UNIVERSIDAD TÉCNICA DEL NORTE

Facultad de Ingeniería En Ciencias Aplicadas

Carrera de Software

Detección de somnolencia en conducción diurna y nocturna utilizando la red neuronal convolucional (CNN) MobileNet

Trabajo de grado presentado ante la Ilustre Universidad Técnica del Norte previo a la obtención del título de Ingeniero en Software

Autor:
Esteban David Enríquez Gallegos

Director:
PhD. Iván Danilo García Santillán

Ibarra – Ecuador
2025

UNIVERSIDAD TÉCNICA DEL NORTE
BIBLIOTECA UNIVERSITARIA

AUTORIZACIÓN DE USO Y PUBLICACIÓN

A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

1. IDENTIFICACIÓN DE LA OBRA

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

DATOS DE CONTACTO			
CÉDULA DE IDENTIDAD:	1004803472		
APELLIDOS Y NOMBRES:	Esteban David Enríquez Gallegos		
DIRECCIÓN:	Luis Mideros y Río Blanco, Casa #32		
EMAIL:	edenriquezg@utn.edu.ec / esteban.enriquez543@gmail.com		
TELÉFONO FIJO:		**TELÉFONO MÓVIL:**	0962876318

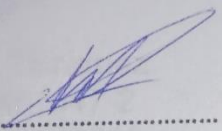
DATOS DE LA OBRA	
TÍTULO:	"Detección de somnolencia en conducción diurna y nocturna utilizando la red neuronal convolucional (CNN) MobileNet
AUTOR (ES):	Esteban David Enríquez Gallegos
FECHA: DD/MM/AAAA	25/02/2025
SOLO PARA TRABAJOS DE GRADO	
PROGRAMA:	■ **PREGRADO** ☐ **POSGRADO**
TITULO POR EL QUE OPTA:	Ingeniero en Software
DIRECTOR:	PhD. Iván Danilo García Santillán
ASESOR:	MsC. Cosme MacArthur Ortega Bustamante

2. CONSTANCIAS

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

Ibarra, a los 25 días del mes de febrero de 2025

EL AUTOR:



(Firma).....
Nombre: Esteban David Enríquez Gallegos
CI: 1004803472

CERTIFICACIÓN DIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR

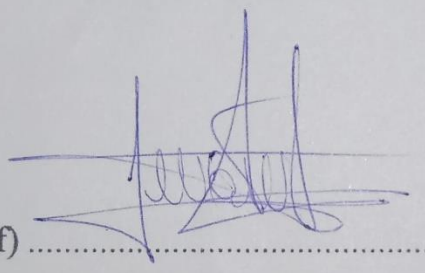
Ibarra, 25 de febrero de 2025,

PhD. Iván Danilo García Santillán

DIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR

CERTIFICA:

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



(f) .....

PhD. Iván Danilo García Santillán

C.C.: 1002292603

DEDICATORIA

Este trabajo se lo dedico a mi familia.

Isaura y Edvio, mis padres, quienes me demostraron con su sacrificio y dedicación que todo es posible y me han brindado su apoyo incondicional desde el momento que elegí esta carrera como mi proyecto de vida.

A Andrés y Vane, mis hermanos, por ser un motor de sueños, creer en mi y alentarme siempre a seguir a adelante.

Y a Sirius, mi amigo fiel de cuatro patas que ha sido un rayo de alegría en los momentos más grises.

Enríquez Gallegos Esteban David

AGRADECIMIENTO

Quiero extender mi agradecimiento al PhD. Iván García mi director de tesis por su orientación, compromiso y apoyo a lo largo de este arduo proceso.

Al Msc. Cosme Ortega, mi asesor de tesis por todos los conocimientos impartidos a lo largo de la carrera.

Agradezco a mi familia por su amor, compañía y paciencia.

Y por último a mis amigos que peses a la distancia jamás dejaron de estar pendientes de mí y fueron un refugio e impulso para seguir adelante.

Enríquez Gallegos Esteban David

RESUMEN

La fatiga y la somnolencia son factores críticos en los accidentes de tráfico, representando una de las principales causas de siniestros viales con víctimas. Para abordar esta problemática, este trabajo propone una solución basada en inteligencia artificial para la detección de somnolencia en conductores, utilizando una red neuronal convolucional (CNN) basada en la arquitectura MobileNetV3.

El modelo fue entrenado siguiendo la metodología KDD (Knowledge Discovery in Databases) y se utilizó un conjunto de datos personalizado, compuesto por imágenes propias y muestras seleccionadas de 12 conjuntos de datos públicos. Para mejorar la precisión del sistema, se integró el cálculo del Eye Aspect Ratio (EAR) a partir de 68 puntos faciales (landmarks) y el etiquetado de las imágenes en dos clases, somnoliento (drowsy) y alerta (non-drowsy).

El modelo fue evaluado en distintos escenarios controlados, incluyendo rostros descubiertos con 97% accuracy, uso de gafas convencionales 94%, gafas de sol 92%, iluminación diurna 94% y nocturna 91%. Para garantizar su implementación en sistemas embebidos, se optimizó el modelo utilizando TensorFlow Lite, logrando un tiempo de inferencia de 120 milisegundos por fotograma (ms/f) con 8 fotogramas por segundo (FPS). La implementación en una Raspberry Pi 4 se realizó siguiendo el criterio de adaptabilidad definido en la norma ISO 25010, asegurando su portabilidad y eficiencia en entornos con recursos limitados.

Palabras Clave: Redes Neuronales Convolucionales, Deep Learning, MobileNet, somnolencia, conductores, sistemas embebidos, clasificación, KDD, ISO 25010, Eye Aspect Ratio, EAR, PERCLOS.

ABSTRACT

Fatigue and drowsiness are critical factors in traffic accidents, representing one of the leading causes of road crashes with victims. To address this issue, this work proposes an artificial intelligence-based solution for drowsiness detection in drivers using a convolutional neural network (CNN) based on the MobileNetV3 architecture.

The model was trained following the KDD (Knowledge Discovery in Databases) methodology and utilized a customized dataset composed of proprietary images and selected samples from 12 public datasets. To enhance the system's accuracy, the calculation of the Eye Aspect Ratio (EAR) from 68 facial landmarks was integrated, and image labeling was performed into two classes: drowsy and non-drowsy.

The model was evaluated in various controlled scenarios, achieving 97% accuracy for uncovered faces, 94% for conventional glasses, 92% for sunglasses, 94% for daytime lighting, and 91% for nighttime lighting. To ensure its implementation in embedded systems, the model was optimized using TensorFlow Lite, achieving an inference time of 120 milliseconds per frame (ms/f) with 8 frames per second (FPS). The implementation on a Raspberry Pi 4 was carried out following the adaptability criteria defined in the ISO 25010 standard, ensuring portability and efficiency in resource-limited environments.

Keywords: Convolutional Neural Networks, Deep Learning, MobileNet, drowsiness, drivers, embedded systems, classification, KDD, ISO 25010, Eye Aspect Ratio, EAR, PERCLOS.

ÍNDICE

DEDICATORIA V
AGRADECIMIENTO VI
RESUMEN VII
ABSTRACT VIII
Introducción 1
Tema 1
Problema 1
Objetivos 2
Alcance 2
Metodología 4
Justificación 5
Capítulo 1: Marco Teórico 6
1.1 Somnolencia en conductores 6
1.2 Metodología KDD 7
1.2.1 Selección 8
1.2.2 Preprocesamiento 8
1.2.3 Transformación 8
1.2.4 Data Mining 8
1.2.5 Interpretación 9
1.3 ISO 25010 9
1.3.1 Portabilidad 10
1.4 Red neuronal convolucional (CNN) MobileNet 11
1.4.1 Clasificación de la familia MobileNet 13
1.4.2 MobileNetV3 14
1.5 Sistemas embebidos 17
1.5.1 Jetson Nano 18
1.5.2 Raspberry Pi 4 18
1.6 Trabajos Relacionados 18
CAPÍTULO 2: DESARROLLO 22
2.1 Selección de la arquitectura MobileNetV3 22

2.2 Selección de Dataset 23
2.2.1 Búsqueda de datasets 23
2.2.2 Recolección de imágenes 24
2.3 Preprocesado 29
2.3.1 Data Cruda 29
2.3.2 Etiquetado de los Landmarks faciales 31
2.3.3 Clasificación de las imágenes 36
2.3.4 Aumento de imágenes 38
2.4 Transformación 39
2.4.1 Dataset final 39
2.5 Minería de datos 40
2.5.1 Enfoques 40
2.5.2 Plataforma de entrenamiento 42
2.5.3 Pre-Entrenamiento 44
2.5.4 Entrenamiento Final 48
2.6 Aplicativo en el sistema embebido 52
2.6.1 Aplicativo en Python para la detección de tiempo real 54
CAPÍTULO 3: RESULTADOS 59
3.1 Métricas de evaluación 59
3.2 Modelos evaluados 62
3.3 Conjuntos de pruebas para la evaluación 66
3.3.1 Datasets con accesorios faciales 67
3.3.2 Datasets con condiciones de iluminación 68
3.3.3 Elección del mejor modelo 69
3.4 Comparación con otros modelos 70
3.5 Evaluaciones en condiciones reales 72
3.6 Portabilidad usando la ISO 25010 74
Limitaciones 77
Discusión 78
Conclusiones 83
Recomendaciones 84
Referencias 85
ANEXOS 90

Anexo 1: 90
Anexo 2: 90
Anexo 3: 90
Anexo 4: 90

ÍNDICE FIGURAS

Figura 1..... 1
Figura 2..... 2
Figura 3..... 4
Figura 4..... 7
Figura 5..... 12
Figura 6..... 14
Figura 7..... 15
Figura 8..... 16
Figura 9..... 24
Figura 10..... 27
Figura 11..... 28
Figura 12..... 28
Figura 13..... 29
Figura 14..... 30
Figura 14..... 32
Figura 15..... 33
Figura 16..... 34
Figura 17..... 35
Figura 18..... 36
Figura 19..... 37
Figura 20..... 37
Figura 21..... 38
Figura 22..... 44
Figura 23..... 45
Figura 24..... 45
Figura 25..... 48
Figura 26..... 53
Figura 27..... 54
Figura 28..... 55
Figura 29..... 56
Figura 30..... 57
Figura 31..... 58

Figura 32 63
Figura 33 64
Figura 34 65
Figura 35 66
Figura 36 74

ÍNDICE TABLAS

Tabla 1 6
Tabla 2 16
Tabla 3 23
Tabla 4 25
Tabla 5 31
Tabla 6 39
Tabla 7 42
Tabla 8 49
Tabla 9 50
Tabla 10 52
Tabla 11 52
Tabla 12 62
Tabla 13 67
Tabla 14 67
Tabla 15 68
Tabla 16 69
Tabla 17 69
Tabla 18 70
Tabla 19 71
Tabla 20 73
Tabla 21 73
Tabla 22 75
Tabla 23 79
Tabla 24 81

Introducción

Tema

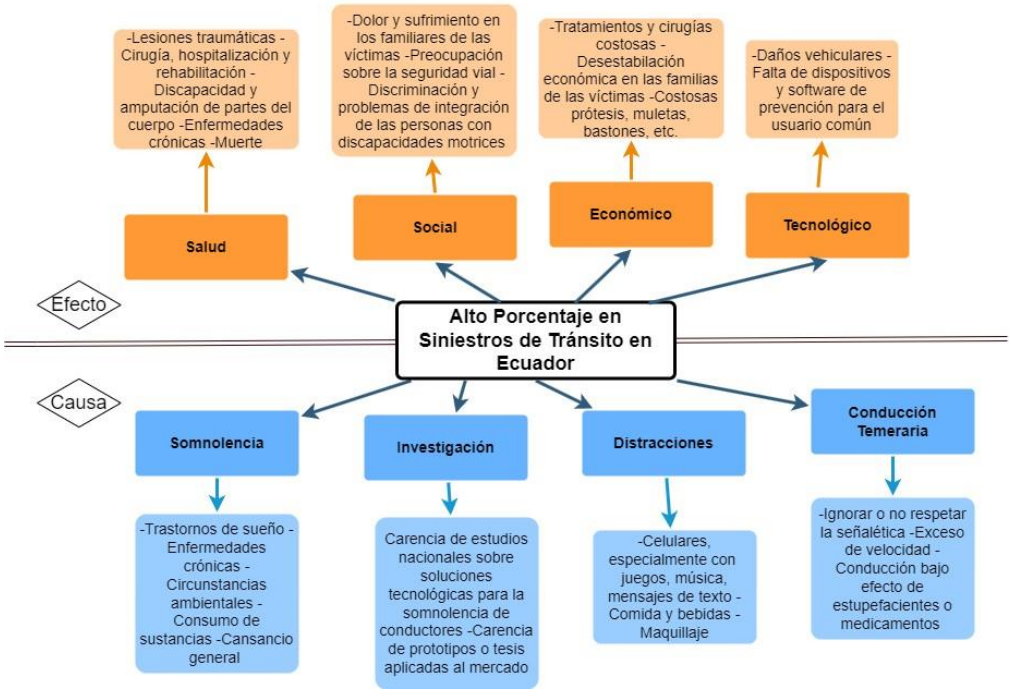
Detección de somnolencia en conducción diurna y nocturna utilizando la red neuronal convolucional (CNN) MobileNet.

Problema

Según MOVIDELNOR, la fatiga y la falta de sueño son causas primordiales de accidentes de tráfico con víctimas, siendo esta una de las 5 principales causas. La fatiga reduce la atención, aumenta los errores al volante y aumenta el riesgo de accidentes, contribuyendo a aproximadamente el 20-30% de los siniestros (MOVIDELNOR, 2023). En respuesta a esta problemática y teniendo en cuenta la figura 1, la inteligencia artificial (IA) puede desempeñar un papel crucial en la detección de somnolencia en los conductores a través del análisis de datos de cámaras, sensores y otros dispositivos. Estos datos permiten realizar análisis automáticos en tiempo real, como evaluar la frecuencia de parpadeo, la posición de los ojos y la cabeza, así como cambios en la expresión facial del conductor (Florez, Quispe, & Coaquira, 2022).

Figura 1

Árbol de problema causa – efecto



Objetivos

Objetivo General

Detectar somnolencia en conductores utilizando la red neuronal convolucional (CNN) MobileNet.

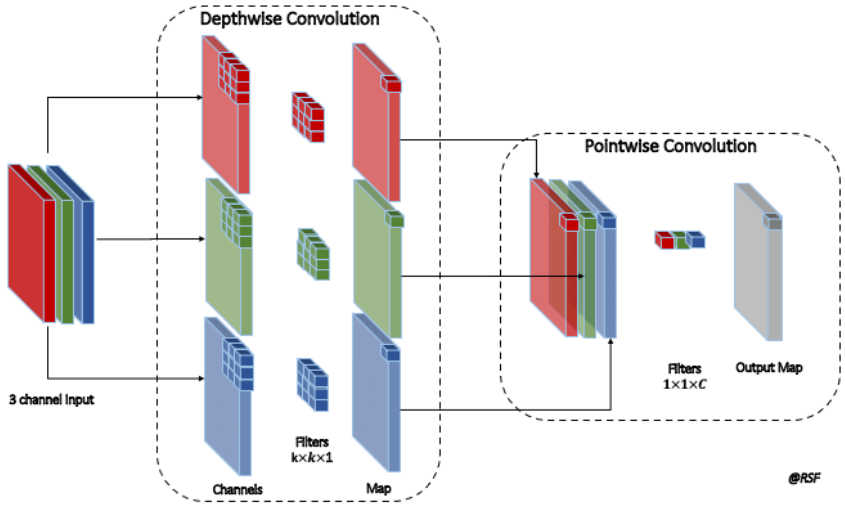
Objetivos Específicos

- Elaborar un marco teórico respecto a redes neuronales profundas en detección de somnolencia.
- Adquirir y etiquetar imágenes de conductores basadas en landmarks faciales.
- Entrenar la red neuronal profunda MobileNet usando Tensorflow y Keras.
- Validar el algoritmo con métricas cuantitativas y gráficas estadísticas.

Alcance

La presente Tesis busca implementar un sistema para detectar la somnolencia en el rostro de las personas a través del reconocimiento de las expresiones en los ojos a través de landmarks faciales. Esto lo lograremos a través del Deep Learning y las redes neuronales convolucionales CNN la cual es un tipo especial de red neuronal usado para la detección de objetos y la clasificación de imágenes (IBM, 2020). Usaremos como base la arquitectura de red neuronal de Google CNN MobileNet, el cual está diseñado para dispositivos móviles o embebidos al usar, cómo se puede observar en la figura 2, una convolución separable por profundidad y solo dos hiper parámetros: un multiplicador de longitud y un multiplicador de resolución para obtener un equilibrio entre precisión y latencia (Aguilar G. , Contreras, Eleuterio, & Lara, 2022).

Figura 2
Arquitectura de Inteligencia Artificial CNN MobileNet

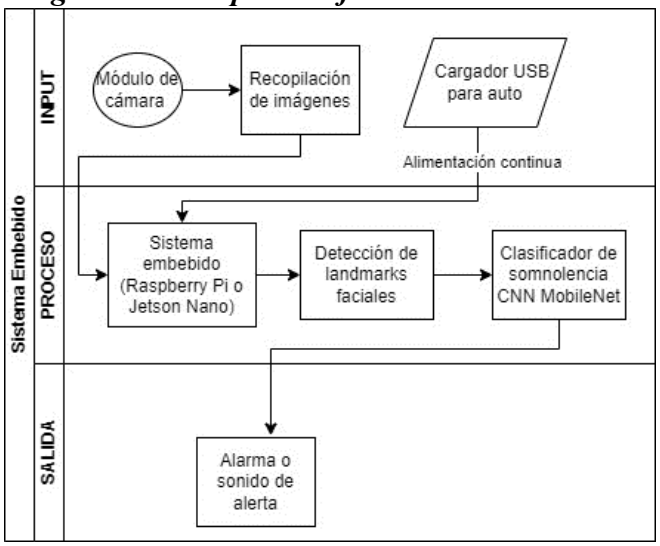


Nota: El gráfico representa la convolución en la arquitectura CNN MobileNet (Cheuque C. , Querales, Leon, Salas, & Torres, 2022).

A su vez se usará librerías de OpenCV y TensorFlow con Keras en Python para implementar técnicas de detección y patrones de parpadeos. El modelo entrenado se implementará en un dispositivo embebido Jetson Nano o Raspberry Pi 4, con lo cual podremos implementar una cámara para monitorear en tiempo real la apertura y cierre de los ojos de los conductores y determinar si sobrepasa un umbral permitido, clasificando el estado del conductor como somnoliento (drowsy) o alerta (awake), como se ve en la figura 3. Se usará la metodología KDD (Knowledge Discovery in Databases) para extraer los patrones de los datos y se aplicará la característica de portabilidad de la ISO 25010 para la implementación en el sistema embebido.

Figura 3

Diagrama de bloques del funcionamiento del sistema embebido



Metodología

Al trabajar con un algoritmo de Deep Learning, como lo es la red neuronal convolucional MobileNet, se usará la investigación de tipo aplicada, pues su elección se busca generar conocimiento con aplicación directa y a mediano plazo en la sociedad. Este proceso metodológico constará de las siguientes etapas fundamentales:

Investigación y entendimiento de artículos científicos: En esta fase inicial, se hará uso de trabajos de titulación anteriores de la UTN sobre somnolencia, además se llevará a cabo una revisión exhaustiva de la literatura científica existente, usando principalmente la biblioteca de Scopus, en el campo de la detección de somnolencia y el uso de Deep Learning con redes neuronales convolucionales. Se identificarán algoritmos, arquitecturas y conceptos relevantes que servirán como base para el desarrollo del proyecto (LOZADA, 2014).

Identificación de Necesidades y Desarrollo de la Tecnología: Se usará el dataset de somnolencia recopilado por el Ing. Iván García y se lo ampliará con datos recopilados de autoría propia usando la metodología KDD para el preprocesamiento y recopilado de datos de los landmarks faciales. Para el entrenamiento de la CNN MobileNet se hará uso de una computadora laptop con gráfica RTX 3050 TI y/o de Google Collab y se usará Python conjunto con las librerías de OpenCV, Tensorflow/Keras.

Desarrollo de Prototipos y Aplicación del Modelo Entrenado: La etapa final del proceso investigativo se centrará en la creación de un prototipo en un sistema embebido Jetson nano o Raspberry pi 4 que materialice el modelo previamente entrenado en el anterior paso. Este prototipo se diseñará con la finalidad de ser transportable, para garantizar esta característica se usará conceptos de portabilidad de la ISO 25010. Con esto, se buscará que el modelo sea de fácil instalación para el uso de los conductores en general.

Justificación

Dentro de los objetivos de desarrollo sostenible (ODS) de las naciones unidas, el presente proyecto apunta al "Objetivo 9 Industria, Innovación e Infraestructura", pues este busca fomentar el desarrollo de tecnologías emergentes y de proporcionar un acceso universal a dichas tecnologías (United Nations, 2023), incluyendo la detección automática de somnolencia en conductores es un desafío fundamental para la seguridad vial de todos los conductores y peatones. En consecuencia y teniendo en cuenta el plan "Creación de oportunidades 2021 Ecuador", este proyecto apunta al "Eje de seguridad integral", específicamente al "Objetivo 9 Garantizar la seguridad ciudadana, orden pública y gestión de riesgos" pues una de las aristas de este objetivo es minimizar la inseguridad vial, siendo esta la problemática principal del presente proyecto (Consejo Nacional de Planificación, 2021).

Algunas justificaciones del proyecto son:

Justificación Tecnológica: Al usar tecnologías actuales y avanzadas de inteligencia artificial como Deep Learning CNN MobileNet, se está proporcionando avances en la aplicación de esta tecnología como a su vez una infraestructura vial que ofrece más seguridad al promover una conducción segura.

Justificación Social: Este proyecto tiene el potencial de minimizar accidentes de tránsito lo que contribuye a disminuir el alto índice de muerte, discapacidades y pérdidas económicas.

Los beneficiaros del presente estudio son los conductores de vehículos privados principalmente de la provincia de Imbabura, según estimaciones de (MOVIDELNOR, 2023) la cantidad de vehículos matriculados en la ciudad de Ibarra para el año 2020 era de 33524.

Capítulo 1: Marco Teórico

1.1 Somnolencia en conductores

Según la agencia nacional de tránsito ANT, los siniestros de tránsitos están dentro de las 10 principales causas de muerte en el Ecuador (ANT, 2022), además de que según estimaciones de MOVILDERNOR (MOVILDELNOR, 2022) entre el 20 y el 30% de los accidentes tienen su origen en la fatiga, la cual puede ser producida por muchos factores como se puede apreciar en la Tabla 1. Es por tanto que la somnolencia al volante emerge como un factor crítico que compromete la atención, el juicio y la capacidad de reacción de los conductores, generando un incremento sustancial en el riesgo de colisiones vehiculares.

En todo el mundo, casi la mitad de las muertes por accidentes de tránsito ocurren en países de bajos y medianos ingresos. En estos países, las lesiones por accidentes de tránsito son la principal causa de muerte para personas de 15 a 29 años. Esto también implica un impacto económico significativo, pues los costos asociados con los accidentes de tránsito, como la atención médica y los daños a la propiedad, representan alrededor del 3% del Producto Interno Bruto (PIB) de los países afectados (Road Safety, 2022).

Tabla 1
Causantes de la somnolencia

Causa	Descripción
Falta de sueño	Se produce cuando no se duerme lo suficiente durante la noche. La cantidad de sueño recomendada para los adultos es de 7-8 horas.
Trastornos del sueño	Algunas de las causas de la somnolencia diurna son el insomnio, la apnea del sueño y desórdenes alimenticios.
Medicamentos	Algunos medicamentos, como los sedantes, los antidepresivos y los antihistamínicos, pueden causar somnolencia como efecto secundario.
Condiciones médicas	Depresión, la diabetes y la enfermedad de Parkinson son algunas de las enfermedades que pueden causar somnolencia diurna.

Estilo de vida	Tener una vida ajetreada que conlleve a condiciones como el estrés, el consumo de alcohol y la falta de ejercicio, pueden contribuir a la somnolencia diurna.

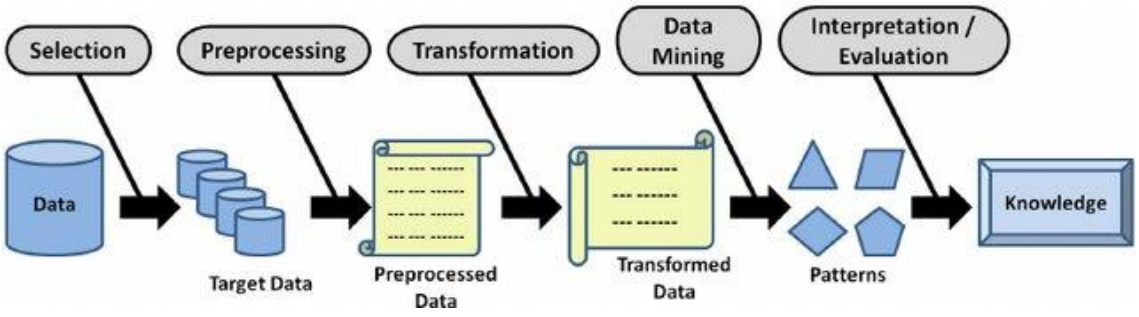
Nota: Adaptado de (MOVILDELNOR, 2022).

1.2 Metodología KDD

El descubrimiento de conocimiento en base de datos, KDD por sus siglas en inglés Knowledge Discovery in Databases (Fayyad, Piatetsky, & Padhraic, 1996), es un proceso de análisis de datos que tiene como objetivo identificar patrones, modelos y tendencias que sean relevantes, útiles y comprensibles para una organización. Requiere un conocimiento profundo del área de estudio y una comprensión de las necesidades de la organización (Conexion ESAN, 2018). Al ser una metodología muy eficaz al momento de trabajar con grandes cantidades de datos esta se usa frecuentemente en temas de machine learning, Deep learning y aplicaciones generales de Big Data.

KDD surge como una herramienta esencial en el análisis de conjuntos observacionales extensos, en donde se utiliza el análisis de conjuntos de datos para revelar relaciones insospechadas y resumir la información de manera comprensible y útil. Esto, como se muestra en la figura 4, es posible debido a su proceso claramente definido en múltiples etapas (Bello, 2023).

Figura 4
Metodología KDD



Nota: El gráfico muestra los pasos que sigue KDD para generar conocimiento (Fayyad, Piatetsky, & Padhraic, 1996).

1.2.1 Selección

La recopilación de datos es el proceso de reunir información de diferentes fuentes y almacenarla en un único repositorio, comúnmente llamado data warehouse. Esta información puede provenir de fuentes internas, como sistemas de información de la empresa, o de fuentes externas, como datos de mercado.

El data warehouse debe ser seguro y confiable para garantizar la integridad de los datos. Los datos deben estar protegidos de accesos no autorizados y de pérdidas. También deben estar organizados de forma que se puedan recuperar y analizar fácilmente (Servicios Digitales, 2021).

1.2.2 Preprocesamiento

Debido a que es común en una data warehouse tener valores inexistentes, incompletos o erróneos se debe preparar los datos para su posterior análisis. De esta forma data con valores inexistentes o incompleta se pueden eliminar o reemplazar con estimaciones mientras que los valores erróneos se pueden corregir o eliminar (Servicios Digitales, 2021).

1.2.3 Transformación

Los datos se convierten a un formato que sea adecuado para el análisis posterior. Esto puede incluir la normalización de los datos, la codificación de los datos y la creación de nuevas variables. El resultado de esta fase es un conjunto de datos limpios y transformados que están listos para ser analizados, una vista minable.

1.2.4 Data Mining

Se aplican algoritmos de minería de datos a la vista minable para extraer patrones y modelos. Un modelo es una representación simbólica y resumida de los datos analizados con el fin de extraer información necesaria para generar conclusiones de manera cómoda y eficaz (Bello, 2023).

Entre todos los algoritmos de minería de datos nos enfocaremos en este grupo:

Algoritmos de aprendizaje automático: Como el Deep Learning, aprenden de los datos para la creación de modelos que realizan tareas automáticas. Por ejemplo, se pueden utilizar para

predecir la probabilidad de que un cliente compre un producto, la probabilidad de que un paciente padezca una enfermedad o la somnolencia en conductores.

1.2.5 Interpretación

En esta fase, se evalúan los resultados de la minería de datos para determinar su validez, utilidad y aplicación práctica. Esto se hace teniendo en cuenta los siguientes aspectos:

- Coherencia: Los resultados deben ser consistentes con los datos originales.
- Realismo: Los resultados deben ser plausibles y apegados a la realidad.
- Utilidad: Los resultados deben ser útiles para los propósitos previstos.
- Aplicación: Los resultados deben ser aplicables a diversas situaciones.

Una vez que se han evaluado los resultados estos se expresan de manera que sean comprensibles para las personas que utilizarán los datos. Finalmente, los resultados se extrapolan a los casos que ya se tengan contemplados los cuales deberán ser situaciones reales que ayuden a determinar su validez y utilidad (Bello, 2023).

1.3 ISO 25010

La norma ISO 25010, titulada "Sistemas y software de calidad: Requisitos y evaluación de la calidad", es un estándar internacional desarrollado por la Organización Internacional de Normalización (ISO) que establece un marco para la evaluación y gestión de la calidad en sistemas y software. Esta norma, parte integral de la serie ISO/IEC 25000, se centra en proporcionar un conjunto de requisitos y directrices para la calidad del producto software con el fin de poder evaluar la calidad a lo largo de su ciclo de vida, promoviendo de esta forma la entrega de productos que cumplan con las expectativas y necesidades de los usuarios (ISO 25000, 2011). El modelo de calidad definido por esta norma se compone de ocho características de calidad:

- Funcionalidad: La capacidad del producto para proporcionar las funciones y características requeridas.
- Usabilidad: La facilidad con la que los usuarios pueden aprender a usar el producto y realizar sus tareas.

- Eficiencia: El grado en que el producto utiliza los recursos de manera efectiva.
- Fiabilidad: La capacidad del producto para funcionar sin fallas.
- Mantenibilidad: La facilidad con la que el producto puede ser modificado o adaptado.
- Portabilidad: La capacidad del producto para ser usado en diferentes entornos.
- Seguridad: La capacidad del producto para proteger los datos y los sistemas (International Standard, 2011).

1.3.1 Portabilidad

La norma ISO 25010 aborda la portabilidad de un producto o sistema a través de tres aspectos fundamentales: adaptabilidad, instalabilidad y reemplazabilidad.

La adaptabilidad representa el grado en que un sistema se puede ajustar de manera efectiva y eficiente a cambiantes entornos de hardware, software, u otros entornos operativos de uso. Esta capacidad es crucial para garantizar la compatibilidad continua del sistema a medida que evolucionan las tecnologías subyacentes y las necesidades de los usuarios.

La instalabilidad es el grado de efectividad y eficiencia con que un sistema puede instalarse y/o desinstalarse con éxito en un entorno específico. Un sistema con alta instalabilidad se puede implementar de manera rápida y sencilla, minimizando los tiempos de inactividad y los costos de implementación (Ormeño, 2019).

La reemplazabilidad denota el grado en que un producto se puede por otro que cumpla con el mismo propósito en el mismo entorno. Esto es importante para la migración a nuevas plataformas o para la integración con otros sistemas pues permite la transición sin problemas entre sistemas, minimizando las interrupciones, cambios de entorno y el impacto en los usuarios.

En conjunto, estos tres aspectos de portabilidad proporcionan un marco integral para evaluar la capacidad de un producto o sistema para adaptarse, instalarse y sustituirse de manera efectiva en diversos contextos, contribuyendo así a la excelencia en la calidad del software y la ingeniería de sistemas (Paneiva, 2023).

1.4 Red neuronal convolucional (CNN) MobileNet

Desde el 2012, y gracias a los avances en el hardware de la computación, se ha logrado generar nuevos métodos y algoritmos para el aprendizaje profundo. Centrándonos únicamente en el aprendizaje automático por reconocimiento de imágenes, se ha evidenciado un gran desarrollo en cuanto a su rendimiento y eficacia gracias a las redes neuronales convolucionales, CNN por sus siglas en ingles Convolutional Neuronal Network (Civik & Yuzgec, 2023).

Estas redes incorporan múltiples capas para calcular la salida en función de un conjunto de datos determinado (Villanueva, 2022), asumen que las entradas corresponden a imágenes y orienta su estructura para trabajarla como una matriz de datos. Al trabajar con las imágenes de esta forma logra generar modelos eficaces para el reconocimiento de formas, como rostros, animales u objetos, esto gracias a que incrementa el nivel de abstracción en cada capa, transitando de patrones y características simples, como la detección de bordes, hacia elementos más complejos, como texturas de pelo, razas de animales o modelos de vehículos. Ejemplos de estas redes son LSTM, VGG16, InceptionV3, DenseNet, MobileNets, ResNet, SimpleCNN, etc.

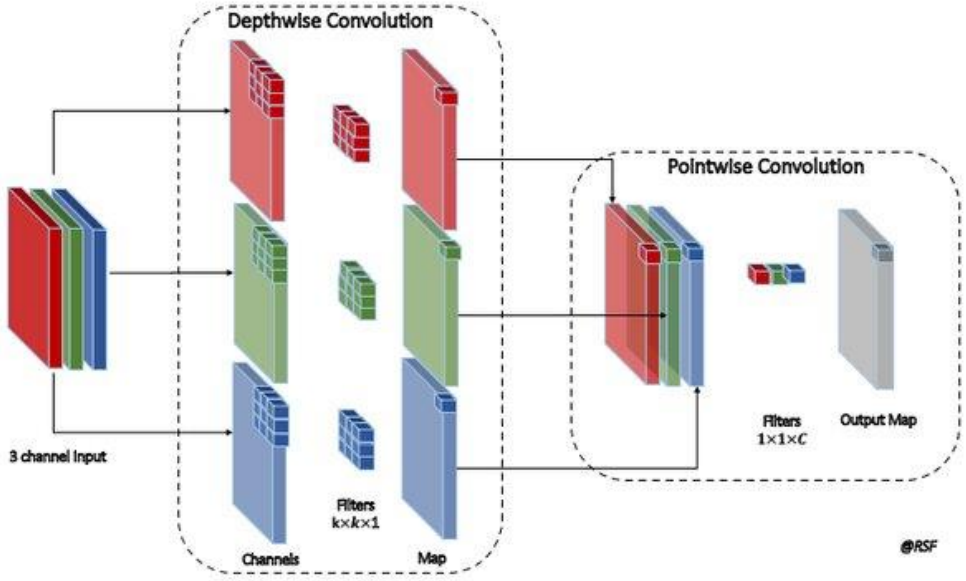
En los últimos años ha existido una tendencia para mejorar estos modelos, haciéndolos más complejos, exactos y profundos con el fin de lograr un incremento significativo en el accuracy de las predicciones, sin embargo, estos avances tienden a ignorar las limitaciones de la tecnología, dejando en un segundo plano la eficiencia computacional, la velocidad de entrenamiento, velocidad de procesamiento y el peso de los modelos (Wang, y otros, 2017). Dentro de este contexto MobileNet utiliza una arquitectura de red especialmente diseñada para la eficiencia pues esta se basa en convoluciones separables por profundidad o, en su término en inglés, depthwise separable convolution, que factorizan una convolución estándar en dos capas más eficientes:

Convolución por profundidad (depthwise convolution): aplica un filtro a cada canal de entrada individualmente.

Convolución de punto a punto (pointwise convolution): aplica una combinación lineal de los resultados de la convolución por profundidad para crear nuevos canales de salida (Cheuque, Querales, & Torres, An Efficient Multi-Level Convolutional Neural Network Approach for White Blood Cells Classification, 2022).

Como se aprecia en la figura 5, al combinar estos hiper parámetros MobileNet puede construir modelos de reconocimiento de imágenes con favorables niveles de precisión y eficiencia, como por ejemplo somnolencia en conductores.

Figura 5
Depthwise separable convolution



Nota: En la figura se muestra el funcionamiento de una convolución separable por profundidad con sus capas dephtwise y pointwise usado en MobileNetV1 (Cheuque, Querales, & Torres, An Efficient Multi-Level Convolutional Neural Network Approach for White Blood Cells Classification, 2022).

Los pilares y la filosofía de MobileNet se basa en la eficiencia y en la precisión. Esto es debido a que MobileNet está diseñado para ser eficiente en términos de tamaño y recursos computacionales gracias a su técnica de convolución separable por profundidad, esto sin descuidar la precisión pues, a pesar de su tamaño liviano, logra una precisión comparable a las arquitecturas más pesadas en términos de coste computacional.

Eficiencia: Las convoluciones separables por profundidad son una forma de factorizar una convolución estándar en dos operaciones más eficientes: una convolución por profundidad y una convolución de punto por punto. La convolución por profundidad aplica un filtro a cada canal de entrada individualmente, mientras que la convolución de punto por punto aplica una combinación lineal de los resultados de la convolución por profundidad para crear nuevos canales de salida (Aguilar G. , Contreras, Eleuterio, & Lara, 2024).

Esta factorización reduce significativamente el costo computacional de las convoluciones estándar. Por ejemplo, una convolución estándar de 3 × 3 con 64 filtros tiene un costo computacional de 14400 operaciones de multiplicación y suma. En cambio, una convolución separable por profundidad de 3 × 3 con 64 filtros tiene un costo computacional de 6400 operaciones de multiplicación y suma.

Precisión: La arquitectura de MobileNet es relativamente simple, pero está diseñada para ser eficiente y precisa. La red consta de 28 capas, incluyendo 13 convoluciones separables por profundidad, 12 capas de agrupación por promedio y una capa totalmente conectada.

MobileNet se entrena con un conjunto de datos de entrenamiento grande y diverso, que incluye imágenes de diferentes clases, tamaños y condiciones de iluminación. Este conjunto de datos ayuda a la red a aprender a generalizar a nuevas imágenes (Aurelien, 2023).

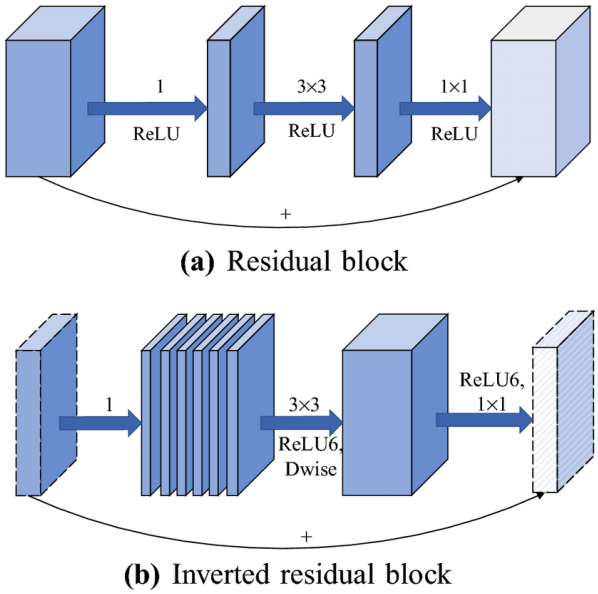
1.4.1 Clasificación de la familia MobileNet

La primera versión, MobileNetV1, fue la propuesta de Google de crear un modelo poderoso, versátil y ligero que logra satisfacer las demandas de aplicaciones de reconocimiento de imágenes en entornos móviles y embebidos sin sacrificar la precisión que ofrecen modelos más pesados (Wang, y otros, 2017). Es también en esta generación que se introduce la convolución separable por profundidad anteriormente mencionada y que podemos observar en la Figura 2.

MobileNetV2 llegó para reemplazar MobileNetV1, en esta generación trabaja con el concepto de residual block, modificándolo para introducir una nueva característica denominada inverted residual blocks representados en la Figura 6. Un residual block es usado para crear conexiones directas de una capa a otra, reduciendo problemas en la gradiente que propician sobreajustes en el entrenamiento (Sandler, Howard, Zhu, Zhmoginov, & Chen, 2018). Con el fin de incrementar la eficiencia de este concepto, MobileNetV2 invierte la estructura, de aquí viene el nombre de inverted residual block, en donde las conexiones y la extracción de características se las realiza en las últimas capas usando depthwise y pointwise convolution, lo que conlleva a generar modelos más eficientes en cuanto a la memoria usada (Machine Learning Reply GmbH, 2023).

Figura 6

Inverted Residual Block



Nota: En la figura se muestra el funcionamiento de un inverted residual block implementados en MobileNetV2 para mejorar su rendimiento en contraste con un residual block normal (Sandler, Howard, Zhu, Zhmoginov, & Chen, 2018).

MobileNetV3 es la última arquitectura de la familia MobileNets, incorporando mejoras generales de rendimiento en su arquitectura, ofreciendo un rendimiento superior en tareas de clasificación de imágenes (Howard, y otros, 2019). Por tal motivo, la convierte en la versión escogida para el proceso de data mining del presente trabajo.

1.4.2 MobileNetV3

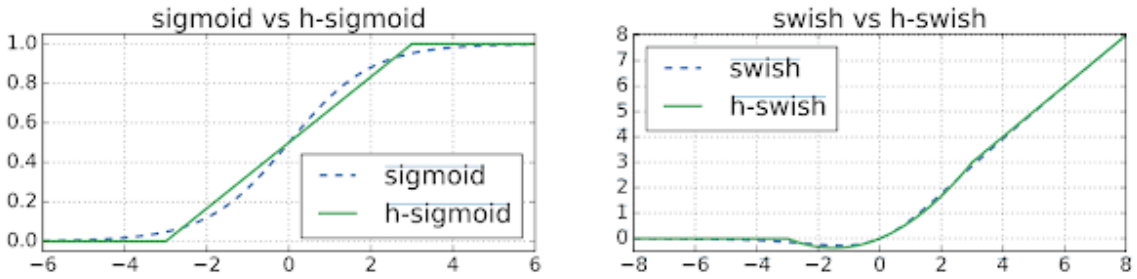
Este modelo incorpora mejoras en la arquitectura gracias a que se ajustó usando una combinación de reconocimiento de hardware con una búsqueda de arquitectura de red, NAS por sus siglas en inglés network architecture search, la cual es una técnica de aprendizaje automático que consiste en explorar dentro de un rango arquitecturas los hiper parámetros ideales, como el número y tamaño de capas, con el fin de encontrar diseños para redes neuronales eficientes y precisas (Andrew & Suyog, 2019).

Además, MobileNetV3 se desarrolló aplicando una nueva función de activación llamada hard-swish, que reemplazan a las tradicionales sigmoides, además como se observa en la figura 7

esta función h-swish está basada en swish. El principal problema de este tipo de funciones es lo ineficiente que son al computarse en dispositivos móviles, esto se soluciona con el desarrollo de un nuevo bloque llamado squeeze and excitation que se adiciona a los inverted residual block introducidos en MobileNetV2 | (Howard, y otros, 2019).

Figura 7

Comportamiento de la función hard swish



Nota: En la figura se muestra el crecimiento de la función hard swish y hard sigmoid con sus contrapartes swish y sigmoide respectivamente (Howard, y otros, 2019).

Todas estas mejoras y cambios ayudaron a desarrollar un modelo con un alto estado del arte, el cual se divide en dos tipos:

MobileNetV3Small: Versión compacta de MobileNetV3. Tiene 22 capas y 1,4 millones de parámetros, lo que la hace ideal para dispositivos móviles con recursos limitados, comparado con MobileNetV2 ofrece un rendimiento superior por 6.6% en cuanto a accuracy con un tamaño reducido y similar nivel de latencia (Jain V. , 2019).

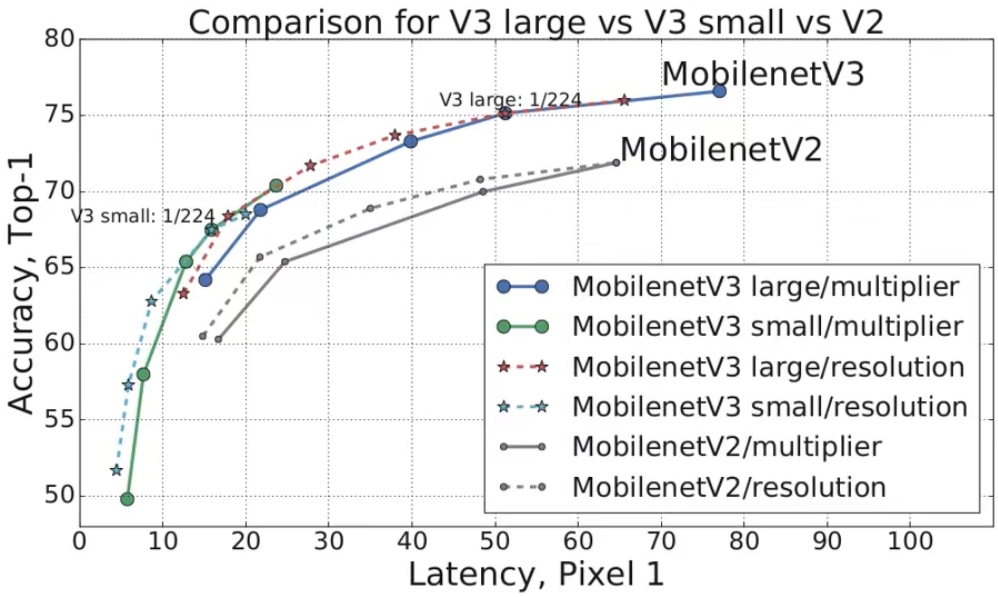
MobileNetV3Large: Versión más grande de MobileNetV3. Tiene 108 capas y 10,4 millones de parámetros, lo que la hace adecuada para dispositivos móviles con más recursos. Ofrece un rendimiento superior en tareas como la clasificación de imágenes y la detección de objetos. En comparación con MobileNetV2, MobileNetV3Large ofrece un rendimiento superior con un 20% menos de latencia.

Como se puede apreciar en la figura 8, para casos que requieren bajo nivel computacional MobileNetV3Small sobrepasa a MobileNetV2 e incluso a su versión Large en casos con baja resolución, mientras que en caso de un nivel computacional más alto MobileNetV3Large es el

modelo que logra un rendimiento superior en cuanto latencia y precisión sin sacrificar la ligereza que caracteriza a esta familia de modelos (Vryniotis & Massa, 2021).

Figura 8

Precisión vs Latency de los modelos V3Large y V3Small



Nota: En la figura se muestra el desempeño de accuracy vs latencia de ambas versiones de MobileNetV3 con distintas resoluciones. (Howard, y otros, 2019).

Tomando en cuenta la sección 1.4.2, la tabla 2 sintetiza las diferencias clave entre estos dos modelos, tomando especial énfasis en aspectos como el tamaño del modelo, la cantidad de parámetros, el costo computacional, la precisión y otros factores relevantes para ayudar a comprender sus respectivas aplicaciones y ventajas.

Tabla 2

Diferencias entre versiones Small y Large

Característica	**MobileNetV3Small**	**MobileNetV3Large**
Tamaño del Modelo (Depende del dataset usado)	Alrededor de 5Mb	Alrededor de 15 Mb
Número de Parámetros (Depende del dataset usado)	Alrededor de 1 millón	Alrededor de 10 millón
Costo Computacional	Menor	Mayor

Número de Capas	22	108
Tamaño de la Imagen de Entrada	Diseñado para aprovechar resoluciones pequeñas 128 x 128 224 x 224	Además de las resoluciones de Small admite resoluciones más grandes como 512 x 512 1024 x 1024
Precisión	Gran precisión en modelos livianos pero menor precisión en modelos complejos	Mayor precisión en escenarios de clasificación complejos
Velocidad/Eficiencia	Inferencia más rápida	Inferencia más lenta
Caso de Uso	Tareas livianas como inferencia de imágenes en tiempo real	Tareas más complejas, aplicaciones de alto rendimiento
Latencia	Menor latencia con resoluciones pequeñas	Mayor latencia que Small en resoluciones pequeñas pero una latencia menor con resoluciones más grandes

Nota: Adaptado de (Howard, y otros, 2019).

1.5 Sistemas embebidos

Es un sistema informático que, a diferencia de un computador convencional, está basado en microprocesadores y microcontroladores con el propósito de centrarse en una función en específico, como puede ser un sistema de control de acceso, electrónica en lavadoras, taxímetro o en la propuesta de este trabajo de titulación, un dispositivo de detección de somnolencia. Son en simples términos computadoras de bajo consumo de energía, que pueden conectarse con otros dispositivos, incluso con otros sistemas embebidos entre sí, para formar parte de un sistema más complejo (INCIBE, 2018).

En el ámbito de la implementación de inteligencia artificial en sistemas embebidos tenemos múltiples aplicaciones como es el reconocimiento por visión artificial combinado con el aprendizaje automático usado para el análisis de datos, clasificación y detección de patrones en

imágenes. En este contexto surgen factores cruciales a considerar, entre ellos se encuentran la precisión, velocidad de cálculo, tamaño compacto del modelo y eficiencia energética.

1.5.1 Jetson Nano

Para la propuesta de un sistema de detección de somnolencia la GPU NVIDIA Jetson Nano se destaca como una opción eficiente para pruebas y evaluación debido a su asequibilidad y uso extendido en campos como inteligencia artificial y Deep learning.

Una Jetson Nano estándar cuenta con una GPU Maxwell de 128 núcleos, una CPU ARM A57 de cuatro núcleos a 1.43GHz, memoria ram de 4gb, un bajo consumo de energía de 10 Watts y sistema operativo basado en Linux. Todas estas características la vuelven ideal para desarrollar sistemas de IA potentes y eficientes a un bajo consumo (Ulatas, Sahin, & Ozbay, 2023).

1.5.2 Raspberry Pi 4

Siendo una SBC, Single-Board Computer, una Raspberry Pi es una placa de desarrollo robusta con una potencia comparada al de un computador modesto. Cuenta con un procesador de cuatro núcleos de 1.5 GHz y 4gb de Ram, lo que le da la potencia suficiente para ejecutar modelos de IA entrenados para tareas complejas como el procesamiento de imágenes y el aprendizaje automático. Esto sumado a su facilidad y compatibilidad con librerías de inteligencia artificial como TensorFlow, PyTorch y OpenCV la vuelve una opción muy viable para trabajar con modelos entrenados (Raspberry Pi Foundation, 2023).

1.6 Trabajos Relacionados

El método revisado de (Phan, Trieu, & Phan, 2023), aborda las limitaciones de los sistemas de detección de somnolencia existentes al considerar múltiples características de fatiga más allá de los rasgos faciales, como la inclinación de la cabeza, el parpadeo y el bostezo. Se uso redes neuronales profundas basadas en LSTM, VGG16, InceptionV3 y DenseNet, logrando una alta precisión del 98% en escenarios generales. Además, el sistema incorpora un modelo de detección de emociones para mejorar aún más la precisión y reducir los costos computacionales. Cuando el nivel de somnolencia del conductor alcanza un umbral crítico, el sistema activa una alerta para advertir al conductor. El sistema tiene complicaciones cuando se trata de detectar estados de

somnolencia con mascarillas y especialmente con gafas, pues en este escenario la precisión baja hasta el 83%.

El artículo de (Bekhouche, Ruichek, & Drnaika, 2022) propone un nuevo marco para la detección de la somnolencia del conductor utilizando una selección híbrida de características profundas. Este marco de trabajo consta de cuatro etapas principales: detección de rostros, extracción de características profundas, agregación de características temporales y selección y clasificación de características. El modelo se evaluó con el conjunto de datos de video de detección de conductores somnolientos de NTHU-DDD (Weng, Lai, & Lai, 2016) logrando un accuracy del 87% en general, pero con un 83% en el escenario de conductores con lentes, en contra parte en conductores con lentes nocturnos se obtuvo un 90% de accuracy. Como último, el modelo tiene limitaciones en cuanto a visiones end-to-end de tipo 2D y 3D.

Los autores (Moujahid, Dornaika, & Carreras, 2021) proponen un nuevo descriptor facial compacto para la detección de la somnolencia del conductor. Este método comienza con la detección y alineación de rostros utilizando el algoritmo ERT. A continuación, la imagen facial se representa mediante una representación piramidal multiescala, y las características se extraen de cada bloque utilizando un enfoque multiescala multinivel. Finalmente, el conductor se clasifica como somnoliento o no somnoliento utilizando un clasificador SVM. El marco propuesto se evaluó en el conjunto de datos de detección de conductores somnolientos de NTHU-DDD (Weng, Lai, & Lai, 2016) logrando un 80% de accuracy general, en escenarios específicos alcanzo un 78% en conductores con lentes y un 74% en lentes nocturnos. Su principal contra es que todas las pruebas se realizaron en un ambiente controlado y no en tiempo real.

Este artículo (Bezerra, Chagas, Marques, & Didier, 2020) presenta un sistema de detección de somnolencia en tiempo real que utiliza la relación de aspecto del ojo (EAR) y un modelo de perceptrón multicapa (MLP). El sistema detecta los ojos de los usuarios en los flujos de video, calcula sus EAR (la relación entre el ancho y la altura del ojo), los clasifica en estados de parpadeo y utiliza el modelo de aprendizaje automático MLP para detectar la somnolencia basándose en la secuencia de EAR. Este sistema alcanzó una precisión de prueba del 94,9% usando el conjunto de datos DROZY (Massoz, Langohr, Francois, & Verly, 2016) el cual contiene datos con diversas características, como el género, vello facial y el uso o no de gafas. Sin embargo, al usarlo en tiempo real caídas de los fotogramas en los videos puede afectar el rendimiento.

Los autores (Almazaroi, Alqarni, Aslam, & Shah, 2023) proponen un sistema basado en CNN para detectar la somnolencia del conductor, bostezos, cinturón de seguridad y distracciones. Primero detecta la cara del conductor utilizando el predictor de formas de Dlib. Luego, extrae características faciales como los ojos, la boca y la posición de la cabeza utilizando OpenCV. Estas características se alimentan a un modelo CNN para clasificar el estado del conductor como somnoliento, bostezando, con cinturón de seguridad o distraído. El sistema logró una precisión del 97%. Sin embargo, el sistema no ha sido probado en condiciones de conducción reales. También es posible que el sistema se vea afectado por factores como la iluminación y las condiciones de la carretera.

El artículo de (Civik & Yuzgec, 2023) propone un modelo basado en CNN para la detección de fatiga de conductores. Primero detecta la región facial en el fotograma de vídeo capturado por la cámara. Posteriormente, se extraen las regiones locales de los ojos y la boca. El sistema utiliza dos modelos de aprendizaje profundo para detectar rápidamente los estados de los ojos y la boca en tiempo real. El sistema se entrenó y probó en el conjunto de datos YawDD (Abtahi, Omidyeganeh, & Hariri, 2020) teniendo condiciones variables de luz y ejemplos con gafas, alcanzó una precisión del 93,6% y del 94,5% para los modelos de ojos y boca, respectivamente. Sin embargo, para obtener buenos resultados se necesita tener buena iluminación además de que la postura no está tomada en cuenta.

Los autores (Hou, Wang, Zhao, Ni, & Kong, 2021) proponen un modelo basado en MobileNetV2 para la detección de anomalías en conductores con las siguientes características: movimiento anormal, detección de cubrebocas, detección de gafas y somnolencia. Primero usa un algoritmo EDVR para incrementar la calidad de videos, luego usa un modelo TSN dual-stream para detectar movimientos anómalos y finalmente los combina con Eye Aspect Ratio EAR para la detección de fatiga junto con MobileNetV2. Usa diversos datasets como WIDER FACE, FDDB (Jain & Miller, 2010) dataset y videos recolectados por los autores los cuales logran un accuracy del 95% y comprueban que la restauración de video usando EDVR eleva el accuracy hasta el 98%. Como limitaciones se destaca que usar un dataset más grande puede requerir una mayor optimización.

El artículo de (Hossain, Rahman, Islam, & Akhter, 2022) propone un modelo de detección de distracciones en conductores usando un modelo CNN con transfer learning de VGG-16,

SimpleCNN, ResNet50 y MobileNetV2. Los autores hacen uso del dataset Farm Distracted Detection competition de Kaggle (Montoya, Holman, Smith, & Kan, 2016), el cual contiene 6000 imágenes de conducción normal, usando el teléfono, tomando café y maquillándose. Luego de ajustar los parámetros y preprocesar las imágenes se procede a entrenar las 4 redes antes mencionadas. Los resultados muestran que MobileNet tiene un rendimiento superior al resto de arquitecturas teniendo un promedio de 98% de acurracy en cada escenario, exceptuando en la del café donde obtiene un 79%. Sin embargo, el modelo no ha sido probado en tiempo real, por lo tanto, se requiere mayores pruebas para comprobar su efectividad en la vida real.

CAPÍTULO 2: DESARROLLO

El presente trabajo propone implementar un sistema de detección de somnolencia en tiempo real, basado en una red neuronal convolucional MobileNetV3 Small/Large, ejecutable en plataformas embebidas como Jetson Nano o Raspberry Pi 4. Siguiendo la metodología KDD, el desarrollo se enfocará en cumplir con los objetivos planteados anteriormente siguiendo a cabo un proceso de desarrollo que incluirá la adquisición y preprocesamiento de datos, la extracción de características relevantes mediante la detección de landmarks faciales y el cálculo de la relación aspecto de los ojos (EAR), y el entrenamiento de un modelo de clasificación. Se definirán regiones de interés (ROI) para focalizar el análisis en las zonas faciales más informativas. El proyecto está planificado con una duración de 6 meses.

2.1 Selección de la arquitectura MobileNetV3

La elección de MobileNetV3 como arquitectura base para el modelo de detección de somnolencia en conductores fue motivada por su balance entre eficiencia computacional y rendimiento predictivo. A diferencia de sus predecesores, MobileNetV1 y MobileNetV2, la versión V3 introduce optimizaciones significativas en la estructura del modelo, incorporando bloques Sandglass y la activación h-swish, lo que permite una mejor representación de patrones complejos sin incrementar considerablemente el costo computacional (Howard, y otros, 2019).

En comparación con MobileNetV1, MobileNetV3 logra una mayor precisión al integrar una búsqueda neural automática para optimizar la arquitectura del modelo. Mientras que MobileNetV1 se basa exclusivamente en convoluciones separables para reducir la complejidad, MobileNetV3 complementa esta estrategia con el uso de capas squeeze-and-excitation, mejorando la capacidad del modelo para adaptarse a variaciones en los datos de entrada. Además, respecto a MobileNetV2, MobileNetV3 incorpora modificaciones que reducen la redundancia en las conexiones entre capas, lo que resulta en un modelo más ligero y rápido en términos de inferencia.

Además, MobileNetV3 cuenta con dos variantes: Small y Large. La versión Small está diseñada como una alternativa extremadamente liviana, ideal para ejecutarse en dispositivos con recursos aún más limitados que los considerados en el resto de la familia MobileNet. Esta variante incorpora un menor número de parámetros y capas más ligeras, lo que reduce significativamente el tiempo de inferencia y el consumo de memoria. Su diseño prioriza la velocidad y la eficiencia,

aunque con una ligera reducción en su capacidad para reconocer patrones complejos, haciéndola adecuada para tareas menos exigentes en términos de precisión (Andrew & Suyog, 2019).

Por otro lado, la versión Large es más grande y compleja, incorporando un mayor número de parámetros y capas más profundas. Está optimizada para tareas que requieren una mayor precisión y capacidad de modelado, permitiéndole capturar patrones más detallados en los datos. A pesar de su complejidad, MobileNetV3 Large conserva el enfoque de eficiencia característico de esta arquitectura, destacándose frente a otros modelos más pesados en términos de rendimiento computacional y consumo energético (Andrew & Suyog, 2019).

Dado que ambas variantes están diseñadas para ser altamente eficientes en dispositivos embebidos, se decidió entrenar el modelo con las dos versiones para identificar el mejor equilibrio entre precisión y eficiencia.

2.2 Selección de Dataset

2.2.1 Búsqueda de datasets

Para garantizar la calidad y la variedad en las imágenes, como se observa en la tabla 3, se optó por hacer un recopilado de distintos dataset etiquetados con landmarks con el fin de obtener una amplia gama de condiciones de iluminación, expresiones faciales, ángulos de cámara y condiciones lumínicas. Además, se recopilaron imágenes de autoría propia en conducción real donde se hizo énfasis en conductores que lleven lentes y gafas oscuras, aspectos que usualmente bajan la detección en este tipo de modelos.

Tabla 3

Selección de distintos datasets

Dataset	Cantidad de imágenes	Ángulos	Condiciones lumínicas	Expresiones faciales	Uso de lentes oscuros
300VW	2442	Variados	Variado	Variado	Limitado
300W	1547	Limitado	Limitado	Variado	Limitado
Helen	2330	Variado	Variado	Variado	Limitado
AFW	337	Variado	Variado	Variado	Variado
CEW	560	Limitado	Variado	Limitado	Limitado

IBUG	185	Variado	Variado	Variado	Limitado
FRGC	4950	Limitado	Limitado	Variado	Limitado
IFPW	224	Variado	Variado	Variado	Limitado
Menpo2D	1777	Variado	Variado	Variado	Limitado

Nota: Autoría propia

De todo este recopilado de imágenes se seleccionaron muestras para formar un dataset final el cual abarca una amplia gama de condiciones, incluyendo ángulos, expresiones faciales, iluminación, y el uso de lentes. Esta diversidad garantiza que el modelo sea capaz de generalizar a una variedad de situaciones de conducción y evita que se creen sesgos al momento de entrenar.

2.2.2 Recolección de imágenes

En cuanto a las imágenes recolectadas, como se puede ver en la figura 9, se priorizó capturarlas en condiciones de conducción en tiempo real y con iluminación variable, lo que permite al modelo aprender a manejar situaciones tanto de luz óptima, como las mañanas y las tardes, como de baja iluminación, como ocurre en la noche. Esto dado que las condiciones de iluminación pueden afectar significativamente la detección de los conductores en las imágenes y, por tanto, la precisión del modelo. Además, se simularon escenarios que representan casos complejos, como la conducción con gafas o lentes de sol.

Figura 9

Ejemplos de imágenes con somnolencia recolectada

Nota: En la figura se muestran ejemplos de imágenes somnolientas recolectadas en distintos escenarios. Autoría Propia

Para obtener imágenes de alta calidad bajo condiciones de baja iluminación, se utilizó una cámara infrarroja conectada a una Raspberry Pi. Las cámaras infrarrojas son particularmente efectivas en ambientes oscuros, ya que no dependen de la luz visible y pueden captar detalles precisos en escenarios con poca visibilidad. Para el presente trabajo, como se muestra en la Tabla 4, se utilizaron diversas herramientas que facilitaron la captura de imágenes en las distintas condiciones previamente mencionadas.

Tabla 4

Herramientas usadas para la captura de imágenes

Herramienta	Descripción
Celular	Usado para la captura de videos e imágenes en distintos ángulos y como punto hotspot
Punto Hotspot	Usado para establecer conexión a través de la red
RealVNC Viewer	Aplicación usada para acceder remotamente a la raspberry pi
Portátil	Usado en conjunto con RealVNC Viewer
Módulo de cámara NIR modelo OV5647	Módulo de cámara infrarrojo para la captura de imágenes tanto de día como de noche
Raspberry Pi 4	Minicomputadora en la que está instalado el módulo de cámara
Cable con conectores tipo USB-A/USB-C	Usado para conectar la raspberry pi con una pantalla en miniatura
Conexión de 5V para vehículos	Usado para alimentar la raspberry pi dentro del vehículo
Pantalla de 2.8 pulgadas	Pantalla de la raspberry pi

Nota: Autoría propia

El proceso de adquisición de imágenes se llevó a cabo mediante dos metodologías distintas. El primero se empleó la cámara del teléfono como fuente para la grabación de los videos. Mientras que en el segundo se utilizó un módulo de cámara infrarrojo integrado en una raspberry pi, siendo

esta última la principal forma de grabación. Ambas sesiones de adquisición de imágenes se caracterizaron por los siguientes parámetros:

Accesorios: Cada sujeto fue grabado bajo tres condiciones de uso de accesorios: sin accesorios (cara descubierta), con gafas y con gafas de sol, denominadas en adelante como bareface, glasses y sunglasses respectivamente. Se tuvo especial cuidado en seleccionar gafas de sol que no fueran completamente opacas, asegurando un nivel de transparencia suficiente para permitir la visualización de los ojos.

Duración: Las sesiones de grabación tuvieron una duración de entre uno y un minuto y medio, utilizando una velocidad de captura de 25 fotogramas por segundo. Se solicitó a los sujetos que simularan situaciones de bostezo, somnolencia general y movimientos de cabeza en diversas direcciones, con el objetivo de obtener una muestra variada de expresiones faciales. Las grabaciones realizadas con el módulo de cámara infrarroja se llevaron a cabo a una resolución de 1920x1080 píxeles, mientras que las grabaciones con el teléfono móvil se realizaron a una resolución de 720x1280 píxeles

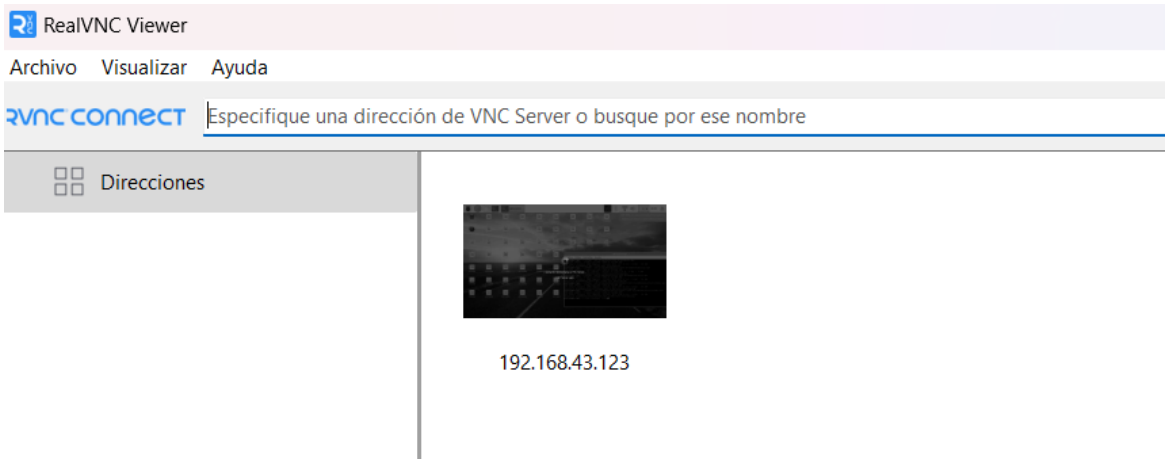
Iluminación: Se implementaron dos condiciones de iluminación principales: grabaciones diurnas, realizadas entre las 15:00 y las 17:30 horas, abarcando las tres condiciones de accesorios (bareface, glasses y sunglasses); y grabaciones nocturnas, realizadas entre las 19:15 y las 21:30 horas, abarcando las condiciones bareface y glasses. Para las grabaciones nocturnas, el uso de la cámara infrarroja fue fundamental debido a las condiciones de baja luminosidad ambiental mientras que durante el día se usaron ambas cámaras, la infrarroja y mediante el teléfono.

Debido a las limitaciones de tamaño de la pantalla de 2.8 pulgadas de la Raspberry Pi, se optó por utilizar una computadora portátil para controlar remotamente el dispositivo durante la captura de video NIR. Esto se realizó mediante el protocolo Virtual Network Computing (VNC).

Para establecer la conexión remota, se utilizó la funcionalidad de hotspot del teléfono móvil para crear una red entre la computadora portátil y la Raspberry Pi. Una vez establecida la red, se utilizó la aplicación RealVNC Viewer en la computadora portátil para conectarse a la Raspberry Pi a través de su dirección IP, como se muestra en la Figura 10.

Figura 10

Panel de control remoto de RealVNC Viewer



Nota: Autoría Propia

Las grabaciones iniciales se almacenaron en formato H.264 debido a su configuración predeterminada. Sin embargo, para facilitar el procesamiento y la manipulación de los videos, se realizó una conversión a un formato estándar más compatible, específicamente MP4.

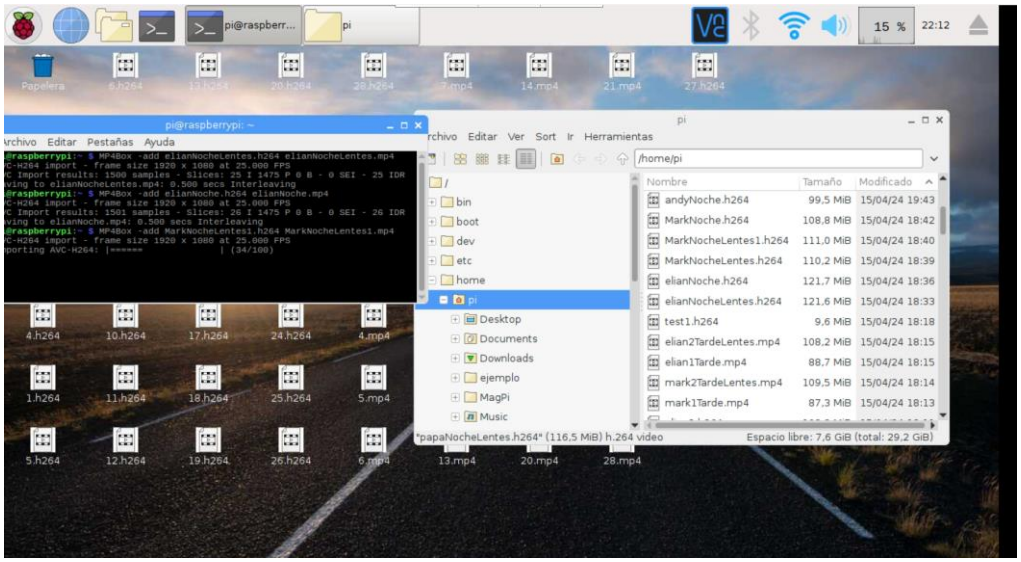
Este proceso de transcodificación se llevó a cabo utilizando la herramienta MP4Box, que se encontraba preinstalada en la Raspberry Pi. Se ejecutó el siguiente comando en la línea de comandos:

'MP4Box -add nombreArchivo.h264 nombreArchivo.mp4'

Cómo se muestra en la figura 11, la opción -add especifica la adición de una nueva pista al archivo contenedor, en este caso, se toma el archivo de video codificado en formato H.264 (nombreArchivo.h264) y se empaqueta dentro del contenedor MP4, creando el archivo nombreArchivo.mp4.

Figura 11

Transformación de formato mediante MP4Box

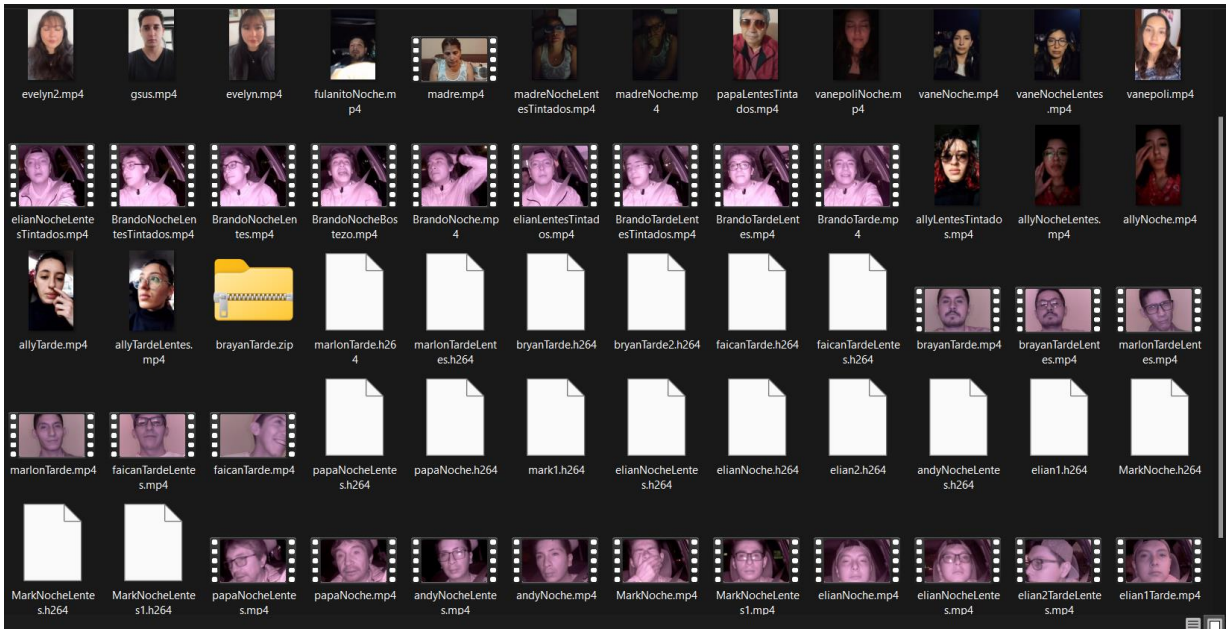


Nota: Autoría Propia

En la figura 12, se pueden apreciar los videos realizados utilizando todas las herramientas y técnicas mencionadas

Figura 12

Recolección de imágenes en crudo



Nota: Autoría Propia

2.3 Preprocesado

2.3.1 Data Cruda

Una vez finalizada la adquisición de video, definida como data cruda, se procedió a la selección y limpieza de las imágenes relevantes para el presente estudio. Para ello, se implementó un script en Python utilizando la librería OpenCV, como se ve en la Figura 13, para extraer fotogramas clave de cada video. El script procesa la data cruda y genera archivos JPG conteniendo los fotogramas extraídos. Para optimizar el proceso y evitar la redundancia de datos, el script solo guarda una selección de fotogramas por segundo, en lugar de los 25. Evitando llenarnos de datos que inevitablemente terminarán como basura digital.

Figura 13
Captura de fotogramas de la data cruda

```
    # Calcula la cantidad de fotogramas a saltar para obtener 8 fotogramas por segundo
    frame_skip = int(round(fps / frame_rate))

    # Lee y guarda los fotogramas
    count = 0
    success = True
    while success:
        success, image = video_capture.read()
        if count % frame_skip == 0:
            if success:
                # Guarda el fotograma en el directorio de salida
                frame_name = f"{os.path.splitext(os.path.basename(video_path))[0]}_{count // frame_skip + 1}.jpg"
                output_path = os.path.join(output_dir, frame_name)
                cv2.imwrite(output_path, image)
        count += 1

    video_capture.release()

if __name__ == "__main__":
    # Ruta del video de entrada
    video_path = "../Propios/alexLentesTintados.mp4"

    # Directorio de salida para los fotogramas
    output_directory = "fotogramas"

    # Crea el directorio de salida si no existe
    os.makedirs(output_directory, exist_ok=True)

    # Extrae los fotogramas del video
    extract_frames(video_path, output_directory)
```

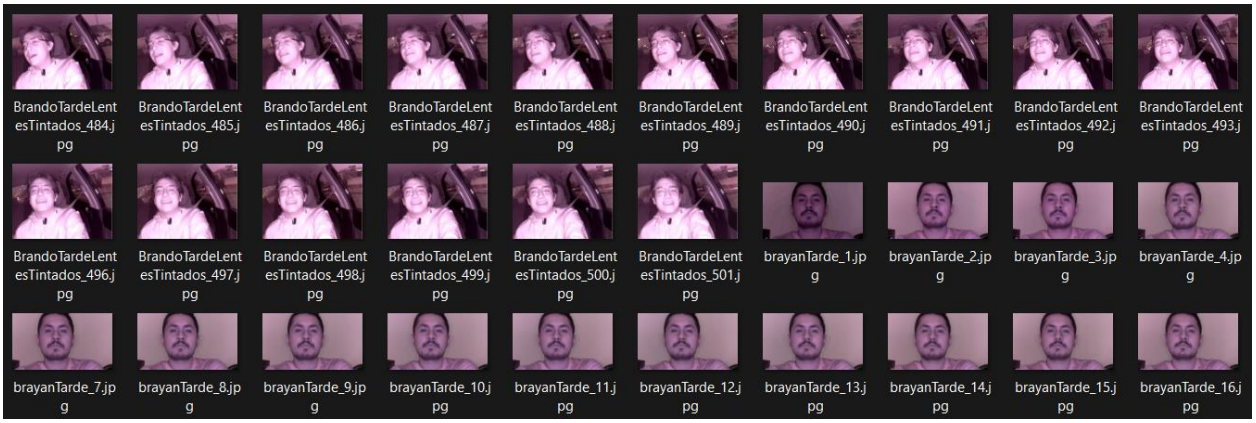
Nota: Autoría propia

El algoritmo implementado utiliza dos variables principales: count y frame_skip. La variable count realiza un seguimiento del número de fotograma actual que se lee del video,

mientras que frame_skip determina el intervalo de fotogramas que se omiten antes de guardar uno. El script verifica si el número de fotograma actual (count) es divisible por frame_skip. Si la condición se cumple, el fotograma se considera relevante y se procede a su almacenamiento. Antes de guardar el fotograma, se realiza una comprobación para asegurar que la lectura del fotograma se haya realizado correctamente. Finalmente, se construye el nombre del archivo JPG combinando el nombre del video y un número de secuencia, y como se puede ver en la figura 14 se guarda el fotograma en el directorio especificado. Con este método de cada video obtenemos alrededor de 500 imágenes.

Figura 14

Muestra de los fotogramas obtenidos de la data cruda



Nota: Autoría Propia

Terminado la transformación de la data cruda el siguiente paso es la selección de imágenes, el criterio de selección se basó en tres factores principales.

Claridad: Se seleccionaron imágenes que presentaran una buena nitidez y resolución, minimizando la presencia de errores o distorsiones que pudieran afectar la calidad del análisis.

Diversidad de condiciones: Se buscó una muestra diversa de imágenes que abarcara diferentes condiciones de iluminación (diurna y nocturna) y de los estados bareface, glasses y sunglasses.

Variedad de expresiones faciales: Se seleccionaron imágenes que representaran una amplia gama de expresiones faciales, incluyendo ojos cerrados, entrecerrados, mirada dirigida en distintas

direcciones y ángulos, con el fin de evitar sesgos y sobreajustes del modelo al momento de aprender las expresiones faciales.

Se procedió manualmente a elegir las mejores fotografías para conformar el dataset, para evitar la repetición de rostros, el cual podría provocar otro sesgo al momento de entrenar, se decidió elegir una cierta cantidad de fotogramas por cada sujeto de pruebas como se puede ver en la tabla 5.

Tabla 5
Detalle de las imágenes por sujeto de prueba

CONDICIÓN	HORA DEL DÍA	OJOS ABIERTOS	OJOS CERRADOS	TOTAL
SOLO EL ROSTRO	Día	2	2	4
CON LENTES	Día	2	2	4
CON GAFAS DE SOL	Día	2	2	4
SOLO EL ROSTRO	Noche	2	2	4
CON LENTES	Noche	2	2	4
CON GAFAS DE SOL	Noche	2	2	4
TOTAL, POR PERSONA		**12**	**12**	**24**

Nota: Autoría Propia

2.3.2 Etiquetado de los Landmarks faciales

Teniendo en cuenta el problema de detección facial, se optó por utilizar dos enfoques, clasificación de estados el Eye Aspect Ratio EAR.

Clasificación: Este enfoque se centra en la clasificación del estado de los ojos como indicador de somnolencia. Se entrena un modelo de aprendizaje automático para clasificar imágenes en dos categorías: drowsy (clase 0) y non-drowsy (clase 1).

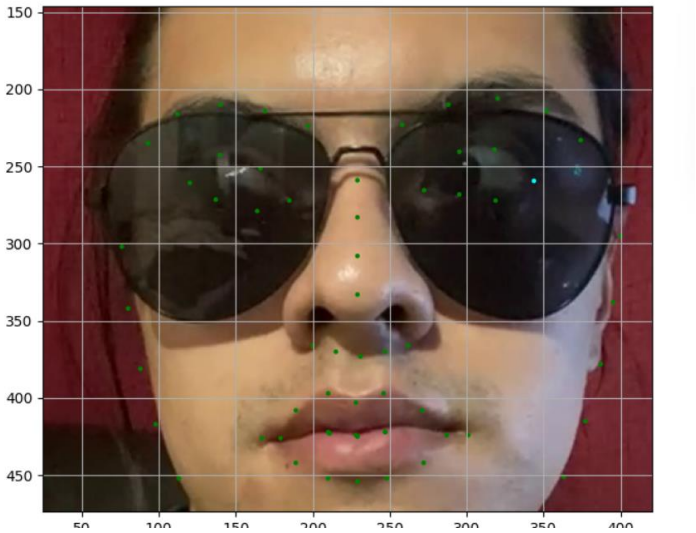
Posterior la arquitectura MobileNetV3 recibe como entrada imágenes etiquetadas y aprende a asociar características visuales con cada clase. Durante la inferencia, el modelo produce una clasificación binaria, donde los valores más cercanos a 0 indican la clasificación del estado como drowsy, y los valores más cercanos a 1 indican el estado non-drowsy.

Eye Aspect Ratio EAR: La métrica EAR se emplea para monitorear el grado de apertura ocular. El modelo se entrena como un problema de regresión, donde la salida consiste en 12 coordenadas que representan los puntos de los ojos (6 por ojo). A partir de estas coordenadas, se calcula el EAR para cada ojo. Además, se utiliza la métrica PERCLOS siglas para Percentage of Eye Closure, que determina el porcentaje de tiempo en que los ojos permanecen cerrados a lo largo de una secuencia de imágenes, lo que permite detectar signos de somnolencia (Pradeep, Namratha, Tellis, & Mayengbam, 2023).

Usando un script de Python obtendremos los landmarks faciales. El código utiliza la biblioteca dlib para detectar rostros y extraer los 68 puntos de faciales en imágenes contenidas en un directorio. Para ello, emplea un modelo preentrenado de predicción de landmarks, shape_predictor_68_face_landmarks.dat. Para cada imagen en la carpeta especificada, se detectan los rostros presentes y, si se encuentra al menos uno, se extraen sus 68 puntos. Como se observa en la figura 14.

Figura 14

Preprocesado de los Landmarks Faciales

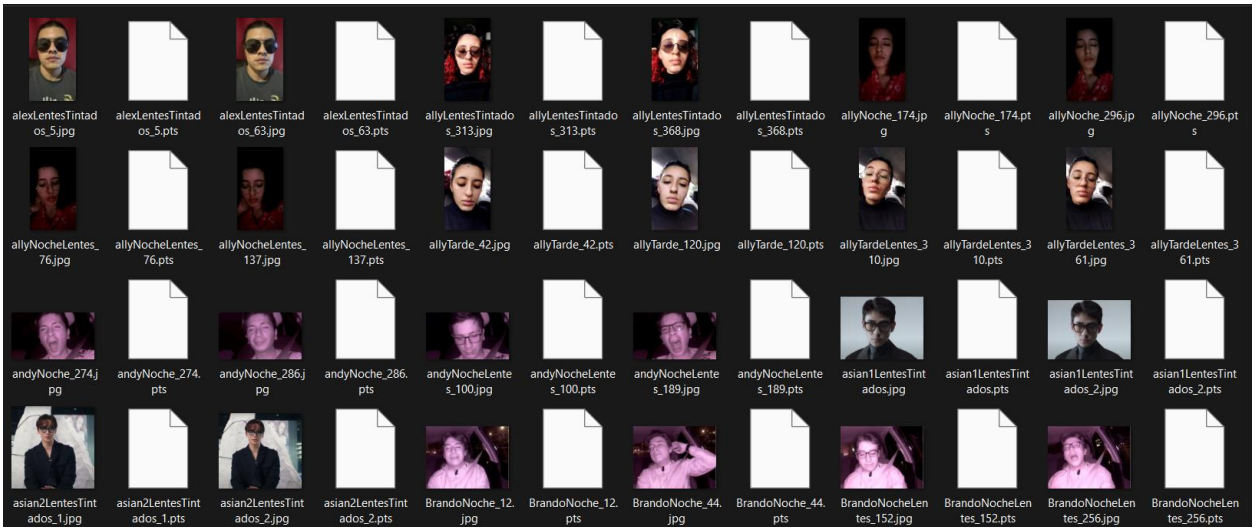


Nota: En la figura se muestran los 68 landmarks faciales correctamente ubicados en una imagen recolectada. Autoría Propia

Una vez obtenidos los landmarks de la primera cara detectada, estos se guardan en un archivo pts con el mismo nombre que la imagen de origen. Este archivo sigue un formato estructurado, incluyendo la versión, el número de puntos y sus coordenadas en el plano de la imagen. El script itera sobre todas las imágenes del directorio especificado, asegurándose de procesar únicamente archivos con extensión .jpg o .png, generando así un archivo pts correspondiente para cada imagen procesada como se puede apreciar en la figura 15.

Figura 15

Carpeta contenedora con todas las imágenes y sus respectivos landmarks



Nota: Autoría Propia

El archivo pts sigue un formato estándar, usado en los demás datasets con este enfoque. Como se puede ver en la figura 16 comienza con la línea version: 1, que indica la versión del formato. Luego, n_points: 68 especifica que el archivo contiene 68 puntos de referencia. A continuación, dentro de llaves {}, se listan las coordenadas (x, y) de cada punto en pixeles, una por línea, donde la primera columna representa la posición horizontal (x) y la segunda la posición vertical (y). Estos puntos siguen un orden predefinido que describe la estructura facial, incluyendo el contorno del rostro, cejas, ojos, nariz y boca, lo que permite su uso en tareas como detección de expresiones, seguimiento facial o análisis de somnolencia.

Figura 16
Estructura del archivo pts

```
version: 1
n_points: 68
{
137 196
141 217
145 237
151 256
160 273
173 287
190 297
209 304
230 303
251 298
267 287
281 274
290 257
295 239
298 218
```

Nota: Autoría Propia

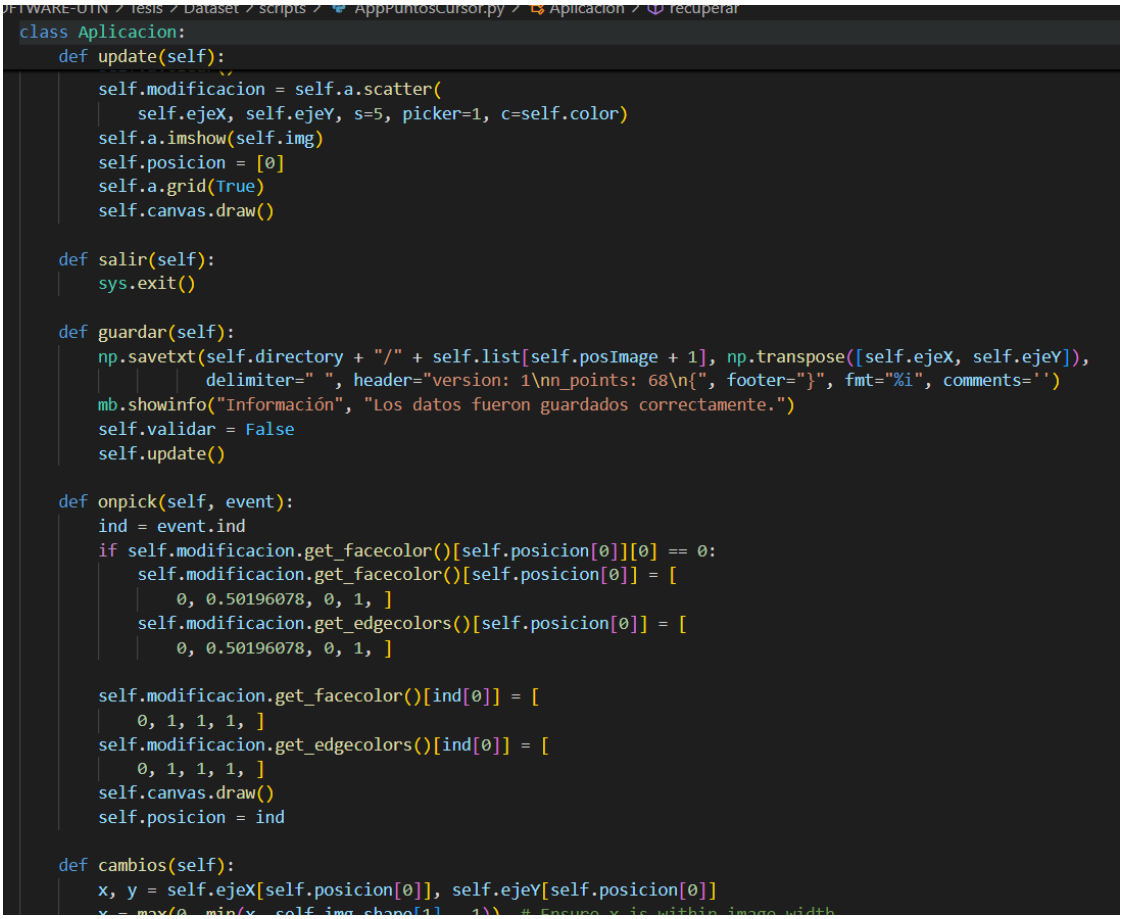
Para garantizar la integridad de las etiquetas y minimizar errores en la anotación, se implementó un script en Python que permitió manipular visualmente los landmarks detectados y ajustarlos manualmente en los casos donde la detección automática no fue completamente precisa. De este modo, se logró eliminar datos incorrectos que podrían haber interferido negativamente en el proceso de entrenamiento, reduciendo el ruido en los datos y mejorando la calidad general del conjunto.

El código implementa una aplicación de escritorio en Tkinter que permite visualizar imágenes y modificar manualmente los landmarks faciales de 68 coordenadas almacenadas en archivos pts. Como se puede observar en la figura 17 la interfaz proporciona herramientas para cargar directorios de imágenes, navegar entre ellas y editar la posición de los puntos mediante interacción con el ratón y teclas de dirección. Utiliza Matplotlib para representar gráficamente la

imagen junto con los puntos de referencia, permitiendo su manipulación en tiempo real. Además, se pueden guardar los cambios en archivos pts, asegurando que la edición se registre correctamente.

Figura 17

Parte del código para la edición de los landmarks



```
class Aplicacion:
    def update(self):
        self.modificacion = self.a.scatter(
            self.ejeX, self.ejeY, s=5, picker=1, c=self.color)
        self.a.imshow(self.img)
        self.posicion = [0]
        self.a.grid(True)
        self.canvas.draw()

    def salir(self):
        sys.exit()

    def guardar(self):
        np.savetxt(self.directory + "/" + self.list[self.posImage + 1], np.transpose([self.ejeX, self.ejeY]),
                   delimiter=" ", header="version: 1\nn_points: 68\n{", footer="}", fmt="%i", comments='')
        mb.showinfo("Información", "Los datos fueron guardados correctamente.")
        self.validar = False
        self.update()

    def onpick(self, event):
        ind = event.ind
        if self.modificacion.get_facecolor()[self.posicion[0]][0] == 0:
            self.modificacion.get_facecolor()[self.posicion[0]] = [
                0, 0.50196078, 0, 1, ]
            self.modificacion.get_edgecolors()[self.posicion[0]] = [
                0, 0.50196078, 0, 1, ]

        self.modificacion.get_facecolor()[ind[0]] = [
            0, 1, 1, 1, ]
        self.modificacion.get_edgecolors()[ind[0]] = [
            0, 1, 1, 1, ]
        self.canvas.draw()
        self.posicion = ind

    def cambios(self):
        x, y = self.ejeX[self.posicion[0]], self.ejeY[self.posicion[0]]
```

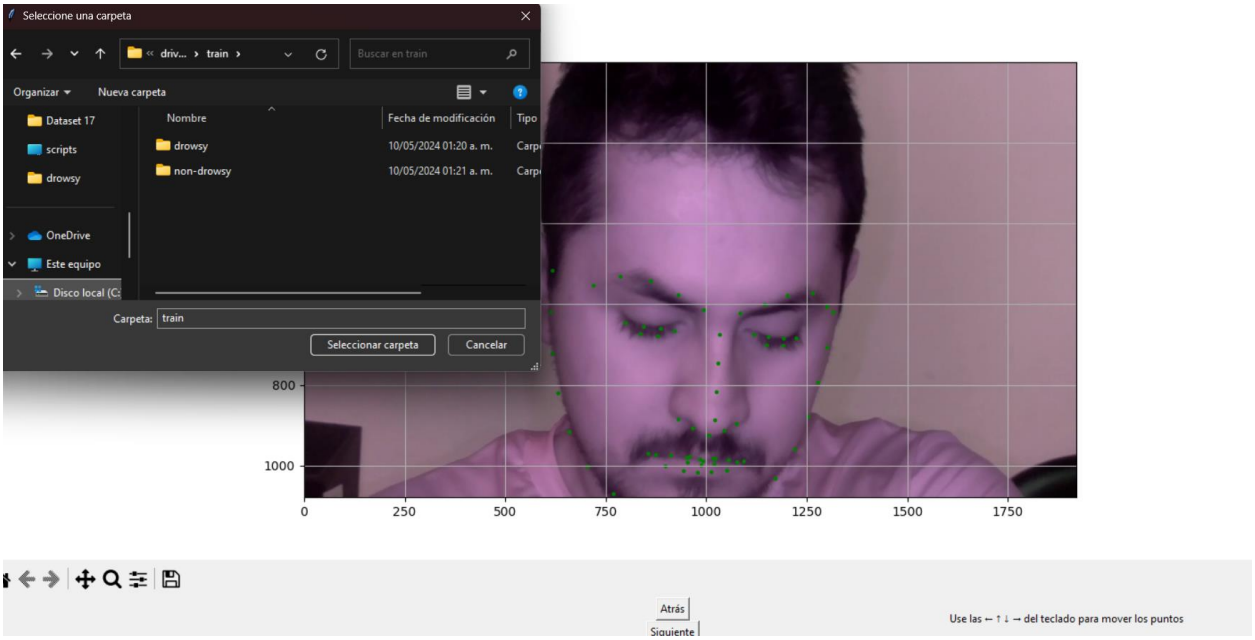
Nota: Autoría propia

El código además define la clase Aplicacion, que gestiona la ventana principal, los menús y la lógica de navegación de imágenes. Implementa métodos como openImg para cargar imágenes y sus correspondientes puntos, guardar para almacenar modificaciones y update para actualizar la visualización. También incorpora eventos de ratón y teclado (mouse_press, mouse_move, left, right, etc.) para ajustar dinámicamente la posición de los landmarks. Se incluyen controles adicionales como una barra de herramientas de Matplotlib (NavigationToolbar2Tk) y validaciones para evitar errores al recorrer imágenes o modificar puntos fuera de los límites.

La Figura 18 muestra la interfaz gráfica que permite realizar ajustes precisos a los puntos de referencia facial (landmarks). Al seleccionar una imagen o una carpeta completa, podemos manipular individualmente cada landmark utilizando el cursor o las teclas de dirección. Los cambios realizados se almacenan automáticamente en el archivo pts correspondiente. Dado que nuestro análisis se centra en la región ocular, únicamente se modificarán los landmarks asociados a los ojos.

Figura 18

Interfaz para la modificación de landmarks



Nota: Autoría Propia

2.3.3 Clasificación de las imágenes

Con los landmarks faciales detectados, se seleccionaron específicamente los puntos correspondientes a los ojos, que se utilizaron para calcular la Relación de Aspecto de los Ojos (EAR) mediante la ecuación 1, en donde los puntos P1 hasta P6 representan los puntos de los ojos, permitiendo medir su apertura o cierre. Este enfoque también será una de las métricas fundamentales empleadas durante la etapa de entrenamiento.

$$EAR = \frac{||P2-P6|| + ||P3-P5||}{2||P1-P4||} \quad (1)$$

Se estableció un umbral de EAR de 0.20 como criterio para clasificar un estado de somnolencia. Como se observa en la Figura 19, valores de EAR inferiores a este umbral indican una apertura ocular mínima, lo cual es característico de un estado de somnolencia. Este valor umbral fue determinado de manera empírica a partir de la observación en una muestra de imágenes.

Figura 19
Muestra de distintos EAR

EYE								
EAR	0.40	0.39	0.34	0.33	0.31	0.29	0.21	0.12

Nota: Tomado de (Çakar, Yildiz, & Demir, 2021).

A partir del valor de EAR y del umbral establecido, se realizó una clasificación binaria de las imágenes en dos categorías: alerta (0) y somnoliento (1). Esta decisión de etiquetado se basa en la literatura existente, donde términos como 'drowsy/non-drowsy' o 'alerta/somnoliento' son comúnmente utilizados. Un script de Python se empleó para automatizar este proceso de clasificación y agregar una etiqueta (0 o 1) a cada imagen, la cual se almacenó en el archivo pts correspondiente, como se puede ver en la figura 20.

Figura 20
Etiquetas binarias

```
142.0 196.0
136.0 196.0
130.0 195.0
123.0 191.0
122.0 182.0
130.0 175.0
135.0 175.0
141.0 176.0
157.0 182.0
141.0 186.0
135.0 186.0
129.0 185.0
}
label: 0
```

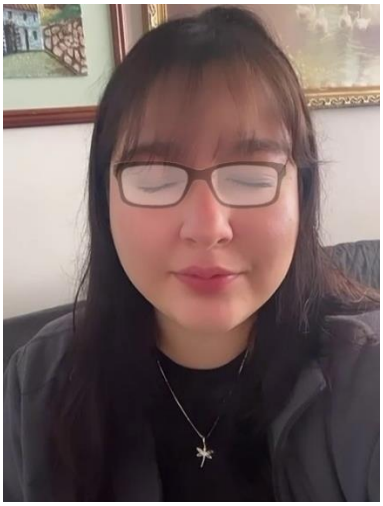
```
308.647 304.005
288.172 304.482
270.455 302.137
250.06699 288.189
237.20999 266.292
272.202 263.917
287.749 268.505
306.865 265.956
357.255 269.096
307.701 285.278
288.694 286.563
270.737 283.107
}
label: 1
```

2.3.4 Aumento de imágenes

Con el fin de mejorar la generalización del modelo y su capacidad para detectar la somnolencia en conductores que utilizan lentes, se realizó un aumento de datos específico. Se desarrolló un script en Matlab que permitió superponer de forma aleatoria imágenes de lentes con diferentes características (opacidad, reflejos) sobre las imágenes originales, como se puede observar en la figura 21. Esta técnica se la realizó con el propósito de que el modelo aprenda a discriminar entre los signos de somnolencia y las variaciones causadas por el uso de lentes, sin que aprenda patrones ocasionados por el reflejo de los lentes.

Figura 21

Aumento de fotos con lentes editados



Nota: En la figura se observa un ejemplo de la aplicación de lentes superpuestos. Autoría propia

Para cada rostro, el código calcula los puntos medios de los ojos, los cuales se utilizan como referencia para alinear las gafas. A continuación, se carga una imagen de gafas, que se rota según el ángulo calculado entre los puntos de los ojos. Posteriormente, se redimensionan las gafas de acuerdo con el tamaño del rostro, ajustando la escala en función de la distancia entre puntos clave del rostro. Después de este ajuste, las gafas se superponen sobre el rostro utilizando la transparencia de la imagen de las gafas y combinándola con la imagen del rostro mediante un proceso de interpolación.

2.4 Transformación

Para garantizar la compatibilidad con la arquitectura MobileNetV3 Small/Large, se aplicaron transformaciones específicas a las imágenes antes de ser procesadas. En primer lugar, las imágenes originales fueron redimensionadas a una resolución de 224x224 píxeles, un tamaño estándar para esta familia de arquitecturas. A continuación, se normalizaron los valores de los píxeles dividiéndolos entre 255, lo que escala los valores al rango [0, 1]. Esta normalización no solo facilita la estabilidad numérica durante el entrenamiento, sino que también mejora la convergencia de los algoritmos de optimización.

En cuanto a las etiquetas de las clases, se utilizó una estrategia basada en la conversión al formato one-hot encoding para garantizar que el modelo interprete adecuadamente las categorías. En este esquema, la etiqueta label: 0, que corresponde a "somnolencia", fue representada como el vector [1, 0], mientras que la etiqueta label: 1, correspondiente a "no somnolencia", se transformó en el vector [0, 1]. Este formato asegura la compatibilidad con la función de pérdida categorical cross-entropy, empleada durante el entrenamiento para evaluar la diferencia entre las predicciones del modelo y las etiquetas reales.

2.4.1 Dataset final

La Tabla 6 muestra la composición del dataset final, el cual fue elaborado a partir de una variedad de muestras recopiladas en la etapa 2.2.1 y con las imágenes procesadas mencionadas en los anteriores capítulos.

Tabla 6
Descripción detallada del dataset final

DATASET	**ESTADO NON-DROWSY**		**ESTADO DROWSY**	
	Numero de imágenes	Porcentaje %	Numero de imágenes	Porcentaje %
300W	201	3	409	7
300WV	0	0	2295	39
AFW	110	2	152	3
CEW	0	0	555	9
COFW	80	1	15	0
FRGC	907	15	48	1

HELEN	1396	24	693	12
IBUG	44	1	53	1
IFPW	522	9	76	1
MEMPO	687	12	366	6
XM2VTS	461	8	40	1
PROPIO	1463	25	1169	20
TOTAL	5871	100	5871	100

Fuente: Adaptado de

La estructura de datos final consta de imágenes y sus respectivos archivos pts, organizados en tres carpetas: entrenamiento, validación y prueba. Para esto se realizó una división aleatoria en donde el 80% de las imágenes corresponden al entrenamiento, validación y prueba contienen 10% de imágenes cada uno.

2.5 Minería de datos

2.5.1 Enfoques

Con el fin de obtener una visión integral del problema y maximizar el rendimiento del modelo, se optó por realizar múltiples entrenamientos. Cada entrenamiento se caracterizó por tener una combinación entre diferentes características como son:

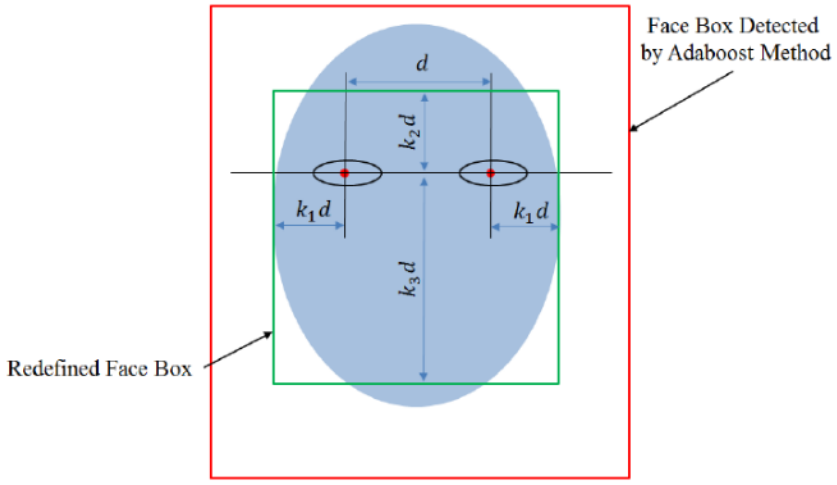
Eye Aspect Ratio: Se utilizó el EAR, una métrica comúnmente empleada para detectar el grado de apertura de los ojos. El EAR se calcula a partir de landmarks faciales alrededor de los ojos y ofrece un indicador continuo que puede diferenciar estados de ojos abiertos o cerrados (Çakar, Yildiz, & Demir, 2021). Este indicador se utilizó tanto como una referencia para la clasificación inicial de las muestras como para validar los resultados de las predicciones del modelo. Además, al ser un indicador geométrico, complementó las características aprendidas por el modelo.

Áreas de Interés (ROI): Con el objetivo de optimizar el entrenamiento del modelo y aumentar su capacidad de generalización, se introdujo la segmentación de imágenes basada en áreas de interés del rostro como se puede ver en la figura 10. Se definieron tres ROI: los ojos izquierdo y derecho, y el rostro completo. Estas regiones fueron extraídas de cada imagen utilizando landmarks faciales. Al centrar el entrenamiento en estas áreas, se reduce

significativamente la cantidad de datos irrelevantes, lo que conlleva una disminución del ruido y una mejora en la convergencia del modelo.

Figura 10

Áreas de interés en el rostro



Nota: Metodología de redefinición de la región de interés (ROI) de la cara basada en los resultados de la detección de los ojos y la cara. Adaptado de (Nguyen, Cho, & Park, 2015).

Arquitectura de MobileNetV3: Se seleccionó la arquitectura MobileNetV3, específicamente en sus variantes Small y Large, debido a su alta eficiencia computacional y su capacidad para ser implementada en dispositivos con recursos limitados, como los sistemas embebidos comúnmente utilizados en vehículos. Esta elección se alinea con el objetivo de desarrollar un modelo de detección de somnolencia que pueda funcionar en tiempo real y en dispositivos con hardware limitado.

Entrenamientos previos: Dado el gran tamaño del conjunto de datos de imágenes y los recursos computacionales limitados, se optó por una estrategia de entrenamiento en dos etapas. En la primera etapa, se realizaron entrenamientos previos con un subconjunto aleatorio de 2300 imágenes para identificar rápidamente una configuración inicial de hiper parámetros prometedora. Esta estrategia permitió ahorrar tiempo y recursos computacionales, y garantizar que los entrenamientos posteriores con el conjunto de datos completo fueran más eficientes y tuvieran mayores probabilidades de éxito.

2.5.2 Plataforma de entrenamiento

Una vez finalizado el preprocesamiento de los datos se inició la fase de entrenamiento del modelo. Para esta etapa, se eligió la plataforma de Kaggle, aprovechando su entorno de desarrollo colaborativo y su acceso gratuito a recursos computacionales de alto rendimiento, como se observa en la tabla 7. Estos recursos facilitaron la ejecución de los experimentos y también aceleraron significativamente el entrenamiento, permitiendo probar diferentes configuraciones del modelo de manera eficiente. Adicional a esto, el uso de Kaggle proporcionó un entorno controlado para el manejo y almacenamiento de datos, asegurando que los procesos fueran reproducibles y escalables.

Tabla 7

Especificaciones del entorno virtual de Kaggle

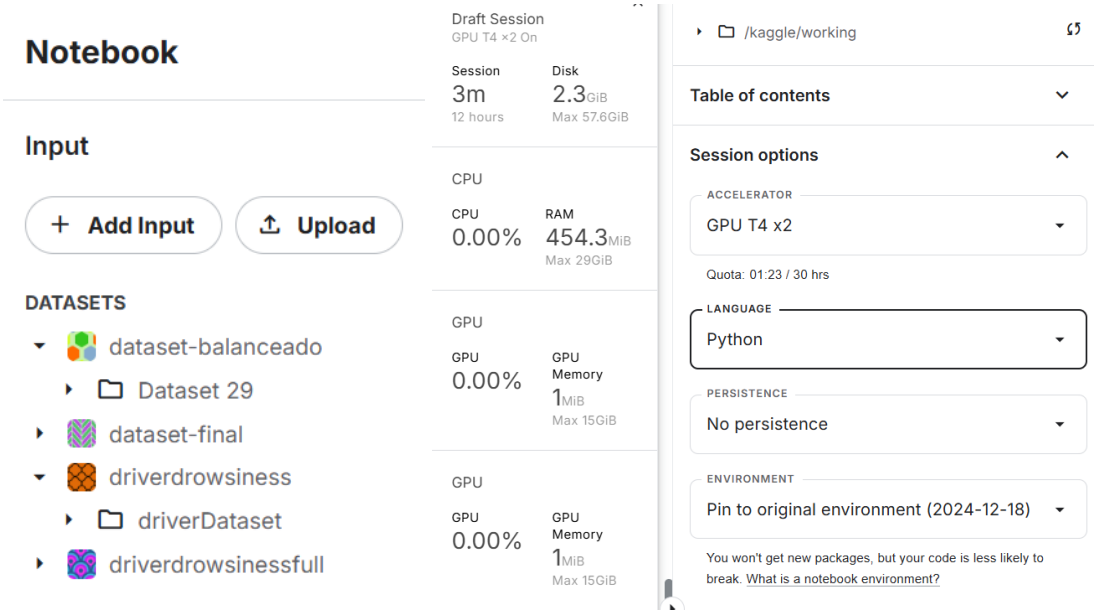
Característica	**Valor**	**Notas**
Tiempo de ejecución	12 horas (CPU/GPU), 9 horas (TPU)	Tiempo máximo para la ejecución de una sesión en el notebook.
Almacenamiento	20 GB (autoguardado, output) 59 GB (dedicado para archivos y data sets, input)	Espacio para código, datos y resultados dentro de /kaggle/working.
Espacio temporal	Disponible	Almacenamiento adicional no guardado automáticamente.
Notebooks	Jupyter Notebooks	Entorno interactivo para escribir y ejecutar código (Python o R) y texto (Markdown).
CPU	4 núcleos	Puede variar (Intel Skylake, Broadwell, AMD), pero las submissions son siempre en Intel Skylake.
RAM (CPU)	30 GB	Memoria disponible para tareas de CPU.

GPU (P100)	1 x Nvidia Tesla P100	GPU de alto rendimiento para acelerar el aprendizaje profundo.
RAM (P100)	29 GB	Memoria disponible para tareas aceleradas por GPU.
GPU (T4)	2 x Nvidia Tesla T4	Dos GPUs de alto rendimiento para acelerar el aprendizaje profundo.
RAM (T4)	29 GB	Memoria disponible para tareas aceleradas por GPU.
TPU	VM v3-8	Unidad de procesamiento tensorial en la nube para aprendizaje automático acelerado.
RAM (TPU)	330 GB	Alta capacidad de memoria para tareas de aprendizaje automático a gran escala.
CPU (TPU)	96 núcleos	Potente CPU para soportar operaciones de TPU.
Tiempo de inactividad	20 minutos	Las sesiones interactivas expiran si no hay actividad.
Opciones de guardado	Guardado rápido, Guardar y ejecutar todo	Guardado rápido para capturas instantáneas; Guardar y ejecutar todo para versiones reproducibles.

Nota: Adaptado de (Kaggle, 2024).

Como se observa en la Figura 22, se configuró el entorno de Kaggle seleccionando Python como lenguaje de programación y cargando los conjuntos de datos en el entorno de pruebas virtual. Para optimizar el rendimiento, se eligió la GPU T4 x2 como hardware de procesamiento, entre las opciones disponibles, que incluían una GPU P100 y una TPU VM v3-8.

Figura 22
Entorno virtual de kaggle



Nota: Autoría Propia

2.5.3 Pre-Entrenamiento

Se decidió trabajar con TensorFlow/Keras para el entrenamiento, esto debido a varios factores: la disponibilidad de una implementación pre-entrenada de MobileNetV3, la facilidad para definir y entrenar modelos complejos, y la amplia gama de herramientas y recursos disponibles en la comunidad de Keras. Además, la capacidad de TensorFlow para ejecutarse sin modificación previa en el entorno de entrenamiento de kaggle facilitó el proceso de entrenamiento.

Como se mencionó anteriormente, debido a que los recursos en Kaggle están limitados por el tiempo de sesión y el uso de la GPU, se decidió seleccionar una muestra de 2300 imágenes del conjunto de datos para realizar entrenamientos preliminares para tunear los hiperparámetros.

Antes de iniciar el entrenamiento del modelo como tal, se llevó a cabo el preprocesamiento de los datos. Como se muestra en la Figura 23, se implementó un método para leer los archivos pts, extrayendo la información de los landmarks y las etiquetas (labels), los cuales se almacenaron en arrays. Posteriormente, estos datos se asociaron con sus imágenes correspondientes para su uso en el entrenamiento:

Figura 23

Método para leer el archivo pts.

```
def read_pts_file(pts_path):
    """Leer un archivo .pts y extraer los landmarks y el label."""
    with open(pts_path, "r") as file:
        lines = file.readlines()

    # Buscar las líneas que contienen "{" y "}"
    start_idx = next(i for i, line in enumerate(lines) if "{" in line) + 1
    end_idx = next(i for i, line in enumerate(lines) if "}" in line)

    # Leer los puntos entre las líneas identificadas
    landmarks = np.array([list(map(float, line.strip().split())) for line in lines[start_idx:end_idx]])

    # Extraer la etiqueta del archivo .pts
    label_line = [line for line in lines if line.startswith("label:")]
    label = int(label_line[0].split(":")[1].strip()) if label_line else None

    return landmarks, label
```

Nota: Autoría Propia

Posteriormente, como se muestra en la Figura 24, se llevó a cabo un preprocesamiento de las imágenes. Utilizando landmarks, se calcularon los EAR y se definió una región de interés, que puede ser el rostro o las regiones de los ojos. A continuación, se redimensionaron las imágenes ROI a una resolución de 224x224 píxeles, en formato RGB, para adaptarlas a las especificaciones de entrada del modelo MobileNet.

Figura 24

Carga y procesamiento de imágenes

```
def process_image(image_path, pts_path, ear_threshold=None):
    # Cargar la imagen
    image = cv2.imread(image_path)

    # Leer los landmarks y el label
    landmarks, label_from_pts = read_pts_file(pts_path)

    # Extraer el ROI del rostro
    face_roi = crop_face(image, landmarks)

    if face_roi is None:
        raise ValueError(f"El ROI del rostro no pudo ser calculado para la imagen: {image_path}")

    # Redimensionar el ROI a 224x224
    face_resized = cv2.resize(face_roi, (224, 224))

    # Extraer los puntos de los ojos
    left_eye = landmarks[36:42]
    right_eye = landmarks[42:48]

    # Calcular EAR
    left_ear = calculate_ear(left_eye)
    right_ear = calculate_ear(right_eye)
    ear = (left_ear + right_ear) / 2.0
```

A continuación, se estableció un umbral de EAR en 0.2 para clasificar el estado de los ojos y se aplicó one-hot encoding a las etiquetas de las imágenes, transformándolas en un formato adecuado para el entrenamiento del modelo. Con estas configuraciones, se procedió a realizar los entrenamientos preliminares. Durante este proceso de experimentación, se probaron diversas técnicas:

Modificación dinámica del learning rate: Mediante la reducción gradual de su valor en intervalos específicos durante el entrenamiento, es una técnica de optimización que busca mejorar la convergencia y mitigar el sobreajuste. Usualmente al disminuir el learning rate en las últimas etapas del entrenamiento, se permite al modelo realizar ajustes más finos en los parámetros, favoreciendo la exploración de regiones más locales del espacio de parámetros y evitando así quedar atrapado en mínimos locales (Bhattbhatt, 2024).

Variación en la cantidad de capas densas: La elección del número de capas densas en una red neuronal es un factor crucial que influye directamente en su capacidad para aprender representaciones complejas y generalizar a nuevos datos. Al variar la cantidad de capas densas, se modifica la capacidad del modelo para capturar patrones de mayor o menor complejidad. Un número excesivo de capas puede conducir al sobreajuste, mientras que un número insuficiente puede limitar la capacidad del modelo para aprender las relaciones subyacentes en los datos.

Incorporación de capas de dropout: Para prevenir el sobreajuste y mejorar la generalización del modelo, se implementó la técnica de dropout. Esta técnica consiste en desactivar y activar aleatoriamente por un corto periodo de tiempo un conjunto de neuronas durante el entrenamiento, lo que obliga a la red a distribuir la información de manera más uniforme entre las neuronas restantes. En ciertos casos se pueden obtener modelos más robustos y menos propensos a memorizar el conjunto de entrenamiento (databasecamp, 2023).

Transferencia de aprendizaje: Con el fin de aprovechar los patrones aprendidos por un modelo previamente entrenado, ajustándolo a una nueva tarea mediante el congelamiento y descongelamiento de capas. Al congelar las capas inferiores, se preservan los pesos asociados a características de bajo nivel. Esto evita que el modelo "olvide" estos patrones durante el reentrenamiento. Por otro lado, al descongelar las capas superiores, se permite que el modelo adapte sus representaciones a las características específicas de la nueva tarea, mejorando así su capacidad de generalización en tareas más pesadas.

Aplicación de regularizadores L1 y L2: La regularización L1 y L2 se incorporaron a la función de pérdida del modelo para controlar la complejidad y prevenir el sobreajuste. La regularización L1, también conocida como Lasso, tiende a producir modelos dispersos, donde muchos pesos se vuelven cero, lo que puede ser útil para la selección de características. Por otro lado, la regularización L2, también conocida como Ridge, tiende a reducir el tamaño de todos los pesos, lo que mejora la estabilidad numérica del modelo y reduce el riesgo de sobreajuste (codelabsacademy, 2024).

Se llevaron a cabo múltiples entrenamientos preliminares utilizando diversas combinaciones de las técnicas mencionadas anteriormente. A partir de estos experimentos, se identificaron los hiperparámetros que ofrecieron el mejor rendimiento del modelo. En particular, un learning rate de 0.0001 demostró ser el más eficaz, favoreciendo una convergencia estable sin comprometer la capacidad del modelo para generalizar. En cuanto al número de épocas, se observó que los modelos con capas preentrenadas congeladas alcanzaban su mejor desempeño entre las épocas 10 y 15, mientras que aquellos con capas descongeladas requerían entre 15 y 25 épocas para obtener resultados óptimos, en ambos escenarios entrenar al modelo durante más épocas conducía a un estancamiento en la generalización del modelo o incluso un desaprendizaje.

En lo que respecta al uso de regularizadores L1 y L2, así como a la implementación de un learning rate variable, estos no aportaron mejoras significativas durante los entrenamientos preliminares. De hecho, en varios casos, estas técnicas no solo no favorecieron la generalización del modelo, sino que también introdujeron ruido en el proceso de aprendizaje, dificultando la convergencia hacia soluciones óptimas. La regularización excesiva mediante L1 y L2 tendió a restringir demasiado la capacidad del modelo para aprender patrones relevantes. Estos resultados sugieren que, para el presente conjunto de datos y la arquitectura utilizada, una configuración más simple sin regularización adicional favoreció un desempeño más robusto y consistente

Posteriormente, como se muestra en la Figura 25, se implementaron funciones para visualizar los resultados obtenidos mediante gráficos y tablas que representan el rendimiento del modelo a lo largo del tiempo. Para evaluar el desempeño, se utilizaron métricas como accuracy, loss, precisión, recall, MSE (Error Cuadrático Medio), MAE (Error Absoluto Medio), RMSE (Raíz del Error Cuadrático Medio) y la matriz de confusión.

Figura 25

Funciones para graficar los resultados.

```
# Función para graficar las métricas de entrenamiento
def plot_training_history(history):
    """Graficar el historial de entrenamiento y validación."""
    plt.figure(figsize=(15, 12))

    # Graficar precisión
    plt.subplot(3, 2, 1)
    plt.plot(history.history['accuracy'], label='Train Accuracy', color='blue')
    plt.plot(history.history['val_accuracy'], label='Validation Accuracy', color='orange')
    plt.xlabel('Epochs')
    plt.ylabel('Accuracy')
    plt.title('Precisión en Entrenamiento y Validación')
    plt.legend()

    # Graficar recall
    plt.subplot(3, 2, 2)
    plt.plot(history.history['recall'], label='Train Recall', color='blue')
    plt.plot(history.history['val_recall'], label='Validation Recall', color='orange')
    plt.xlabel('Epochs')
    plt.ylabel('Recall')
    plt.title('Recall en Entrenamiento y Validación')
    plt.legend()
```

Nota: Autoría Propia

2.5.4 Entrenamiento Final

Tras completar los preentrenamientos, se procedió con el entrenamiento utilizando el conjunto de datos completo. Como se detalló en el apartado 2.5.1, este proceso se centró en dos enfoques principales:

Entrenamiento con EAR: En este enfoque, se calcularon los valores de los EAR a partir de los landmarks faciales extraídos de cada imagen y se hizo uso de un umbral EAR establecido en 0.2. Estos valores EAR se integraron como características adicionales al modelo, complementando la información visual proporcionada por las imágenes. Para optimizar el rendimiento, las imágenes fueron recortadas utilizando una región de interés centrada en el rostro, asegurando que el modelo se enfocara en las áreas más relevantes para la detección de somnolencia.

Se seleccionó MobileNetV3Small como arquitectura base, congelando sus capas iniciales con los pesos de ImageNet. Los hiperparámetros completos se presentan en la Tabla 8.

Tabla 8
Hiperparámetros del entrenamiento con EAR

Aspecto	**Descripción**
Entradas	- **Imagen**: Tensor de forma (224, 224, 3) para imágenes RGB redimensionadas. - **Índice EAR**: Tensor unidimensional de forma (1,) para el valor EAR.
Red Base	MobileNetV3 Small preentrenada en ImageNet. Se excluyó la capa superior y se utilizó Global Average Pooling para reducción de dimensionalidad.
Congelación de capas	Las capas base de MobileNetV3 fueron congeladas para evitar modificaciones durante el entrenamiento.
Combinación de entradas	Las salidas de MobileNetV3 se concatenaron con el índice EAR para fusionar información visual y del comportamiento ocular.
Capas densas	Se añadieron dos capas densas de 128 neuronas, activadas con ReLU, para aprender combinaciones no lineales.
Capa de salida	Capa de salida con 2 neuronas y activación softmax, para clasificar entre dos clases: somnolencia y no somnolencia.
Compilación del modelo	- Optimizador: Adam con tasa de aprendizaje inicial de 0.0001. - Función de pérdida: Categorical Cross Entropy. - Métricas: Precisión, recall, MSE y MAE.
Etiquetado de datos	Las etiquetas se generaron utilizando un umbral EAR de 0.2: EAR inferior a 0.2 → "somnolencia" (label: 0); EAR superior a 0.2 → "no somnolencia" (label: 1).

Codificación de etiquetas	Las etiquetas se transformaron a formato one-hot encoding para ajustarse al esquema de salida del modelo.
Estrategias de entrenamiento	- EarlyStopping: Detención del entrenamiento si no hay mejoras en la validación después de 6 épocas. - ModelCheckpoint: Guardado de los mejores pesos. - ValidationLogger: Registro de predicciones en validación al final de cada época.
Tamaño de lote y épocas	- Tamaño de lote: 8. - Épocas: Máximo de 25, dependiendo de los resultados. - Tiempo de entrenamiento: Registrado para evaluar eficiencia.
Guardar el modelo	El modelo se guarda en la época con mejor desempeño en validación gracias a ModelCheckpoint.

Nota: Autoría Propia

Entrenamiento usando solo las etiquetas: En este enfoque, se emplearon las etiquetas correspondientes a cada imagen, las cuales se encuentran especificadas en la última línea de los archivos pts con el formato label: 0 o label: 1, representando los estados de somnolencia y no somnolencia, respectivamente. Estas etiquetas fueron codificadas mediante one-hot encoding para ser integradas al modelo. Adicionalmente, las imágenes fueron preprocesadas aplicando un ROI centrado en el rostro, asegurando nuevamente, que el modelo solo aprenda patrones referentes al rostro. Para este caso se eligió MobileNetV3Large con las capas descongeladas. Los hiperparámetros completos se pueden observar en la tabla 9.

Tabla 9

Hiperparámetros del entrenamiento con etiquetas

Aspecto	**Descripción**
Entradas	- Imagen: Tensor de forma (224, 224, 3) para imágenes RGB redimensionadas.

Red Base	MobileNetV3 Large preentrenada en ImageNet, excluyendo la capa superior y utilizando Global Average Pooling para reducción de dimensionalidad.
Capas adicionales	- Capas densas**:** 256, 128, y 128 neuronas activadas con ReLU. - Capa de salida: 2 neuronas con activación softmax para clasificación.
Congelación de capas	Las capas base de MobileNetV3 fueron descongeladas (trainable=True) para permitir que se actualicen durante el entrenamiento.
Compilación del modelo	- Optimizador: Adam con tasa de aprendizaje de 0.0001. - Función de pérdida: Categorical Cross Entropy. - Métricas: Precisión, Recall, MSE, MAE.
Callbacks de entrenamiento	- EarlyStopping: Detención si no hay mejoras en la validación después de 7 épocas. - ModelCheckpoint: Guardado del modelo con el mejor desempeño en validación. - ValidationLogger: Registra las predicciones de validación en un archivo CSV al final de cada época.
Tamaño de lote y épocas	- Tamaño de lote: 8. - Épocas: 35.
Entrenamiento	El modelo fue entrenado utilizando solo las imágenes como entrada y las etiquetas (one-hot encoded) como salida. Se registró el tiempo de entrenamiento.
Guardar el modelo	El mejor modelo se guarda gracias al ModelCheckpoint.
Tiempo de entrenamiento	Se registró el tiempo total de entrenamiento utilizando time.time(), y el modelo final fue guardado al finalizar el proceso.

Nota: Autoría Propia

2.6 Aplicativo en el sistema embebido

Una vez entrenados los dos modelos, se implementaron códigos en Python optimizados para su ejecución en sistemas embebidos, con el objetivo de realizar la clasificación de somnolencia en tiempo real. Para este proyecto, se empleó una Raspberry Pi 4 como plataforma principal. Las características técnicas de este dispositivo se presentan en la Tabla 10.

Tabla 10
Especificaciones Raspberry Pi 4

Característica	Descripción
Procesador	Broadcom BCM2711, quad-core Cortex-A72 (ARM v8) 64-bit SoC a 1.5GHz
Memoria RAM	4GB LPDDR4 SDRAM
GPU	VideoCore VI, capaz de decodificar H.265 (HEVC) de 4kp60 y H.264 (AVC) de 1080p60
Video	2 puertos micro-HDMI 2.0 para salida de video dual
Alimentación	Micro USB
GPIO	General Purpose Input Output de 40 pines para conectar un zumbador
Entrada de cámara	Puerto CSI para cámara
Módulo de cámara	Modelo de cámara infrarroja NIR 0V5647
Sistema Operativo	Raspberry Pi OS, sistema basado en GNU/Linux

Nota: Adaptado de (Raspberry Pi Ltd, 2024).

Adicional, las librerías y versiones principales que se utilizaron para hacer funcionar el programa en la Raspberry Pi 4 se presentan en la tabla 11

Tabla 11
Especificaciones de las versiones

Componente	Versión	Función
Sistema operativo	Raspberry Pi OS 32-bit (2021-12-02)	Base del sistema, proporciona entorno de ejecución

IDE	Geany 1.33	Editor de código para escribir y ejecutar scripts Python
Lenguaje de programación	Python 3.7.3	Lenguaje principal para la lógica de detección
Librerías	NumPy 1.21.6	Manipulación de arrays numéricos
	OpenCV 4.1.0	Procesamiento de imágenes
	TensorFlow Lite Runtime 2.11.0	Ejecución de modelos de aprendizaje profundo
	Imutils 0.5.4	Acceso a la cámara y funciones de visión por computadora
	RPi.GPIO 0.7.1	Control de los pines GPIO de la Raspberry Pi

Nota: Autoría Propia

Con el objetivo de garantizar la portabilidad y evitar conflictos de versiones, se desarrolló un ejecutable, como se ve en la Figura 26, que crea un entorno de ejecución independiente con todas las librerías requeridas.

Figura 26

Ejecutable para instalar las dependencias necesarias.

```
#!/bin/bash

# IMPORTANTE
# Correr el siguiente comando UNA SOLA VEZ
# para instalar las dependencias del proyecto: chmod +x setup_env.sh ./setup_env.sh

################################################################################################################################################################

# Nombre del entorno virtual
ENV_NAME="drowsiness_env"

# Actualizar el sistema
sudo apt-get update && sudo apt-get upgrade -y

# Instalar dependencias necesarias para Python y virtualenv
sudo apt-get install -y python3-venv python3-dev python3-pip libatlas-base-dev libjpeg-dev libtiff5-dev libjasper-dev libpng-dev libavcodec-dev
libgtk2.0-dev libgtk-3-dev libcanberra-gtk* libxvidcore-dev libx264-dev libopenblas-dev liblapack-dev libhdf5-dev libhdf5-serial-dev libqtgui4

# Crear entorno virtual
python3 -m venv $ENV_NAME

# Activar entorno virtual
source $ENV_NAME/bin/activate

# Actualizar pip dentro del entorno virtual
pip install --upgrade pip

# Instalar librerías con versiones específicas
pip install numpy==1.21.6
pip install opencv-python==4.1.0.25
pip install imutils==0.5.4
pip install RPi.GPIO==0.7.1

# Instalar TensorFlow Lite Runtime
pip install https://github.com/google-coral/pycoral/releases/download/release-feb-2023/tflite_runtime-2.11.0-cp37-cp37m-linux_armv7l.whl

# Verificación de instalaciones
echo "\nVerificando las instalaciones:"
pip show numpy opencv-python imutils RPi.GPIO tflite_runtime
```

2.6.1 Aplicativo en Python para la detección de tiempo real

Para el aplicativo final de Python se desarrollaron dos versiones, una que trabaja con el modelo con EAR y otra versión que trabaja solamente con los labels.

Versión con EAR: Se emplean bibliotecas como OpenCV para la manipulación de imágenes, dlib para la detección de rostros y landmarks faciales, y TensorFlow para cargar y utilizar un modelo previamente entrenado. Adicionalmente, se utiliza pygame para reproducir una alarma sonora en caso de detectar somnolencia, esto para el script en la computadora, en la versión final para el sistema embebido se reemplaza esto con RPi.GPIO para encender o apagar el zumbador.

Se carga el modelo MobileNetV3Small que ha sido entrenado para reconocer signos de somnolencia a partir de imágenes faciales y la métrica EAR que se calcula mediante una función que mide la relación entre las distancias verticales y horizontales de los ojos, en concordancia con la metodología empleada durante el entrenamiento. Como se ve en la figura 27 se identifica el valor EAR mínimo entre ambos ojos, considerándolo como el indicador preferencial, ya que el cierre de un solo ojo puede ser suficiente para comprometer la seguridad vial.

Figura 27
Obtención de los landmarks y cálculo del EAR

```
for face in faces:
    x, y, w, h = (face.left(), face.top(), face.width(), face.height())
    cv2.rectangle(frame, (x, y), (x + w, y + h), (255, 0, 0), 2)

    # Obtener landmarks del rostro
    landmarks = predictor(gray, face)
    left_eye = [(landmarks.part(i).x, landmarks.part(i).y) for i in range(36, 42)]
    right_eye = [(landmarks.part(i).x, landmarks.part(i).y) for i in range(42, 48)]

    # Dibujar landmarks de los ojos (más pequeños)
    for point in left_eye + right_eye:
        cv2.circle(frame, point, 1, (0, 255, 0), -1)

    # Calcular EAR de ambos ojos
    left_ear = calculate_ear(np.array(left_eye))
    right_ear = calculate_ear(np.array(right_eye))

    # Ajustar el EAR según el ojo más cerrado
    ear = min(left_ear, right_ear)
```

Nota: Autoría Propia

Se carga el detector de rostros dlib que identifica las caras en cada frame capturado desde la cámara web, y el predictor de landmarks localiza los puntos clave alrededor de los ojos. Estos landmarks se utilizan para calcular el EAR de ambos ojos y para recortar la región de interés y la redimensiona para que se ajuste al formato de entrada del modelo.

Previo a la etapa de predicción, se implementó una función para calcular el EAR promedio de un individuo en un intervalo de 3 segundos como se ve en la figura 28. Esta función se diseñó para mitigar el sesgo introducido por personas con ojos rasgados o adultos mayores, quienes tienden a presentar una menor apertura ocular promedio, lo que podría generar falsos positivos en la detección de somnolencia. Durante este período de 3 segundos, se requiere que el usuario mantenga la mirada fija en la cámara con los ojos abiertos de forma natural, sin parpadear. Se capturan y almacenan los valores de EAR en un arreglo, para posteriormente calcular el promedio mediante la función mean() de la librería NumPy. Este promedio se multiplica por un factor de 0.7 para determinar el umbral de somnolencia específico para el usuario.

Figura 28

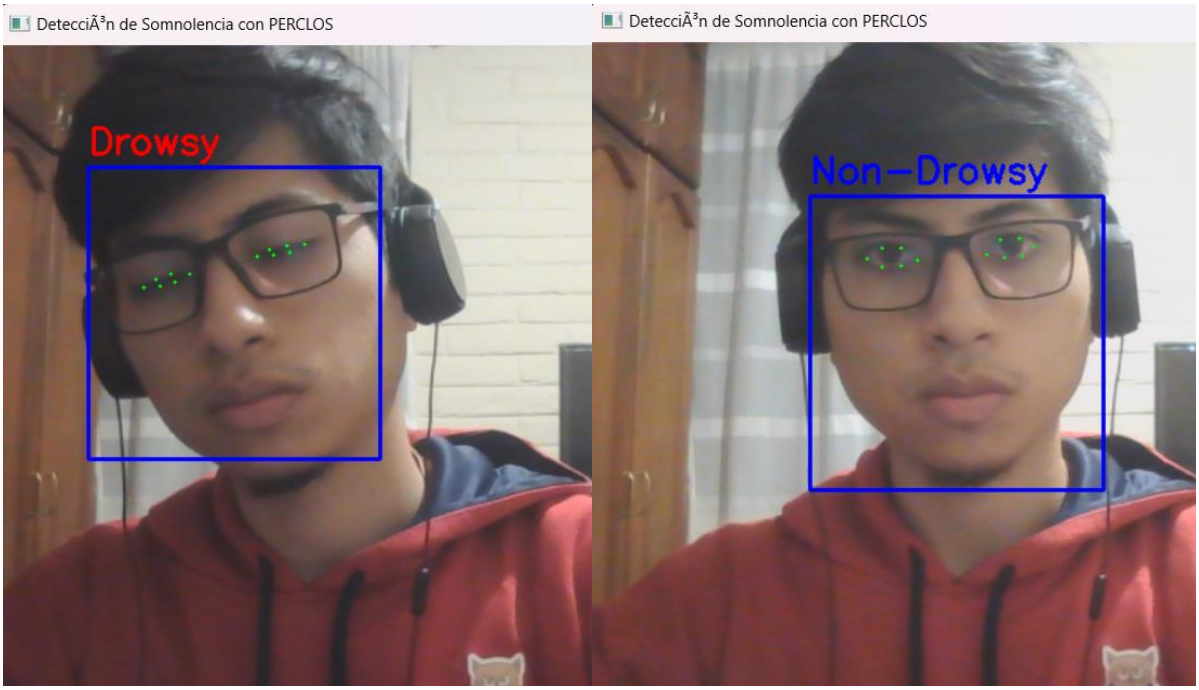
Función para calcular el EAR promedio y definir un umbral

```
def calculate_initial_ear_average(cap, detector, predictor, duration=3):
    ear_values = []
    start_time = time.time()

    while time.time() - start_time < duration:
        ret, frame = cap.read()
        if not ret:
            break

        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        faces = detector(gray)

        for face in faces:
            landmarks = predictor(gray, face)
            left_eye = [(landmarks.part(i).x, landmarks.part(i).y) for i in range(36, 42)]
            right_eye = [(landmarks.part(i).x, landmarks.part(i).y) for i in range(42, 48)]

            left_ear = calculate_ear(np.array(left_eye))
            right_ear = calculate_ear(np.array(right_eye))
            ear = min(left_ear, right_ear)

            ear_values.append(ear)

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

El modelo predice si la persona está somnolienta o no, y el resultado se muestra en la pantalla mediante un rectángulo y una etiqueta que cambia de color: rojo para "Drowsy" (somnoliento) y azul para "Non-Drowsy" (despierto) como se puede apreciar en la figura 29. Además, se implementa el cálculo del PERCLOS (Percentage of Eye Closure), este indicador mide el porcentaje de tiempo o de frames en los que los ojos de una persona están cerrados durante un intervalo de tiempo específico. Si el PERCLOS supera un umbral predefinido, el sistema activa una alarma sonora para alertar al usuario sobre la posible somnolencia. En la configuración actual, se utiliza un umbral de PERCLOS de 0.3, lo que significa que, si el 30% de los frames analizados en un intervalo de 5 segundos indican un estado de somnolencia, se activa la alarma.

Figura 29

Estados de somnolencia predichos



Nota: Autoría Propia.

Versión con etiquetas: Se utiliza OpenCV para el procesamiento de imágenes en tiempo real, numpy para operaciones matemáticas y tensorflow.keras para trabajar con un modelo MobileNetV3Large, además se definieron las etiquetas Non-Drowsy y Drowsy. Se implementó dlib para la detección del rostro y recortar la imagen a un ROI facial, acorde con el entrenamiento.

Como se ve en la figura 30 se define la función preprocess_frame(frame) que toma un frame de la cámara y lo prepara para la entrada al modelo, esto incluye el redimensionamiento del frame RGB a 224x224 píxeles y normalizarlos.

Figura 30
Función de preprocesado de imágenes

```python
def preprocess_frame(frame):
    """
    Preprocesar un frame para la entrada al modelo.
    - Cambiar tamaño a (224, 224, 3).
    - Normalizar utilizando `preprocess_input`.
    """
    frame_resized = cv2.resize(frame, (224, 224))
    frame_preprocessed = preprocess_input(frame_resized)
    return np.expand_dims(frame_preprocessed, axis=0)

# Inicializar la cámara
cap = cv2.VideoCapture(0)  # Usa la cámara de la laptop (ID 0)
if not cap.isOpened():
    print("Error: No se pudo acceder a la cámara.")
    exit()

print("Presiona 'q' para salir.")

while True:
    # Capturar frame de la cámara
    ret, frame = cap.read()
    if not ret:
        print("Error: No se pudo leer el frame.")
        break

    # Preprocesar el frame completo para la clasificación
    input_frame = preprocess_frame(frame)

    # Hacer predicción con el modelo
    predictions = model.predict(input_frame)
```

Nota: Autoría Propia

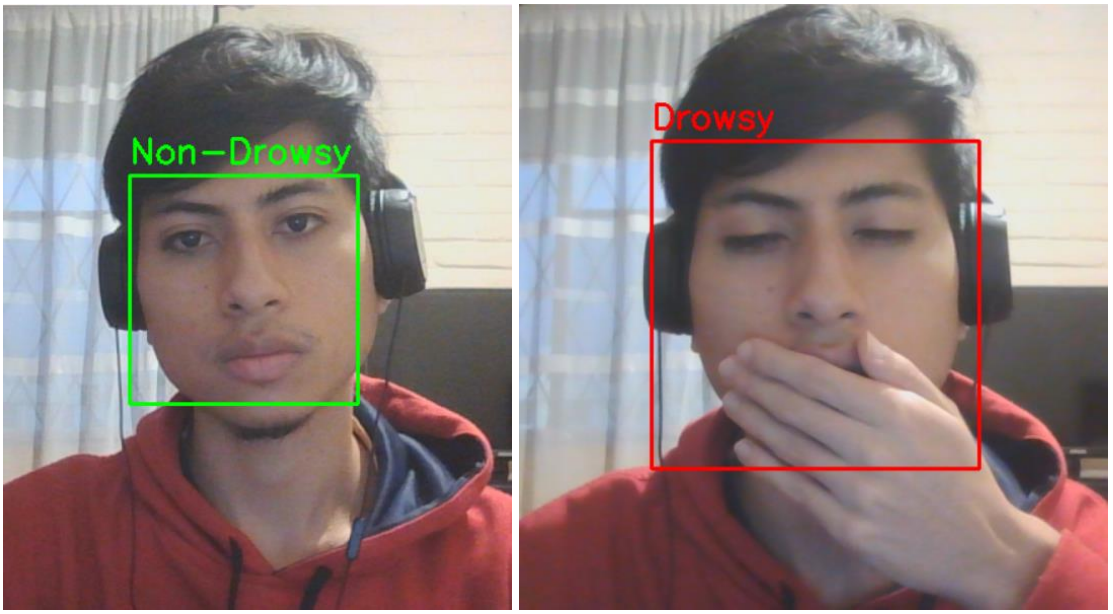
El sistema se conecta a la cámara integrada de la computadora utilizando cv2.VideoCapture(0). Se verifica si la cámara está accesible; de lo contrario, el programa se detiene con un mensaje de error.

Dentro del bucle principal, el código captura continuamente frames de la cámara. Cada frame es preprocesado mediante la función preprocess_frame y luego se pasa al modelo para realizar la predicción. El modelo devuelve una probabilidad para cada clase, y se selecciona la clase con la mayor probabilidad utilizando np.argmax(predictions). Adicionalmente, se implementa el cálculo del PERCLOS para la activación de la alarma en caso de detección de somnolencia. En concordancia con la versión previamente mencionada, se estableció un umbral de PERCLOS de 0.3 en un intervalo de 5 segundos. Esto implica que, si el porcentaje de frames etiquetados como drowsy supera el 30% del total en dicho intervalo, se activa una alarma para alertar al conductor.

Finalmente, y como se aprecia en la figura 31, según la clase predicha, se asigna la etiqueta correspondiente: "Non-Drowsy" o "Drowsy". Además, se utilizan los colores verde y rojo para representar los estados visualmente.

Figura 31

Estados de somnolencia versión con etiquetas



Nota: Autoría Propia.

CAPÍTULO 3: RESULTADOS

Dado que se han llevado a cabo procesos de clasificación y regresión, se aplicarán métricas específicas para evaluar su desempeño. Estas métricas se utilizarán tanto durante la fase de entrenamiento como en diversos escenarios controlados y en tiempo real, con el fin de medir el rendimiento del modelo bajo diferentes condiciones. En el caso del sistema embebido, se evaluará el apartado de portabilidad de la norma ISO 25010, con el objetivo de evaluar su adaptabilidad y correcto funcionamiento en dispositivos embebidos.

3.1 Métricas de evaluación

Las métricas de evaluación para analizar el rendimiento del modelo a utilizar son accuracy, loss, precision, recall y F1-score para la clasificación. Además, se emplearán métricas pérdida como son error absoluto medio (MAE), error cuadrático medio (MSE), y la raíz del error cuadrático medio (RMSE).

Para cuantificar estas métricas es necesario tener en cuenta los conceptos de True Positive (TP), True Negative (TN), False Positive (FP) y False Negative (FN). Estos términos representan los posibles resultados de un modelo de clasificación en relación con las predicciones y los valores reales.

True Positive (TP): Se refiere a los casos en los que el modelo predice correctamente la clase positiva.

True Negative (TN): Corresponde a los casos en los que el modelo predice correctamente la clase negativa.

False Positive (FP): Ocurre cuando el modelo predice incorrectamente la clase positiva, es decir, identifica un negativo como positivo.

False Negative (FN): Sucede cuando el modelo predice incorrectamente la clase negativa, es decir, identifica un positivo como negativo (Diaz, 2020).

El accuracy se calcula utilizando la Ecuación 2 y permite determinar el porcentaje de instancias clasificadas correctamente por el modelo. Sin embargo, para que esta métrica refleje de

manera fiel el rendimiento real del modelo, es recomendable que el conjunto de datos esté balanceado entre sus clases.

$$Accuracy = \frac{TP+TN}{TN+TP+FP+FN} \quad (2)$$

La precisión se calcula como la relación entre el número de muestras positivas clasificadas correctamente, verdaderos positivos, y el total de muestras clasificadas como positivas tanto correcta como incorrectamente, como se ve en la ecuación 3. Esta métrica evalúa la capacidad del modelo para clasificar de manera precisa una muestra como positiva. Cuando el modelo realiza muchas clasificaciones positivas incorrectas o pocas clasificaciones positivas correctas, el denominador aumenta, lo que reduce el valor de la precisión. Por el contrario, la precisión es alta cuando el modelo logra maximizar los verdaderos positivos y minimizar los falsos positivos (Skelton, 2024).

$$Precision = \frac{TP}{TP+FP} \quad (3)$$

El recall se cuantifica como la relación entre el número de muestras positivas clasificadas correctamente y el total de muestras positivas reales. Esta métrica evalúa la capacidad del modelo para detectar correctamente las muestras positivas. Un valor alto de recall indica que el modelo es eficaz para identificar la mayoría de las muestras positivas, lo que significa que se puede confiar en su habilidad para detectarlas. El recall se enfoca exclusivamente en cómo se clasifican las muestras positivas, independientemente de cómo se clasifiquen las negativas (Skelton, 2024). Se calcula mediante la ecuación 4.

$$Recall = \frac{TP}{TP+FN} \quad (4)$$

El F1-score es una métrica que cuantifica el rendimiento de un modelo mediante la combinación de precision y recall en una única medida. Se define como la media armónica de estas dos métricas, lo que permite equilibrarlas y proporcionar una evaluación más representativa del desempeño del modelo, especialmente en escenarios con clases desbalanceadas (Kundi, 2022). Se calcula mediante la ecuación 5.

$$F1Score = \frac{2*Precision*Recall}{Precision+Recall} \quad (5)$$

La función de pérdida o loss, es una métrica que cuantifica la discrepancia entre las predicciones generadas por un modelo y los valores reales. Esta función evalúa qué tan bien o mal está desempeñándose el modelo en un caso específico, y el valor resultante, conocido como pérdida, refleja la precisión de la predicción individual. Un valor de pérdida más bajo indica un mejor ajuste del modelo a los datos, ya que significa que las predicciones están más cerca de los valores reales. Por el contrario, una pérdida alta sugiere que el modelo tiene dificultades para generalizar correctamente (Alake, 2024).

El error cuadrático medio MSE es una función de pérdida que cuantifica la discrepancia promedio entre las predicciones de un modelo de aprendizaje automático y los valores objetivo-reales. Esta métrica se calcula promediando el cuadrado de las diferencias entre las predicciones y los valores reales como se puede ver en la ecuación 6. Al elevar al cuadrado estas diferencias, se penalizan en mayor medida los errores de predicción más grandes, lo que hace que el MSE sea sensible a los valores atípicos (Alake, 2024).

$$MSE = \left(\frac{1}{n}\right) * \Sigma(y_i - \bar{y})^2 \quad (6)$$

Donde, n representa el número total de muestras en el conjunto de datos, y_i corresponde al valor predicho por el modelo para la i-ésima muestra, $\bar{y}$ denota el valor real u objetivo asociado a la i-ésima muestra.

El error absoluto medio (MAE) mide la diferencia promedio en valor absoluto entre las predicciones y los valores reales, asignando el mismo peso a todos los errores. A diferencia del error cuadrático medio (MSE), que eleva los errores al cuadrado y amplifica el impacto de los valores atípicos, el MAE es menos sensible a estos, ya que los trata de manera uniforme. Esto lo hace más adecuado para escenarios donde los valores atípicos no deben penalizarse excesivamente. Este error se lo calcula mediante la ecuación 7.

$$MAE = (1/n) * \Sigma|y_i - \bar{y}| \quad (7)$$

Donde, n representa al número de muestras, $\bar{y}$ corresponde al valor objetivo para la muestra i-ésima, y_i representa el valor previsto en la muestra i-ésima.

3.2 Modelos evaluados

Se evaluaron dos arquitecturas de MobileNetV3: la variante Small, con enfoque basado en EAR y la variante Large, con enfoque basado en etiquetas. Los resultados de las métricas de evaluación se presentan en la Tabla 12. La variante Small alcanzó un accuracy del 98% y una precisión del 97%, mientras que la variante Large obtuvo un 93% tanto en accuracy como en precisión. El mejor desempeño de la variante Small se atribuye a la información adicional proporcionada por los valores de EAR durante su entrenamiento. En cuanto al tiempo de entrenamiento, la variante Small requirió aproximadamente la mitad del tiempo que la variante Large. Esta diferencia se debe a que la variante Large posee una mayor cantidad de parámetros y fue entrenada desde cero, sin congelar capas, a diferencia del modelo Small.

Tabla 12

Métricas en porcentajes y tiempo de los modelos MobileNetV3

Modelo V3	***Loss***	***Accuracy***	***Precision***	***Recall***	***F1 Score***	***MSE***	***MAE***	***RMS***	***Tiempo de entrenamiento***
Small (EAR)	0.05	0.98	0.97	0.98	0.98	0.014	0.03	0.14	9 min 56 s
Large (Labels)	0.21	0.93	0.93	0.94	0.93	0.054	0.075	0.26	17 min 14 s

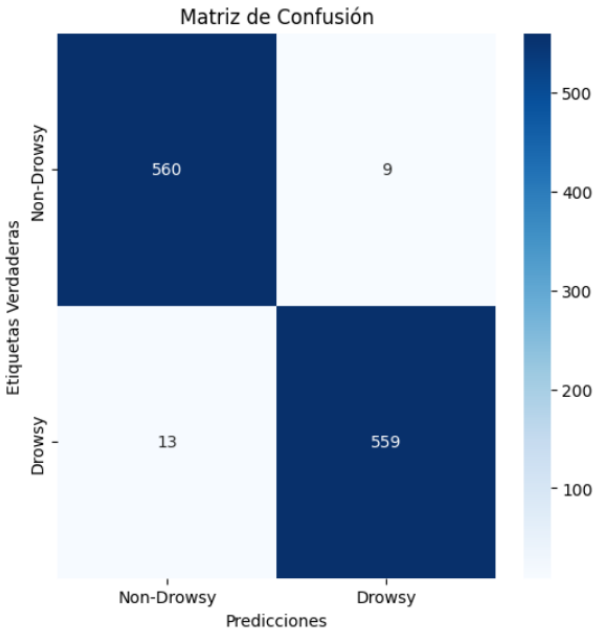
Nota: Autoría Propia.

Para ambas versiones del modelo, se generaron las correspondientes matrices de confusión, así como gráficos de métricas de evaluación que representan la evolución del rendimiento a lo largo del período de entrenamiento. Estas métricas incluyen precisión, recall, F1-score y las funciones de pérdida, las cuales fueron monitoreadas en función de las épocas para analizar la convergencia y el posible sobreajuste del modelo.

Como se puede observar en la Figura 32, se presenta la matriz de confusión correspondiente a la variante MobileNetV3Small. En dicha matriz, el eje y representa las etiquetas verdaderas (ground truth), mientras que el eje x corresponde a las etiquetas predichas por el modelo. En este caso, de un total de 569 instancias pertenecientes a la clase Non-Drowsy, el modelo predijo correctamente 560. Por otro lado, de 572 instancias correspondientes a la clase Drowsy, se

clasificaron correctamente 559 casos, lo que indica un rendimiento equilibrado y consistente en ambas clases.

Figura 32
Matriz de confusión para la variante MobileNetV3Small entrenada con EAR

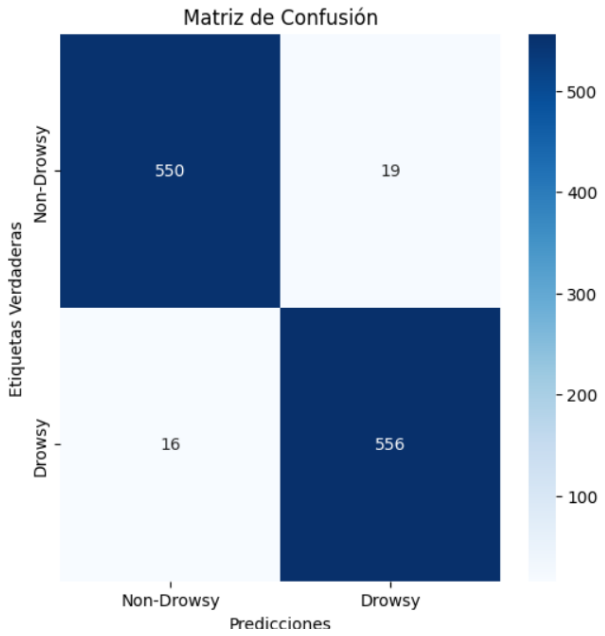


Nota: Autoría Propia

En el caso de la variante MobileNetV3Large entrenada con labels, como se muestra en la Figura 33, de la matriz de confusión podemos inferir que, de un total de 569 instancias pertenecientes a la clase Non-Drowsy, el modelo predijo correctamente 550, lo que representa una ligera disminución en el rendimiento comparado con la variante Small. Por otro lado, para las 572 instancias correspondientes a la clase Drowsy, se clasificaron correctamente 556 instancias, manteniendo un nivel de precisión similar al de la variante Small. En contraste con la variante Small, la variante Large experimentó un leve descenso en el desempeño para la clase Non-Drowsy, lo que podría atribuirse a diferencias en la capacidad de generalización o a la complejidad adicional del modelo.

Figura 33

Matriz de Confusión para la variante MobileNetV3Large



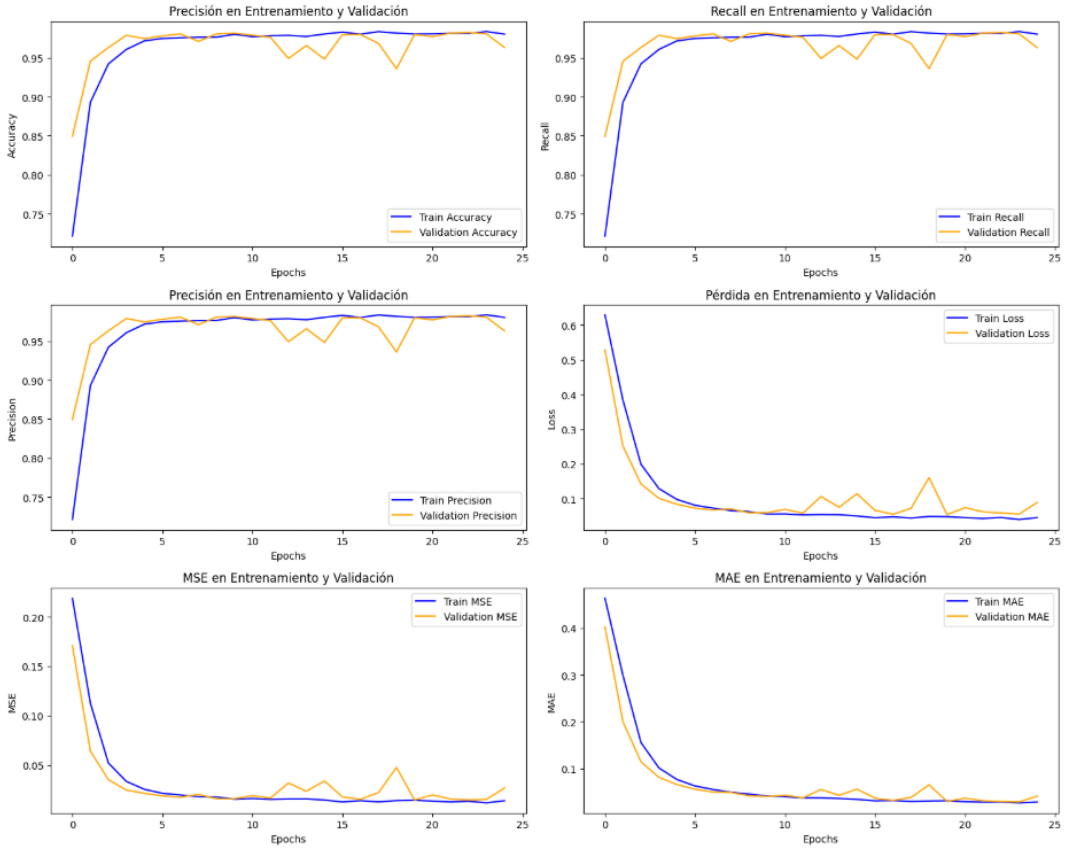
Nota: Autoría Propia.

Durante la fase de entrenamiento, y como se mencionó previamente, se monitorearon y evaluaron las métricas a lo largo del tiempo. Este seguimiento permitió analizar la evolución del rendimiento de los modelos en función de las épocas, proporcionando información sobre su convergencia, estabilidad y posibles casos de sobreajuste.

Como se observa en la Figura 34, las métricas de evaluación de la variante MobileNetV3Small alcanzan su máximo rendimiento entre las épocas 9-11 y 18-21, donde se evidencia una estabilización en los valores de precisión, recall y F1-score. Sin embargo, durante las épocas 12-16 y a partir de la época 24, se aprecia un ligero deterioro en el rendimiento del modelo, lo que sugiere un inicio de desaprendizaje o sobreajuste. Sin embargo, estas variaciones entre épocas son mínimas, con diferencias en los valores de las métricas que oscilan entre 0.02 y 0.04, lo que indica una convergencia sólida y estable de los patrones aprendidos por el modelo.

Figura 34

Evolución durante las épocas de la variante MobileNetV3Small

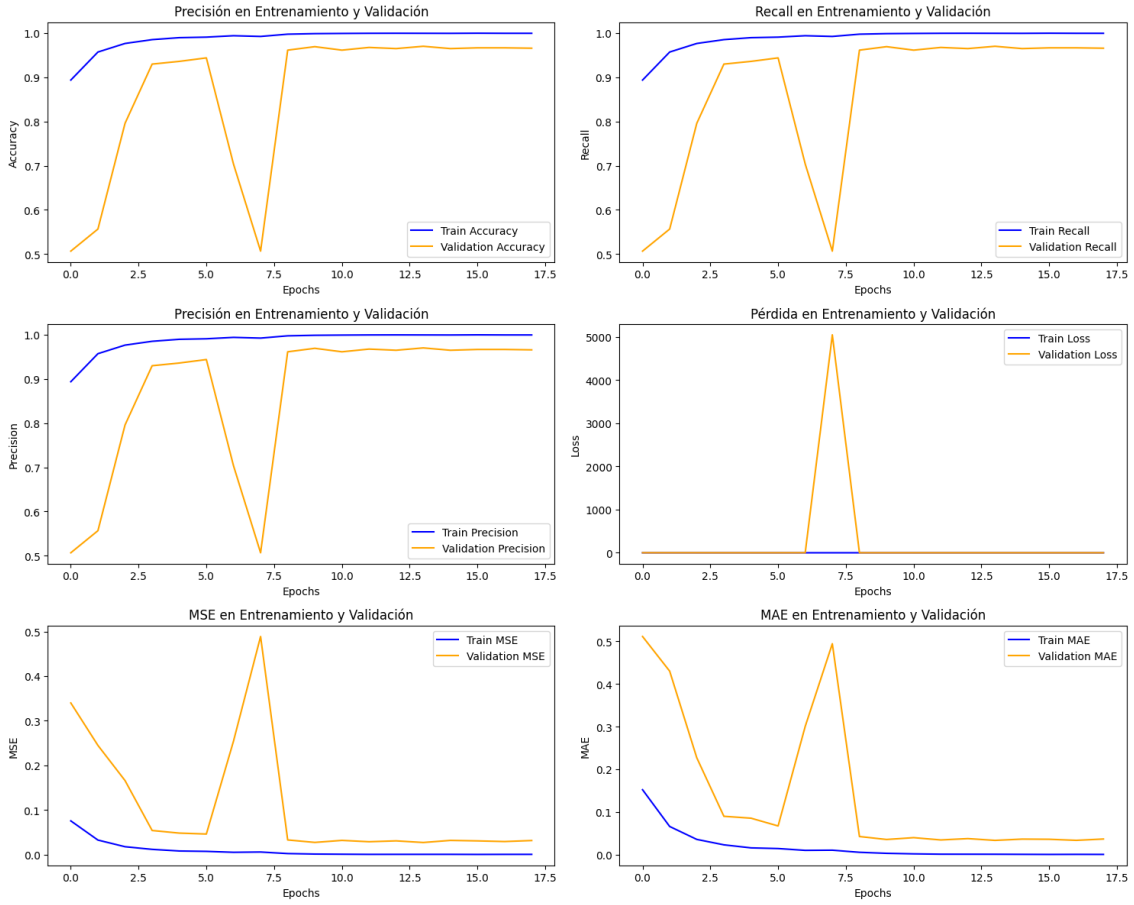


Nota: Autoría Propia

En el caso de la variante MobileNetV3Large, como se muestra en la Figura 35, el modelo presenta dificultades para generalizar durante la época 6, experimentando una caída drástica en su rendimiento. Posteriormente, el modelo logra recuperarse y converge de manera estable durante las épocas siguientes. Sin embargo, a partir de la época 12, su rendimiento se estanca, sin mostrar mejoras significativas. Un aspecto destacable es la marcada disparidad observada en las funciones de pérdida, y sobre todo en la función crossentropy durante la época 6, la cual se corrige rápidamente en las épocas posteriores. Este comportamiento sugiere que el modelo enfrentó problemas para racionalizar los patrones de aprendizaje en esa fase específica, posiblemente debido a la complejidad de la arquitectura o a la sensibilidad ante ciertos patrones en los datos.

Figura 35

Evolución durante las épocas de la variante MobileNetV3Large



Nota: Autoría Propia

3.3 Conjuntos de pruebas para la evaluación

Se sometieron los modelos a pruebas adicionales utilizando distintos conjuntos de datos diseñados para simular diversos escenarios y condiciones. Estos conjuntos incluyeron condiciones de iluminación diurna y nocturna, así como variaciones en la apariencia del rostro, tales como: rostro sin accesorios (bareface), rostro con gafas de sol (sunglasses) y rostro con lentes (glasses). Este enfoque se lo hizo con el fin de analizar el rendimiento de los modelos en contextos más realistas, evaluando su capacidad de generalización y adaptabilidad frente a variaciones en las condiciones de entrada.

3.3.1 Datasets con accesorios faciales

En la Tabla 13 se presentan los resultados comparativos de las métricas de evaluación para el modelo MobileNetV3Small, empleando un conjunto de prueba que comprendía imágenes clasificadas en tres categorías: "glasses", "sunglasses" y "bareface". El modelo alcanzó un accuracy promedio del 94%, observándose una disminución en el rendimiento para la categoría de " sunglasses ", donde el accuracy descendió hasta el 92%. En comparación con el accuracy general del 98% obtenido durante el entrenamiento, estos resultados indican que el modelo enfrenta mayores dificultades al clasificar imágenes con accesorios faciales, especialmente cuando se usan lentes de sol.

Tabla 13

Métricas con accesorios faciales de la variante MobileNetV3Small

Clases	**Loss**	**Accuracy**	**Precision**	**Recall**	**F1Score**	**MSE**	**MAE**	**RMSE**
Bareface	0.51	0.97	0.96	0.95	0.96	0.23	0.38	0.47
Glasses	0.50	0.94	0.94	0.93	0.93	0.31	0.48	0.56
Sunglasses	0.51	0.92	0.91	0.90	0.91	0.39	0.51	0.59
Promedio	**0.51**	**0.94**	**0.94**	**0.93**	**0.93**	**0.31**	**0.46**	**0.54**

Nota: Autoría Propia

En la Tabla 14 se presentan los resultados de las métricas de evaluación para la variante MobileNetV3Large, utilizando el mismo conjunto de pruebas mencionado anteriormente. El modelo alcanzó un accuracy promedio del 73%, con un mínimo del 70% en la clasificación de la categoría sunglasses. En comparación con el accuracy general del 93% obtenido durante el entrenamiento, se observa una disminución significativa en el rendimiento durante la fase de pruebas. Estos resultados sugieren la presencia de sobreajuste, lo que indica que el modelo ha optimizado su desempeño para el conjunto de entrenamiento, pero presenta dificultades para generalizar a datos no vistos.

Tabla 14

Métricas con accesorios faciales de la variante MobileNetV3Large

Clases	Loss	Accuracy	Precision	Recall	F1Score	MSE	MAE	RMSE
Bareface	0.58	0.78	0.78	0.75	0.75	0.33	0.40	0.49
Glasses	0.69	0.71	0.73	0.71	0.71	0.42	0.57	0.63
Sunglasses	0.70	0.70	0.79	0.69	0.70	0.61	0.68	0.75
Promedio	**0.65**	**0.73**	**0.76**	**0.71**	**0.72**	**0.45**	**0.55**	**0.62**

Nota: Autoría Propia

3.3.2 Datasets con condiciones de iluminación

En la Tabla 15 se muestran los resultados comparativos de las métricas de evaluación para el modelo MobileNetV3Small, utilizando un conjunto de pruebas que incluye imágenes categorizadas en condiciones de iluminación diurna y nocturna. El modelo alcanzó un accuracy promedio del 93%, con una disminución marginal hasta el 91% en escenarios de baja luminosidad. Esta reducción en el rendimiento bajo condiciones de poca luz es previsible, dado que la captura de detalles visuales se ve afectada en entornos oscuros. Esta limitación podría mitigarse mediante el uso de cámaras infrarrojas.

Tabla 15

Métricas con condiciones lumínicas de la variante MobileNetV3Small

Clases	Loss	Accuracy	Precision	Recall	F1Score	MSE	MAE	RMSE
Diurno	0.61	0.94	0.94	0.94	0.94	0.24	0.38	0.47
Nocturno	0.62	0.91	0.92	0.92	0.92	0.24	0.40	0.49
Promedio	**0.62**	**0.93**	**0.93**	**0.93**	**0.93**	**0.24**	**0.39**	**0.48**

Nota: Autoría Propia

En la Tabla 16 se presentan los resultados comparativos de las métricas de evaluación para el modelo MobileNetV3Large, utilizando el mismo conjunto de pruebas mencionado anteriormente, el cual incluye imágenes categorizadas en condiciones de iluminación diurna y nocturna. Siguiendo una tendencia similar a la observada en el conjunto de evaluación de accesorios faciales, el modelo experimentó una disminución significativa en su accuracy en comparación con los resultados obtenidos durante el entrenamiento. Específicamente, el modelo alcanzó un accuracy del 77% para imágenes en condiciones diurnas y del 74% para condiciones nocturnas, lo que representa una caída notable respecto al rendimiento en la fase de entrenamiento. Este comportamiento refuerza la hipótesis de un posible sobreajuste durante el entrenamiento.

Tabla 16

Métricas con condiciones lumínicas de la variante MobileNetV3Large

Clases	**Loss**	**Accuracy**	**Precision**	**Recall**	**F1Score**	**MSE**	**MAE**	**RMSE**
Diurno	0.60	0.77	0.78	0.77	0.77	0.24	0.38	0.47
Nocturno	0.64	0.74	0.75	0.74	0.74	0.34	0.47	0.55
Promedio	**0.62**	**0.75**	**0.76**	**0.75**	**0.75**	**0.29**	**0.42**	**0.51**

Nota: Autoría Propia

3.3.3 Elección del mejor modelo

Dado que la variante MobileNetV3Large, entrenada con labels, muestra una caída significativa en accuracy, precision, recall y F1-score en comparación con los resultados obtenidos durante el entrenamiento, podemos concluir que el modelo experimentó sobreajuste. Esto indica que su capacidad de generalización es limitada y tiene dificultades para clasificar correctamente la somnolencia, por lo que se descarta esta variante para su implementación.

En contraste, la variante MobileNetV3Small, entrenada con los valores EAR, presenta una ligera disminución en accuracy, mostrando dificultades en condiciones de baja iluminación y con gafas de sol. Sin embargo, mantiene un rendimiento superior al 91% en todas las condiciones evaluadas como se ve en la tabla 17. Específicamente, el modelo alcanzó un 98% de accuracy durante el entrenamiento, un 94% en presencia de accesorios faciales y un 93% en distintas condiciones lumínicas, desglosando con un desempeño del 92% para lentes de sol y 91% en entornos nocturnos de poca luz. Además, las métricas de accuracy, precision, recall y F1-score son consistentes, con una variación máxima del 1% entre ellas. Esto indica una clasificación estable tanto para verdaderos positivos como para verdaderos negativos, lo que sugiere que el modelo es robusto y confiable en la detección de somnolencia en conductores.

Tabla 17

Comparativa de las métricas del modelo MobileNetV3Small con los distintos dataset de prueba

Clases	**Loss**	**Accuracy**	**Precision**	**Recall**	**F1Score**	**MSE**	**MAE**	**RMSE**
General	0.05	0.98	0.97	0.98	0.98	0.01	0.03	0.14
Accesorios faciales	0.51	0.94	0.94	0.93	0.93	0.31	0.46	0.54

Condiciones de luz	0.62	0.93	0.93	0.93	0.93	0.24	0.39	0.48
Promedio	**0.39**	**0.95**	**0.95**	**0.93**	**0.95**	**0.27**	**0.29**	**0.38**

Nota: Autoría Propia

Con base en los resultados obtenidos, se seleccionó la variante MobileNetV3Small para su implementación en el sistema embebido, así como para la comparativa con los modelos encontrados en la literatura.

3.4 Comparación con otros modelos

Con el objetivo de evaluar el rendimiento del modelo entrenado en relación con el estado del arte, se realizó una búsqueda de artículos científicos relacionados. Los resultados de esta comparación se presentan en la Tabla 18, donde se detallan los valores de accuracy y f1-score obtenidos por nuestro modelo y los modelos de referencia.

Tabla 18
Comparación de métricas generales

Autor	***Modelo***	***Accuracy***	***F1Score***
Nuestro modelo	MobileNetV3Small	0.9807	0.98
*(Priyanka, Shanthi, Kumar, & Praveen, 2024)***.**	CNN-LSTM, LongShort-Term Memory	0.96	0.95
*(Lin, Wang, Yang, & Wei, 2024)***.**	Multi-aware graph Convolutional Network (MAGCN) ResNet50	0.9676	0.9655
(Tang & Guo, 2024)	Yolov8 network	0.948	0.96
*(Keyuan, y otros, 2024)***.**	Hybrid brain network with Wavelet Packet Transform to EEG signals	0.9371	0.9335

(Alzami, Naufal, Al Azies, Winarno, & Soeleman, 2024)	MobileNetV2 con Auto-CLAHE	0.9362	0.9359
	MobileNetV2 CLAHE	0.913	0.9116
	MobileNetV2 AHE	0.9291	0.9287

Nota: Autoría propia

El modelo propuesto en este trabajo, con un 98.07% de accuracy, demuestra un rendimiento superior en términos de accuracy y F1-score en comparación con numerosos estudios relacionados, alcanzando resultados comparables con los trabajos de (Lin, Wang, Yang, & Wei, 2024) y (Priyanka, Shanthi, Kumar, & Praveen, 2024) los cuales representan los mejores resultados reportados con 96.76% y 96% de accuracy respectivamente. Es importante destacar que estos estudios emplean enfoques computacionalmente más costosos, como el uso de Long Short-Term Memory (LSTM) y MAGCN con ResNet50, modelos que, debido a su complejidad y alto consumo de recursos, presentan dificultades para su implementación en sistemas embebidos. En contraste, nuestro enfoque se centra en la optimización de la detección de ojos utilizando MobileNetV3, una red neuronal ligera que no solo logra un rendimiento ligeramente superior a los modelos mencionados, sino que también supera los resultados obtenidos con MobileNet por (Alzami, Naufal, Al Azies, Winarno, & Soeleman, 2024).

En la Tabla 19 se presenta una comparación detallada de los valores de accuracy desglosados por las categorías bareface, glasses y sunglasses, contrastando los resultados reportados en la literatura con los obtenidos por el modelo propuesto en este trabajo.

Tabla 19

Comparación en el accuracy con las categorías bareface, glasses y sunglasses.

Autor	***Modelo***	***Bareface***	***Glasses***	***Sunglasses***
Modelo propuesto	MobileNetV3Small	0.97	0.94	0.92
(Tang & Guo, 2024).	Yolov8 network	0.948	-	0.9080
(JARNDAL, TAWFIK, SIAM,	Vision Transformers-based Driver	0.9897	0.9893	0.9884

ALSYOUF, & CHEAITOU, 2024)	Drowsiness Detection (ViT-DDD)			
	CNN	0.9582	0.9521	0.9374
(Ziryawulawo, y otros, 2024)	CNN based on Haar Cascade Classifier	0.95	0.93	-
(Eddine, Ruichek, & Dornaika, 2022)	ResNet-50 VGGFace 2	0.8822	0.8822	0.8822
	HOG-NB	0.8935	0.8031	0.7630

Nota: Autoría Propia

El modelo propuesto muestra un rendimiento sólido en diversos escenarios en comparación con estudios existentes, destacando su enfoque de bajo costo computacional al utilizar MobileNetV3Small. Los resultados obtenidos alcanzan un accuracy del 97% para rostros sin accesorios (bareface), 94% para rostros con lentes (glasses) y 92% para rostros con gafas de sol (sunglasses). Estos valores reflejan un enfoque competitivo frente al modelo de (JARNDAL, TAWFIK, SIAM, ALSYOUF, & CHEAITOU, 2024), que, si bien mantiene un accuracy consistente del 98% en las tres categorías, requiere un modelo computacionalmente más costoso, lo que podría limitar su implementación en sistemas de bajos recursos. Cabe resaltar que se observa una tendencia común en todos los modelos evaluados: el accuracy disminuye ligeramente en presencia de lentes y experimenta una reducción más significativa con gafas de sol, en comparación con los rostros sin accesorios.

3.5 Evaluaciones en condiciones reales

Para validar el modelo propuesto en condiciones reales, se implementó el sistema embebido, integrando la cámara NIR, en un vehículo Chevrolet D-Max y un Hyundai Tucson. Se evaluaron diez personas en sesiones de un minuto, replicando las condiciones utilizadas para la captura de los conjuntos de datos originales. De cada sesión, se seleccionaron 100 fotogramas, de los cuales 50 correspondían a estados de somnolencia y 50 a estados de alerta, con el objetivo de garantizar un conjunto de datos balanceado para la construcción de la matriz de confusión y el

cálculo posterior de las métricas de evaluación. En la Tabla 20 se presenta la matriz de confusión correspondiente al sujeto de prueba 1.

Tabla 20
Matriz de confusión para el sujeto de prueba 1

Sujeto de prueba 1		**Valores Predichos**	
		Awake	Drowsiness
Valores Reales	Awake	47	3
	Drowsiness	2	48

Nota: Autoría Propia

A partir de las matrices de confusión generadas, se procedió a calcular las métricas de evaluación, incluyendo accuracy, precision, recall y F1-score, para cada uno de los 10 sujetos de prueba. Estos valores se resumen en la Tabla 21.

Tabla 21
Métricas de los sujetos de pruebas con el modelo MobileNetV3Small

SUJETO DE PRUEBA	**ACCURACY**	**RECALL**	**PRECISION**	**F1SCORE**
1	0.95	0.92	0.94	0.93
2	0.96	0.95	0.97	0.96
3	0.94	0.91	0.93	0.92
4	0.97	0.96	0.98	0.97
5	0.92	0.91	0.92	0.91
6	0.98	0.97	0.98	0.98
7	0.93	0.92	0.94	0.93
8	0.95	0.94	0.95	0.94
9	0.96	0.95	0.96	0.95
10	0.94	0.92	0.94	0.93
PROMEDIO	**0.95**	**0.94**	**0.95**	**0.94**

Nota: Autoría Propia

El modelo presenta métricas de rendimiento consistentes con los resultados obtenidos en los distintos conjuntos de datos de prueba previamente mencionados, lo que sugiere que el sistema

es robusto y mantiene un desempeño estable en diversas condiciones reales. Es importante destacar que el uso de la cámara NIR mitiga significativamente la problemática asociada a las gafas oscuras, ya que esta tecnología permite captar la luz infrarroja sin obstrucciones, transformando las gafas de sol en lentes transparentes para la cámara, como se ilustra en la Figura 36. Sin embargo, se debe considerar que las fuentes de luz infrarroja externas pueden generar refracciones no deseadas, lo que podría afectar la calidad de la imagen y, en consecuencia, comprometer la precisión de la clasificación.

Figura 36
Ejemplo de fotograma capturado con la cámara NIR

Nota: Autoría Propia.

3.6 Portabilidad usando la ISO 25010

En el presente trabajo, se evaluó la portabilidad del sistema embebido utilizando el estándar ISO 25010, el cual establece un marco de referencia para la calidad del software. Para este análisis, se hizo especial énfasis en tres características clave definidas por la norma: instalabilidad, que se refiere a la facilidad con la que el sistema puede ser implementado en diferentes entornos; remplazabilidad, que evalúa la capacidad del sistema para ser sustituido por otro software que cumpla con los mismos requisitos funcionales; y adaptabilidad, que mide la flexibilidad del sistema para ajustarse a cambios en el entorno o en los requisitos sin necesidad de modificaciones significativas (ISO 25000, 2011).

Para llevar a cabo esta evaluación, se utilizará la Tabla 22, la cual enumera las características clave a considerar. Cada característica será calificada utilizando una escala Likert de cinco puntos, donde:

- 1 representa rendimiento muy bajo,
- 2 indica rendimiento bajo,
- 3 corresponde a rendimiento moderado,
- 4 refleja rendimiento alto, y
- 5 significa rendimiento muy alto

Tabla 22
Evaluación de la adaptabilidad con la ISO 25010

Criterio	***Preguntas clave***	***Respuestas***	***Calificación (1-5)***
Adaptabilidad	¿Qué tan fácil es cambiar a otro sistema? ¿Cuánto esfuerzo requiere la migración?	El aplicativo puede correr en distintos sistemas, al usar Python es fácil su instalación en dispositivos basados en Linux como RaspberryOS	4
Dependencia de hardware: cámara NIR y zumbador (Facilidad a ser reemplazado)	¿El programa funciona con diferentes modelos de cámaras NIR y zumbadores? ¿Es necesario modificar el código si cambia el hardware?	No se necesita modificar el aplicativo en caso de tener otro módulo de cámara diferente al usado para este experimento, lo mismo con el zumbador.	5
Facilidad de instalación y configuración	¿Qué tan sencillo es instalar y configurar el software en diferentes	Se necesita instalar el sistema embebido en el automóvil, ubicar la cámara encima del tablero y	2

	dispositivos? ¿Se requieren ajustes manuales después de la instalación?	conectar el Raspberry con un alimentador para autos. Se hace uso de un script para descargar todas las librerías con sus respectivas versiones. Se necesita estar conectado a internet durante este paso.	
Flexibilidad en la configuración del entorno	¿El software permite ajustes en parámetros de configuración sin modificar el código?	Se modifica dinámicamente el parámetro del EAR para cada conductor, por lo que no es necesario modificar el código. El tiempo de duración del zumbido no está habilitado para que el usuario final lo modifique.	3
Automatización de la instalación	¿La instalación es completamente automática?	El proceso de instalación se lo realiza a través de un script, el único paso manual extra que podría necesitar el usuario es la de correr el script con permisos de administrador	4
Uso eficiente de los recursos	¿El software funciona correctamente en dispositivos con diferentes capacidades de CPU, RAM y almacenamiento?	El aplicativo carga el modelo MobileNetV3Small que pesa alrededor de 5MB. Al ser una arquitectura pensada para dispositivos móviles y sistemas embebidos no tiene problemas en correr en dispositivos limitados	5

Nota: Autoría Propia

Considerando todas las características evaluadas, se obtuvo un promedio de 4 en la escala de Likert, lo que indica un rendimiento general alto en términos de adaptabilidad. Sin embargo, el principal desafío se presenta durante la fase de instalación, ya que el sistema, al estar en un entorno

de pruebas y no ser un producto final destinado al usuario común, puede generar dificultades para personas sin experiencia en el uso de dispositivos electrónicos.

Limitaciones

Para el presente trabajo, se realizó una revisión exhaustiva de la literatura relacionada, seleccionando 12 conjuntos de datos que incluían tanto imágenes como archivos pts con los 68 puntos de referencia faciales (landmarks). A estos conjuntos se añadieron imágenes de creación propia. Inicialmente, se consideró utilizar la totalidad de las imágenes disponibles, que sumaban cerca de 30,000; sin embargo, debido a la presencia de desequilibrios en la distribución de clases en algunos conjuntos de datos y con el fin de evitar sesgos al utilizar un número elevado de imágenes de un mismo sujeto, se optó por realizar un muestreo representativo de cada conjunto. Esto permitió construir un conjunto de datos balanceado compuesto por 11,742 imágenes, donde 5,871 corresponden al estado drowsy y 5,871 al estado non-drowsy.

Otra limitante fue el uso de ambas variaciones de MobileNetV3, Small y Large. Inicialmente, se planeaba entrenar ambos enfoques con ambas versiones y luego evaluarlas en los distintos conjuntos de pruebas para determinar qué variantes ofrecían el mejor rendimiento. Sin embargo, solo se obtuvieron resultados positivos durante el entrenamiento con MobileNetV3-Small utilizando el enfoque EAR y con MobileNetV3-Large utilizando el enfoque de las etiquetas. El modelo MobileNetV3-Large tuvo un desempeño deficiente en los distintos conjuntos de pruebas, por lo que se descartó para su implementación en el sistema embebido, quedando únicamente el modelo MobileNetV3-Small, el cual demostró consistencia en todas las etapas de evaluación.

Discusión

El trabajo propuesto implementa una red neuronal convolucional (CNN) basada en la arquitectura MobileNetV3 para la detección de somnolencia en conductores. El modelo fue entrenado utilizando un conjunto de datos personalizado, compuesto por imágenes propias y muestras seleccionadas de 12 conjuntos de datos públicos. Adicionalmente, se incorporó como entrada el valor del Eye Aspect Ratio (EAR), calculado a partir de los 68 puntos faciales (landmarks). Para evaluar el rendimiento del modelo, se emplearon métricas de clasificación como accuracy, recall, precision y F1-score, junto con funciones de pérdida como categorical crossentropy, error absoluto medio (MAE), error cuadrático medio (MSE) y la raíz del error cuadrático medio (RMSE).

Se entrenaron dos modelos distintos: el primero utilizó los valores del EAR como entrada y se basó en la arquitectura MobileNetV3Small, mientras que el segundo modelo se entrenó directamente con las etiquetas de las imágenes (drowsy y non-drowsy), empleando la variante MobileNetV3Large. Ambos modelos fueron evaluados utilizando diversos conjuntos de datos de prueba para validar su rendimiento en ambientes controlados. Durante esta fase de pruebas, la variante MobileNetV3Large mostró un desempeño significativamente inferior, experimentando una caída de aproximadamente 20 puntos en las métricas de evaluación en comparación con los resultados obtenidos durante el entrenamiento. Debido a esta discrepancia y a su bajo rendimiento en condiciones de prueba, se decidió descartar esta variante y centrar los esfuerzos en la optimización y uso de la variante MobileNetV3Small, la cual fue constante y consistente en todas las etapas de prueba.

Se utilizó el entorno virtual de Kaggle para el entrenamiento y prueba del modelo, ya que ofrecía mejores características y reducía el tiempo de procesamiento en comparación con otros entornos disponibles. Durante el entrenamiento, el modelo alcanzó un 98% de accuracy, 97% de recall, 98% de precision y 98% de F1-Score. En cuanto a las funciones de pérdida, se obtuvieron los siguientes valores: 0.5% en categorical crossentropy, 1.4% en MAE, 3% en MSE y 14% en RMSE.

Adicionalmente, se llevaron a cabo pruebas en escenarios y condiciones controladas para evaluar el rendimiento del modelo en diversos contextos. Para ello, se utilizaron cinco conjuntos de datos de prueba, diseñados para simular las siguientes condiciones:

- Bareface: Rostros sin accesorios,
- Glasses: Rostros con lentes convencionales,
- Sunglasses: Rostros con gafas de sol,
- Luz diurna: Condiciones de iluminación diurna,
- Luz nocturna: Condiciones de iluminación nocturna.

En términos de accuracy, el modelo demostró un rendimiento consistente y robusto, obteniendo los siguientes resultados: 97% para bareface, 94% para glasses, 92% para sunglasses, 94% para luz diurna y 91% para luz nocturna.

En la Tabla 23 se presenta un análisis comparativo de los resultados obtenidos en el presente estudio y los resultados reportados en la literatura para experimentos similares. La comparación se estructura en torno a los resultados del entrenamiento general, así como los resultados específicos para las condiciones de rostros descubiertos (bareface), con gafas (glasses) y con gafas de sol (sunglasses). Además, se realizó una comparación con el estudio de (Inlago, 2025), el cual fue desarrollado bajo la supervisión y tutoría del PhD. Iván García, quien también es tutor del presente trabajo. Al pertenecer a la misma línea de investigación, la comparación resulta más pertinente, ya que ambos estudios utilizaron enfoques metodológicos similares, así como los mismos conjuntos de datos tanto para el entrenamiento como para las pruebas.

Tabla 23
Comparación de métricas con trabajos relacionados a somnolencia.

Autor	Modelo	Accuracy (%)
General		
Nuestro Modelo	MobileNetV3Small	98.07
(Inlago, 2025)	ResNext-50	93.06
(Priyanka, Shanthi, Kumar, & Praveen, 2024).	CNN-LSTM, LongShort-Term Memory	96
(Lin, Wang, Yang, & Wei, 2024).	Multi-aware graph Convolutional Network (MAGCN) ResNet50	96.76

(Tang & Guo, 2024)	Yolov8 network	94.8
Bareface		
Nuestro Modelo	MobileNetV3Small	97
(Inlago, 2025)	ResNext-50	93.50
(Tang & Guo, 2024).	Yolov8 network	94.8
(JARNDAL, TAWFIK, SIAM, ALSYOUF, & CHEAITOU, 2024)	Vision Transformers-based Driver Drowsiness Detection (ViT-DDD)	98.97
(Ziryawulawo, y otros, 2024)	CNN based on Haar Cascade Classifier	95
Glasses		
Nuestro Modelo	MobileNetV3Small	94
(Inlago, 2025)	ResNext-50	92.25
(Tang & Guo, 2024).	Yolov8 network	-
(JARNDAL, TAWFIK, SIAM, ALSYOUF, & CHEAITOU, 2024)	Vision Transformers-based Driver Drowsiness Detection (ViT-DDD)	98.93
(Ziryawulawo, y otros, 2024)	CNN based on Haar Cascade Classifier	93
Sunglasses		
Nuestro Modelo	MobileNetV3Small	92
(Inlago, 2025)	ResNext-50	88.68
(Tang & Guo, 2024).	Yolov8 network	90.80
(JARNDAL, TAWFIK, SIAM, ALSYOUF, & CHEAITOU, 2024)	Vision Transformers-based Driver Drowsiness Detection (ViT-DDD)	98.84
(Ziryawulawo, y otros, 2024)	CNN based on Haar Cascade Classifier	-

Nota: Autoría propia

Dado que los estudios analizados utilizaron distintos conjuntos de pruebas, compararlos únicamente en función de las métricas podría generar un sesgo al determinar qué modelo tiene un mejor desempeño, ya que un mismo modelo puede comportarse de manera diferente según el conjunto de datos utilizado. Teniendo esto en cuenta, puede afirmarse que el presente trabajo

supera a (Inlago, 2025) **e**n todos los escenarios de prueba, ya que ambos modelos fueron entrenados y evaluados con los mismos conjuntos de datos.

Adicionalmente, se realizó una evaluación comparativa del tiempo de inferencia del modelo y su capacidad para procesar fotogramas por segundo (FPS). El tiempo de inferencia se define como el período que el modelo tarda en procesar una entrada y generar una salida. Es importante destacar que estos valores varían significativamente según el hardware utilizado, especialmente en sistemas embebidos, donde las limitaciones de recursos pueden afectar el rendimiento.

Dado que el aplicativo está destinado a ser implementado en un sistema embebido, se utilizó TensorFlow Lite, una versión optimizada de TensorFlow diseñada específicamente para dispositivos con recursos limitados. Durante las pruebas realizadas en una laptop Ryzen 5 con GPU RTX 3050 Ti, el modelo demostró un tiempo de inferencia de 10 milisegundos por fotograma (ms/f) y una tasa de procesamiento superior a 70 FPS, con el mismo equipo, pero usando la GPU integrada el rendimiento es de 20 ms/f a 40 FPS. En la tabla 24 se puede observar la comparación con otros modelos.

Tabla 24

Diferentes tiempos de inferencia entre modelos de clasificación de somnolencia

Autor	**Modelo**	**Inferencia ms/f**	**FPS**	**Hardware**
Nuestro modelo	MobileNetV3 Small	10	70	Laptop Ryzen 5, 16 GB Ram, GPU RTX 3050 Ti
		20	37	Laptop Ryzen 5, 16 GB Ram, GPU integrada
		130	8	RaspBerry Pi 4
(Inlago, 2025).	ResNext-50	666	1.5	Raspberry Pi 3 modelo B+ con CPU a 1.4 GHz y 1 GB RAM
		50	20	Laptop con CPU RYZEN 7, GPU RTX 3060, 16GB RAM

(Lin, Wang, Yang, & Wei, 2024).	Multi-aware graph Convolutional Network (MAGCN) ResNet50	81	8	CPU Intel(R) 8-Core (TM) i7-8700K a 3.70 GHz GPU Titan RTX 24 GB RAM: 64GB
(Tang & Guo, 2024).	Yolov8 network	150	25	Laptop Intel(R) Core (TM) i5-10400F CPU @ 2.90GHz GPU RTX 3060TI, RAM 32GB
(JARNDAL, TAWFIK, SIAM, ALSYOUF, & CHEAITOU, 2024)	Vision Transformers-based Driver Drowsiness Detection (ViT-DDD)	-	4	Raspberry Pi 4B
(Fresta, Bellotti, Bochenko, & Lazzaroni, 2025)	ResNet-18	860	-	Jetson Nano quad-core ARM Cortex A57 CPU 1.43 GHz, NVIDIA Maxwell GPU, 4GB RAM
		250	-	Raspberry Pi 5 quad-core ARM Cortex A76 CPU 2.4 GHz 8 GB of RAM

Nota: Autoría propia

Como se puede observar, el modelo propuesto con 130 ms/f de inferencia a 8 FPS, aunque limitado en FPS, supera en tiempo de inferencia a varios modelos de referencia, incluso en comparación con dispositivos de gama alta como laptops, modelos embebidos superiores como Raspberry Pi 5 y Jetson Nano, y también supera al estudio de (Inlago, 2025). Este alto rendimiento se debe principalmente a la arquitectura MobileNetV3Small, la cual está optimizada para operar de manera eficiente en dispositivos embebidos. Además, el modelo presenta un tamaño reducido, con un peso de 5 MB, el cual se reduce a 2 MB cuando se utiliza la versión optimizada con la función Tensorflow.Lite. Estas características permiten una inferencia rápida y un bajo consumo de recursos, lo que lo hace ideal para su implementación en sistemas embebidos con limitaciones de hardware, sin comprometer su rendimiento en comparación con dispositivos más potentes.

Conclusiones

- Los experimentos revelaron que la arquitectura MobileNetV3Small superó a MobileNetV3Large en la tarea de detección de somnolencia cuando se incorporó el valor del Eye Aspect Ratio (EAR) como entrada. Por otro lado, MobileNetV3Large presentó mejores métricas al trabajar únicamente con las etiquetas de las imágenes. Sin embargo, esta variante experimentó una disminución notable en el rendimiento durante las pruebas, mientras que MobileNetV3Small con EAR mostró una gran eficiencia y consistencia durante todas las pruebas. Esta diferencia de desempeño sugiere que, para la detección de somnolencia, la complejidad reducida de MobileNetV3Small podría ser más beneficiosa, probablemente debido a su capacidad para prevenir el sobreajuste en conjuntos de datos de entrenamiento relativamente pequeños.

- La inclusión del Eye Aspect Ratio (EAR) como una característica de entrada, en combinación con la arquitectura MobileNetV3Small, resultó en mejoras significativas en la precisión y estabilidad del modelo. El EAR es una medida que cuantifica la apertura de los ojos y ha demostrado ser un indicador fiable de somnolencia. Al integrar esta información directamente en el modelo, mejoramos su capacidad para discernir entre estados de alerta y somnolencia.

- El modelo propuesto demostró un alto rendimiento en una variedad de condiciones, incluyendo diferentes expresiones faciales (bareface), el uso de gafas (glasses, sunglasses) y variaciones en la iluminación ambiental (luz diurna y luz nocturna). Sin embargo, observamos una ligera disminución en el rendimiento en presencia de gafas de sol y en condiciones de iluminación nocturna.

- La compatibilidad del modelo con TensorFlow Lite, su tamaño reducido y su bajo consumo de recursos lo hacen especialmente adecuado para aplicaciones en tiempo real, como la detección de somnolencia en sistemas embebidos dentro de vehículos. Estos hallazgos sugieren que el modelo propuesto tiene el potencial de contribuir a la seguridad vial al proporcionar una solución práctica y eficiente para la detección de somnolencia en conductores.

Recomendaciones

- Dado que el modelo está destinado a implementarse en sistemas embebidos, el uso de TensorFlow Lite es altamente recomendable para mejorar la eficiencia y reducir el uso de recursos. En línea con este pensamiento se recomienda realizar pruebas exhaustivas en diferentes plataformas embebidas, evaluando el rendimiento en términos de tiempo de inferencia y FPS para garantizar la viabilidad de su futura implementación en automóviles.
- La inclusión del Eye Aspect Ratio (EAR) como una característica de entrada, en combinación con la arquitectura MobileNetV3Small, resultó en mejoras significativas en la precisión y estabilidad del modelo. El EAR es una medida que cuantifica la apertura de los ojos y ha demostrado ser un indicador fiable de somnolencia. Al integrar esta información directamente en el modelo, mejoramos su capacidad para discernir entre estados de alerta y somnolencia.
- Es recomendable probar el modelo en un rango más amplio de escenarios del mundo real, como diferentes condiciones de iluminación, uso de más accesorios (mascarillas, pelo largo) y distractores (teléfonos, bostezos, bebidas, etc). Estas pruebas pueden revelar limitaciones y áreas de mejora, además de permitir la adaptación del modelo a situaciones del mundo real.
- Para investigaciones futuras, se recomienda extender este estudio evaluando arquitecturas alternativas basadas en MobileNet, incluyendo EfficientNet, ShuffleNet, MobileNet-SSD y la recientemente desarrollada MobileNetV4

Referencias

Referencias

Abtahi, S., Omidyeganeh, M., & Hariri, B. (2020). YawDD: Yawning Detection Dataset. *IEEE Dataport*. doi:10.21227/e1qm-hb90

Aguilar, G., Contreras, A., Eleuterio, A., & Lara, R. (2022). Estudio comparativo de tres Redes Neuronales Convolucionales pre-entrenadas en la detección de violencia contra las mujeres. *Ciencia Ergo Sum, III*.

Aguilar, G., Contreras, A., Eleuterio, A., & Lara, R. (2024). A Comparative Study of Three Pre-trained Convolutional Neural Networks in the Detection of Violence Against Women. *Ciencia Ergo-Sum*.

Alake, R. (29 de Abril de 2024). Obtenido de Datacamp: https://www.datacamp.com/es/tutorial/loss-function-in-machine-learning

Almazaroi, A., Alqarni, M., Aslam, N., & Shah, R. (2023). Real-Time CNN-Based Driver Distraction & Drowsiness Detection System. *Intelligent Automation & Soft Computing* , 2153-2174. doi:doi.org/10.32604/iasc.2023.039732

Alzami, F., Naufal, M., Al Azies, H., Winarno, S., & Soeleman, M. (2024). Time Distributed MobileNetV2 with Auto-CLAHE for Eye Region Drowsiness Detection in Low Light Conditions. *International Journal of Advanced Computer Science and Applications*, 488-500. doi:10.14569/IJACSA.2024.0151146

Andrew, H., & Suyog, G. (13 de Noviembre de 2019). *Google LLC*. (Alphabet) Obtenido de Google Research: https://research.google/blog/introducing-the-next-generation-of-on-device-vision-models-mobilenetv3-and-mobilenetedgetpu/

ANT. (2022). *ant.gob.ec.* Obtenido de ant.gob.ec: https://www.ant.gob.ec/anuario/

Aurelien, G. (2023). *Aprendizaje automático práctico con Scikit-Learn, Keras y TensorFlow* (Tercera ed.). Farnham: O'Reilly Media. Obtenido de https://oreilly.com/catalog/errata.csp?isbn=9781492032649

Aydemir, G., Kurnaz, O., Bekiryazıcı, T., Avci, A., & Kocakulak, M. (2021). Driver Drowsiness Detection using MobileNets and Long Short-term Memory. *2021 13th International Conference on Electrical and Electronics Engineering (ELECO)*, (págs. 220-223). Bursa. doi:10.23919/ELECO54474.2021.9677724

Bekhouche, S., Ruichek, Y., & Drnaika, F. (2022). Driver drowsiness detection in video sequences using hybrid selection of deep features. *Knowledge Based Systems*. doi:https://doi.org/10.1016/j.knosys.2022.109436

Bello, E. (31 de Octubre de 2023). *IEBS*. Obtenido de IEBS: https://www.iebschool.com/blog/data-mining-mineria-datos-big-data/

Bezerra, C., Chagas, M., Marques, J., & Didier, I. (2020). Real-time classification for autonomous drowsiness detection using eye aspect ratio. *Expert Systems with Applications, 158*. doi:doi.org/10.1016/j.eswa.2020.113505.

Bhattbhatt, V. (24 de Enero de 2024). Obtenido de Medium: https://medium.com/thedeephub/learning-rate-and-its-strategies-in-neural-network-training-270a91ea0e5c

Çakar, M., Yildiz, K., & Demir, Ö. (2021). Thumbnail Selection with Convolutional Neural Network Based on Emotion Detection. *International Journal of Advances in Engineering and Pure Sciences*, 88-93. doi:10.7240/jeps.900561

Cheuque, C., Querales, M. L., & Torres, R. (Enero de 2022). An Efficient Multi-Level Convolutional Neural Network Approach for White Blood Cells Classification. *MDPI*. doi:10.3390/diagnostics12020248

Cheuque, C., Querales, M., Leon, R., Salas, R., & Torres, R. (2022). An Efficient Multi-Level Convolutional Neural Network Approach for White Blood Cells Classification. *MDPI*. doi:doi.org/10.3390/diagnostics12020248

Civik, E., & Yuzgec, U. (2023). Real-time driver fatigue detection system with deep learning on a low-cost embedded system. *Microprocessors and Microsystems, 99*. doi:doi.org/10.1016/j.micpro.2023.104851.

codelabsacademy. (5 de Junio de 2024). Obtenido de codelabsacademy: https://codelabsacademy.com/es/blog/the-role-of-l1-and-l2-regularization-in-preventing-overfitting-and-enhancing-model-generalization

Conexion ESAN. (9 de Agosto de 2018). *esan.edu.pe.* Obtenido de esan.edu.pe: https://www.esan.edu.pe/conexion-esan/mineria-de-datos-en-que-consiste-el-knowledge-discovery-in-databases

Consejo Nacional de Planificación. (20 de Septiembre de 2021). *observatorioplanificacion.cepal.org*. Obtenido de observatorioplanificacion.cepal.org: https://observatorioplanificacion.cepal.org/sites/default/files/plan/files/Plan-de-Creaci%C3%B3n-de-Oportunidades-2021-2025-Aprobado_compressed.pdf

databasecamp. (5 de Abril de 2023). Obtenido de databasecamp: https://databasecamp.de/en/ml/dropout-layer-en

Diaz, R. (2020). Recuperado el Enero de 2025, de The Machine Learners: https://www.themachinelearners.com/metricas-de-clasificacion/#AUC

Eddine, S., Ruichek, Y., & Dornaika, F. (2022). Driver drowsiness detection in video sequence susing hybrid selection of deepfeatures. *Knowledge-BasedSystems, 252*. doi:10.1016/j.knosys.2022.109436

Fayyad, U., Piatetsky, G., & Padhraic, S. (1996). From Data Mining to Knowledge Discovery in Databases. *AI Magazine, 17*. Obtenido de http://www.aaai.org/Conferences/National/1997/aaai97.html

Florez, R., Quispe, F., & Coaquira, R. (2022). A CNN-Based Approach for Driver Drowsiness Detection by Real-Time Eye State Identification. *Computer Vision and Pattern Recognition Based on Deep Learning*. doi:doi.org/10.3390/app13137849

Fresta, M., Bellotti, F., Bochenko, I., & Lazzaroni, L. (12 de Febrero de 2025). Deep Learning-Based Real-Time Driver Cognitive Distraction Detection. *IEEE Access, 13*. doi:10.1109/ACCESS.2025.3539392

Hossain, U., Rahman, A., Islam, M., & Akhter, A. (2022). Automatic driver distraction detection using deep convolutional neural networks. *Intelligent Systems with Applications, XIV*. doi:10.1016/j.iswa.2022.200075

Hou, M., Wang, M., Zhao, W., Ni, Q., & Kong, X. (2021). A lightweight framework for abnormal driving behavior detection☆. *Computer Communications*, 128-136. doi:10.1016/j.comcom.2021.12.007

Howard, A., Sandler, M., Chen, B., Wang, W., Chen, L.-C., Tan, M., . . . Vasudevan, V. (20 de Noviembre de 2019). Searching for MobileNetV3. *International Conference on Computer Vision (ICCV)*, 1314-1324. doi:10.1109/ICCV.2019.00140

INCIBE. (8 de Febrero de 2018). *INCIBE*. Obtenido de Instituto Nacional de Ciberseguridad: https://www.incibe.es/incibe-cert/blog/introduccion-los-sistemas-embebidos

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

International Standard. (1 de Marzo de 2011). *cdn.standards.iteh.ai.* Obtenido de cdn.standards.iteh.ai: https://standards.iteh.ai/catalog/standards/sist/51129631-2caf-4c9e-b352-5028346adc93/iso-iec-25010-2011

ISO 25000. (2011). *iso25000.com.* Obtenido de iso25000.com: https://iso25000.com/index.php/normas-iso-25000/iso-25010/19-iso-iec-25010

Jain, V. (22 de Noviembre de 2019). *Medium.* Obtenido de Medium: https://towardsdatascience.com/everything-you-need-to-know-about-mobilenetv3-and-its-comparison-with-previous-versions-a5d5e5a6eeaa

Jain, V., & Miller, E. (2010). FDDB: A Benchmark for Face Detection in Unconstrained Settings. *UMass Amherst technical report*, 1-11. Obtenido de https://vis-www.cs.umass.edu/fddb/

JARNDAL, TAWFIK, SIAM, ALSYOUF, & CHEAITOU. (2 de Diciembre de 2024). A Real-Time Vision Transformers-Based System for Enhanced Driver Drowsiness Detection and Vehicle Safety. *IEEE Access, 13*. doi:10.1109/ACCESS.2024.3522111

Kaggle. (Marzo de 2024). *Kaggle*. Obtenido de Google LLC: https://www.kaggle.com/docs/notebooks

Keyuan, Z., Duanpo, Qinqin, Fang, Junbiao, Lurong, & Yixuan. (12 de Abril de 2024). Algorithm for drowsiness detection based on hybrid brain network parameter optimization. *Biomedical Signal Processing and Control*. doi:10.1016/j.bspc.2024.106344

Kundi, R. (16 de Diciembre de 2022). Obtenido de V7 Labs: https://www.v7labs.com/blog/f1-score-guide

Lin, L., Wang, S., Yang, J., & Wei, F. (15 de Octubre de 2024). A multi-aware graph convolutional network for driver drowsiness detection. *Knowledge-Based Systems*. doi:10.1016/j.knosys.2024.112643

LOZADA, J. (2014). Investigación Aplicada: Definición, Propiedad Intelectual e Industria. *TIC's y Sociedad*.

Machine Learning Reply GmbH. (28 de Agosto de 2023). *Linkedin.* Obtenido de Linkedin: https://www.linkedin.com/pulse/unleashing-mobilenetv2-efficient-cnn-insights/

Massoz, Q., Langohr, T., Francois, C., & Verly, J. (2016). The ULg Multimodality Drowsiness Database (called DROZY) and Examples of Use. *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 1-7. doi:10.1109/WACV.2016.7477715

Montoya, A., Holman, D., Smith, T., & Kan, W. (2016). *Kaggle.* Obtenido de Kaggle: https://www.kaggle.com/competitions/state-farm-distracted-driver-detection

Moujahid, A., Dornaika, F., & Carreras, I. (2021). Efficient and compact face descriptor for driver drowsiness detection. *Expert Systems with Applications, 168*. doi:https://doi.org/10.1016/j.eswa.2020.114334.

MOVIDELNOR. (25 de Mayo de 2023). *www.movidelnor.gob.ec.* Obtenido de www.movidelnor.gob.ec: https://www.movidelnor.gob.ec/webepm/uniportalepm/2023/05/25/fatiga-un-factor-de-riesgo-frente-al-volante/

MOVILDELNOR. (2022). *movidelnor.gob.ec.* Obtenido de movidelnor.gob.ec: https://www.movidelnor.gob.ec/webepm/uniportalepm/2023/05/25/fatiga-un-factor-de-riesgo-frente-al-volante/

Nguyen, D., Cho, S., & Park, K. (15 de Junio de 2015). Age Estimation-Based Soft Biometrics Considering Optical Blurring Based on Symmetrical Sub-Blocks for MLBP. (S. Odintsov, Ed.) *Symmetry*, 1882-1913. doi:10.3390/sym7041882

Ormeño, N. (15 de Mayo de 2019). *Medium*. Obtenido de Medium: https://normeno.medium.com/iso-25010-y-el-desarrollo-de-software-112393a4b341

paho.org. (2022). Obtenido de paho.org: https://www.paho.org/en/topics/road-safety

Paneiva, B. (10 de Noviembre de 2023). *Lumiform*. Obtenido de Lumiform: https://lumiformapp.com/es/checklists-recursos/certificacion-iso-25010

Phan, A., Trieu, T., & Phan, T. (2023). Driver drowsiness detection and smart alerting using deep learning and IoT. *Internet Of Things, XXII*(100705). doi:10.1016/j.iot.2023.100705

Pradeep, Namratha, Tellis, N., & Mayengbam. (2023). A Review on Eye Aspect Ratio Technique. *International Journal of Advanced Research in Science, Communication and Technology (IJARSCT), 3*. doi:10.48175/IJARSCT-7843

Priyanka, Shanthi, Kumar, & Praveen. (27 de Agosto de 2024). Data fusion for driver drowsiness recognition: A multimodal perspective. *Egyptian Informatics Journal*. doi:10.1016/j.eij.2024.100529

Raspberry Pi Foundation. (20 de Diciembre de 2023). *Raspberry Pi*. Obtenido de Raspberry Pi: https://www.raspberrypi.com/documentation/

Raspberry Pi Ltd. (2024). *Raspberry Pi Ltd.* Obtenido de https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. *The IEEE Conference on Computer Vision and Pattern Recognition*, 4510-4520. doi:10.48550/arXiv.1801.04381

Servicios Digitales. (29 de Abril de 2021). *laboratoriodecertificacion.es.* Obtenido de laboratoriodecertificacion.es: https://www.laboratoriodecertificacion.es/breve-explicacion-del-proceso-kdd/

Skelton, J. (28 de Agosto de 2024). Obtenido de Digital Ocean: https://www-digitalocean-com.translate.goog/community/tutorials/deep-learning-metrics-precision-recall-accuracy?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc&_x_tr_hist=true

Tang, & Guo. (2024). Fatigue Driving Detection Methods Based on Drivers Wearing Sunglasses. *IEEE Access*, 70946-70962. doi:10.1109/ACCESS.2024.3394218

Ulatas, H., Sahin, E., & Ozbay, M. (Marzo de 2023). Application of a novel deep learning technique using CT images for COVID-19 diagnosis on embedded systems. *Alexandria Engineering Journal*, 345-358. doi:10.1016/j.aej.2023.05.036

Villanueva, J. (2022). *Diseño y entrenamiento de una red neuronal para reconocimiento de imagenes.* Universidad de Sevilla. Sevilla: Universidad de Sevilla.

Vryniotis, V., & Massa, F. (26 de Mayo de 2021). *PyTorch.* Obtenido de PyTorch: https://pytorch.org/blog/torchvision-mobilenet-v3-implementation/

Wang, W., Howard, A., Zhu, M., Chen, B., Kaleniechenko, D., & Andreetto, M. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *Google*.

Weng, C., Lai, Y., & Lai, S. (2016). Driver Drowsiness Detection via a Hierarchical Temporal Deep Belief Network. *Computer Vision – ACCV 2016 Workshops*, 117-133. doi:https://doi.org/10.1007/978-3-319-54526-4_9

Ziryawulawo, Kirabo, Mwikirize, Serugunda, Mugume, & Miyingo. (2024). Machine Learning Based Driver Monitoring System: A Case Study for the Kayoola EVS. *IEEE Access*, 40-48. doi:10.23919/SAIEE.2023.10071976

ANEXOS

Anexo 1:

Notebook de entrenamiento en Kaggle: https://www.kaggle.com/code/stalfos1/drowsy-non-drowsy

Anexo 2:

Notebook con los dataset de pruebas en Kaggle: https://www.kaggle.com/code/stalfos1/bareface-glasses-sunglasses

Anexo 3:

Repositorio con el aplicativo y los modelos entrenados: https://github.com/Stalfos1/Drowsiness-Detection-MobileNetV3Small/tree/main/Aplicativo

Anexo 4:

Repositorio con scripts varios usados principalmente para la etiquetación de landmarks: https://github.com/Stalfos1/Drowsiness-Detection-MobileNetV3Small/tree/main/Scripts%20adicionales