.imshow('Camara', frame)
    # Sale del bucle si se presiona la tecla 'q'
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
# Libera la cámara y cierra la ventana
cap.release()
cv2.destroyAllWindows()
```

Anexo 15: Código - detección de somnolencia en sistema embebido

```
import cv2
import time
import math
import vlc
import numpy as np
import mediapipe as mp
import onnxruntime
# Definición de una función para calcular la distancia entre dos puntos
def Distance(p1, p2):
  x1, y1 = p1
  x2, y2 = p2
  dist = math.hypot(x2 - x1, y2 - y1)
  return dist
# Definición de una función para calcular el índice de relación de aspecto
   de la boca (MAR)
def mouth_aspect_ratio(p1, p2, p3, p4, p5, p6, p7, p8):
     d_A = Distance(p2, p8)
     d_B = Distance(p3, p7)
     d_C = Distance(p4, p6)
     d_D = Distance(p5, p1)
     return (d_A + d_B + d_C) / (3 * d_D)
# Configuración de las bibliotecas y los parámetros iniciales
mp_drawing = mp.solutions.drawing_utils
mp_face_mesh = mp.solutions.face_mesh
drawing_spec = mp_drawing.DrawingSpec(color=(255, 0, 0), thickness=1,
   circle_radius=1)
cap = cv2.VideoCapture(0)
cap.set(3, 640)
cap.set(4, 480)
left_x, left_y, left_w, left_h = 0, 0, 0, 0
right_x, right_y, right_w, right_h = 0, 0, 0, 0
contador_mouth = 0
```

```
contador_s = 0
Ptime = 0
# Configuración del reproductor de alarma de sonido
alarm = vlc.MediaPlayer("sound.mp3")
# Definición de un diccionario de clases para etiquetar los resultados
clases = {0: 'Undetected', 1: 'DETECTED'}
# Inicialización de Onnxruntime para cargar el modelo de red neuronal
providers = ['TensorrtExecutionProvider', 'CUDAExecutionProvider']
session = onnxruntime.InferenceSession("model_best_ddaig.onnx", providers=
    providers)
session.get_inputs()[0].shape
session.get_inputs()[0].type
input_name = session.get_inputs()[0].name
output_name = session.get_outputs()[0].name
# Inicialización de la detección de puntos clave facial con Mediapipe
    FaceMesh
with mp_face_mesh.FaceMesh(
    max_num_faces=1,
    min_detection_confidence=0.75,
    min_tracking_confidence=0.5) as face_mesh:
  while True:
    # Captura de un frame del video
    success, vidcap = cap.read()
    if not success:
      print("Ignoring␣empty␣camera␣frame.")
      continue
    frame = cv2.flip(vidcap, 1)
    frame1 = frame.copy()
    frame2 = frame.copy()
    roi = frame1.copy()
    frame_rgb = cv2.cvtColor(frame1, cv2.COLOR_BGR2RGB)
    # Procesamiento de los puntos clave faciales
    results = face_mesh.process(frame_rgb)
    puntos = []
    # Cálculo del FPS
    cTime = time.time()
    fps = 1 / (cTime - Ptime)
    Ptime = cTime
    cv2.putText(frame1, f'FPS:{int(fps)}', (10, 30), cv2.
        FONT_HERSHEY_COMPLEX_SMALL, 1, (255, 255, 255), 2)
    if results.multi_face_landmarks:
      for face_landmarks in results.multi_face_landmarks:
        punto = []
        for id, lm in enumerate(face_landmarks.landmark):
          ih, iw, ic = frame1.shape
```

```
73            x, y = int(lm.x * iw), int(lm.y * ih)
74            punto.append([x, y])
75          puntos.append(punto)
76           # Detección de características faciales (ojos, boca, etc.)
77          if puntos:
78            puntos = puntos[0]
79            countourId = [175, 140, 172, 93, 162, 104, 151, 333, 389, 323, 397,
                  378]
80            eyes = [159, 145, 33, 133, 386, 374, 362, 263]
81            mouth = [61, 73, 11, 303, 291, 404, 16, 180]
82            # Dibujar puntos en los contornos, ojos y boca
83            for ids in countourId:
84              cv2.circle(frame1, tuple(puntos[ids]), 2, (209, 204, 208), cv2.
                    FILLED)
85            for ids in mouth:
86              cv2.circle(frame1, tuple(puntos[ids]), 4, (255, 40, 0), cv2.
                    FILLED)
87              cv2.circle(frame1, tuple(puntos[ids]), 2, (5, 255, 194), cv2.
                    FILLED)
88            for ids in eyes:
89              cv2.circle(frame1, tuple(puntos[ids]), 4, (255, 40, 0), cv2.
                    FILLED)
90              cv2.circle(frame1, tuple(puntos[ids]), 2, (5, 255, 194), cv2.
                    FILLED)
91            # Cálculo de la distancia y dimensiones de los ojos
92            ojoder1 = puntos[336]
93            ojoder2 = puntos[293]
94            ojoder3 = puntos[346]
95            ojoizq1 = puntos[63]
96            ojoizq2 = puntos[107]
97            ojoizq3 = puntos[117]
98            dedW = Distance(ojoder1, ojoder2)
99            dedH = Distance(ojoder2, ojoder3)
100           deiW = Distance(ojoizq1, ojoizq2)
101           deiH = Distance(ojoizq1, ojoizq3)
102           xed = puntos[336][0]
103           yed = puntos[336][1]
104           xei = puntos[63][0]
105           yei = puntos[63][1]
106           # propuesta de la corrección de la región de interes (ROI) de los
                ojos
107           right_x, right_y, right_w, right_h = xed, yed, int(dedW), int(dedH)
108           left_x, left_y, left_w, left_h = xei, yei, int(deiW), int(deiH)
109           if (left_x > right_x):
110             start_x, end_x = right_x, (left_x + left_w)
```

```
111             else:
112               start_x, end_x = left_x, (right_x + right_w)
113             if (left_y > right_y):
114               start_y, end_y = right_y, (left_y + left_h)
115             else:
116               start_y, end_y = left_y, (right_y + right_h)
117             if ((end_x - start_x) > 50 and (end_y - start_y) < 200):
118               start_x, start_y, end_x, end_y = start_x - 10, start_y - 10,
                     end_x + 10, end_y + 10
119               cv2.rectangle(frame1, (start_x, start_y), (end_x, end_y), (246,
                     255, 0), 2)
120               face_roi = roi[start_y:end_y, start_x:end_x]
121               # Detección de somnolencia visual
122               face_roi = cv2.resize(face_roi, (64, 64))
123               face_roi_gray = cv2.cvtColor(face_roi, cv2.COLOR_BGR2GRAY)
124               img_rsp = face_roi_gray.reshape(1, 64, 64, 1)
125               img_rsp = (img_rsp / 255.0).astype(np.float32)
126               # Realizar inferencia en el modelo de red neuronal
127               prediction = session.run([output_name], {input_name: img_rsp})
128               predicted_class_onnx = np.argmax(prediction, axis=None)
129               prediction_label_onnx = clases[predicted_class_onnx]
130               nmax = np.max(prediction)
131               prob = (round(nmax, 3) * 100)
132               # Mostrar información en pantalla
133               cv2.putText(frame1, "Microsleep:", (10, 60), cv2.
                     FONT_HERSHEY_COMPLEX_SMALL, 1, (255, 0, 0), 1)
134               cv2.putText(frame1, "Yawning:", (10, 110), cv2.
                     FONT_HERSHEY_COMPLEX_SMALL, 1, (255, 0, 0), 1)
135               cv2.putText(frame1, "Alarm:", (10, 160), cv2.
                     FONT_HERSHEY_COMPLEX_SMALL, 1, (255, 0, 0), 1)
136               cv2.putText(frame1, "Time␣(s):", (10, 210), cv2.
                     FONT_HERSHEY_COMPLEX_SMALL, 1, (255, 0, 0), 1)
137               # Activar o desactivar la alarma según las condiciones
138               if predicted_class_onnx == 1 and nmax >= 0.95:
139                 contador_s += 1
140               if predicted_class_onnx == 0 and nmax >= 0.50:
141                 contador_s -= 1
142               if contador_s >= (0.3 * fps):
143                 contador_s = 8
144                 alarm.play()
145                 cv2.putText(frame1, prediction_label_onnx, (10, 85), cv2.
                       FONT_HERSHEY_COMPLEX_SMALL, 1, (0, 0, 255), 1)
146                 cv2.putText(frame1, "ON", (10, 185), cv2.
                       FONT_HERSHEY_COMPLEX_SMALL, 1, (0, 0, 255), 1)
```

```
147                 cv2.putText(frame1, str(round(contador_s / 30, 2)), (10, 235),
                        cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0, 0, 255), 1)
148               elif (contador_s > 0 and contador_s < (0.3 * fps)):
149                 alarm.stop()
150                 cv2.putText(frame1, prediction_label_onnx, (10, 85), cv2.
                        FONT_HERSHEY_COMPLEX_SMALL, 1, (0, 255, 0), 1)
151                 cv2.putText(frame1, "OFF", (10, 185), cv2.
                        FONT_HERSHEY_COMPLEX_SMALL, 1, (0, 255, 0), 1)
152                 cv2.putText(frame1, str(round(contador_s / 30, 2)), (10, 235),
                        cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0, 255, 0), 1)
153               elif contador_s < 0:
154                 contador_s = 0
155                 cv2.putText(frame1, prediction_label_onnx, (10, 85), cv2.
                        FONT_HERSHEY_COMPLEX_SMALL, 1, (0, 255, 0), 1)
156                 cv2.putText(frame1, "OFF", (10, 185), cv2.
                        FONT_HERSHEY_COMPLEX_SMALL, 1, (0, 255, 0), 1)
157                 cv2.putText(frame1, str(round(contador_s / 30, 2)), (10, 235),
                        cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0, 255, 0), 1)
158             # Cálculo del índice de relación de aspecto de la boca (MAR)
159             p1 = puntos[mouth[0]]
160             p2 = puntos[mouth[1]]
161             p3 = puntos[mouth[2]]
162             p4 = puntos[mouth[3]]
163             p5 = puntos[mouth[4]]
164             p6 = puntos[mouth[5]]
165             p7 = puntos[mouth[6]]
166             p8 = puntos[mouth[7]]
167             MAR = int(mouth_aspect_ratio(p1, p2, p3, p4, p5, p6, p7, p8) * 100)
168             if MAR >= 55:
169               contador_mouth += 1
170             if MAR < 55:
171               contador_mouth -= 1
172             # Detectar somnolencia por el MAR
173             if contador_mouth >= (5 * fps):
174               contador_mouth = 15
175               alarm.play()
176               cv2.putText(frame1, "DETECTED", (10, 135), cv2.
                      FONT_HERSHEY_COMPLEX_SMALL, 1, (0, 0, 255), 1)
177             elif (contador_mouth > 0 and contador_mouth < (5 * fps)):
178               alarm.stop()
179               cv2.putText(frame1, "Undetected", (10, 135), cv2.
                      FONT_HERSHEY_COMPLEX_SMALL, 1, (0, 255, 0), 1)
180             elif contador_mouth < 0:
181               contador_mouth = 0
```

```
182             cv2.putText(frame1, "Undetected", (10, 135), cv2.
                    FONT_HERSHEY_COMPLEX_SMALL, 1, (0, 255, 0), 1)
183     # Configuración de la pantalla y visualización
184     cv2.rectangle(frame2, (5, 10), (240, 310), (0, 0, 0), -1)
185     frame3 = cv2.addWeighted(frame2, 0.3, frame1, 0.7, 0)
186     frame4 = cv2.resize(frame3, (1024, 600))
187     cv2.imshow('detect␣somno', frame4)
188     # Esperar a que se presione la tecla 'Esc' para salir
189     key = cv2.waitKey(33)
190     if key == 27:
191       break
192 # Liberar la cámara y cerrar las ventanas de visualización
193 cap.release()
194 cv2.destroyAllWindows()
```

Anexo 16: Tabla - Costo y presupuesto

El costo de los componentes utilizados en el sistema detector de somnolencia en conductores de vehículos se encuentra detallado en la siguiente Tabla y ha sido financiado por el autor de la investigación.

Componentes	Cantidad	Valor U. (S/.)	Valor Total (S/.)
Kit NVIDIA Jetson Nano B01	1	1273.2	1273.2
Cámara infrarroja ELP-KL36IR	1	223.43	223.43
Pantalla tactil 7-inch	1	222.77	222.77
DeskPi - carcasa	1	126.39	126.39
Convertidor DC-DC Step-Down LM2596	1	3	3
Adaptador de carro	1	10	10
otros componentes	1	8	8
Precio Total:			S/ 1866.79

Tabla de costo y presupuesto del sistema