UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE INGENIERÍA EN SISTEMAS, ELECTRÓNICA E INDUSTRIAL

CARRERA DE TELECOMUNICACIONES

Tema:

SISTEMA DE RECONOCIMIENTO DE INDICADORES DE SOMNOLENCIA MEDIANTE INTELIGENCIA ARTIFICIAL

Trabajo de titulación modalidad: Proyecto de Investigación, presentado previo a la obtención del título de Ingeniera en Telecomunicaciones

ÁREA: Comunicaciones

LÍNEA DE INVESTIGACIÓN: Tecnología de la Información y Sistemas de Control

AUTOR: Mayra Dennise Altamirano Guerra

TUTOR: Ing. Edgar Patricio Córdova Córdova, Mg.

Ambato – Ecuador

agosto - 2023

APROBACIÓN DEL TUTOR

En calidad de tutor del trabajo de titulación con el tema: SISTEMA DE RECONOCIMIENTO DE INDICADORES DE SOMNOLENCIA MEDIANTE INTELIGENCIA ARTIFICIAL, desarrollado bajo la modalidad Proyecto de Investigación por la señorita Mayra Dennise Altamirano Guerra, estudiante de la Carrera de Telecomunicaciones, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, de la Universidad Técnica de Ambato, me permito indicar que el estudiante ha sido tutorado durante todo el desarrollo del trabajo hasta su conclusión, de acuerdo a lo dispuesto en el Artículo 17 del Reglamento para la Titulación de Grado en la Universidad Técnica de Ambato y el numeral 6.3 del instructivo del reglamento referido.

Ambato, agosto 2023.

Ing. Edgar Patricio Córdova Córdova, Mg.

TUTOR

AUTORÍA

El presente trabajo de titulación titulado: SISTEMA DE RECONOCIMIENTO DE INDICADORES DE SOMNOLENCIA MEDIANTE INTELIGENCIA ARTIFICIAL es absolutamente original, auténtico y personal y ha observado los preceptos establecidos en la Disposición General Quinta del Reglamento para la Titulación de Grado en la Universidad Técnica de Ambato. En tal virtud, el contenido, efectos legales y académicos que se desprenden del mismo son de exclusiva responsabilidad del autor.

Ambato, agosto 2023.

Mayra Dennise Altamirano Guerra

C.C. 1850998061

AUTOR

DERECHOS DE AUTOR

Autorizo a la Universidad Técnica de Ambato para que reproduzca total o parcialmente este trabajo de titulación dentro de las regulaciones legales e institucionales correspondientes. Además, cedo todos mis derechos de autor a favor de la institución con el propósito de su difusión pública, por lo tanto, autorizo su publicación en el repositorio virtual institucional como un documento disponible para la lectura y uso con fines académicos e investigativos de acuerdo con la Disposición General Cuarta del Reglamento para la Titulación de Grado en la Universidad Técnica de Ambato.

Ambato, agosto 2023.

Mayra Dennise Altamirano Guerra

C.C. 1850998061

AUTOR

APROBACIÓN DEL TRIBUNAL DE GRADO

En calidad de par calificador del informe final del trabajo de titulación presentado por la señorita Mayra Dennise Altamirano Guerra, estudiante de la Carrera de Telecomunicaciones, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, bajo la Modalidad Proyecto de Investigación, titulado SISTEMA DE RECONOCIMIENTO DE INDICADORES DE SOMNOLENCIA MEDIANTE INTELIGENCIA ARTIFICIAL, nos permitimos informar que el trabajo ha sido revisado y calificado de acuerdo al Artículo 19 del Reglamento para la Titulación de Grado en la Universidad Técnica de Ambato y el numeral 6.4 del instructivo del reglamento referido. Para cuya constancia suscribimos, conjuntamente con la señora Presidente del Tribunal.

Ambato, agosto 2023.

Firmado electrónicamente por:
ELSA PILAR URRUTIA URRUTIA

Ing. Elsa Pilar Urrutia Urrutia, Mg.
PRESIDENTE DEL TRIBUNAL

Firmado electrónicamente por:
JAIME RODRIGO GUILCAPI MOSQUERA

Ing. Jaime Rodrigo Guilcapi Mosquera
PROFESOR CALIFICADOR

Firmado electrónicamente por:
FABIAN RODRIGO SALAZAR ESCOBAR

Ing. Fabián Rodrigo Salazar Escobar
PROFESOR CALIFICADOR

DEDICATORIA

El presente proyecto de investigación se lo dedico a dos pilares fundamentales en mi vida, mis queridos padres, Marco y Cecilia. Su presencia y apoyo incondicional han sido mi mayor fortaleza en este camino académico.

A mis amados hermanos, Verónica, Alexandra, Elizabeth, Marcos y Sarahi, les dedico este logro con cariño y gratitud. Su compañía, afecto y ánimo han sido un núcleo esencial en mi vida.

A mis adorables sobrinos, Sasha y Emilio, quienes han sido una parte tan significativa en mi vida. Su cariño y alegría han iluminado mi camino y han llenado mi corazón de felicidad.

Mayas

AGRADECIMIENTO

En primer lugar, agradezco a Dios, por concederme salud, sabiduría y la fortaleza necesaria para enfrentar los desafíos y seguir siempre adelante en la búsqueda de mis sueños.

A mis padres, les debo un agradecimiento infinito. Su amor incondicional y su ejemplo de superación han sido la base sobre la cual he construido mi camino hacia el éxito. Cada paso que he dado, lo he dado sabiendo que ellos estaban ahí para apoyarme y alentarme en cada decisión.

A mis hermanas y hermano, gracias por ser mi pilar de apoyo a lo largo de mi vida universitaria.

A mis amigos por darme la fuerza y la motivación necesarias para seguir adelante en este camino, en especial a Fredy por su apoyo incondicional.

Agradezco de manera especial a mi tutor de tesis, Ing. Patricio Córdova. Su apoyo, orientación y sabiduría en todo este proceso de investigación han sido fundamentales para alcanzar este logro.

Mayas

ÍNDICE GENERAL DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN EJECUTIVO

La falta de sueño no solo afecta la seguridad, sino que también aumenta el riesgo de otros problemas de salud. La somnolencia, causada principalmente por la privación de sueño, impacta negativamente las funciones diarias del ser humano, incluyendo la capacidad de reacción, el rendimiento y la atención, lo cual provoca una disminución en el nivel de alerta y concentración. En este contexto, se llevó a cabo un estudio con el objetivo de implementar un sistema basado en inteligencia artificial para reconocer indicadores de somnolencia y emitir alertas a las personas que se encuentren en ese estado, con el fin de restablecer su atención y permitirles continuar con sus actividades. El sistema consta de cuatro etapas: adquisición, procesamiento, entrenamiento y visualización. En la etapa de adquisición, se utilizó la cámara Pi Noir V2 para capturar imágenes o videos en tiempo real. Los datos adquiridos se enviaron al NVIDIA Jetson Nano para su procesamiento. Se emplearon redes neuronales para entrenar un modelo capaz de reconocer de manera precisa los indicadores de somnolencia. Para su despliegue y uso práctico, se implementó en un entorno de servicio de alojamiento en la nube. El algoritmo del sistema se desarrolló en Python debido a la variedad de librerías disponibles, y se utilizó la librería OpenCV para el procesamiento de imágenes debido a su amplia gama de comandos. Los resultados de las pruebas mostraron que el sistema procesa y envía la información en un tiempo promedio de 2.38 milisegundos en video en tiempo real.

Palabras clave: Indicadores somnolencia, inteligencia artificial, redes neuronales

ABSTRACT

The lack of sleep not only affects safety but also increases the risk of other health problems. Sleepiness, mainly caused by sleep deprivation, negatively impacts daily human functions, including reaction time, performance, and attention, leading to a decrease in alertness and concentration. In this context, a study was conducted with the aim of implementing an artificial intelligence-based system to recognize signs of sleepiness and issue alerts to individuals in that state, in order to restore their attention and allow them to continue with their activities. The system consists of four stages: acquisition, processing, training, and visualization. In the acquisition stage, the Pi Noir V2 camera was used to capture real-time images or videos. The acquired data was sent to the NVIDIA Jetson Nano for processing. Neural networks were used to train a model capable of accurately recognizing indicators of sleepiness. For practical use and deployment, the system was implemented in a cloud hosting environment. The system's algorithm was developed in Python due to the variety of available libraries, and the OpenCV library was used for image processing due to its wide range of commands. Test results showed that the system processes and sends information at an average time of 2.38 milliseconds for real-time video.

Keywords: Sleepiness indicators, artificial intelligence, neural networks

CAPÍTULO I

MARCO TEÓRICO

1.1. Tema de investigación

SISTEMA DE RECONOCIMIENTO DE INDICADORES DE SOMNOLENCIA MEDIANTE INTELIGENCIA ARTIFICIAL

1.1.1. Planteamiento del problema

La privación del sueño es uno de los problemas más habituales entre las personas. Según la Organización Mundial de la Salud, aproximadamente el 40% de la población tiene dificultades para dormir correctamente. Dormir menos de lo recomendado tiene un impacto directo en nuestro cuerpo y puede ocasionar diversos trastornos tanto a nivel físico como mental, incluyendo el estrés y la ansiedad. Además, esta falta de sueño también tiene repercusiones negativas en nuestro rendimiento diario y bienestar al día siguiente [1]. Estudios realizados en diferentes países han revelado que la somnolencia es un problema frecuente, con prevalencias que oscilan entre el 0,3% y el 25% de la población [2].

Según una investigación reciente llevada a cabo por la revista Philips, se reveló que aproximadamente el 75% de los encuestados en Latinoamérica, incluyendo países como Argentina, Brasil, Colombia y México, experimentan alguna condición que afecta negativamente su calidad de sueño, entre las cuales se destacan el insomnio, los ronquidos, la apnea y la somnolencia. De manera alarmante, un significativo número de individuos que padecen estas condiciones no están recibiendo la atención médica necesaria para hacer frente a estos problemas [3].

Según estadísticas del Instituto Nacional de Estadísticas y Censos (INEC) de Ecuador, la población ecuatoriana duerme, en promedio, unas 55.3 horas a la semana. Esta cantidad

varía ligeramente según la ubicación geográfica y el entorno urbano o rural. En las áreas urbanas, se duerme un promedio de 54.8 horas a la semana, mientras que, en las áreas rurales, el promedio aumenta a 56.6 horas. En regiones como la sierra y Galápagos, la media de sueño es de 54.2 horas, en la amazonia es de 55.3 horas y en la costa es de 56.2 horas a la semana. Estos promedios equivalen a aproximadamente 7 horas de sueño al día. De forma destacada, se observa que las zonas de la sierra y las áreas urbanas son donde los ecuatorianos duermen menos en comparación con otras regiones del país [4].

Los trastornos del sueño abarcan diversas alteraciones en el patrón y la calidad del descanso, y pueden tener consecuencias significativas en la salud general, la seguridad y la calidad de vida de las personas. La privación del sueño, en particular, puede tener un impacto negativo en la seguridad y aumentar el riesgo de padecer otros problemas de salud. La falta de sueño suele ser la causa más frecuente de somnolencia, afectando negativamente las funciones cotidianas del ser humano, lo que se traduce en una disminución en la capacidad de reacción, pérdida del rendimiento y falta de atención por parte del individuo [1] [2].

La falta de sueño y los trastornos del sueño, como la somnolencia, están estrechamente vinculados. Por esta razón, es crucial abordar los malos hábitos que pueden afectar la calidad del descanso. Entre estos hábitos se encuentran el consumo de bebidas con cafeína, el ejercicio nocturno y la falta de horarios regulares para acostarse y despertarse. Estas prácticas pueden alterar significativamente el sueño, llevando a que la persona se quede dormida durante sus actividades diarias, ya que el cuerpo intenta compensar las horas de descanso perdidas [1] [3].

El sueño es un proceso esencial que ocupa alrededor de un tercio de nuestra vida, por lo tanto, es de suma importancia que esté adecuadamente regulado. Sin embargo, en la actualidad, hemos experimentado una drástica disminución en el tiempo dedicado al descanso. Esta reducción en el sueño tiene consecuencias negativas que afectan el comportamiento de la persona. Es crucial tener en cuenta que los trastornos del sueño están vinculados a diversas enfermedades médicas y psiquiátricas [1] [5].

1.2. Antecedentes investigativos

La investigación utilizó información de repositorios de instituciones educativas nacionales e internacionales, así como de artículos científicos, trabajos de investigación, libros y reportes relacionados con el tema.

En el año 2020, Cristian Patricio Baiza Lovato, Quito, Ecuador, presentó su trabajo de investigación titulado "Sistema de detección y alerta del estado de somnolencia de conductores mediante visión artificial", en el cual desarrolló un prototipo con la capacidad de detectar el estado de somnolencia en los conductores. Para lograr esta detección, utilizó un método de inteligencia artificial conocido como visión artificial, el cual se basó en analizar el parpadeo de los usuarios en intervalos de tiempo predeterminados. De esta manera, el sistema pudo identificar con precisión posibles estados de somnolencia. El proyecto fue desarrollado mediante el uso de un módulo Raspberry Pi y librerías OpenCV para el reconocimiento facial, todo programado en el lenguaje Python. Los resultados obtenidos fueron significativos, ya que permitieron determinar una de las principales señales de somnolencia que experimentan los conductores cuando están en un estado de fatiga causado por un desequilibrio en su sueño [6].

En el año 2021, Eduard Antonio Suárez Buitrago, Satander, Colombia, presentaron un trabajo de investigación titulado "Sistema de detección de somnolencia en conductores de automóviles empleando técnicas de procesamiento de imágenes y machine learning". El objetivo principal del proyecto fue emplear técnicas de machine learning para clasificar imágenes y determinar el grado de somnolencia en conductores de automóvil. A partir de estos resultados, se implementó un sistema de alerta de precaución adecuado para cada caso específico. Además, se evaluó el rendimiento de diferentes técnicas utilizadas en el estudio, entre ellas el Algoritmo k-Nearest Neighbor (KNN), el algoritmo Support Vector Machine (SVM) y una red neuronal convolucional (CNN). Para llevar a cabo la investigación, se analizó la base de datos MRL Eye Dataset, que proporcionó información sobre las distintas propiedades de las imágenes utilizadas. Posteriormente, se realizó un resumen de las categorías presentes en este conjunto de datos. Al evaluar el rendimiento de las técnicas KNN y SVM, se observó que, al utilizar el conjunto de datos de prueba para entrenamiento y validación, la técnica KNN mostró una mayor cantidad de imágenes

clasificadas correctamente en comparación con SVM, como se reflejó en una matriz de confusión. El proyecto representó un paso significativo en la detección temprana de somnolencia en conductores [2].

En el año 2021, Ángel Ismael Velecela Santander, Azogues, Ecuador, presentó su trabajo de investigación titulado "Implementación de un sistema de detección de Sueño, en el vehículo eléctrico de la Universidad católica de Cuenca". El proyecto se basó en el uso de una cámara web que permite el monitoreo y control del rostro del usuario para detectar el estado de somnolencia. Para lograrlo, se realizó un procesamiento en tiempo real de las imágenes capturadas, analizando los rasgos característicos de cada usuario. El sistema utilizó la librería OpenCV para recibir la información desde la webcam y establecer parámetros que permitieran determinar la ubicación del usuario. Luego, se conectó a un módulo GSM SIM800L para enviar mensajes a través del puerto serial de la Raspberry. Al utilizar tecnologías como el procesamiento de imágenes en tiempo real y la comunicación a través de módulos GSM, el sistema puede alertar a los conductores sobre su estado de somnolencia y fomentar una conducción más segura y consciente. [8].

En el año 2022, Hugo Fernández Solís, San Cristóbal de la Laguna, España, presentó su trabajo de investigación titulado "Sistema de detección de somnolencia". Desarrolló un sistema altamente efectivo para detectar y clasificar imágenes utilizando Redes Neuronales y modelos de Deep Learning. La implementación del proyecto se llevó a cabo utilizando el lenguaje de programación Python, y se hizo uso de diversas librerías que facilitaron la interacción con el modelo, así como la captura y manipulación de imágenes. Entre estas librerías, se destaca opencv-python, que permitió la detección precisa de ojos y rostros en las imágenes. En cuanto al control de versiones y el almacenamiento del proyecto, se utilizaron herramientas confiables como Google Drive, GitHub y Git. Los resultados obtenidos fueron impresionantes, ya que el sistema alcanzó una tasa de acierto entre el 99% y el 96%, lo que demuestra una precisión y confiabilidad en la detección de la somnolencia en los conductores. Para futuras investigaciones, se plantea implementar este sistema en algún tipo de microprocesador que pueda ser incorporado en los vehículos, lo que permitiría su funcionamiento sin necesidad de un hardware complejo [9].

En el año 2022, Kevin Daniel Delgado Egas y Marco Antonio Yandún Velasteguí, Tulcán, Ecuador, presentó su trabajo de investigación titulado “Sistema de detección de somnolencia para conductores de taxis en la ciudad de Tulcán”. El propósito central del proyecto consistió en crear un sistema altamente eficaz para la detección de la somnolencia en conductores, empleando una variedad de herramientas electrónicas. En particular, se utilizó un sensor de obstáculo infrarrojo, diseñado para detectar cuándo el conductor cierra los ojos durante un período de tiempo determinado. Además, se incorporó un dispositivo Buzzer que emite una alarma en caso de que se detecte somnolencia. Para establecer la conexión entre la aplicación móvil y la parte electrónica, se empleó un módulo de comunicación Bluetooth. Para complementar el sistema, se desarrolló una aplicación móvil que permitió gestionar la parte electrónica, guardar los resultados obtenidos y administrar a los usuarios del sistema. El funcionamiento y eficiencia del prototipo tipo gafas se probaron con 20 usuarios voluntarios. Durante estas pruebas, se evaluó el momento en que los usuarios cerraron los ojos y se verificó la correcta emisión de la alarma en esos casos. Los resultados fueron muy alentadores, ya que, en todos los casos, las pruebas resultaron exitosas [10].

1.3. Fundamentación teórica

1.3.1. El sueño

El sueño es un estado de reposo y recuperación vital para el funcionamiento saludable del organismo. Durante el sueño, el cuerpo realiza una serie de actividades fisiológicas esenciales para nuestra salud y bienestar, como la consolidación de la memoria, la reparación de tejidos, la regulación del estado de ánimo y el fortalecimiento del sistema inmunológico [11] [12].

1.3.1.1. Ciclo del sueño

El ciclo del sueño está compuesto por distintas etapas. La primera etapa, conocida como somnolencia, nos sumerge en un estado de relajación en el que todavía somos conscientes de los estímulos externos [13]. Durante esta fase, la persona se encuentra en transición entre la vigilia y el sueño. A continuación, se adentra en el sueño profundo, en el cual la

actividad cerebral disminuye y se llevan a cabo procesos de reparación y recuperación física. Posteriormente, se llega a la fase del sueño REM (movimiento rápido de los ojos), caracterizada por sueños vívidos, mientras que el cuerpo experimenta una parálisis temporal para evitar que actuemos los sueños. Finalmente, se vuelve a la etapa de sueño ligero y el ciclo se repite [14].

1.3.2. Introducción a la somnolencia

Se puede definir como un trastorno del sueño, en el cual los sentidos pierden su agudeza, siendo un estado entre la lucidez y el sueño profundo, el mismo que puede causar perturbaciones en la conciencia [15]. Es un estado de sueño ligero, que provoca tendencia a disminuir la capacidad para realizar tareas diarias.

1.3.2.1. Factores desencadenantes de somnolencia

La somnolencia, esa sensación de adormecimiento y necesidad de dormir, puede ser causada por diversos factores. Una de las causas más comunes es la falta de sueño adecuado y de calidad. Cuando no se duerme lo suficiente o el sueño es interrumpido, es probable que se experimente somnolencia durante el día. Otro factor frecuente es el ritmo de vida acelerado y el estrés, que pueden generar un desequilibrio en el ciclo del sueño y causar somnolencia excesiva. Además, ciertos trastornos del sueño, como la apnea del sueño, pueden contribuir a la somnolencia diurna. El consumo de alcohol, medicamentos sedantes o drogas también puede tener un impacto en la somnolencia. Enfermedades como la depresión, el síndrome de fatiga crónica o la narcolepsia también pueden ser causas de somnolencia persistente. Entre otras causas están los cambios hormonales, factores ambientales, estilo de vida y hábitos [14].

1.3.3. Indicadores de somnolencia

La somnolencia se manifiesta a través de una serie de indicadores que nos alertan sobre nuestro estado de vigilia y alerta. Estos indicadores pueden variar de una persona a otra, podemos encontrar factores visuales, tales como, la reducción de los reflejos, cabeceos,

bostezo frecuente, variación de la apertura de los ojos, los cuales proporcionan información para la detección de este estado [16].

Existen patrones relacionados, los movimientos de la cabeza son significativamente menos frecuentes; el número de veces que el individuo se toca la barbilla, la cara, la cabeza, la oreja y los ojos se incrementa; se inclina ligeramente la cabeza hacia un lado a consecuencia de la relajación muscular del cuello; la actividad de los ojos se incrementa radicalmente; se producen episodios de cabeceo con más frecuencia y se tiende a adoptar posturas de relajación [17].

Los patrones de parpadeo y el umbralado, definido como el porcentaje de tiempo que los ojos se encuentran cerrados por debajo del 80 % de su nivel base, son las medidas más aceptadas en la literatura para la detección de fatiga o somnolencia [17]. En la tabla 1 se describen algunos indicadores de somnolencia.

Tabla 1. Características de indicadores de somnolencia

Bostezos frecuentes	Exceso de bostezos repetitivos y constantes
Ojos llorosos	Sensación de ojos húmedos o lágrimas frecuentes
Parpadeo lento	Reducción en la velocidad de los parpadeos
Concentración	Incapacidad para enfocarse en tareas o actividades
Fatiga	Sensación general de cansancio y falta de energía
Nivel de apertura de los ojos	Sensación de pesadez en los párpados y dificultad para mantener los ojos abiertos
Microsueños	Episodios cortos de sueño involuntario
Cambios en la velocidad de reacción	Reducción en la capacidad de reaccionar rápidamente a estímulos o situaciones

Elaborado por el investigador en base a [17].

1.3.3.1. Desempeño humano frente a la somnolencia

El desempeño humano se ve significativamente afectado por la somnolencia. Cuando el individuo está somnoliento, la capacidad cognitiva, la concentración y la memoria se ven disminuidas. La toma de decisiones se vuelve más lenta y menos precisa, lo que puede tener consecuencias negativas en situaciones que requieren atención y reacción rápida, como conducir o realizar tareas que demandan habilidades motoras. Además, la somnolencia puede afectar nuestro estado de ánimo, incrementando la irritabilidad y la falta de paciencia.

En general, la somnolencia compromete nuestra capacidad para funcionar de manera óptima y segura en diversas áreas de nuestra vida. Por lo tanto, es esencial abordar la somnolencia y garantizar un sueño adecuado y de calidad para mantener un desempeño humano óptimo [17].

Figura 1. Estado de somnolencia [17]

1.3.4. Características visuales detección de somnolencia

1.3.4.1. Detección de rostro

La detección de rostro para la detección de somnolencia es una técnica utilizada para identificar los signos visuales de somnolencia en el rostro de una persona. Este enfoque se basa en el análisis de características faciales y expresiones que pueden indicar niveles de somnolencia o fatiga [18].

En la tabla 2 se describen los comportamientos más comunes asociados con la somnolencia y la fatiga.

Tabla 2. Comportamientos más comunes en la detección de rostro [19]

Bostezos	Es un proceso reflejo que implica abrir ampliamente la boca y respirar profundamente. Se produce cuando el cuerpo intenta aumentar el flujo de oxígeno al cerebro para mantenerse despierto.
Cabeceos	Son movimientos involuntarios de la cabeza hacia adelante y hacia atrás, o hacia los lados, que se observan cuando una persona lucha por mantenerse despierta, suelen ir acompañados de períodos cortos de microsueños.

Estos gestos son señales visibles de que una persona está luchando contra la necesidad de dormir y puede indicar un nivel elevado de somnolencia [18]. Los cabeceos suelen aparecer en un estado avanzado de somnolencia, mientras que los bostezos se producen en un estado temprano de somnolencia.

1.3.4.2. Detección de ojos

Dentro de las características visuales de la somnolencia se encuentra la detección de ojos, el cual se basa en el análisis e identificación de características o patrones para medir la frecuencia de parpadeo y el nivel de apertura, con el objetivo de determinar de determinar si una persona experimenta somnolencia [20] [21].

En la tabla 3 se describen los comportamientos más comunes asociados con la somnolencia y la fatiga.

Tabla 3. Comportamientos más comunes en la detección de ojos [22] [23]

Frecuencia de parpadeo	En condiciones normales, una persona parpadea aproximadamente de 15 a 20 veces por minuto. No obstante, cuando alguien experimenta somnolencia, esta frecuencia tiende a disminuir de manera significativa en comparación con lo habitual.
Nivel de apertura del ojo	Se refiere al grado de abertura de los párpados y proporciona indicios visibles de somnolencia. Es una medida objetiva y no invasiva que se puede registrar de forma continua. En comparación con el estado normal de vigilia y alerta, la apertura del ojo tiende a disminuir.

Una herramienta utilizada para evaluar el nivel de apertura ocular es el PERCLOS (Percentage of Eyelid Closure over the Pupil Over Time), que refleja los cierres lentos de los párpados. En condiciones normales, el porcentaje de apertura debe estar por encima del 80%. Si el valor del PERCLOS es inferior a este umbral, indica un indicador de cansancio o somnolencia. El PERCLOS permite evidenciar y medir el parpadeo de una persona [24].

El PERCLOS es un método eficaz para monitorizar la somnolencia y la fatiga en diferentes contextos, como la conducción, la vigilancia en el trabajo y los estudios sobre el sueño. Al analizar el porcentaje de cierre de los párpados en relación con la pupila a lo largo del tiempo, se obtiene una medida cuantitativa que ayuda a identificar y prevenir la somnolencia, proporcionando una herramienta útil para mantener un estado de alerta y seguridad [20].

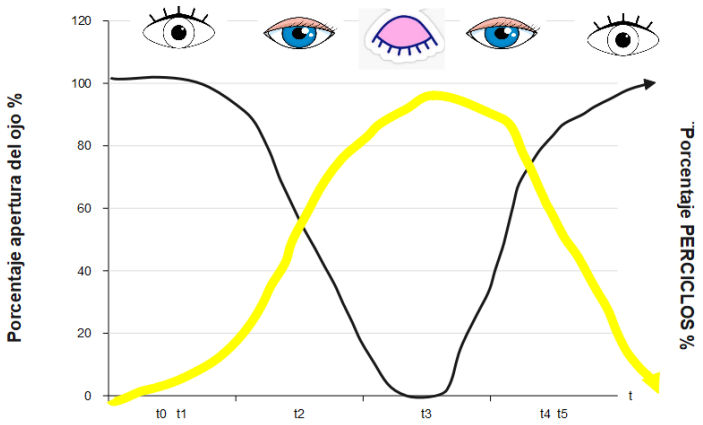


Figura 2. Representación gráfica del nivel de apertura del ojo [25]

1.3.5. Procesamiento Digital de Señales

El procesamiento digital de señales abarca técnicas y algoritmos utilizados para manipular, analizar y transformar señales en formato digital mediante el uso de herramientas informáticas. Proporciona una amplia variedad de operaciones, como filtrado, modulación, compresión, análisis espectral y detección de patrones. Estas técnicas permiten mejorar la calidad de las señales, extraer información relevante y facilitar el análisis de los datos digitales [26].

Un procesador digital de señales (DSP) adquiere señales del mundo real, como voz, video, temperatura, presión o posición, y las procesa a alta velocidad y en tiempo real después de haber sido digitalizadas. El DSP toma estas señales analógicas, las convierte en formato digital y aplica algoritmos y operaciones matemáticas para manipularlas, analizarlas y transformarlas según los requerimientos específicos. Gracias a su capacidad de procesamiento rápido, el DSP permite realizar tareas complejas en tiempo real, lo que lo convierte en una herramienta esencial en aplicaciones como comunicaciones, control industrial, procesamiento de audio y video, entre otros [26] [27].

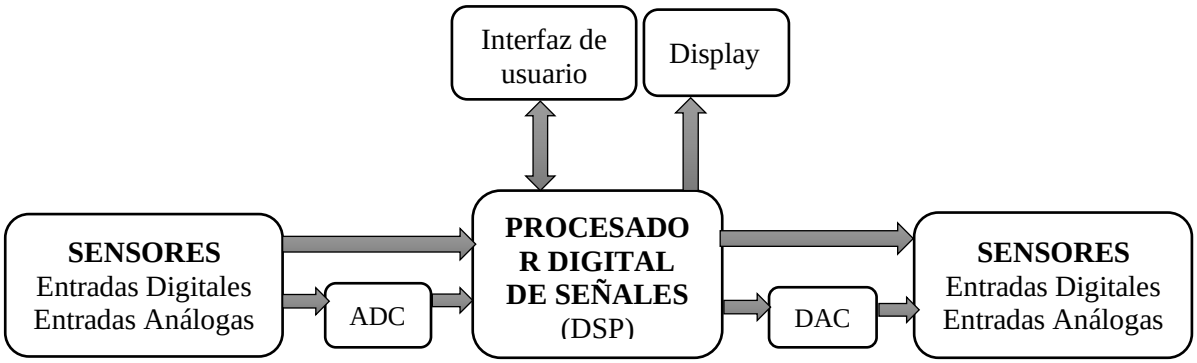


Figura 3. Sistema de Procesamiento Digital de Señales [26] [27]

La figura 3 ilustra un sistema típico de Procesamiento Digital de Señales (DSP) que consta de un DSP y componentes externos como los Convertidores Análogos-Digitales (ADC), encargados de convertir señales continuas en formato digital (1's y 0's), y los Convertidores Digitales-Análogos (DAC), que transforman señales discretas a continuas. Aunque las señales del mundo real pueden ser procesadas en su forma analógica y continua, el procesamiento de señales digitalizadas ofrece ventajas en términos de velocidad y precisión [26] [27].

1.3.5.1. Integración de inteligencia artificial en el procesamiento digital de señales

La integración de la inteligencia artificial (IA) en el procesamiento digital de señales (DSP) ha abierto nuevas oportunidades y mejoras en diversos campos, como el procesamiento de audio, el procesamiento de imágenes, la comunicación inalámbrica, la medicina, entre otros. La IA permite automatizar y optimizar tareas complejas de análisis de señales que anteriormente requerían métodos manuales o algoritmos específicos [26].

Algunas de las técnicas de IA utilizadas en el DSP incluyen redes neuronales, algoritmos de aprendizaje automático supervisado y no supervisado, algoritmos de procesamiento de imágenes y reconocimiento de patrones. Estas técnicas permiten a los sistemas de PDS

analizar señales en tiempo real, identificar características relevantes, reducir el ruido y mejorar la calidad de la señal [26]. A continuación, en la tabla 4, se describe la integración de inteligencia artificial en el procesamiento digital de señales en diversas áreas.

Tabla 4. Aplicación de la integración de inteligencia artificial en el procesamiento digital de señales [26]

Procesamiento de imágenes y video	Reconocimiento de objetos, detección de patrones, análisis de imágenes y videos en tiempo real.
Procesamiento de voz y audio	Reconocimiento y síntesis de voz, cancelación de ruido, mejora de la calidad del sonido, transcripción automática de voz.
Procesamiento de señales biomédicas	Análisis de señales biomédicas como ECG, EEG, imágenes médicas, detección de enfermedades, diagnóstico médico.
Procesamiento de señales de comunicaciones	Mejora de la calidad de la señal, cancelación de interferencias, detección de patrones de modulación.
Procesamiento de señales en sistemas inteligentes	Análisis y toma de decisiones basada en señales en tiempo real en sistemas inteligentes y automatizados.

1.3.6. Fundamentos del procesamiento digital de imágenes

El procesamiento digital de imágenes se fundamenta en principios matemáticos y probabilísticos, pero la toma de decisiones respecto a las técnicas empleadas a menudo involucra la intuición y el análisis subjetivo por parte de los especialistas. Esta elección suele basarse en juicios visuales y experiencia previa. Por lo tanto, adquirir un conocimiento básico sobre la percepción humana resulta relevante y necesario para comprender mejor cómo los seres humanos interpretan y procesan las imágenes, lo que a su vez puede enriquecer el desarrollo y selección de técnicas de procesamiento digital más efectivas y adecuadas.

El procesamiento digital de imágenes se refiere al conjunto de técnicas y algoritmos utilizados para manipular y analizar imágenes digitales. A continuación, en la tabla 5, se describen brevemente los fundamentos del procesamiento digital de imágenes.

Tabla 5. Fundamentos del procesamiento digital de imágenes

Representación de imágenes	Las imágenes se representan digitalmente utilizando una cuadrícula de píxeles, donde cada píxel almacena información sobre el color y la intensidad de esa ubicación en la imagen.
Muestreo y cuantización	El muestreo implica la captura de muestras de la imagen en puntos discretos de la cuadrícula de píxeles. La cuantización consiste en asignar valores discretos a las muestras tomadas, lo que permite la representación digital de la imagen.
Mejora de imágenes	Este proceso incluye técnicas para mejorar la calidad de las imágenes, como el ajuste del contraste, la reducción de ruido, la nitidez y la mejora del detalle.
Transformaciones espaciales	Las transformaciones espaciales son operaciones que se aplican directamente a los píxeles de una imagen. Estas operaciones incluyen el ajuste del brillo y el contraste, el cambio de escala, la rotación y la filtración espacial, que pueden resaltar o suprimir características específicas de una imagen.
Segmentación de imágenes	La segmentación implica dividir la imagen en regiones o objetos más significativos, utilizando criterios como el color, la textura, el contraste o la forma.
Extracción de características	La extracción de características consiste en identificar y extraer atributos específicos de una imagen que son relevantes para una tarea particular. Esto puede incluir la detección de bordes, la textura, el color, la forma u otras características descriptivas de la imagen.
Reconocimiento de patrones	Se utilizan algoritmos y técnicas de aprendizaje automático para identificar y clasificar patrones en las imágenes, lo que permite la detección de objetos, reconocimiento facial, seguimiento de objetos, entre otros.

Elaborado por el investigador en base a [28]

1.3.7. Sistemas electrónicos para el procesamiento de imágenes

Los sistemas electrónicos para el procesamiento de imágenes son dispositivos diseñados específicamente para adquirir, procesar y analizar imágenes de manera eficiente y precisa. Estos sistemas suelen incluir componentes como sensores de imagen, circuitos de captura

y conversión analógico-digital, unidades de procesamiento de imágenes y dispositivos de almacenamiento. Los sistemas electrónicos emplean algoritmos y técnicas avanzadas para realizar diversas operaciones, como filtrado, segmentación, reconocimiento de patrones y mejora de calidad de imagen [28].

1.3.7.1. Etapas de los sistemas electrónicos para el procesamiento de imágenes

Comprende una serie de componentes y etapas que trabajan en conjunto para analizar y procesar imágenes de manera eficiente y efectiva.

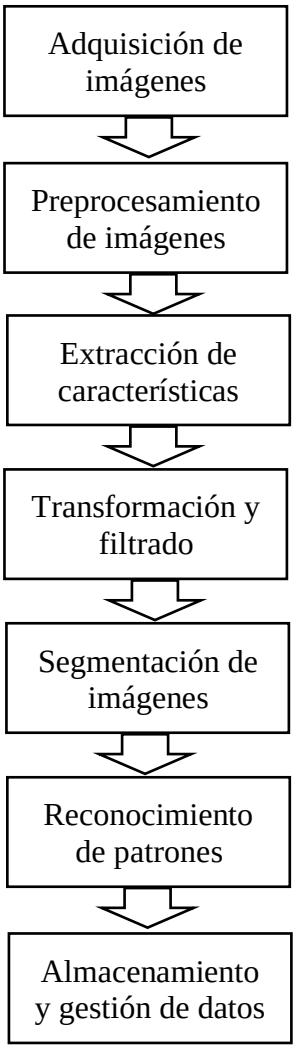


Figura 4. Etapas de los sistemas electrónicos para el procesamiento de imágenes [29]

Cada una de estas etapas desempeña un papel fundamental en el procesamiento de imágenes y permite obtener información valiosa a partir de los datos visuales. A continuación, en la tabla 6, se describen los elementos clave de esta arquitectura [29].

Tabla 6. Etapas de la arquitectura de sistemas electrónicos para el procesamiento de imágenes enfocada a IA

Etapa	Descripción
Adquisición de imágenes	Captura de imágenes utilizando cámaras, escáneres u otros dispositivos de entrada.
Preprocesamiento de imágenes	Mejora de la calidad y claridad de las imágenes mediante ajustes de contraste, eliminación de ruido, etc.
Extracción de características	Identificación y extracción de características relevantes de las imágenes, como bordes, texturas y formas.
Transformación y filtrado	Aplicación de técnicas de transformación y filtrado para modificar y mejorar las imágenes.
Segmentación de imágenes	Separación de regiones de interés u objetos del fondo mediante algoritmos de segmentación.
Reconocimiento de patrones	Separación de regiones de interés o objetos del fondo mediante algoritmos de segmentación.
Almacenamiento y gestión de datos	Almacenamiento y gestión eficiente de imágenes y datos relacionados en sistemas de almacenamiento adecuados.

Elaborado por el investigador en base a [28]

1.3.8. Sistemas de detección

Los sistemas de detección de somnolencia son herramientas diseñadas para identificar y alertar sobre el nivel de somnolencia de una persona. Estos sistemas utilizan diferentes técnicas y sensores para detectar signos de somnolencia y prevenir accidentes relacionados con la falta de atención debido a la fatiga [20] [22].

Estos sistemas utilizan sensores y algoritmos para monitorear y analizar los signos fisiológicos y comportamentales asociados con la somnolencia.

1.3.8.1. Reconocimiento de patrones

El sistema de medición por parámetros de somnolencia es una herramienta utilizada para evaluar y monitorear el nivel de somnolencia en una persona. Se basa en la medición de diferentes parámetros fisiológicos y comportamentales, que pueden indicar la presencia de somnolencia y alertar sobre el riesgo de quedarse dormido o cometer errores debido a la falta de atención [22] [30].

Algunos de los parámetros comunes que se pueden medir incluyen [22]:

- Frecuencia de parpadeo
- Movimiento ocular
- Actividad cerebral
- Variabilidad de la frecuencia cardíaca:
- Respuestas cognitivas
- Cabeceos

1.3.8.2. Reconocimiento facial

El sistema por reconocimiento facial para detectar la somnolencia es una tecnología que utiliza algoritmos y análisis de imágenes para identificar y evaluar los rasgos faciales asociados con la somnolencia. Estos sistemas utilizan cámaras o sensores para capturar imágenes del rostro y luego analizan diferentes características faciales para determinar el nivel de somnolencia de una persona [30] [31].

Algunas de las características faciales que se pueden analizar incluyen [31]:

- Cierre de ojos
- Movimientos de los párpados
- Expresiones faciales
- Postura de la cabeza
- Bostezos

1.3.9. Inteligencia artificial en el reconocimiento de indicadores de somnolencia

La inteligencia artificial ha demostrado ser una herramienta poderosa en el reconocimiento de indicadores de somnolencia. Mediante el uso de algoritmos de aprendizaje automático y redes neuronales, la inteligencia artificial puede analizar datos como patrones de parpadeo, movimientos oculares, cambios en el tono vocal y variaciones en la actividad cerebral para identificar signos de somnolencia. Estos modelos de inteligencia artificial se entrenan con conjuntos de datos etiquetados, lo que les permite reconocer patrones y realizar predicciones precisas sobre el estado de somnolencia de una persona [32].

1.3.9.1. Machine Learning

El aprendizaje automático (en inglés, machine learning) es uno de los enfoques principales de la inteligencia artificial. En pocas palabras, se trata un aspecto de la informática en el que los ordenadores o las máquinas tienen la capacidad de aprender sin estar programados para ello. Un resultado típico serían las sugerencias o predicciones en una situación particular [33].

1.3.9.2. Tipos de aprendizaje de Machine Learning

Dentro del campo del aprendizaje automático, existen diferentes enfoques o tipos de aprendizaje que se utilizan para abordar distintos tipos de problemas. Cada tipo de aprendizaje tiene sus propias características y aplicaciones, y la elección del enfoque adecuado depende del problema que se desea resolver y de la disponibilidad de datos etiquetados o información contextual [34] [35].

Es importante conocer la clasificación de los tipos de algoritmos de aprendizaje de Machine Leaning. En la tabla 7, se describe los tipos de aprendizaje de Machine Learning.

Tabla 7. Tipos de aprendizaje de Machine Learning

Tipos de aprendizaje de Machine Learning	
Tipo	**Descripción**
Aprendizaje supervisado	Un algoritmo de aprendizaje supervisado está diseñado para aprender con una gran cantidad de datos etiquetados. En otras palabras, se enseña al algoritmo cómo realizar su trabajo. Su objetivo es aprender a partir de datos etiquetados para predecir una respuesta correcta. Aquí podemos encontrar redes neuronales, regresión lineal, árboles de decisión, Naïve Bayes, SVM, entre otros.
Aprendizaje no supervisado	A diferencia del aprendizaje supervisado, el algoritmo no supervisado trabajo con datos no etiquetados, es decir, con ejemplos en los cuales se desconoce la respuesta correcta. Su objetivo es descubrir y presentar interesantes disposiciones de los datos. Aquí podemos encontrar factorización matricial, análisis de componentes principales (PCA), mezclas Gaussianas, clustering, entre otros.
Aprendizaje por refuerzo	Es un tipo de aprendizaje automático en el que no hay capacitación con datos etiquetados o no etiquetados, este algoritmo e entrenado a través de la interacción con un entorno simulado, en donde, por cada acierto, el algoritmo recibirá un refuerzo, algo parecido a la conducta humana o la usada para entrenar robots, esta recompensa le permitirá al algoritmo ser cada vez más inteligente y capaz de mejorar la toma de decisiones. Aquí podemos encontrar programación dinámica, Q – Learning , SARSA, entre otros.

Elaborado por el investigador en base a [34] [36]

1.3.10. Redes Neuronales

Las redes neuronales son modelos computacionales inspirados en el funcionamiento del cerebro humano. Están compuestas por capas de neuronas artificiales interconectadas, donde cada neurona procesa información y transmite señales a otras neuronas. Estas redes aprenden a través de la adaptación de los pesos de las conexiones entre las neuronas, permitiendo reconocer patrones complejos y realizar tareas como clasificación, reconocimiento de imágenes, procesamiento de lenguaje natural y más. Las redes

neuronales han demostrado ser efectivas en el campo del aprendizaje automático y la inteligencia artificial, y su capacidad para aprender y generalizar a partir de ejemplos los convierte en una herramienta poderosa en diversas aplicaciones [32].

1.3.11. Capas de una Red Neuronal

Una red neuronal es un sistema compuesto por capas de nodos o neuronas interconectadas, las cuales se encargan de procesar información. El número de capas presentes en una red neuronal varía según la complejidad de la función que se intenta modelar. La forma más sencilla de una red neuronal es la de alimentación directa, que puede constar de una o varias capas, donde la información fluye en una dirección, de entrada, a salida, sin retroalimentación. Si bien las capas de entrada y salida siempre están presentes, las capas ocultas pueden o no estar incluidas, dependiendo de la configuración de la red [37]. A continuación, se presenta la tabla 8 que resume las características de cada capa:

Tabla 8. Capas de una Red Neuronal típica

Capa de Entrada	Esta capa es la primera de la red y está formada por neuronas que reciben datos o señales del entorno.
Capas Ocultas	Las capas ocultas no tienen conexión directa con el entorno. Procesan la información recibida de la capa de entrada y la transforman para ser utilizada por la capa de salida. Pueden ser precedidas por otras capas ocultas o la capa de entrada.
Capa de Salida	Es la última capa de la red y está compuesta por neuronas que proporcionan la solución de la red neuronal. Es responsable de proporcionar la respuesta final basada en la información procesada en las capas anteriores.

Elaborado por el investigador en base a [37]

Las capas en una red neuronal son elementos esenciales de su estructura, y están conformadas por neuronas que se encargan de procesar la información proveniente de la

capa anterior. Estas neuronas transforman los datos de manera que puedan ser adecuadamente utilizados por la capa siguiente en el proceso de la red [38].

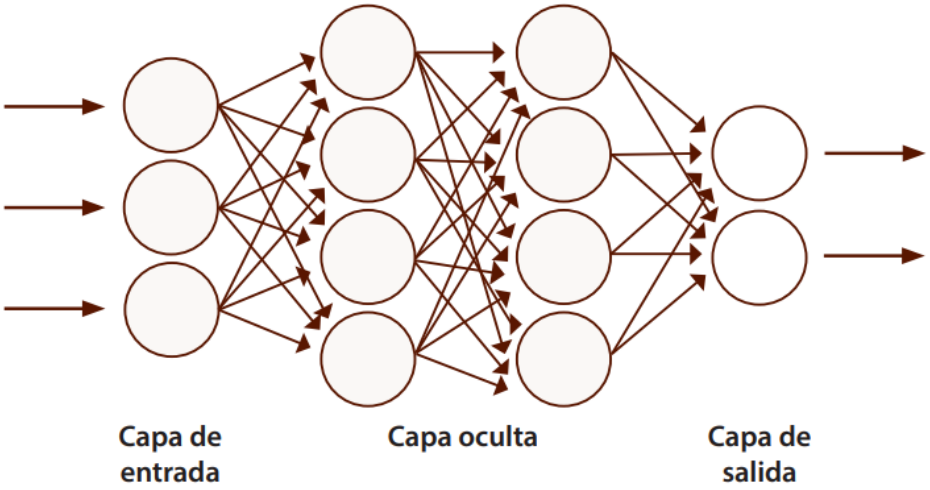


Figura 5. Arquitectura típica de una red neuronal [38]

1.3.12. Construcción modelo

Para la construcción de un modelo se tiene en cuenta las siguientes etapas:

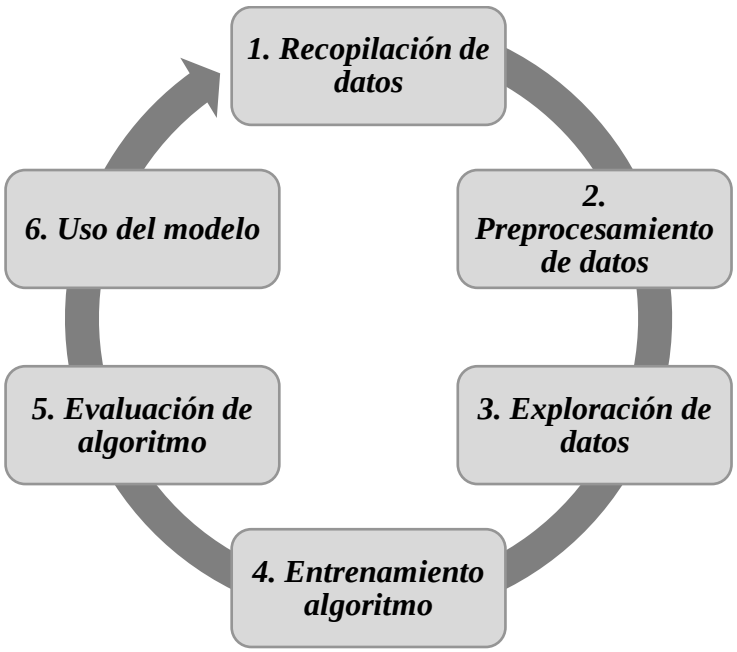


Figura 6. Construcción de un modelo [34]

Recolectar los datos: Investigar y obtener datos que se utilizarán para alimentar el modelo. Esto puede implicar la recopilación de datos nuevos o utilizar conjuntos de datos existentes de fuentes tal como un sitio web, utilizando una API o una base de datos [34].

Preprocesamiento de los datos: Consiste en preparar los datos antes de entrenar el modelo para garantizar que estén en un formato adecuado y que contengan la información más importante y comprensible para alimentar el algoritmo de aprendizaje. Por lo general se tiene que realizar varias tareas de preprocesamiento antes de poder usar los datos [34].

Explorar los datos: En este punto, se deben detectar valores atípicos; o encuentre las características que tienen más influencia para hacer una predicción. Es importante hacer imputaciones de valores nulos, agregaciones, eliminación de columnas, entre otros ajustes necesarios para limpiar los datos [34].

Seleccionar y Entrenar el algoritmo: Elige el algoritmo o modelo de Machine Learning adecuado para abordar el problema. Puedes seleccionar entre diferentes tipos de modelos, como regresión lineal, árboles de decisión, SVM, redes neuronales, entre otros. Los algoritmos de aprendizaje se alimentan con los datos que se procesaron en las etapas anteriores. La idea es que los algoritmos pueden extraer información útil de los datos iniciales y luego hacer las predicciones [34].

Evaluación y pruebas del algoritmo: Evalúa el rendimiento del modelo utilizando métricas adecuadas para el tipo de problema que se está resolviendo. Evalúa el rendimiento final del modelo y realiza los ajustes necesarios si se observan problemas o discrepancias significativas [34].

Implementación: Implementar el modelo en un entorno de producción y monitorear su rendimiento continuamente [34].

1.4. Objetivos

1.4.1. Objetivo General

Implementar un sistema de reconocimiento de indicadores de somnolencia mediante inteligencia artificial

1.4.2. Objetivos Específicos

- Documentar los métodos para la adquisición de indicadores de somnolencia.
- Elaborar algoritmos de visión artificial para la detección de estado de somnolencia
- Diseñar un sistema de indicadores de somnolencia.
- Realizar las pruebas de funcionamiento del sistema de detección del estado de somnolencia que permitan la validación de los resultados.

CAPÍTULO II

METODOLOGÍA

2.1. Materiales

En el desarrollo del proyecto de investigación, se llevó a cabo un análisis de varios dispositivos relacionados con la aplicación de visión artificial, con el propósito de seleccionar aquel que mejor se adaptara a las necesidades requeridas.

2.1.1. Microcomputador

Son dispositivos electrónicos fundamentales en la era digital actual. Se trata de pequeños circuitos integrados que incorporan una CPU, memoria y periféricos en un solo chip. Estos dispositivos son diseñados para ejecutar tareas específicas y controlar diversos sistemas y dispositivos electrónicos.

2.1.2. Raspberry Pi 4

El Raspberry Pi 4 es un ordenador de placa única (SBC) que ofrece capacidades adecuadas para aplicaciones de inteligencia artificial (IA). Está equipado con un procesador Broadcom BCM2711 de 64 bits, que consta de cuatro núcleos Cortex-A72 a 1.5 GHz. Con opciones de memoria RAM de 2, 4 u 8 GB, el Raspberry Pi 4 proporciona un rendimiento escalable para cargas de trabajo de IA. La GPU VideoCore VI integrada permite el procesamiento de gráficos y aplicaciones de IA aceleradas. Además, cuenta con puertos USB 3.0, puertos Ethernet Gigabit, Wi-Fi y Bluetooth, lo que facilita la conexión de periféricos y la comunicación en red necesaria para implementaciones de IA. Este ordenador es ampliamente utilizado en proyectos de IA a pequeña escala, como reconocimiento de objetos, clasificación de imágenes y tareas de aprendizaje automático de nivel básico [39].

Figura 7. Raspberry pi 4 [40]

2.1.3. NVIDIA Jetson Nano

El kit de desarrollo NVIDIA Jetson Nano es una computadora de inteligencia artificial para creadores, estudiantes y desarrolladores que lleva el poder de la inteligencia artificial moderna a una plataforma fácil de usar y de bajo consumo. Tiene la el rendimiento y las capacidades necesarias para ejecutar cargas de trabajo de IA modernas. Es una computadora pequeña y poderosa que te permite ejecutar múltiples redes neuronales en paralelo para aplicaciones como clasificación de imágenes, detección de objetos, segmentación y procesamiento de voz. Con su capacidad de procesamiento de alto rendimiento y su enfoque en la IA en el borde, el Jetson Nano es adecuado para aplicaciones como sistemas autónomos, robótica, visión por computadora y análisis de datos en tiempo real [41].

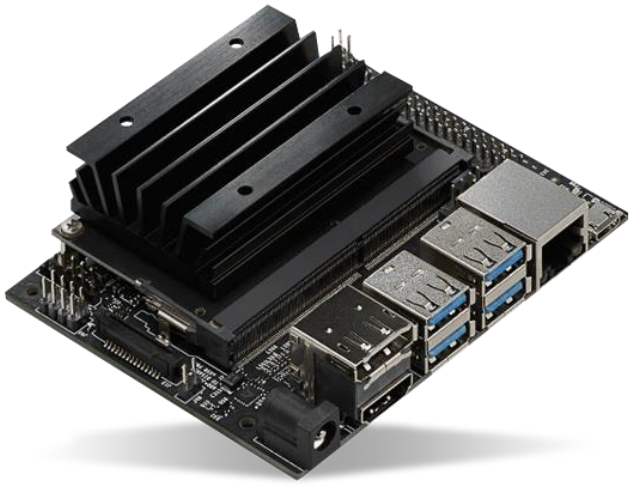


Figura 8. NVIDIA Jetson Nano [41]

2.1.4. Brix Celeron BPCE 3455C

El Brix Celeron BPCE 3455C es un mini PC compacto y de bajo consumo energético fabricado por Gigabyte. Está equipado con un procesador Intel Celeron BPCE 3455C, que consta de cuatro núcleos x86 con una velocidad de reloj de hasta 2.2 GHz. Ofrece un rendimiento gráfico adecuado para aplicaciones cotidianas y reproducción multimedia. Con su tamaño compacto y bajo consumo energético, el Brix Celeron BPCE 3455C es una opción popular para aplicaciones de escritorio de bajo perfil, señalización digital, servidores domésticos y otras aplicaciones que requieren un equilibrio entre rendimiento y eficiencia energética. [42]

Figura 9. Brix Celeron BPCE 3455C

2.1.5. Comparativa de Microcomputadores

El Raspberry Pi 4 ofrece capacidades de IA adecuadas para proyectos a pequeña escala, como reconocimiento de objetos y clasificación de imágenes básica. Por otro lado, el NVIDIA Jetson Nano proporciona un rendimiento significativamente mayor y capacidades de aprendizaje profundo, lo que lo hace adecuado para aplicaciones más exigentes en términos de IA. Por último, el Brix Celeron BPCE 3455C se posiciona como una opción más asequible y equilibrada en términos de precio y rendimiento, especialmente para tareas de IA más sencillas.

En la tabla 9, se muestra las características principales para la selección del microordenador.

Tabla 9. Características principales de microordenadores

Tipo	Raspberry Pi 4	Jetson Nano	Brix Celeron BPCE 3455C
Fabricante	Raspberry Pi	NVIDIA	GIGABYTE
GPU	VideoCore VI 500 MHz	Maxwell de 128 núcleos	-
CPU	1.5-GHz, 4-core Broadcom BCM2711 (Cortex-A72)	Quad-Core ARM A57 @ 1.43 GHz	Procesador Intel® Celeron® J4105 hasta 2.5GHz
Puertos USB	2x USB 3.0 2x USB 2.0 2x micro HDMI	4x USB 3.0 1x USB 2.0 Micro B	4x USB 3.0 1 Puerto Micro SD
Otros puertos	2 carriles para cámara MIPI. 2 × puertos microhdmi (admite hasta 4kp60)	2 carriles para cámara MIPI CSI2 Puerto de salida HDMI	1 Puerto HDMI y 1 Puerto VGA
Conectividad	Puerto Gigabit Ethernet. Wi-Fi 802.11ac Bluetooth 5.0	Puerto Gigabit Ethernet M.2 Key E	Puerto Gigabit Ethernet. Wi-Fi 802.11ac Bluetooth 4.0
Memoria	4 GB LPDDR4-3200	4GB 64-bit LPDDR4 @ 1600MHz | 25.6 GB/s	4 GB SO-DIMM DDR4 2400 MH
Medidas	22.00 cm x 11 cm x 6 cm	6.91 cm x 4.50 cm x 4.50 mm	11.6 cm x 10.3 cm x 5.6 cm
Almacenamiento	microSD (no incluye)	microSD (no incluye)	Disco duro de 2.5 pulgadas mecánico o SSD (no incluye)
Precio	$260.00	$180.00	$185.00

Elaborado por el investigador en base a [43] [40] [44]

Considerando los requerimientos específicos del proyecto, se optó por el Jetson Nano NVIDIA como la opción más adecuada, debido a su capacidad para procesar algoritmos, lo que se refleja en su excelente rendimiento en tareas relacionadas con la Inteligencia Artificial.

2.1.6. Cámara

2.1.7. Cámara web digital D-Bugg M26

Una cámara web digital es un dispositivo de captura de imágenes y video que se conecta a través de USB a un ordenador o dispositivo compatible. Estas cámaras están diseñadas para facilitar la comunicación en línea y permitir la captura de imágenes en tiempo real. En el contexto de la inteligencia artificial (IA), las cámaras web digitales pueden utilizarse como una fuente de entrada de video para aplicaciones de visión por computadora y reconocimiento de objetos [42].

Figura 10. Cámara web digital D-Bugg M26 [42]

2.1.8. Cámara Pi Noir V2

La cámara Pi Noir V2 es una cámara específicamente diseñada para su uso con las placas Raspberry Pi y compatible con Jetson Nano. Esta cámara utiliza un sensor de imagen Sony IMX219 de 8 megapíxeles, pero a diferencia de las cámaras convencionales, está optimizada para capturar imágenes en condiciones de baja luminosidad y sin luz visible. La cámara Pi Noir V2 no cuenta con un filtro de infrarrojos, lo que la hace ideal para aplicaciones de visión nocturna y monitoreo en entornos con poca luz. En el ámbito de la inteligencia artificial, la cámara Pi Noir V2 es ampliamente utilizada en proyectos de visión por computadora que requieren captura de imágenes en entornos oscuros [45].

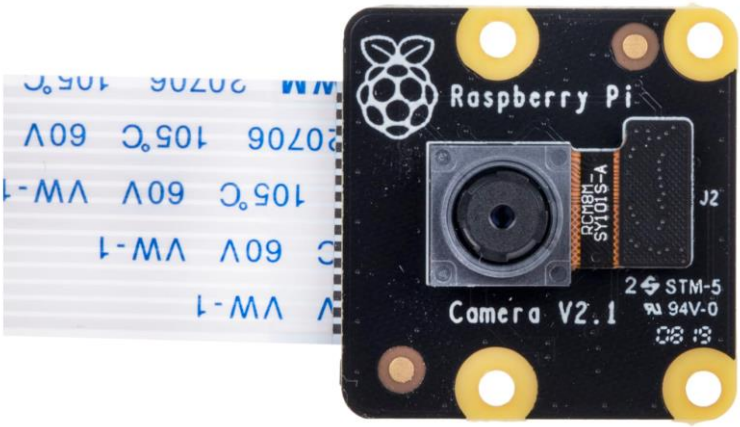


Figura 11. Cámara Pi Noir V2 [40]

2.1.9. Cámara Logitech C920 HD PRO WEBCAM

Esta cámara está diseñada con componentes ópticos y sensores de alta calidad para capturar imágenes nítidas y de alta resolución. En el contexto de la inteligencia artificial, las cámaras Logitech son utilizadas como dispositivos de entrada de video para aplicaciones de visión por computadora y reconocimiento de objetos. La cámara Logitech es adecuada para tareas como reconocimiento facial, seguimiento de objetos, análisis de emociones y más [46].

Figura 12. Cámara Logitech C920 HD PRO WEBCAM [46]

2.1.10. Comparativa de Cámara

En la tabla 10, se muestra las características principales para la selección de la cámara.

Tabla 10. Características principales de cámara.

TIPO	WebCam D-Bugg M26	Pi Noir V2	Logitech C920 HD PRO WEBCAM
Tamaño	19 x 7,6 x 12,2 cm	Alrededor de 25 × 24 × 11,5 mm	21 x 7,6 x 15,2 cm
Resolución fija	90 fps 4096 px x 2160 Px	8 megapíxeles	15.0 megapíxeles
Sensor	No	Sony IMX219	-
Profundidad de campo	No	Aproximadamente 10 cm a ∞	71 mm
Longitud focal	28 mm	4,74 mm	-
Costo	$ 15.00	$ 25.00	$ 60.00

Elaborado por el investigador en base a [46, 40]

Después de considerar los requerimientos específicos del proyecto, se decidió elegir el módulo de Cámara Pi Noir V2, debido a su velocidad para realizar capturas y procesamiento de imagen, así como por su compatibilidad con el microordenador seleccionado. En definitiva, el módulo de Cámara Pi Noir V2 se consideró la mejor opción para satisfacer las necesidades específicas del proyecto.

2.1.11. Periféricos de entrada y salida

Mouse: Es un dispositivo de entrada que se utiliza para trabajar en la interfaz de la computadora. Su función principal es señalar, seleccionar o mover datos o archivos, así como desplazarse por las diferentes aplicaciones [47].

Teclado: Es un dispositivo de entrada con el que el usuario interactúa y puede introducir datos en el ordenador de manera instantánea [47].

Monitor: Es un dispositivo que muestra al usuario tanto imágenes como texto, y se conecta a través de una tarjeta gráfica conocida como adaptador o tarjeta de vídeo [47].

2.1.12. Software

Este aspecto adquiere gran relevancia, ya que todo el funcionamiento del sistema está íntimamente ligado a la correcta programación, instalación y configuración del software en el dispositivo destinado a llevar a cabo las tareas requeridas. En la tabla 11 se menciona algunos de los lenguajes más comunes:

Tabla 11. Software más común para el procesamiento de imágenes [34].

Python	Es uno de los lenguajes más populares y ampliamente utilizado en el campo del Machine Learning. Ofrece una amplia variedad de bibliotecas y frameworks, como TensorFlow, Keras, PyTorch y scikit-learn, que facilitan la implementación de algoritmos de Machine Learning y el procesamiento de datos
Matlab	Es un software de programación y entorno de desarrollo utilizado en aplicaciones de inteligencia artificial (IA). Proporciona una amplia gama de herramientas y funciones especializadas para el análisis, procesamiento y visualización de datos, lo que lo convierte en una herramienta poderosa para el desarrollo de algoritmos de IA
Julia	Es un lenguaje de programación diseñado específicamente para el análisis numérico y científico. Tiene un rendimiento comparable a C++ y Python, y cuenta con bibliotecas como Flux y MLJ para el desarrollo de modelos de Machine Learning. Los programas escritos en Julia se compilan eficientemente en código nativo en diversas plataformas gracias al esquema desarrollado por LLVM (Low-Level Virtual Machine)

2.1.13. Comparativa de Software

En la tabla 12, se realiza la comparación de las características técnicas de software con enfoque a Inteligencia Artificial

Tabla 12. Comparación de características técnicas de software con enfoque a Inteligencia Artificial

CARACTERÍSTICAS TÉCNICAS	MATLAB	JULIA	PYTHON
Aprendizaje Automático	Amplia colección de funciones y toolboxes	Paquetes y librerías como Flux, MLJ, Scikit-learn	Bibliotecas populares como TensorFlow, PyTorch
Procesamiento de Imágenes	Toolbox de Procesamiento de Imágenes	Paquetes como Images.jl, JuliaImages	Bibliotecas como OpenCV, scikit-image, PIL
Procesamiento de Texto	Funciones de procesamiento de texto.	Librerías como TextAnalysis.jl, NLP.jl	Bibliotecas populares como NLTK, SpaCy, Gensim
Visión por Computadora	Funciones de procesamiento y análisis de imágenes	Paquetes como ImageFeature.jl, JuliaCV	Bibliotecas como OpenCV, scikit-image
Redes Neuronales	Toolbox de Redes Neuronales	Paquetes como Flux, Knet, TensorFlow.jl	Bibliotecas populares como TensorFlow, PyTorch
Procesamiento de Datos	Funciones para manipulación y análisis de datos	Paquetes como DataFrames.jl, Query.jl	Bibliotecas populares como pandas, NumPy, SciPy
Rendimiento	Rápido y eficiente para cálculos matemáticos	Rendimiento competitivo	Rendimiento competitivo
Comunidad y Soporte	Gran comunidad y soporte de la industria	Comunidad creciente	Gran comunidad y amplio soporte

Elaborado por el investigador

Una vez realizada la comparación de características técnicas de software con enfoque a Inteligencia Artificial, se concluye que, Matlab es conocido por su enfoque en el análisis numérico y tiene una buena integración con herramientas de visualización. Julia es un lenguaje rápido y escalable, con una comunidad en crecimiento y librerías de IA. Python tiene un amplio ecosistema de IA y es muy popular en el campo, ofreciendo una combinación de facilidad de uso y rendimiento.

Python es la elección ideal para cumplir con los requerimientos debido a su capacidad para ejecutarse de manera más rápida en comparación con otros lenguajes de programación. Esto se debe a su enfoque en el uso de scripts para la ejecución, lo que resulta en un menor tiempo de procesamiento. Esta característica única de Python se convierte en una ventaja significativa para el sistema de monitoreo, ya que permite el procesamiento de imágenes en un período de tiempo considerablemente menor en comparación con otro software de programación disponibles.

2.1.14. Algoritmos de la red neuronal

Para el desarrollo de la red neuronal del sistema de reconocimiento de indicadores de somnolencia se han utilizado específicamente tres algoritmos de IA clave, los cuales se especifican en la tabla 13.

Tabla 13. Algoritmos de la red neuronal

Eigenfaces	Técnica de reconocimiento facial basada en el análisis de componentes principales. Representa rostros como un conjunto de "eigenfaces" que capturan las variaciones más significativas en la apariencia facial. Es eficiente, pero puede ser sensible a variaciones en iluminación y expresiones faciales.
Local Binary Patterns (LBP)	Método de descripción de texturas utilizado en el reconocimiento facial. Extrae patrones locales de una imagen y los representa como histogramas de patrones binarios. Es robusto a variaciones en iluminación y pose, y puede capturar características locales relevantes en un rostro.
Redes neuronales convolucionales (CNN)	Modelos de aprendizaje profundo utilizados en el reconocimiento facial. Estas redes están compuestas por capas convolucionales que aprenden características jerárquicas en los rostros. Son altamente efectivas para la clasificación de imágenes y han logrado grandes avances en el reconocimiento facial debido a su capacidad para aprender representaciones complejas y su robustez ante variaciones.

Elaborado por el investigador

2.1.15. Cloud Computing

El Cloud Computing, o computación en la nube, es una tecnología revolucionaria que ha transformado la forma en que se almacena, procesa y accede a datos e información. En lugar de depender de recursos locales, como servidores y discos duros, el Cloud

Computing utiliza una red de servidores remotos y poderosos que se encuentran distribuidos en diferentes ubicaciones geográficas [48].

2.1.16. Microsoft Azure

Microsoft Azure es una plataforma de computación en la nube que ofrece una amplia gama de servicios y herramientas para el desarrollo, implementación y gestión de aplicaciones de inteligencia artificial (IA). Proporciona servicios de infraestructura, almacenamiento, bases de datos y análisis de datos, así como servicios específicos para IA, como Azure Machine Learning y Cognitive Services. Estos servicios permiten a los usuarios construir, entrenar y desplegar modelos de aprendizaje automático, así como implementar aplicaciones inteligentes que utilizan técnicas como el reconocimiento de voz, la visión por computadora y el procesamiento del lenguaje natural [49] [50].

2.1.17. Google Cloud Platform (GCP)

(GCP) es una plataforma de computación en la nube que brinda una variedad de servicios y herramientas para aplicaciones de inteligencia artificial (IA). Ofrece servicios de infraestructura escalables, almacenamiento y bases de datos, así como servicios específicos para IA, como Google Cloud Machine Learning Engine y TensorFlow. Estos servicios permiten a los usuarios desarrollar, entrenar y desplegar modelos de IA, así como aprovechar algoritmos preentrenados y API de aprendizaje automático de Google para tareas como reconocimiento de imágenes, traducción de idiomas y análisis de sentimientos. Ofrece capacidades avanzadas de procesamiento de datos y análisis, lo que permite la extracción de información valiosa de grandes volúmenes de datos para impulsar la IA [50].

2.1.18. Amazon Web Services (AWS)

AWS es una plataforma líder en servicios en la nube que ofrece una amplia gama de soluciones para aplicaciones de inteligencia artificial (IA). Entre sus servicios, destacan cómputo escalable, almacenamiento, bases de datos, análisis de datos y servicios

específicos para IA, como Amazon SageMaker y Amazon Rekognition. Estas herramientas permiten a los usuarios desarrollar, entrenar y desplegar modelos de IA, aprovechando la infraestructura escalable de AWS para un procesamiento eficiente. Además, AWS brinda servicios de automatización, orquestación y procesamiento de datos avanzados, lo que posibilita el análisis de grandes volúmenes de información para impulsar la IA y la toma de decisiones informadas [50].

2.1.19. Comparativa de plataformas Cloud Computing

Se llevó a cabo un análisis de los principales parámetros para la creación de servidores web en la nube con el fin de instalar el servidor encargado de recopilar los datos enviados desde el controlador central. En la Tabla 14, se presentan en detalle las características principales de cada una de estas plataformas.

Tabla 14. Cuadro comparativo de plataformas Cloud Computing

Parámetros	Microsoft Azure	Google Cloud Platform (GCP)	Amazon Web Services (AWS)
Almacenamiento	Azure Blob Storage Azure Disk Storage Azure Files	Google Cloud Storage Google Cloud Persistent Disk Google Cloud Filestore	Amazon S3 (Simple Storage Service) Amazon EBS (Elastic Block Store) Amazon Glacier
Máquinas virtuales	Azure ofrece Azure Virtual Machines	GCP ofrece Google Compute Engine	AWS proporciona Elastic Compute Cloud (EC2)
Bases de datos	Azure SQL Database Azure Cosmos DB (base de datos NoSQL) Azure Synapse Analytics (data warehouse) Azure Database for MySQL/PostgreSQL	Cloud SQL (MySQL y PostgreSQL) Cloud Firestore (base de datos NoSQL) BigQuery (data warehouse) Cloud Spanner (base de datos relacional globalmente distribuida).	Amazon RDS (Relational Database Service) Amazon DynamoDB (base de datos NoSQL) Amazon Redshift (data warehouse) Amazon Aurora (base de datos relacional).
Sistemas operativos	Windows Server Linux macOS FreeBSD	Windows Server Linux macOS	Windows Server Linux macOS FreeBSD

Elaborado por el investigador

Después de considerar los requerimientos específicos del proyecto, se decidió elegir el servicio de Cloud Computing de Microsoft Azure, ya que es una mejor opción en comparación con Google Cloud Platform (GCP) o Amazon Web Services (AWS) para el proyecto, debido a su enfoque integral en soluciones de inteligencia artificial y aprendizaje automático. Microsoft Azure ofrece una ventaja única al permitir el registro con una cuenta de correo institucional, lo que otorga un crédito de $100 durante un año, además ofrece servicios de aprendizaje automático, procesamiento de lenguaje natural, visión por computadora y más, que permiten a los desarrolladores crear y desplegar modelos de inteligencia artificial de manera eficiente.

2.1.20. Modo de ejecución- API de reconocimiento facial de Azure

Para utilizar el reconocimiento facial en Azure, debes seguir los siguientes pasos generales [49]:

- Crear una cuenta de Azure y obtener las credenciales para acceder al API.
- Configurar tu entorno de desarrollo para interactuar con el API, utilizando el lenguaje de programación de tu elección.
- Enviar solicitudes HTTP al API para realizar tareas específicas, como detección de rostros, comparación o entrenamiento de modelos.
- Procesar las respuestas del API para obtener los resultados deseados, como los rostros detectados, los descriptores faciales o los puntajes de coincidencia.

2.1.21. Herramientas de desarrollo y gestión de datos

2.1.22. Jupyter

Jupyter es una plataforma de código abierto que permite crear y compartir documentos interactivos que combinan código, visualizaciones y texto. Los usuarios pueden escribir y ejecutar fragmentos de código en diferentes lenguajes de programación, ver los resultados y agregar explicaciones en formato de texto enriquecido [51].

2.1.23. Anaconda

Anaconda es una plataforma de distribución y gestión de paquetes de código abierto utilizada principalmente en el campo de la ciencia de datos y la computación científica. Proporciona un entorno de desarrollo completo que incluye el lenguaje de programación Python, así como una amplia gama de bibliotecas y herramientas populares para el análisis de datos, el aprendizaje automático y la visualización. Con Anaconda, los usuarios pueden instalar, actualizar y administrar fácilmente paquetes y entornos de Python, lo que facilita la configuración y la colaboración en proyectos de ciencia de datos [52].

2.1.24. Herramientas para desarrollar el entrenamiento

Para desarrollar el entrenamiento de modelos de aprendizaje automático, se utilizan diversas herramientas y recursos que facilitan el proceso.

2.1.25. TensorFlow Playground

Es una herramienta interactiva en línea desarrollada por Google para experimentar y aprender sobre redes neuronales sin necesidad de escribir código. Permite ajustar los hiperparámetros, como el número de capas, unidades y funciones de activación, y ver cómo estos cambios afectan el comportamiento del modelo en tiempo real. Es una excelente opción para principiantes que deseen comprender los conceptos básicos de las redes neuronales [53].

2.1.26. Amazon SageMaker

Amazon SageMaker es un servicio de aprendizaje automático completamente administrado por AWS (Amazon Web Services). Proporciona un entorno de desarrollo integrado y escalable para construir, entrenar e implementar modelos de aprendizaje automático, incluyendo redes neuronales. SageMaker ofrece capacidades avanzadas de optimización, selección de modelos y despliegue en producción, lo que lo convierte en una opción sólida para proyectos más grandes que requieran infraestructura y recursos escalables en la nube [54].

2.1.27. Colaboratory (Colab)

Google Colab es un entorno de bloc de notas Jupyter en la nube que permite crear y ejecutar código Python, incluyendo modelos de redes neuronales con TensorFlow y otras bibliotecas de aprendizaje automático [55].

2.1.28. Comparativa de herramientas para desarrollar el entrenamiento

Se llevó a cabo un estudio de las principales características de herramientas para desarrollar el entrenamiento de redes neuronales. En la Tabla 15, se presentan en detalle las características principales de cada una de estas plataformas.

Tabla 15. Cuadro comparativo de herramientas para desarrollar el entrenamiento

TensorFlow Playground	• Interfaz gráfica interactiva y amigable para ajustar los hiperparámetros de la red neuronal. • Visualización en tiempo real del comportamiento del modelo con diferentes configuraciones. • No se requiere conocimiento previo de programación.
Amazon SageMaker	• Ambiente completamente administrado para entrenamiento de modelos de aprendizaje automático. • Soporte para múltiples frameworks de aprendizaje automático, incluyendo TensorFlow y PyTorch. • Escalabilidad y flexibilidad para manejar grandes conjuntos de datos y modelos complejos. • Opciones de implementación en producción para desplegar modelos en aplicaciones en tiempo real.
Colaboratory (Colab)	• Entorno de desarrollo interactivo basado en bloc de notas Jupyter en la nube. • Acceso gratuito a GPU y TPU para acelerar el entrenamiento de modelos. • Facilita la colaboración y el intercambio de código con otros usuarios. • Integración con Google Drive para almacenar y cargar datos y modelos.

Elaborado por el investigador en base a [53] [54] [55]

Tras analizar detenidamente los requisitos particulares del proyecto, se destacó Amazon SageMaker, como la mejor opción, debido a su amplia gama de características avanzadas y su enfoque completo en el aprendizaje automático a escala empresarial. Su soporte para múltiples frameworks, incluyendo TensorFlow y PyTorch, permite una mayor

flexibilidad en la elección de las tecnologías de desarrollo. Además, la escalabilidad y la capacidad de manejar grandes conjuntos de datos y modelos complejos en la nube son atributos fundamentales para el proyecto, ya que requieren un procesamiento intensivo.

2.1.29. Entrenamiento utilizando los servicios de Amazon Web Services (AWS)

El entrenamiento utilizando los servicios de Amazon Web Services (AWS) brinda a las organizaciones y desarrolladores una plataforma potente y versátil para el aprendizaje automático y la inteligencia artificial. AWS ofrece una amplia gama de herramientas y servicios, como Amazon SageMaker, que simplifican y agilizan todo el proceso de entrenamiento de modelos, desde la preparación y etiquetado de datos hasta la implementación y optimización de algoritmos. En la tabla 16, se describen los pasos generales para entrenar una red neuronal utilizando AWS.

Tabla 16. Pasos del entrenamiento mediante los servicios de Amazon Web Services (AWS)

Crear una cuenta de AWS	Regístrate en AWS en https://aws.amazon.com/es/ y configura tu cuenta.
Acceder a la consola	Inicia sesión en la Consola de Administración de AWS.
Seleccionar el servicio de aprendizaje automático	Escoge Amazon SageMaker como el servicio para entrenar tu modelo.
Configurar Amazon SageMaker	Crea una instancia de bloc de notas de SageMaker para desarrollar y ejecutar tu código.
Conectar con el bloc de notas	Accede a la instancia de bloc de notas a través del navegador y prepara tu entorno de desarrollo.
Preparar los datos	Carga tus datos en AWS desde un almacén en línea o desde tus archivos locales.
Desarrollar y entrenar el modelo	Utiliza el entorno de Jupyter Notebook para escribir y ejecutar el código del modelo.
Configurar hiperparámetros	Ajusta los hiperparámetros del modelo para el entrenamiento.
Iniciar el entrenamiento	Ejecuta el entrenamiento del modelo y supervisa su progreso.
Optimizar y ajustar	Realiza iteraciones para mejorar el rendimiento del modelo ajustando hiperparámetros y arquitectura.
Guardar y desplegar el modelo	Guarda el modelo entrenado y prepáralo para su implementación en producción.
Gestión de costos	Monitorea los recursos utilizados y cierra instancias cuando no las necesites para controlar costos.

Elaborado por el investigador

2.2. Métodos

2.2.1. Modalidad de investigación

A continuación, se describe los diferentes tipos de investigaciones que se aplicaron para la ejecución del proyecto.

El trabajo investigativo que se presenta tiene la característica de enfocarse a la investigación aplicada, debido a que el mismo busca dar solución al problema que se ha planteado, empleando los conocimientos teóricos y prácticos para poder diseñar el dispositivo electrónico.

Investigación Bibliográfica, se justifica porque nació la necesidad de buscar información principalmente referente a la somnolencia y las distintas formas para obtener una medida de la misma, así como la determinación de diagnósticos preventivos y correctivos.

Investigación de Campo, se aplicó debido a que para poder dar cabida al proyecto es necesario monitorear al usuario, para poder realizar pruebas, se recopiló información suficiente y necesaria para el desarrollo del presente proyecto.

2.2.2. Recolección de información

Para lograr obtener información se empleó los distintos repositorios de las universidades del país, libros, revistas de difusión científica en línea, y demás bases de datos que sean confiables con la finalidad que puedan aportar para el desarrollo del proyecto.

2.2.3. Procesamiento y análisis de datos

Para poder procesar y analizar los datos obtenidos mediante la investigación se procederá a realizar los siguientes pasos:

- Recolección de la información obtenida.
- Revisión de la información disponible.
- Estudio de las diferentes soluciones existentes para la elaboración de dispositivos fiables que permitan recolectar datos y procesarlos.
- Deducción de la información más importante que ayude al correcto desarrollo del proyecto de investigación y que permita dar una solución adecuada a la propuesta

2.2.4. Desarrollo de Proyecto

Para cumplir con los objetivos planteados en el proyecto se llevó a cabo las siguientes actividades:

Tema: SISTEMA DE RECONOCIMIENTO DE INDICADORES DE SOMNOLENCIA MEDIANTE INTELIGENCIA ARTIFICIAL	
Objetivo General: Implementar un sistema de reconocimiento de indicadores de somnolencia mediante inteligencia artificial	
Objetivo Específico	**Actividades**
Documentar los métodos para la adquisición de indicadores de somnolencia.	• Determinación de los métodos para la adquisición de indicadores de somnolencia. • Análisis del comportamiento del usuario ante la somnolencia.
Elaborar algoritmos de visión artificial para la detección de estado de somnolencia	• Selección el software más apropiado para la adquisición de señales y procesamiento de imágenes. • Selección de algoritmo para el desarrollo del sistema reconocimiento de indicadores de somnolencia. • Adquisición de una base de datos etiquetada.
Diseñar un sistema de indicadores de somnolencia.	• Programación de algoritmo para la interpretación de los datos. • Diseño de interfaz gráfica.
Realizar las pruebas de funcionamiento del sistema de detección del estado de somnolencia que permitan la validación de los resultados	• Corrección de errores. • Pruebas de funcionamiento.

CAPÍTULO III

RESULTADOS Y DISCUSIÓN

3.1. Análisis y discusión de los resultados

El sistema de reconocimiento de indicadores de somnolencia mediante inteligencia artificial utiliza algoritmos para analizar y procesar datos como la apertura de los ojos, movimientos de cabeza, bostezos y expresiones faciales. Su objetivo principal es detectar a personas que se encuentren en un estado de somnolencia y emitir alertas para que puedan restablecer su atención y continuar con sus actividades de manera segura.

El desarrollo de este sistema comenzó con la documentación de los métodos de adquisición de indicadores, lo que permitió identificar factores visuales relevantes para la detección de la somnolencia. Los algoritmos de inteligencia artificial empleados lograron una detección precisa y en tiempo real de los estados de somnolencia, con un procesamiento óptimo de las imágenes. Además, se enfocó en diseñar el sistema para garantizar una integración efectiva y una interfaz amigable para los usuarios.

Los resultados del entrenamiento del algoritmo fueron evaluados y validados, demostrando una precisión del 0.96 y un rendimiento del 97.3%. Esto evidencia un excelente desempeño en la detección y clasificación de los casos positivos y negativos dentro del conjunto de datos proporcionado. No obstante, es importante tener en cuenta que el rendimiento del sistema puede variar en diferentes conjuntos de datos y escenarios de prueba, por lo que es esencial realizar pruebas adicionales para asegurar su confiabilidad en diversas situaciones reales.

3.2. Desarrollo de la propuesta

Para el desarrollo del proyecto se realizó una investigación de modalidad investigativa, bibliográfica y de campo. Donde se definió se documentó los métodos de adquisición de indicadores visuales de somnolencia. Se desarrolla un algoritmo del sistema de

reconocimiento de indicadores de somnolencia, el cual se programó en Python debido a la variedad de librerías disponibles, y se utilizó la librería OpenCV para el procesamiento de imágenes debido a su amplia gama de comandos.

3.2.1. Adquisición de indicadores de somnolencia

Los factores visuales como bostezos, ojos cansados, cabeceos, frotarse los ojos, la reducción de los reflejos, la pesadez de los párpados, el hormigueo de los ojos, el deseo de cerrar los ojos en un momento, la necesidad de estirarse tiene lugar cuando la persona está somnolienta y no solamente cansada.

Para el desarrollo del proyecto se tomó en cuenta los factores visuales mencionados en la tabla 17.

Tabla 17. Factores visuales somnolencia

Nivel de apertura de los ojos	Medición del grado de apertura de los ojos para evaluar cambios en la apertura media y detectar signos de somnolencia.
Análisis de expresiones faciales	Reconocimiento y análisis de las expresiones faciales para detectar signos de fatiga, como el entrecerrar de los ojos.
Seguimiento de movimientos de la cabeza	Monitoreo de los movimientos de la cabeza y detección de patrones de cabeceo o movimientos irregulares asociados a la somnolencia.
Bostezos	Detección de bostezos a través del análisis de los movimientos faciales característicos asociados a la somnolencia.

Elaborado por el investigador en base a [57]

En el mapa de coordenadas utilizado por Dlib, una biblioteca de código usada en Python para la detección y el reconocimiento de rostros, en donde cada ojo se encuentra representado por un conjunto de 6 coordenadas, mientras que la boca está definida por un

conjunto de 20 puntos. Esta configuración de puntos permite una representación detallada y precisa de las características faciales, facilitando así el análisis y reconocimiento de los ojos y la boca en imágenes o videos.

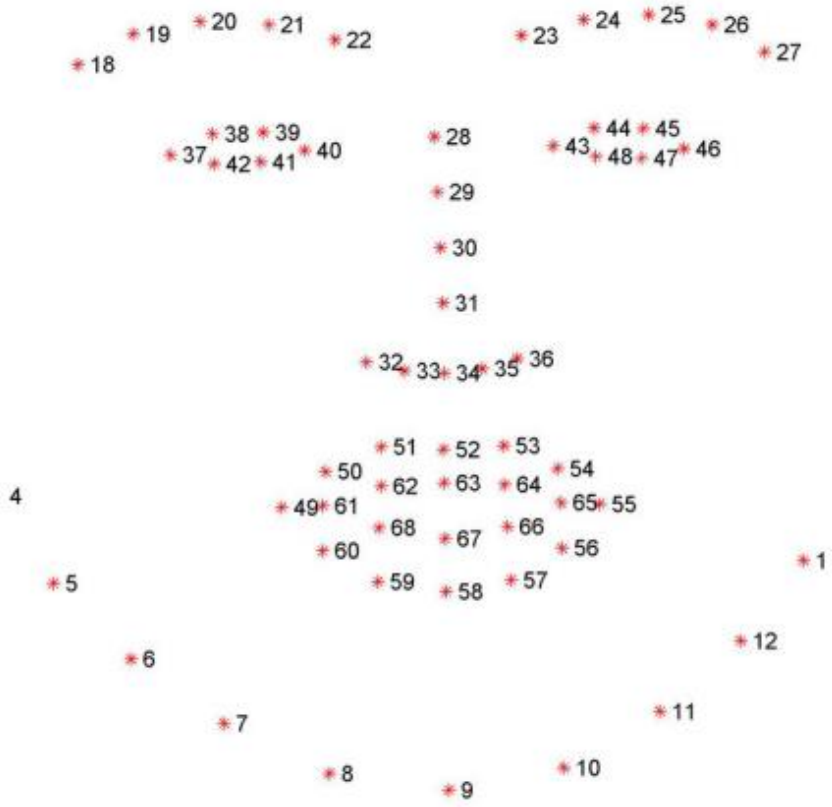


Figura 13. Visualización de las coordenadas faciales Dlib [56]

Bostezos

Al investigar en línea sobre métodos para distinguir entre una boca abierta y una boca cerrada, se descubre la existencia del parámetro MAR (Mouth Aspect Ratio). Este indicador representa la proporción entre la longitud y el ancho de la boca:

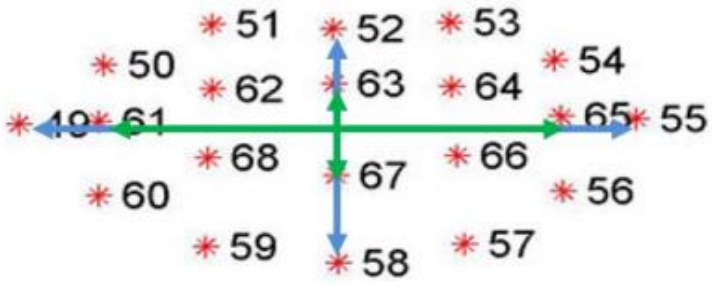


Figura 14. Extracto de las coordenadas de la boca [56]

Figura 15. Extracto de las coordenadas de los ojos [56]

Estos indicadores fueron adquiridos mediante el procesamiento de imágenes capturadas por una cámara y su análisis utilizando algoritmos de inteligencia artificial. La cámara registró el comportamiento visual de la persona y la IA procesó y clasificó los patrones relevantes asociados a la somnolencia.

Análisis de expresiones faciales

Para el análisis de expresiones faciales se tomó en cuenta los puntos clave faciales más comunes generados mediante el mapa de coordenadas de Dlib. El mapa de coordenadas utilizado por Dlib incluye 68 puntos clave faciales. Estos puntos se utilizan para identificar diversas características faciales, como los ojos, las cejas, la nariz, la boca, las mejillas y la mandíbula. Cada punto se representa mediante un par de coordenadas (x, y), que indican su posición en píxeles en la imagen de la cara. El primer punto (índice 0) se encuentra en la esquina izquierda del ojo izquierdo, y los puntos siguientes se numeran en un patrón específico en todo el rostro. En la siguiente tabla 18 se describen los puntos utilizados.

Tabla 18. Análisis de expresiones faciales

Puntos	Descripción
0-16	Representan las cejas y la frente.
17-21	Ubican el ojo izquierdo.
22-26	Ubican el ojo derecho.
27-30	Identifican la nariz.
31-35	Representan la parte superior del labio superior.
36-41	Localizan la parte superior del labio inferior.
42-47	Señalan la parte inferior del labio inferior.
48-67	Definen la mandíbula y las mejillas.

Elaborado por el investigador

Seguimiento de movimientos de la cabeza

Para el desarrollo del proyecto se tomó en cuenta la inclinación y rotación de la cabeza de una persona en relación con un punto de referencia, como el plano horizontal o la gravedad. Los ángulos generalmente se miden en grados o radianes y pueden ayudar a determinar la posición y la orientación de la cabeza. En tabla 19, se describen los ángulos utilizados en el seguimiento de movimientos de la cabeza.

Tabla 19. Seguimiento de movimientos de la cabeza

Ángulo	Rango Típico (grados)	Descripción
Roll	-180° a 180°	Inclinación lateral de la cabeza. 0° cuando la cabeza está vertical, positivo hacia la derecha y negativo hacia la izquierda.
Pitch	-90° a 90°	Inclinación frontal de la cabeza. 0° cuando la cabeza está vertical y mirando hacia adelante. Positivo hacia adelante (flexión), negativo hacia atrás (extensión).

Yaw	-180° a 180°	Giro de la cabeza. 0° cuando la cabeza está mirando directamente hacia adelante. Positivo hacia la derecha, negativo hacia la izquierda.

Elaborado por el investigador

En las figuras 16, 17 y 18 se visualiza los ángulos de seguimiento de movimientos de cabeza.

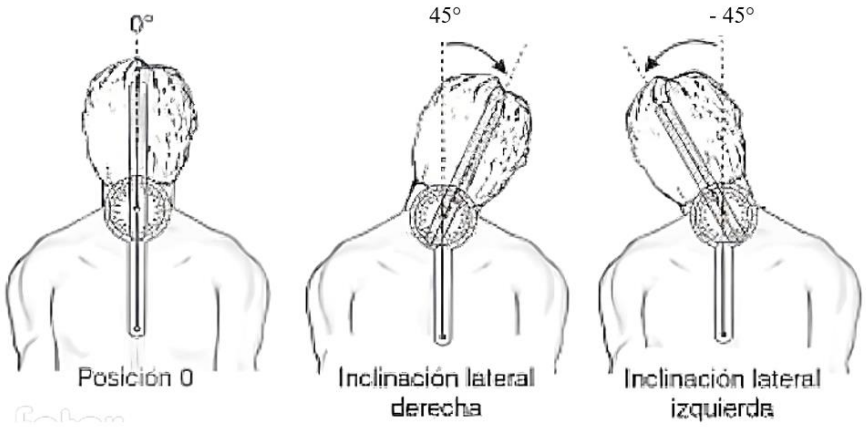


Figura 16. Ángulo de Roll [58]

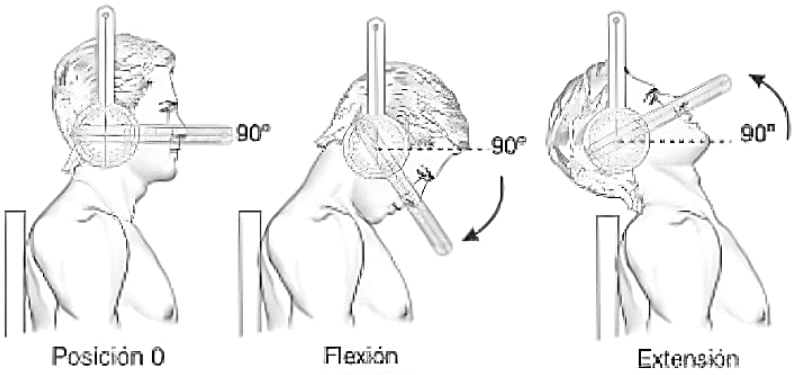


Figura 17. Ángulo de Pitch [58]

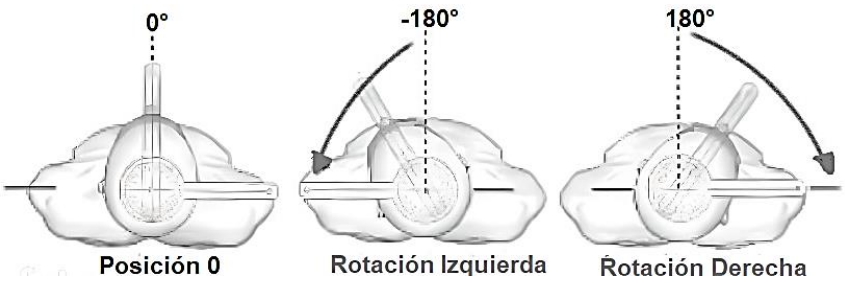


Figura 18. Ángulo de Yaw [58]

3.2.2. Arquitectura del sistema

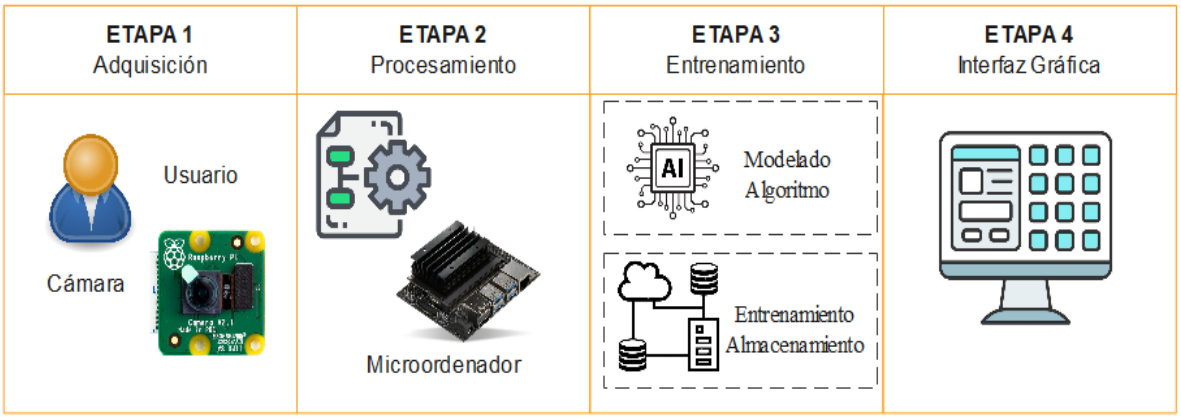


Figura 19. Esquema General Sistema de reconocimiento de indicadores de somnolencia mediante inteligencia artificial

Elaborado por el investigador

En la figura 19, se muestra el esquema general del sistema de reconocimiento de indicadores de somnolencia el cual está conformado por 4 etapas que a continuación, se explican de manera general.

Etapa 1.- El sistema inicia en la adquisición en tiempo real de imágenes o videos mediante una cámara incorporada. La cámara seleccionada captura las imágenes o videos,

Etapa 2.- Los datos adquiridos son enviados al microordenador (Jetson Nano) para su procesamiento.

Etapa 3.- Se emplearon técnicas avanzadas de inteligencia artificial, como redes neuronales, para entrenar un modelo que tiene la capacidad de reconocer de manera precisa los indicadores de somnolencia. Para su despliegue y uso práctico, se implementó en un entorno de servicio de alojamiento en la nube.

Etapa 4.- El modelo realiza la evaluación de nuevas muestras de datos y lleva a cabo una clasificación precisa de la somnolencia del individuo. Los resultados de esta clasificación pueden ser visualizados de manera conveniente a través de una interfaz intuitiva y fácil de usar.

Diagrama de bloques

A continuación, se proporciona una descripción detallada del algoritmo representado en el diagrama de flujo presentado en la figura 21.

El algoritmo del sistema comienza capturando imágenes o videos en tiempo real y aplica un preprocesamiento para detectar rostros. En caso de que no se detecte ningún rostro, el sistema realiza una búsqueda adicional hasta encontrar uno. Si se detecta un rostro, el algoritmo continúa con la detección de la apertura de los ojos, evaluando tanto el retardo del parpadeo como el nivel de apertura de los ojos. Luego, el algoritmo procede a la detección de bostezos, buscando patrones de movimiento característicos asociados a los bostezos. Posteriormente, se realiza la detección de expresiones faciales para identificar indicadores de somnolencia, como cambios en la boca o expresiones faciales. Si se identifica una expresión de somnolencia, el programa activa una alarma auditiva como señal de advertencia. En caso de que en cualquier punto del proceso no se reconozca un rostro, el sistema regresa al inicio para seguir realizando la detección y garantizar una monitorización continua y precisa de los indicadores de somnolencia.

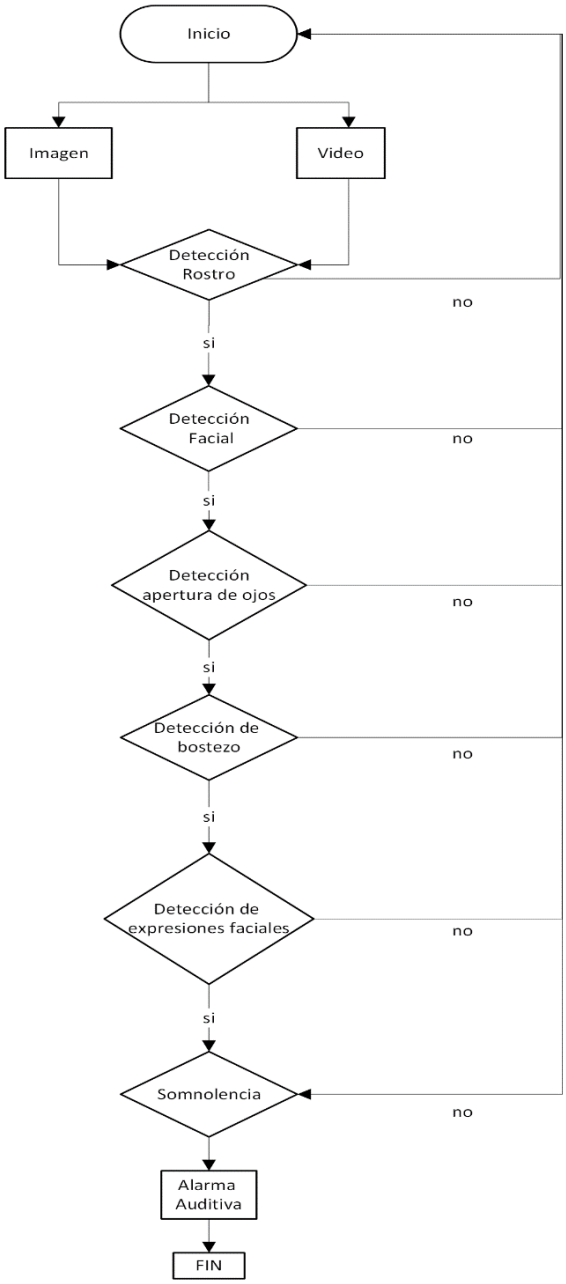


Figura 20. Diagrama general del algoritmo del sistema

Elaborado por: La investigadora

3.2.3. Diseño del sistema

3.2.3.1. Posicionamiento Sistema-Usuario

Para lograr un enfoque total del usuario con la lente estándar incluida en la cámara Pi NoIR V2, es fundamental tener en cuenta que la distancia mínima de enfoque típica de la cámara al usuario es de 30 centímetros (aproximadamente 12 pulgadas). Esto significa que la cámara es capaz de capturar una imagen clara y enfocada del usuario cuando este

se encuentre a una distancia de la pantalla de al menos 30 centímetros de la lente. Esta distancia óptima garantiza que la cámara pueda captar los detalles faciales y expresiones del usuario.

El escenario del sistema de reconocimiento de indicadores de somnolencia es similar a la figura 20.

Figura 21. Escenario del sistema de reconocimiento de indicadores de somnolencia

Elaborado por el investigador

3.2.3.2. Adquisición y procesamiento de datos

Adquisición

La captura de la imagen se realiza utilizando la cámara Pi NoiR. En la figura 22, se muestra el diagrama de flujo que describe el proceso de adquisición de la imagen.

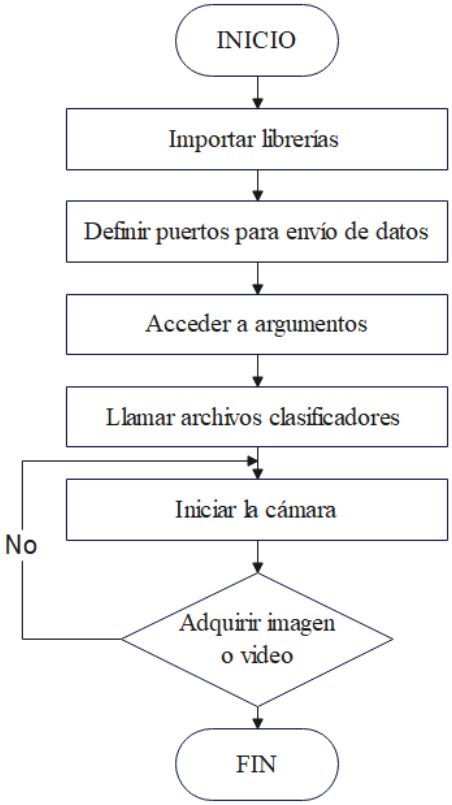


Figura 22. Diagrama de flujo para la adquisición de la imagen o video

Elaborado por: La investigadora

Importar las librerías: Con el fin de realizar la programación en tiempo real, se hizo uso de bibliotecas específicas de Python.

- Pillow
- dlib
- opencv-contrib-python
- flask
- tf-nightly
- tensorflow

Definir puertos para envió de datos: Para habilitar la cámara, se estableció el puerto 5000 como configuración.

```
app.run(host='0.0.0.0', port=5000, debug=True)
```

Acceder a los argumentos: En Python, se accedió a los argumentos pasados a través de la consola mediante la lista sys.argv.

```
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", required=True,
help="path to input image")
```

Llamar a archivos clasificadores:

```
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.models import load_model
```

Inicializar la cámara y configurarla: El programa se adapta a la resolución de la cámara.

```
cap = cv2.VideoCapture(0)
```

Adquisición de la imagen: Una vez que la cámara ha sido inicializada, se lleva a cabo la captura de imágenes o videos. Este proceso implica recibir las imágenes de vista previa y transmitirlas lo más rápido posible para permitir el procesamiento posterior de la imagen.

```
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.models import load_model
```

Procesamiento de datos

Después de capturar la escena, las imágenes se almacenan por defecto en canales RGB (Rojo, Verde y Azul, sus siglas en ingles). A continuación, se inicia el proceso de procesamiento de la imagen, el cual implica convertirla a escala de grises y aplicar un suavizado para mejorar el contraste mediante la ecualización de histogramas. Esta técnica ajusta las intensidades de los píxeles para obtener una imagen más balanceada. Una vez que la imagen ha sido suavizada, se realiza la segmentación para identificar características similares utilizando clasificadores. Posteriormente, se lleva a cabo la detección de signos de somnolencia, como la apertura de los ojos, bostezos el asentamiento de la cabeza y el

uso del teléfono celular. En la figura 23, se muestra el diagrama de flujo del procesamiento de datos

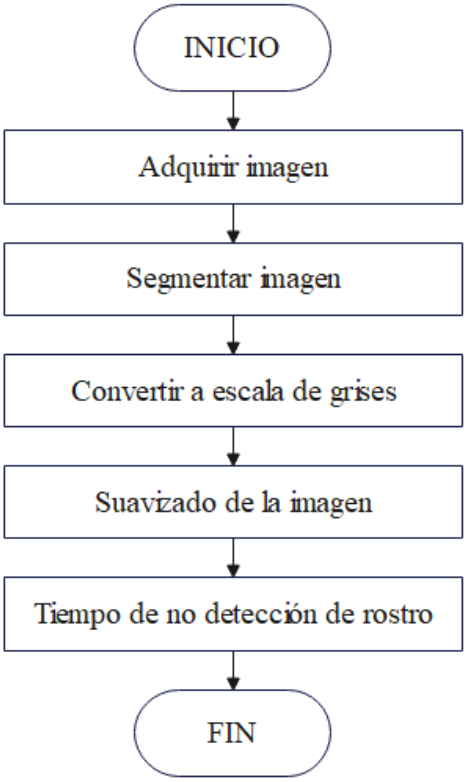


Figura 23. Diagrama de flujo para procesamiento de datos

Elaborado por: La investigadora

Después de capturar la imagen, ésta se almacena por defecto en los canales RGB (Rojo, Verde y Azul, según las siglas en inglés). Sin embargo, para realizar ciertas operaciones es necesario convertir la imagen a escala de grises. Para lograr esto, se utiliza la función cvtColor que permite la conversión de la imagen de un espacio de color a otro. En este caso, se requiere convertir la imagen original del espacio de color RGB a escala de grises. Para ello, se llama a la función cvtColor proporcionándole la imagen original. Para convertir la imagen original a una en escala de grises se utiliza el siguiente comando de OpenCV:

```
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

Siendo:

frame: imagen de entrada al sistema con tres canales de color

gray: imagen convertida a escala de grises

Figura 24. Imagen original RGB frente a imagen convertida a escala de grises.

Elaborado por el investigador

Python ofrece diversas funciones para mejorar el contraste de una imagen, razón por la cual se utiliza la ecualización de histogramas para modificar las intensidades de los píxeles. En OpenCV, se dispone de la función equalizeHist(), la cual permite aplicar fácilmente la ecualización de histogramas a una imagen. Además, esta técnica se puede aplicar de manera sencilla en imágenes en escala de grises.

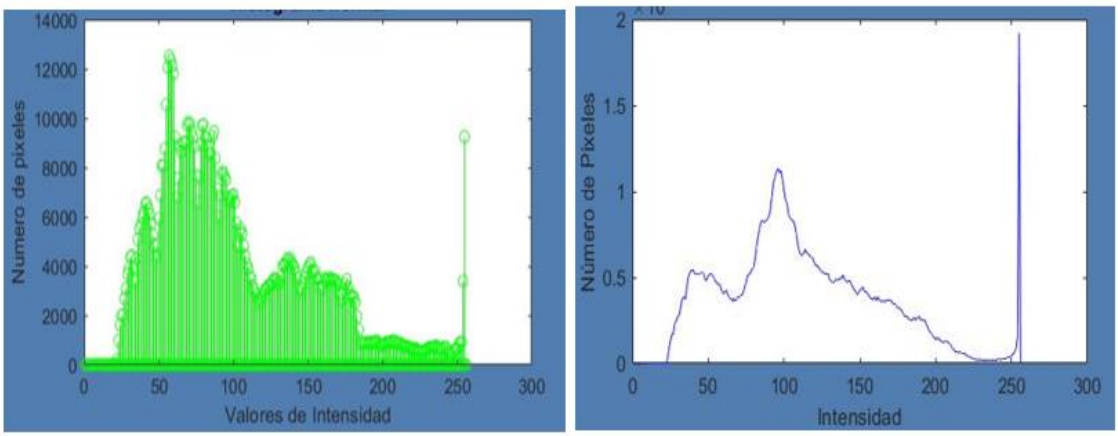


Figura 25. Ecualización de histogramas

Elaborado por el investigador

3.2.3.3. Capas del algoritmo

Esta arquitectura de red neuronal convolucional fue diseñada para extraer y aprender características relevantes de imágenes mediante capas convolucionales y max pooling, y luego clasificarlas utilizando una capa dense. Este tipo de arquitectura ha demostrado ser efectiva para diversas tareas de visión por computadora, como clasificación de imágenes, detección de objetos y segmentación semántica, entre otros. A continuación, se observa en la figura 26 y se describe se explica el propósito y función de cada una de las capas mencionadas:

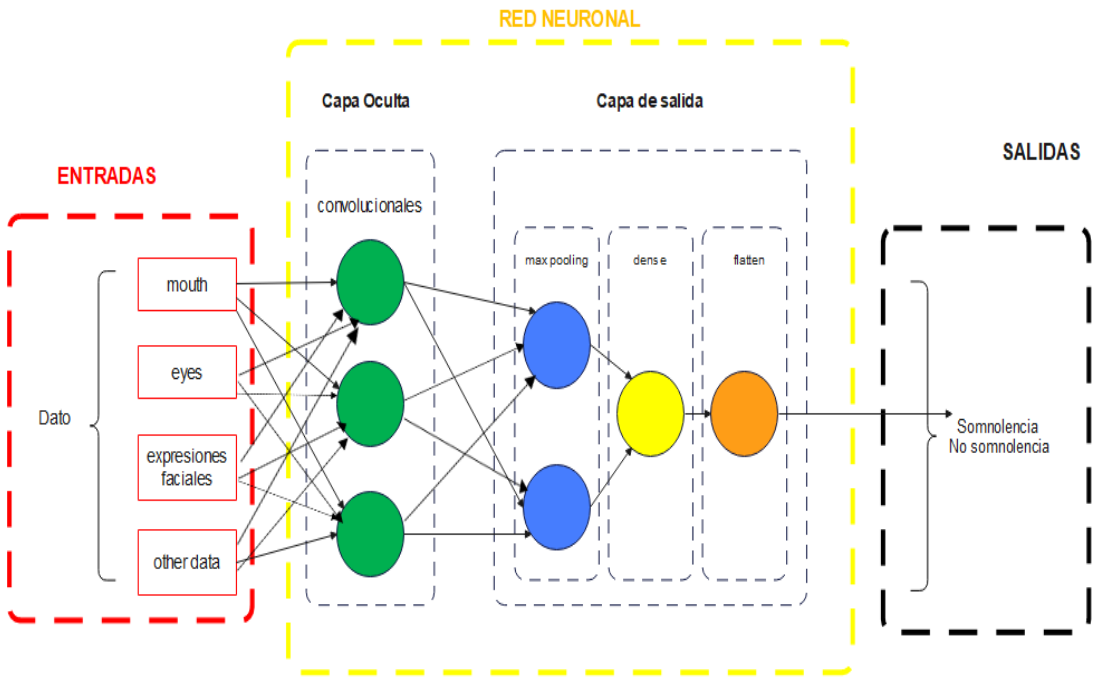


Figura 26. Capas del algoritmo

Elaborado por el investigador

3 capas convolucionales: Cada capa convolucional aplica un conjunto de filtros (kernels) a la imagen de entrada para extraer características relevantes, como bordes, formas o texturas. Cada filtro se desliza por la imagen mediante una operación de convolución, produciendo mapas de características que resaltan patrones específicos. Las capas

convolucionales permiten aprender características más complejas a medida que se profundiza en la red.

2 capas max pooling: Las capas de max pooling se utilizaron para reducir la dimensionalidad espacial de los mapas de características generados por las capas convolucionales. Estas capas toman ventanas y toman el valor máximo dentro de cada ventana, reduciendo así la resolución de la imagen, pero manteniendo las características más importantes.

1 capa dense: La capa dense (totalmente conectada) es una capa tradicional de una red neuronal. En esta capa, todas las neuronas están conectadas a todas las neuronas de la capa anterior. La capa dense se utiliza para realizar tareas de clasificación y toma como entrada las características extraídas por las capas convolucionales y de max pooling. La capa dense aprende patrones más abstractos y realiza la clasificación final del problema.

1 capa flatten: Antes de conectar las capas densas, es necesario aplanar los mapas de características de las capas convolucionales y max pooling para que puedan ser interpretados como un vector unidimensional. La capa flatten se encarga de convertir la información tridimensional en una forma lineal o plana que puede ser alimentada a la capa dense para la clasificación.

3.2.4. Análisis de la base de datos

Para el presente trabajo de investigación, fue necesario contar con una base de datos que incluyera una gran cantidad de imágenes y ciertas propiedades descriptivas asociadas a cada una de ellas. Por lo tanto, se requería tener esta base de datos etiquetada o con la capacidad de ser etiquetada, con el fin de obtener todos los datos necesarios para la clasificación y poder interpretar con precisión lo que cada dato representaba. De esta manera, se pudo obtener valores reales para luego analizar los resultados de manera efectiva. La abundancia de datos ayuda a evitar el sobreajuste o sobreentrenamiento de los modelos durante el proceso de entrenamiento. Además, contar con una gran cantidad

de datos permite realizar una validación cruzada en múltiples iteraciones durante el entrenamiento, lo que contribuye a mejorar la confiabilidad y el rendimiento general del sistema.

Para la base de datos, se usó las siguientes herramientas de acceso libre y gratuito:

MRL Eye Dataset

Este conjunto de datos consiste en una amplia colección de imágenes del ojo humano. Se trata de un dataset de gran escala que ofrece una extensa variedad de imágenes oculares para su análisis y procesamiento.

UC Irvine

Repositorio conocido en el campo de la inteligencia artificial que ofrece diversas bases de datos de diferentes características. Para el presente proyecto se tomó en cuenta bases de datos enfocadas en rostros, bostezos, movimientos de cabeza, expresiones faciales.

3.2.5. División de conjunto de datos

Una vez que se ha creado el dataset de manera adecuada, se procede a generar la estructura de carpetas necesaria para el proyecto. Se organizaron en subdirectorios correspondientes a los conjuntos de entrenamiento, prueba y validación. De esta forma, se logra una disposición ordenada y eficiente de los datos para facilitar el proceso de entrenamiento y evaluación del modelo. Una práctica común según el autor Alberto Peteiro Gándara, es dividir el conjunto de datos en aproximadamente 80% para entrenamiento y 20% para pruebas [58]. La base cuenta con 3269 datos almacenados

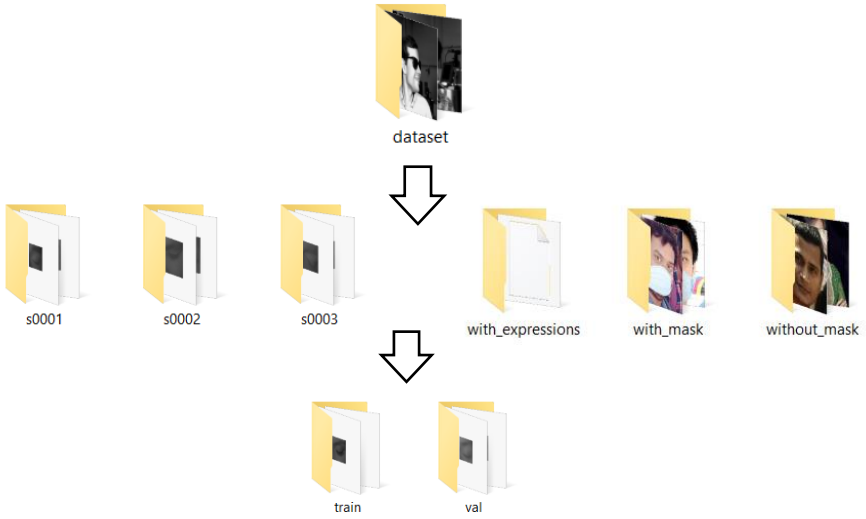


Figura 27. Dataset

Elaborado por el investigador

3.2.6. Programación del algoritmo

Revisar código completo en ANEXO E.

Imagen

El siguiente código es un script de detección facial en imágenes utilizando TensorFlow y OpenCV. Permite detectar rostros en una imagen y clasificarlas. A continuación, se explica el código paso a paso:

1. Importación de paquetes:

Se importa las bibliotecas necesarias para ejecutar el script, incluyendo TensorFlow, OpenCV (cv2), NumPy, argparse y os.

- ***TensorFlow***: Biblioteca de aprendizaje automático de código abierto, utilizada para construir y entrenar modelos de aprendizaje profundo y otras técnicas de inteligencia artificial [34].

- ***OpenCV (cv2):*** Biblioteca de código abierto para visión por computadora y procesamiento de imágenes, que ofrece algoritmos y funciones para análisis de imágenes y videos en tiempo real [34].
- ***NumPy:*** Biblioteca esencial para la computación científica en Python, proporciona estructuras de datos eficientes para trabajar con matrices multidimensionales y realizar operaciones matemáticas y numéricas [34].
- ***argparse:*** Biblioteca estándar de Python para crear interfaces de línea de comandos, permite definir argumentos y opciones de manera sencilla y manejar el análisis de la entrada del usuario [34].
- ***os:*** Biblioteca estándar de Python para interactuar con el sistema operativo, facilita la manipulación de archivos, navegación de directorios y acceso a variables de entorno [34].

```
from tensorflow.keras.applications.mobilenet_v2 import
 preprocess_input
from tensorflow.keras.preprocessing.image import img_t
o_array
from tensorflow.keras.models import load_model
import numpy as np
import argparse
import cv2
import os
```

2. Análisis de argumentos:

Se creó un analizador de argumentos utilizando argparse, que permite al usuario especificar la imagen de entrada, el modelo de detección de rostros, el modelo de detección y un valor de confianza mínimo para filtrar detecciones débiles.

`"-i","--image":` la ruta de la imagen de entrada que se va a analizar.

`"-f","--face"`: la ruta del directorio que contiene el modelo de detección de rostros.

`"-m","--model"`: la ruta del modelo entrenado de detección faciales.

`"-c","--confidence"`: la confianza mínima requerida para filtrar las detecciones débiles.

3. Carga de modelos:

Se cargarón los modelos preentrenados para la detección de rostros y la clasificación de rostros. Estos modelos se encuentran en los archivos "deploy.prototxt" y "res10_300x300_ssd_iter_140000.caffemodel" para la detección de rostros y en el archivo "face_detector.model" para la clasificación de rostros, los cuales se pueden revisar en el ANEXO G.

```
print("[INFO] loading face detector model...")
prototxtPath = os.path.sep.join([args["face"], "deploy.pro
totxt"])
weightsPath = os.path.sep.join([args["face"],
    "res10_300x300_ssd_iter_140000.caffemodel"])
net = cv2.dnn.readNet(prototxtPath, weightsPath)

print("[INFO] loading face face detector model...")
model = load_model(args["model"])
```

4. Lectura y preparación de la imagen:

La imagen de entrada se lee desde el archivo especificado por el usuario y se almacena en la variable ***'image'***.

- Se realiza una copia de la imagen original en la variable orig.
- Se obtienen las dimensiones espaciales de la imagen (h y w).

```
image = cv2.imread(args["image"])
orig = image.copy()
(h, w) = image.shape[:2]
```

5. Preprocesamiento de la imagen:

La imagen se convierte en un objeto binario mediante ***'cv2.dnn.blobFromImage'***, lo que implica normalizar los valores de píxeles y redimensionar la imagen a 300x300 píxeles con valores de substracción.

```
blob = cv2.dnn.blobFromImage(image, 1.0, (300, 300),
    (104.0, 177.0, 123.0))
```

6. **Detección de rostros:**

- La imagen binaria se pasa a través del modelo de detección de rostros, y se obtienen las detecciones.
- Se recorren las detecciones y se filtran aquellas que tienen una confianza mayor al valor especificado por el usuario.

```
net.setInput(blob)
detections = net.forward()
```

7. **Procesamiento de las detecciones:**

- Para cada detección válida, se calculan las coordenadas del cuadro delimitador en la imagen original (***startX, startY, endX, endY***).
- Se extrae el ROI (Region of Interest) del rostro detectado y se prepara para ser clasificado por el modelo de detección.

```
or i in range(0, detections.shape[2]):
    confidence = detections[0, 0, i, 2]

    if confidence > args["confidence"]:
        box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
        (startX, startY, endX, endY) = box.astype("int")

        (startX, startY) = (max(0, startX), max(0, startY))
        (endX, endY) = (min(w - 1, endX), min(h - 1, endY))
```

8. **Clasificación facial:**

- El ROI del rostro se pasa a través del modelo de detección de rostros, y se obtienen las probabilidades.
- La etiqueta de clase (somnoliento o no somnoliento) se determina comparando las probabilidades obtenidas.

- Se asigna un color al cuadro delimitador y al texto de acuerdo a la etiqueta de clase.
- La etiqueta de clase y la probabilidad se muestran sobre el cuadro delimitador en la imagen original.

```
face = image[startY:endY, startX:endX]
face = cv2.cvtColor(face, cv2.COLOR_BGR2RGB)
face = cv2.resize(face, (224, 224))
face = img_to_array(face)
face = preprocess_input(face)
face = np.expand_dims(face, axis=0)
label = "face" if face > withoutface else "No face"
color = (0, 255, 0) if label == "Mask" else (0, 0, 255)

label = "{}:
{:.2f}%".format(label, max(face, withoutface) * 100)
```

9. Mostrar la imagen de salida:

Se muestra la imagen original con los cuadros delimitadores y las etiquetas que indican si se detectó somnolencia o no en cada rostro.

```
cv2.putText(image, label, (startX, startY - 10),
    cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)
cv2.rectangle(image, (startX, startY), (endX, endY), c
olor, 2)
```

10. Esperar la interacción del usuario:

La imagen de salida se muestra hasta que el usuario presione una tecla.

```
cv2.imshow("Output", image)
cv2.waitKey(0)
```

Video

El siguiente código es un script de detección facial en tiempo real a través de la cámara de video. A continuación, se explica el código paso a paso:

1. Importación de paquetes

Se importó las bibliotecas y paquetes necesarios para ejecutar la detección de rostros en tiempo real utilizando la cámara de video.

```
from tensorflow.keras.applications.mobilenet_v2 import pre
process_input
from tensorflow.keras.preprocessing.image import img_to_ar
ray
from tensorflow.keras.models import load_model
import imutils
from imutils.video import VideoStream
import numpy as np
import argparse
import time
import cv2
import os
```

2. Función detect_and_predict_face

Esta función realiza la detección de rostros en un solo fotograma de video. Utiliza los modelos cargados para detectar rostros y predecir si esta somnoliento o no.

```
def detect_and_predict_face(frame, faceNet, faceNet):
    # (El código de la función se omite aquí para
mantener la claridad)
    return (locs, preds)
```

3. Análisis de argumentos

El script utiliza argparse para analizar los argumentos proporcionados en la línea de comandos. Los argumentos son las rutas a los modelos de detección de rostros y clasificación, así como el nivel de confianza mínimo para filtrar las detecciones débiles.

```
ap = argparse.ArgumentParser()
ap.add_argument("-f", "--face", type=str,
    default="face_detector",
    help="path to face detector model directory")
ap.add_argument("-m", "--model", type=str,
    default="mask_detector.model",
    help="path to trained face detector model")
ap.add_argument("-c", "--
confidence", type=float, default=0.5,
    help="minimum probability to filter weak detections")
args = vars(ap.parse_args())
```

4. Carga de modelos

Se cargó los modelos previamente entrenados para la detección de rostros y la clasificación de rostros desde los archivos especificados en los argumentos.

```
print("[INFO] loading face detector model...")
prototxtPath = os.path.sep.join([args["face"], "deploy.pro
totxt"])
weightsPath = os.path.sep.join([args["face"],
    "res10_300x300_ssd_iter_140000.caffemodel"])
faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)

print("[INFO] loading face detector model...")
maskNet = load_model(args["model"])
```

5. Inicialización de la transmisión de video

El script inicializa la transmisión de video utilizando imutils.video.VideoStream y activa el sensor de la cámara. Luego espera unos segundos (2.0) para permitir que la cámara se estabilice antes de comenzar la detección en tiempo real.

```
print("[INFO] starting video stream...")
vs = VideoStream(src=0).start()
time.sleep(2.0)
```

6. Detección y clasificación en tiempo real

El código entra en un bucle while True donde obtiene continuamente fotogramas del flujo de video y realiza la detección y clasificación de mascarillas en cada fotograma utilizando la función detect_and_predict_face. Luego, muestra los resultados en tiempo real mediante cuadros delimitadores y etiquetas sobre los rostros detectados.

```
while True:
    frame = vs.read()
    frame = imutils.resize(frame, height=600, width=800)
    (locs, preds) = detect_and_predict_mask(frame, faceNet
, maskNet)

    # (El código para mostrar las detecciones y
clasificaciones se omite aquí para mantener la claridad)
```

7. Interfaz gráfica

Además de mostrar los cuadros delimitadores y etiquetas, el código también crea una interfaz gráfica combinando imágenes adicionales, como "portada.jpg", "lat.jpg" y "relleno.jpg", para tener un aspecto más estético y visual.

```
 logenc=cv2.imread("portada.jpg")
    loglat=cv2.imread("lat.jpg")
    rell=cv2.imread("relleno.jpg")
    conh=cv2.hconcat([loglat,frame,rell])
    conv=cv2.vconcat([logenc,conh])
    cv2.namedWindow("FISEI", cv2.WINDOW_NORMAL)
    cv2.setWindowProperty("FISEI",cv2.WND_PROP_FULLSCREEN,
cv2.WINDOW_NORMAL)
    cv2.imshow("FISEI", conv)
```

8. Ciclo y terminación

El ciclo while continúa mostrando los resultados hasta que el usuario presione la tecla "q". Cuando eso sucede, se detiene la transmisión de video y se cierran todas las ventanas.

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

vs.stop()
cv2.destroyAllWindows()
os._exit(00)
```

3.2.7. Entrenamiento

Para el entrenamiento de la red neuronal, se empleó dos enfoques diferentes. En primer lugar, se optó por el entrenamiento manual, que involucra el uso de una base de datos interna. Mediante el uso de un script, se accede a dicha base de datos para iniciar el proceso de entrenamiento para cada imagen, siguiendo parámetros preestablecidos. Por otro lado, se utilizó AWS una empresa filial de Amazon, especializada en machine learning.

El uso de AWS presenta varias ventajas significativas en comparación con el enfoque manual. En primer lugar, la base de datos de AWS es considerablemente más amplia, lo que resulta en un menor porcentaje de error. Además, se logra una mayor optimización en la caracterización de los valores durante el proceso de entrenamiento. También, el tiempo de entrenamiento se reduce considerablemente en comparación con el método manual, lo que resulta en una optimización más eficiente del modelo.

Entrenamiento de la Red Manual

El siguiente código es un script que entrena un modelo de detección facial utilizando el conjunto de datos previamente seleccionados. A continuación, se explica el código paso a paso:

1. Importación de paquetes

Se importó todas las bibliotecas y paquetes necesarios para el entrenamiento del modelo, incluyendo TensorFlow, NumPy, Matplotlib, argparse y otros paquetes de utilidades.

```
from tensorflow.keras.preprocessing.image import ImageDataG
enerator
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.layers import AveragePooling2D
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Input
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.applications.mobilenet_v2 import prep
rocess_input
from tensorflow.keras.preprocessing.image import img_to_arr
ay
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
```

2. Análisis de argumentos

Se utilizó argparse para analizar los argumentos proporcionados en la línea de comandos, como el directorio del conjunto de datos, el archivo de trazado de pérdida/precisión y la ruta de salida del modelo entrenado.

```
ap = argparse.ArgumentParser()
ap.add_argument("-d", "--dataset", required=True,
    help="path to input dataset")
ap.add_argument("-p", "--plot", type=str, default="plot.png",
    help="path to output loss/accuracy plot")
ap.add_argument("-m", "--model", type=str,
    default="mask_detector.model",
    help="path to output face mask detector model")
args = vars(ap.parse_args())
```

3. Preparación del conjunto de datos

En esta parte, el código carga las imágenes del conjunto de datos y sus correspondientes etiquetas. Las imágenes se redimensionan a un tamaño de 224x224 y se preprocesan para ser utilizadas con el modelo MobileNetV2. Las etiquetas se codifican utilizando el método *one-hot encoding.*

```
print("[INFO] loading images...")
imagePaths = list(paths.list_images(args["dataset"]))
data = []
labels = []

data = np.array(data, dtype="float32")
labels = np.array(labels)

lb = LabelBinarizer()
labels = lb.fit_transform(labels)
labels = to_categorical(labels)

(trainX, testX, trainY, testY) = train_test_split(data, labels,
    test_size=0.20, stratify=labels, random_state=42)
```

4. Creación del generador de imágenes de entrenamiento

Se utilizó ***ImageDataGenerator´*** para crear un generador de imágenes de entrenamiento que realiza el aumento de datos, lo que ayuda a aumentar la cantidad de datos de entrenamiento y mejorar la generalización del modelo.

```
aug = ImageDataGenerator(
    rotation_range=20,
    zoom_range=0.15,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.15,
    horizontal_flip=True,
   fill_mode="nearest")
```

5. Construcción del modelo

El modelo se construyó utilizando MobileNetV2 preentrenado como base y se agrega una cabeza personalizada con capas completamente conectadas para la clasificación binaria (somnoliento/o no).

```
baseModel = MobileNetV2(weights="imagenet", include_top=Fa
lse,
    input_tensor=Input(shape=(224, 224, 3)))

headModel = baseModel.output
headModel = AveragePooling2D(pool_size=(7, 7))(headModel)
headModel = Flatten(name="flatten")(headModel)
headModel = Dense(128, activation="relu")(headModel)
headModel = Dropout(0.5)(headModel)
headModel = Dense(2, activation="softmax")(headModel)

model = Model(inputs=baseModel.input, outputs=headModel)
```

6. Compilación del modelo

El modelo se compiló utilizando el optimizador Adam y la función de pérdida de entropía cruzada binaria. La métrica de precisión también se utiliza para evaluar el rendimiento del modelo durante el entrenamiento.

```
for layer in baseModel.layers:
    layer.trainable = False

opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
model.compile(loss="binary_crossentropy", optimizer=opt,
   metrics=["accuracy"])
```

7. Entrenamiento del modelo

El modelo se entrenó utilizando el generador de imágenes de entrenamiento y el conjunto de datos de entrenamiento. Se realiza el entrenamiento de la cabeza personalizada, mientras que las capas base de MobileNetV2 permanecen congeladas (no se actualizan).

```
H = model.fit(
    aug.flow(trainX, trainY, batch_size=BS),
    steps_per_epoch=len(trainX) // BS,
    validation_data=(testX, testY),
    validation_steps=len(testX) // BS,
    epochs=EPOCHS
```

8. Evaluación del modelo

Una vez finalizado el entrenamiento, se evaluó el modelo utilizando el conjunto de prueba y se muestra un informe de clasificación que muestra la precisión, recall y f1-score para cada clase (con/sin somnolencia).

```
predIdxs = model.predict(testX, batch_size=BS)
predIdxs = np.argmax(predIdxs, axis=1)
print(classification_report(testY.argmax(axis=1), predIdxs
,
    target_names=lb.classes_))
```

9. Guardado del modelo

El modelo entrenado se guarda en disco en formato h5 para su uso posterior en la detección facial en imágenes o video.

```
model.save(args["model"], save_format="h5")
```

10. Trazado de gráficos de pérdida y precisión

Se traza la pérdida de entrenamiento y precisión a lo largo de las épocas para visualizar el rendimiento del modelo durante el entrenamiento.

```
N = EPOCHS
plt.style.use("ggplot")
plt.figure()
plt.plot(np.arange(0, N), H.history["loss"], label="train_loss")
plt.plot(np.arange(0, N), H.history["val_loss"], label="val_loss
")
plt.plot(np.arange(0, N), H.history["accuracy"], label="train_ac
c")
plt.plot(np.arange(0, N), H.history["val_accuracy"], label="val_
acc")
plt.title("Training Loss and Accuracy")
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")
plt.legend(loc="lower left")
plt.savefig(args["plot"])
```

Entrenamiento modelo AWS

AWS ofrece una variedad de herramientas y servicios que facilitan todo el proceso de entrenamiento y despliegue de modelos de aprendizaje automático en la nube. En la tabla 20, se resume los pasos que se siguieron para el entrenamiento mediante AWS. Revisar código de parámetros del entrenamiento, ANEXO E.

Tabla 20. Pasos a seguir para el entrenamiento mediante AWS

Entrenamiento del modelo	Inicio del proceso de entrenamiento utilizando los datos almacenados y el algoritmo seleccionado.
Ajuste de hiperparámetros	Opcionalmente, ajustar los hiperparámetros del modelo para optimizar el rendimiento.
Evaluación del modelo	Evaluación del modelo utilizando un conjunto de datos de prueba separado para medir su rendimiento.
Despliegue del modelo	Despliegue del modelo en producción para realizar predicciones en tiempo real.
Monitoreo y mantenimiento	Monitoreo y ajustes del modelo en producción para mantener su rendimiento óptimo.

Elaborado por el investigador

1. Importación de datos

Se generó una matriz de datos de reconocimiento facial de 2205 filas x 6000 columnas, en donde las columnas representan los datos binarizados. Se importaron los datos de diferentes parámetros:

- face
- eyes
- mouth
- feelings
- otros datos como movimientos de cabeza y perfil
- datos de predicción

Face_1

	0	1	2	3	4	5	6	7	8	9	...	5990	5991	5992	5993	5994	5995	5996	5997	5998	5999
0	151.47	151.45	151.52	151.27	150.80	150.69	153.89	154.67	152.88	153.82	...	151.16	151.19	151.25	151.16	151.10	151.16	151.14	151.10	151.21	151.19
1	151.11	151.12	151.16	150.92	150.70	150.62	152.40	153.21	152.81	153.53	...	150.82	150.82	150.86	150.80	150.73	150.79	150.84	150.79	150.80	150.86
2	150.81	150.79	150.84	150.65	150.35	150.23	152.03	152.81	152.44	153.27	...	150.49	150.44	150.47	150.46	150.38	150.47	150.50	150.43	150.54	150.62
3	150.48	150.47	150.52	150.31	150.04	149.98	151.63	152.48	152.24	152.94	...	150.34	150.30	150.28	150.38	150.41	150.33	150.31	150.31	150.25	150.28
4	150.41	150.35	150.24	150.12	149.87	149.71	151.64	152.37	151.78	152.68	...	150.31	150.20	150.17	150.28	150.31	150.25	150.27	150.22	150.13	150.19
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2200	151.70	151.83	151.90	151.75	151.62	151.63	153.78	154.73	153.83	154.43	...	151.84	151.75	151.67	151.76	151.81	151.78	151.78	151.83	151.84	151.86
2201	151.90	151.83	151.81	151.77	151.65	151.63	153.97	154.84	153.67	154.37	...	151.81	151.81	151.84	151.83	151.79	151.87	151.87	151.75	151.77	151.78
2202	151.73	151.78	151.84	151.62	151.46	151.49	153.91	154.88	153.69	154.24	...	151.73	151.66	151.73	151.71	151.68	151.79	151.84	151.73	151.73	151.83
2203	151.77	151.77	151.75	151.56	151.47	151.57	154.09	154.80	153.45	154.21	...	151.75	151.71	151.68	151.71	151.73	151.75	151.71	151.68	151.71	151.78
2204	151.83	151.78	151.77	151.65	151.49	151.48	154.00	154.77	153.48	154.29	...	151.70	151.71	151.63	151.64	151.67	151.64	151.70	151.73	151.71	151.70

2205 rows × 6000 columns

```
Importar archivos de reconocimiento facial
```

Figura 28. Importación de datos de reconocimiento facial

Elaborado por el investigador

```
Importar archivos de mouth tracking

mouth_1 = get_files(datos=datos, filename='TS1.txt')
mouth_2 = get_files(datos=datos, filename='TS2.txt')
mouth_3 = get_files(datos=datos, filename='TS3.txt')
mouth_4 = get_files(datos=datos, filename='TS4.txt')

mouth_1
```

	0	1	2	3	4	5	6	7	8	9	...	50	51	52	53	54	55	56	57	58	59
0	35.570	35.492	35.469	35.422	35.414	35.320	35.227	35.242	35.160	35.176	...	36.008	35.984	35.996	36.039	36.008	36.008	36.094	36.102	36.090	36.152
1	36.156	36.094	35.992	36.008	35.992	35.902	35.824	35.820	35.727	35.727	...	37.328	37.324	37.340	37.332	37.316	37.410	37.418	37.422	37.488	37.477
2	37.488	37.391	37.340	37.312	37.223	37.145	37.059	36.973	36.898	36.879	...	38.457	38.461	38.457	38.469	38.469	38.555	38.527	38.543	38.527	38.621
3	38.633	38.535	38.469	38.379	38.297	38.223	38.125	38.062	37.977	37.969	...	39.441	39.363	39.367	39.457	39.461	39.461	39.473	39.441	39.453	39.461
4	39.461	39.461	39.375	39.281	39.203	39.113	39.043	38.969	38.875	38.883	...	40.324	40.320	40.312	40.340	40.320	40.387	40.391	40.391	40.387	40.391
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2200	35.414	35.348	35.254	35.262	35.168	35.172	35.105	35.094	35.000	35.008	...	35.426	35.516	35.437	35.426	35.410	35.414	35.414	35.426	35.414	35.414
2201	35.434	35.332	35.266	35.195	35.262	35.195	35.082	35.105	34.988	35.008	...	35.434	35.434	35.449	35.414	35.422	35.414	35.426	35.437	35.402	35.398
2202	35.434	35.355	35.262	35.187	35.266	35.102	35.105	35.094	35.016	35.016	...	35.500	35.434	35.410	35.434	35.426	35.437	35.441	35.410	35.434	35.437
2203	35.449	35.359	35.277	35.184	35.195	35.184	35.105	35.016	35.012	35.008	...	35.437	35.531	35.426	35.516	35.422	35.441	35.461	35.414	35.426	35.441

Figura 29. Importación de datos de Boca

Elaborado por el investigador

```
perfil = get_files(datos=datos, filename='profile.txt')

perfil
```

	0	1	2	3	4
0	3	100	0	130	1
1	3	100	0	130	1
2	3	100	0	130	1
3	3	100	0	130	1
4	3	100	0	130	1
...	...	...	...	...	...
2200	100	100	0	90	0
2201	100	100	0	90	0
2202	100	100	0	90	0
2203	100	100	0	90	0
2204	100	100	0	90	0

2205 rows × 5 columns

Figura 30. Importación de datos movimiento de cabeza

Elaborado por el investigador

2. Condiciones de instancia

Se generó las condiciones de instancia a cada parámetro mediante un muestreo aleatorio de la base de datos perteneciente a AWS, mostrados en la figura 26. En donde *instancia*: hace referencia a la imagen o datos

```
#face traking / %:
    #3: close to total failure (732 instances)
    #20: reduced effifiency (732 instances)
    #100: full efficiency (741 instances)

#eyed traking/ %:
    #100: optimal behavior(1125 instances)
    #90: small lag (360 instances)
    #80: severe lag (360 instances)
    #73: close to total failure (360 instances)

#mouth traking:
    #0: full efficiency (1221 instances)
    #1: close efficiency (492 instances)
    #2: small efficiency (492 instances)

#feelings traking:
    #130: optimal (599 instances)
    #115: slightly reduced (399 instances)
    #100: severely reduced (399 instances)
    #90: close to total failure (808 instances)

#mask traking:
    #0: conditions were stable (1449 instances)
    #1: static conditions might not have been reached yet (756 instances)
```

```
faceCondition = pd.DataFrame(perfil.iloc[:, 0])
eyedCondition = pd.DataFrame(perfil.iloc[:, 1])
mouthLeak = pd.DataFrame(perfil.iloc[:, 2])
feelingsAcc = pd.DataFrame(perfil.iloc[:, 3])
maskFlag = pd.DataFrame(perfil.iloc[:, 4])
```

```
# promediar los datos del ciclo, como se nos ha entregado varios datos ciclos de entrada
def Mean(DataFrame):
    DataFrame=DataFrame.mean(axis=1)
    return DataFrame
```

Figura 31. Condiciones de instancia

Elaborado por el investigador

3. Función Dataframe

Mediante la función dataframe se realizó en cambio de variable para generar las matrices por cada uno de los parámetros. Se ordenan los valores de 0 a 2205.

```
#convertir todos los datos a su respectiva media

# datos de face

face1=Mean(face_1)
face2=Mean(face_2)
face3=Mean(face_3)
face4=Mean(face_4)
face5=Mean(face_5)
face6=Mean(face_6)

# datos de eyed

eyed1=Mean(eyed_1)
eyed2=Mean(eyed_2)

# datos de mouth
mouth1=Mean(mouth_1)
mouth2=Mean(mouth_2)
mouth3=Mean(mouth_3)
mouth4=Mean(mouth_4)

#datos de feelings

feelingsCE=Mean(feelings_CE)
feelingsCP=Mean(feelings_CP)

# Otros datos

mask_eficiencia=Mean(mask)
movement_1=Mean(movement1)
factorfi=Mean(factor_fi)
```

Figura 32. Cambio de variable

Elaborado por el investigador

4. Clasificación de datos

Los datos se clasifican y separan según el parámetro establecido

```
[19]: Data
```

	PS1	PS2	PS3	PS4	PS5	PS6	FS1	FV2	TS1	TS2	...	CE	CP	EPS1	VS1	SE	PL	CD	VC	HA	Stable
0	160.673492	109.466914	1.991475	0.000000	9.842170	9.728097	6.709815	10.304592	35.621983	40.978767	...	39.601350	1.862750	2538.929167	0.576950	59.157183	0	3	100	130	1
1	160.603320	109.354890	1.976234	0.000000	9.635142	9.529488	6.715315	10.403098	36.676967	41.532767	...	25.786433	1.255550	2531.498900	0.565850	59.335617	0	3	100	130	1
2	160.347720	109.158845	1.972224	0.000000	9.530548	9.427949	6.718522	10.366250	37.880800	42.442450	...	22.218233	1.113217	2519.928000	0.576533	59.543150	0	3	100	130	1
3	160.188088	109.064807	1.946575	0.000000	9.438827	9.337430	6.720565	10.302678	38.879050	43.403983	...	20.459817	1.062150	2511.541633	0.569267	59.794900	0	3	100	130	1
4	160.000472	108.931434	1.922707	0.000000	9.358762	9.260636	6.690308	10.237750	39.803917	44.332750	...	19.787017	1.070467	2503.449500	0.577367	59.455267	0	3	100	130	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2200	161.227572	109.779581	2.001438	10.202473	9.972037	9.850361	6.689930	10.184515	35.313783	40.874800	...	46.628517	2.160600	2543.911033	0.550833	59.033100	0	100	100	90	0
2201	161.206070	109.787481	1.998781	10.197919	9.966184	9.844854	6.692182	10.177767	35.321600	40.868883	...	46.689817	2.151450	2543.411333	0.547483	59.068000	0	100	100	90	0
2202	161.192120	109.756174	1.993436	10.196824	9.964329	9.842628	6.693277	10.176172	35.319183	40.875950	...	46.472300	2.143300	2542.729767	0.545233	59.132350	0	100	100	90	0
2203	161.208917	109.793884	2.007077	10.198588	9.968232	9.846690	6.684128	10.178353	35.324767	40.876067	...	46.544967	2.148483	2544.046333	0.537017	58.970800	0	100	100	90	0

```
[20]: Data.head()
```

	PS1	PS2	PS3	PS4	PS5	PS6	FS1	FV2	TS1	TS2	...	CE	CP	EPS1	VS1	SE	PL	CD	VC	HA	Stable
0	160.673492	109.466914	1.991475	0.0	9.842170	9.728097	6.709815	10.304592	35.621983	40.978767	...	39.601350	1.862750	2538.929167	0.576950	59.157183	0	3	100	130	1
1	160.603320	109.354890	1.976234	0.0	9.635142	9.529488	6.715315	10.403098	36.676967	41.532767	...	25.786433	1.255550	2531.498900	0.565850	59.335617	0	3	100	130	1
2	160.347720	109.158845	1.972224	0.0	9.530548	9.427949	6.718522	10.366250	37.880800	42.442450	...	22.218233	1.113217	2519.928000	0.576533	59.543150	0	3	100	130	1
3	160.188088	109.064807	1.946575	0.0	9.438827	9.337430	6.720565	10.302678	38.879050	43.403983	...	20.459817	1.062150	2511.541633	0.569267	59.794900	0	3	100	130	1
4	160.000472	108.931434	1.922707	0.0	9.358762	9.260636	6.690308	10.237750	39.803917	44.332750	...	19.787017	1.070467	2503.449500	0.577367	59.455267	0	3	100	130	1

5 rows × 22 columns

Figura 33. Clasificación de datos

Elaborado por el investigador

[21]:
```
Data.describe(include="all")
```

[21]:

	PS1	PS2	PS3	PS4	PS5	PS6	FS1	FV2	TS1	TS2	...	CE	CP	EPS1	VS1	SE
count	2205.000000	2205.000000	2205.000000	2205.000000	2205.000000	2205.000000	2205.000000	2205.000000	2205.000000	2205.000000	...	2205.000000	2205.000000	2205.000000	2205.000000	2205.000000
mean	160.485315	109.379906	1.753227	2.600266	9.163320	9.079363	6.198549	9.649453	45.424567	50.365979	...	31.299077	1.808399	2495.509203	0.613315	55.287900
std	4.699425	4.986585	0.251902	4.279355	0.576296	0.549589	1.032883	0.449246	7.991933	7.396254	...	11.575330	0.278263	73.836682	0.060260	8.960189
min	155.391547	104.406307	0.840252	0.000000	8.365800	8.321527	2.018572	8.857513	35.313783	40.859400	...	17.555983	1.062150	2361.747267	0.524367	18.276617
25%	158.100195	106.962382	1.729733	0.000000	8.547239	8.487167	6.391670	9.203397	36.237150	41.864183	...	20.084650	1.550100	2442.933467	0.555100	56.270183
50%	158.960895	107.730169	1.779631	0.000000	9.115781	9.031516	6.576673	9.692270	44.836650	49.780583	...	27.392533	1.739683	2480.926633	0.610183	58.758150
75%	161.000735	109.421612	1.932047	3.503266	9.844351	9.729275	6.657508	10.155008	54.104317	58.584467	...	46.677383	2.148483	2548.211467	0.649850	59.656900
max	180.922708	131.589089	2.023398	10.207068	9.978510	9.856591	6.722707	10.403098	57.899283	61.958467	...	47.903667	2.840100	2740.641000	0.839067	60.755300

8 rows × 22 columns

Figura 34. Clasificación de datos según parámetros.

Elaborado por el investigador

5. Concatenación

Se concatenó en un solo marco los datos parametrizados.

```
# fusionar o concatener en un solo marco de datos (dataFrame)

Data=pd.DataFrame()

# datos de face

Data['PS1']=face1
Data['PS2']=face2
Data['PS3']=pface3
Data['PS4']=face4
Data['PS5']=face5
Data['PS6']=face6

# datos de eyed

Data['FS1']=eyed1
Data['FV2']=eyed2

# datos de temperatura

Data['TS1']=mouth1
Data['TS2']=mouth2
Data['TS3']=mouth3
Data['TS4']=mouth4

#datos de feelings

Data['CE']=feelingsCE
Data['CP']=feelingsCP

# Otros datos

Data['EPS1']=mask_eficiencia
Data['VS1']=movement_1
Data['SE']=factorfi

# datos de Prediccion

Data['PL']=pLeak
Data['CD']=cCondition
Data['VC']=vCondition
Data['HA']=hAcc
Data['Stable']=stable
```

Figura 35. Concatenación de datos

Elaborado por el investigador

Mediante la tabla 21 se explica brevemente los datos parametrizados:

Tabla 21. Parámetros del modelo

PS1 **PS2** **PS3** **PS4** **PS5** **PS6**	Face (rostro)
FS1 **FV2**	Eyes (ojos)
TS1 **TS2** **TS3** **TS4**	Mouth (boca)
CE **CP**	Expresiones faciales
EPS1	Mascarilla
VS1	Movimiento cabeza
SE	perfil

Elaborado por el investigador

6. Histograma de valores

Por medio de la figura 36 se observa el tipo de dato que tiene cada conjunto de parámetros establecidos anteriormente.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2205 entries, 0 to 2204
Data columns (total 22 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   PS1     2205 non-null   float64
 1   PS2     2205 non-null   float64
 2   PS3     2205 non-null   float64
 3   PS4     2205 non-null   float64
 4   PS5     2205 non-null   float64
 5   PS6     2205 non-null   float64
 6   FS1     2205 non-null   float64
 7   FV2     2205 non-null   float64
 8   TS1     2205 non-null   float64
 9   TS2     2205 non-null   float64
 10  TS3     2205 non-null   float64
 11  TS4     2205 non-null   float64
 12  CE      2205 non-null   float64
 13  CP      2205 non-null   float64
 14  EPS1    2205 non-null   float64
 15  VS1     2205 non-null   float64
 16  SE      2205 non-null   float64
 17  PL      2205 non-null   int64
 18  CD      2205 non-null   int64
 19  VC      2205 non-null   int64
 20  HA      2205 non-null   int64
 21  Stable  2205 non-null   int64
dtypes: float64(17), int64(5)
memory usage: 379.1 KB
```

Figura 36. Tipo de datos de cada conjunto

Elaborado por el investigador

De acuerdo al número total de datos hace una comparativa con los valores nulos

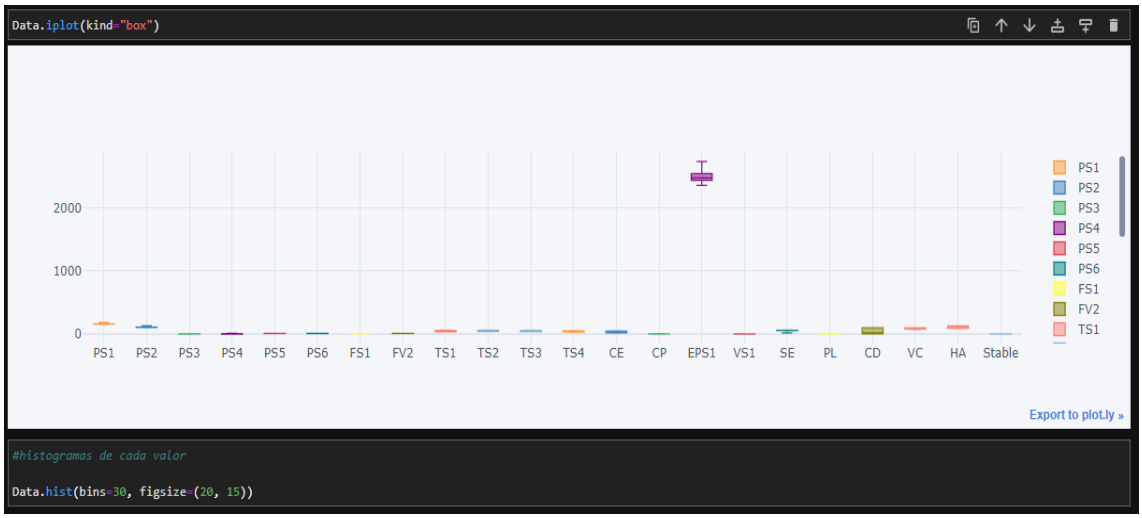


Figura 37. Histograma de cada valor

Elaborado por el investigador

Del histograma de la figura 38 se realizó la media de cada uno de los valores

Figura 38. Media de cada conjunto de valor

Elaborado por el investigador

7. Distribución normal

Mediante los datos obtenido en la figura anterior se generó las gráficas de distribución normal, obteniendo como resultado que ninguno sigue una distribución normal, de acuerdo a cada valor de la base de datos toma parámetros que le conviene,

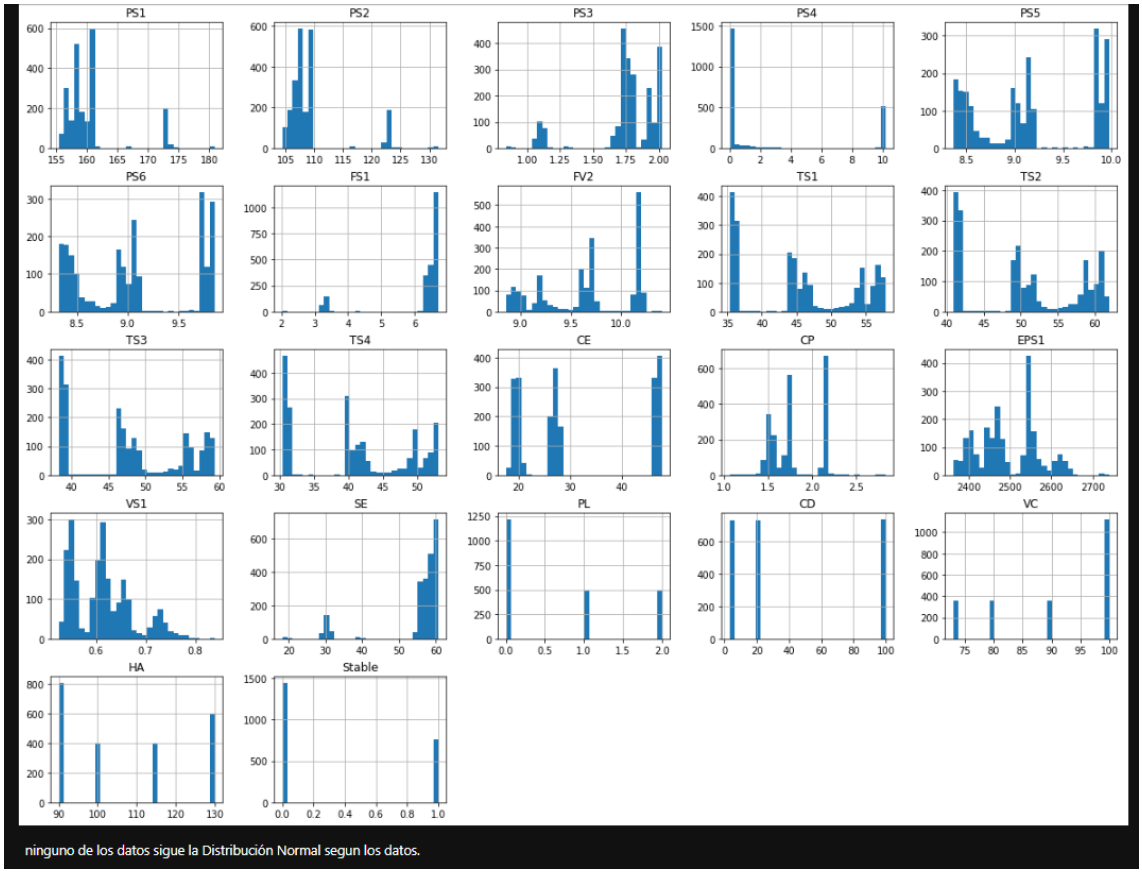


Figura 39. Distribución Normal

Elaborado por el investigador

8. Matriz de correlación

Se realizó la matriz de correlación entre los conjuntos de datos de entrada, decir el total de valores de entrenamiento y paralelamente con el total de valores que identifica cada una de las variables.

Figura 40. Matriz de correlación

Elaborado por el investigador

9. Normalización de los valores en el eje x y y

Se normalizó los valores del eje (x,y), en donde se define que entre mayor sea el números de componente la varianza tiende a 0.

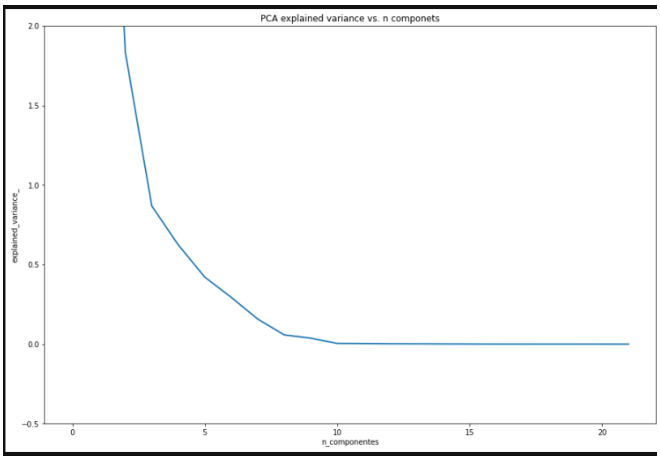


Figura 41. Normalización de valores

Elaborado por el investigador

10. Modelo Generado

Mediante la matriz de correlación se generó el modelo de ganancia de información

```
#Selección de features utilizando la ganancia de información.
from sklearn.feature_selection import mutual_info_classif
importances=mutual_info_classif(X,y)
feat_imp=pd.Series(importances, X.columns[0:len(X.columns)])
print(feat_imp)
feat_imp.plot(kind='barh')
plt.show()
```

```
PS1     0.279421
PS2     0.269172
PS3     0.189773
PS4     0.111586
PS5     0.392179
PS6     0.390753
FS1     0.299641
FV2     0.294070
TS1     0.408269
TS2     0.374336
TS3     0.390098
TS4     0.392197
CE      0.196112
CP      0.122915
EPS1    0.271929
VS1     0.173402
SE      0.285616
dtype: float64
```

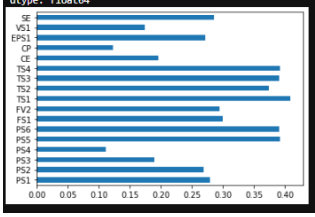


Figura 42. Modelo de ganancia

Elaborado por el investigador

Se generó las validaciones de entrenamiento

```
from sklearn.model_selection import train_test_split
# Train/test 80:20
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2,random_state=2020)
# Train/Validation 90:10
x_train, x_val, y_train, y_val = train_test_split(x_train,y_train, test_size=0.1, random_state=2020)

print("Shape of x_train:",x_train.shape)
print("Shape of x_test:",x_test.shape)
print("Shape of x_val:",x_val.shape)
print("Shape of y_train:",y_train.shape)
print("Shape of y_test:",y_test.shape)
print("Shape of y_val:",y_val.shape)
```

```
Shape of x_train: (1587, 17)
Shape of x_test: (441, 17)
Shape of x_val: (177, 17)
Shape of y_train: (1587,)
Shape of y_test: (441,)
Shape of y_val: (177,)
```

```
print(x_train.shape)
print(y_train.shape)
```

```
(1587, 17)
(1587,)
```

```
y_train = y_train.values.reshape(-1,1)
y_test = y_test.values.reshape(-1,1)
y_val = y_val.values.reshape(-1,1)
```

```
from sklearn.preprocessing import StandardScaler

# scaler for x
scaler = StandardScaler()
scaler.fit(x_train)
x_train_scaled = scaler.transform(x_train)
x_val_scaled = scaler.transform(x_val)
x_test_scaled = scaler.transform(x_test)

# scaler for y
scaler2 = StandardScaler()
scaler2.fit(y_train)
y_train_scaled = scaler2.transform(y_train)
y_val_scaled = scaler2.transform(y_val)
y_test_scaled = scaler2.transform(y_test)
```

Figura 43. Validación del entrenamiento

Elaborado por el investigador

3.2.8. Evaluación de resultados de entrenamiento

Una vez que el modelo ha sido entrenado, se procede a realizar predicciones en el conjunto de prueba. Para cada imagen en el conjunto, se obtiene el índice de la clase con la mayor probabilidad predicha. Utilizando esta información, se genera un informe de clasificación que muestra la precisión, recuperación, puntuación F1 y soporte para cada clase.

En la figura 44 se muestra las pruebas de testeo de acuerdo al número de datos de las 100 épocas

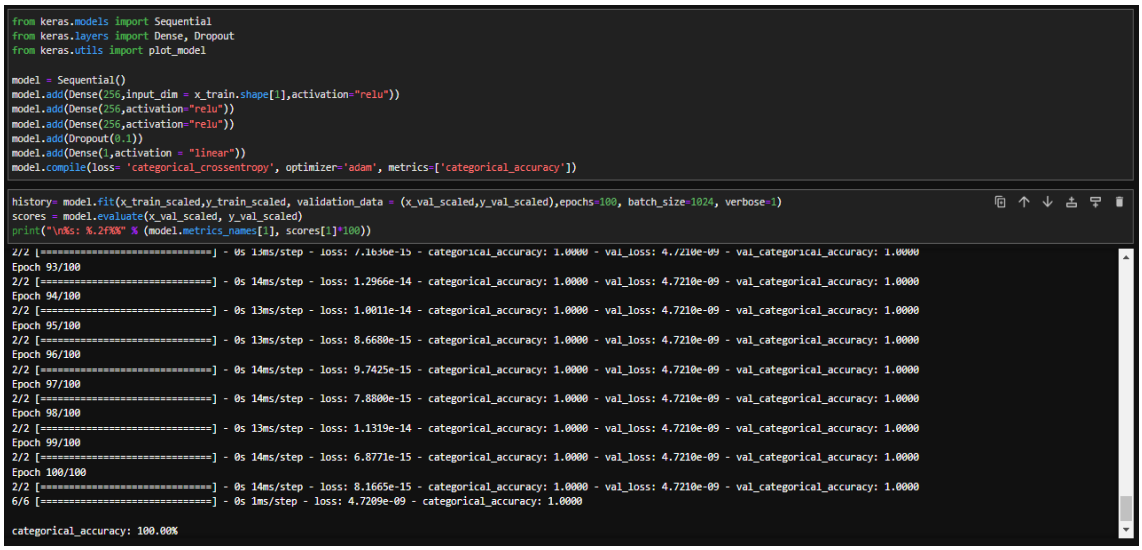


Figura 44. Muestras de testeo 100 épocas

Elaborado por el investigador

Se generó un gráfico de las perdidas por entrenamiento por validación de acuerdo a cada época, mostrando como resultado un desajuste en la validación.

El valor de entrenamiento fue menor al rendimiento de validación del algoritmo

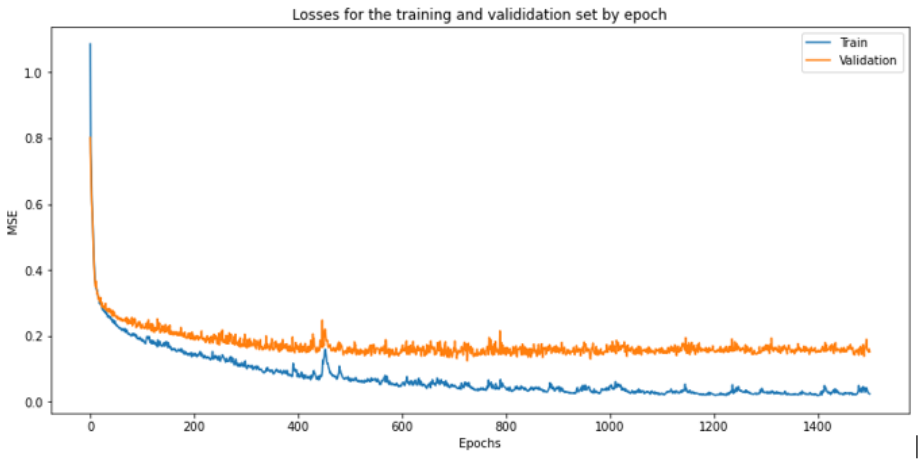


Figura 45. Pérdida por entrenamiento sin ajuste

Elaborado por el investigador

Para ello se ajuste en modelo automáticamente, realizando cada paso mencionado con anterioridad. Teniendo como resultado los siguientes valores:

```
#Selección de features utilizando la ganancia de información.
from sklearn.feature_selection import mutual_info_classif
importances=mutual_info_classif(X2,y2)
feat_imp=pd.Series(importances, X2.columns[0:len(X2.columns)])
print(feat_imp)
feat_imp.plot(kind='barh')
plt.show()
```

```
PS1     0.580035
PS2     0.591373
PS3     0.140037
PS4     0.088636
PS5     0.353748
PS6     0.354115
FS1     0.371377
FV2     0.252797
TS1     0.436819
TS2     0.406083
TS3     0.439246
TS4     0.376826
CE      0.129169
CP      0.062581
EPS1    0.254181
VS1     0.120241
SE      0.415622
dtype: float64
```

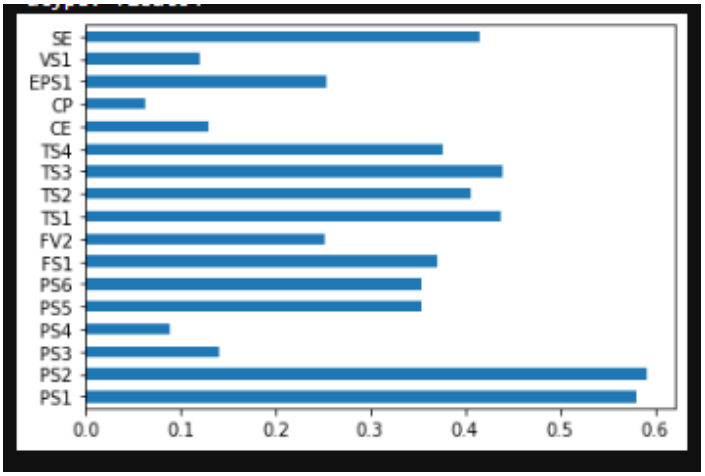


Figura 46. Ajuste del modelo de validación

Elaborado por el investigador

Se obtuvo como resultados un ajuste satisfactorio, este se refleja en el gráfico de la figura 47.

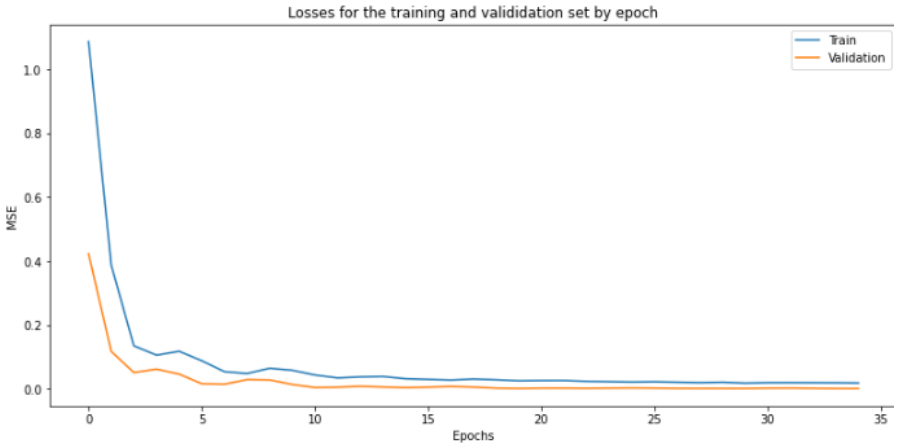


Figura 47. Gráfica de ajuste del modelo

Elaborado por el investigador

En la figura 48 se puede observar un ejemplo del análisis facial de la base de datos dentro de AWS.

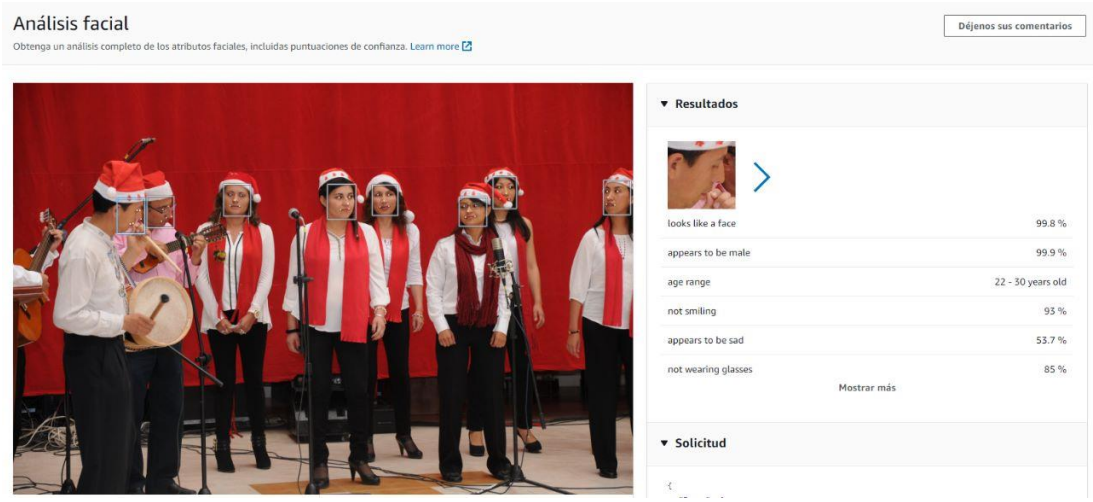


Figura 48. Análisis facial entrenamiento AWS

Elaborado por el investigador

3.2.9. Servidor Microsoft Azure

El servidor ha sido implementado como una máquina virtual en la plataforma de Microsoft Azure, lo que representa una mejora significativa en la infraestructura de alojamiento. En la figura 49, se puede observar la información esencial de la máquina virtual el sistema llamado DreamFaceIA1.

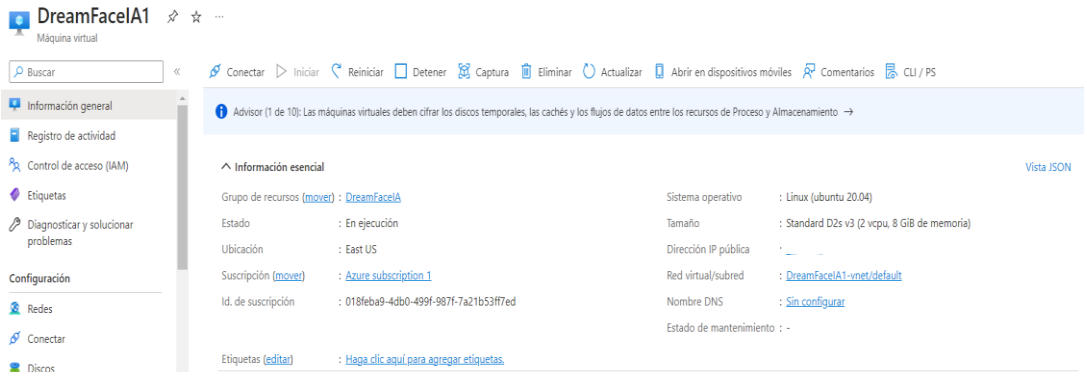


Figura 49. Información General del servidor creado

Elaborado por el investigador

En la figura 50, se muestra el grupo de recursos de la máquina virtual mencionada con anterioridad.

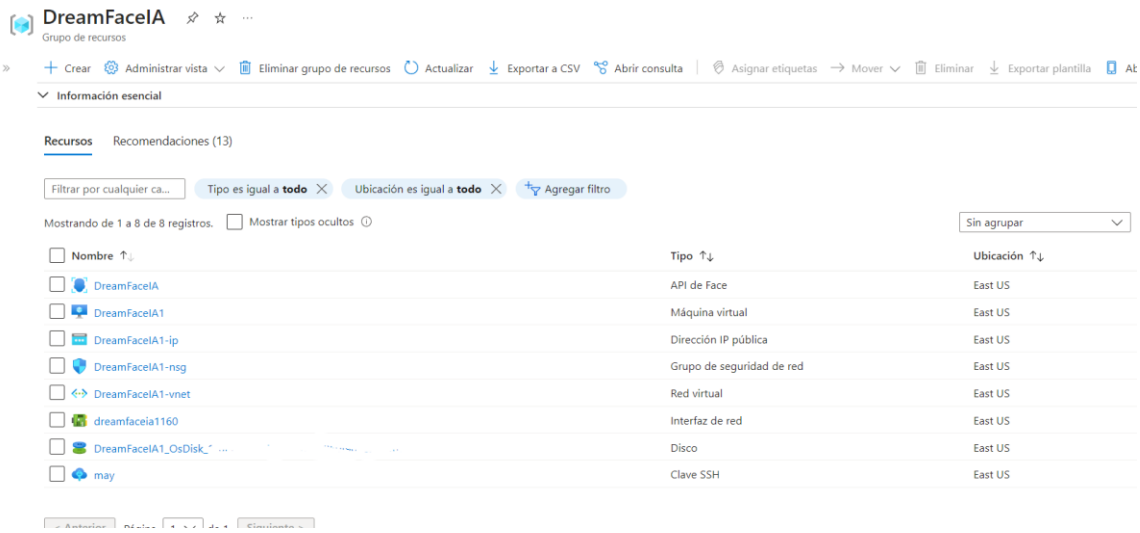


Figura 50. Panel de control del servidor

Elaborado por el investigador

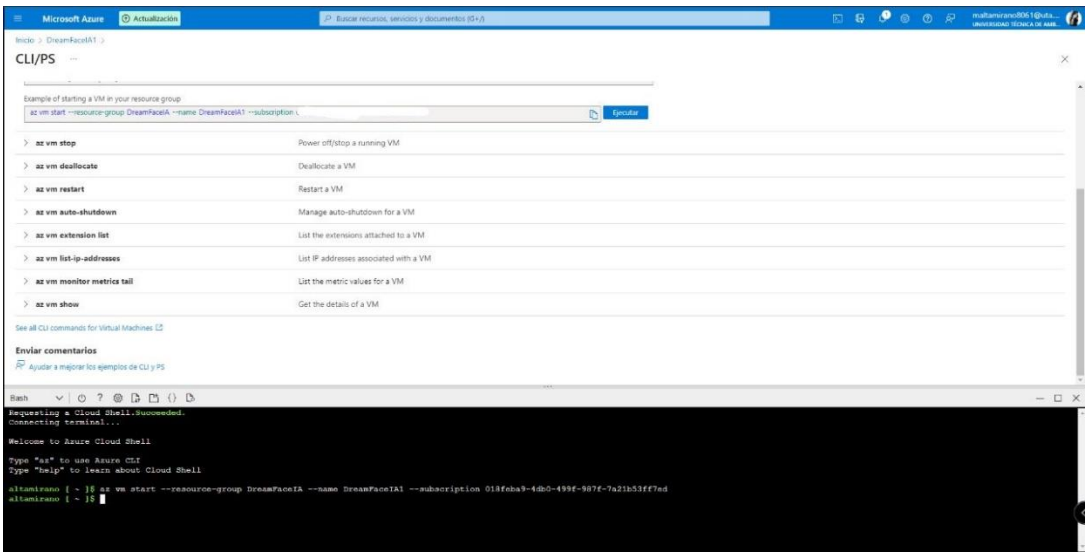


Figura 51. Máquina virtual en funcionamiento

Elaborado por el investigador

3.2.10. Interfaz

La norma ISO 9241-11 es un estándar internacional que proporciona directrices específicas para evaluar la usabilidad de las interfaces de usuario en sistemas interactivos. Enfocándose en la interacción persona-sistema, esta norma busca garantizar que las

interfaces sean diseñadas de manera eficiente y efectiva, proporcionando una experiencia de usuario satisfactoria [58].

Se ha creado una interfaz intuitiva y fácil de usar para la visualización, aprovechando la potencia de HTML para lograr una experiencia de usuario fluida. Al iniciar el programa, se presentan dos opciones: "Inicio" y "Aplicación", como se puede apreciar en la figura 52. Esta disposición clara y concisa permite a los usuarios navegar cómodamente por el sistema y seleccionar la opción deseada de manera sencilla.

Figura 52. Portada principal de la interfaz del sistema de detección.

Elaborado por el investigador

Una vez que se elige la opción "Aplicación", el sistema se activa automáticamente para comenzar con la detección.

3.2.11. Pruebas de funcionamiento

Prueba 1

El proceso de la primera prueba de funcionalidad en el sistema electrónico de detección se lleva a cabo a través de cuatro etapas, que se desarrollan de manera cronológica y detallada a continuación:

1. Ingreso a la interfaz

Para acceder a la interfaz del sistema, el usuario simplemente necesita ingresar a la página en el navegador web. Esta le dará acceso a la interfaz al sistema, tal como se visualiza en la figura 52, donde podrá interactuar y utilizar el sistema de reconocimiento de indicadores de somnolencia mediante inteligencia artificial. La interfaz amigable y accesible facilita la experiencia del usuario y le permite realizar de manera intuitiva las acciones requeridas para la detección y análisis de indicadores de somnolencia en tiempo real.

2. Inicialización del proceso

Una vez que se selecciona la opción "Aplicación", el sistema activa automáticamente la cámara para iniciar la detección. Se observa a detalle en la figura 53 y 54.

Figura 53. Activación de la cámara en la primera prueba de funcionamiento del sistema primera toma

Elaborado por el investigador

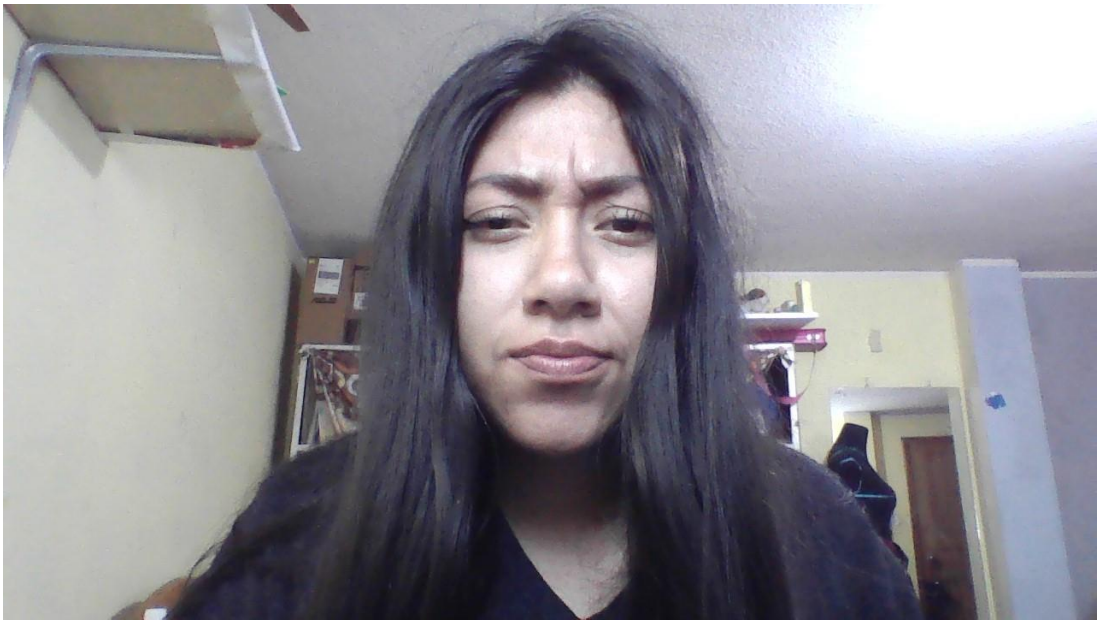


Figura 54. Activación de la cámara en la primera prueba de funcionamiento del sistema segunda toma

Elaborado por el investigador

3. Análisis del estado

La toma de datos en el sistema se realiza de manera periódica, capturando información cada 2 segundos. Esta frecuencia precisa y constante asegura que el sistema esté actualizado con datos relevantes en tiempo real, permitiendo un monitoreo efectivo y una detección oportuna de cualquier cambio o evento significativo.

En esta etapa, el sistema realiza la detección de los puntos del rostro necesarios para el análisis. Para el análisis exhaustivo de las pruebas de funcionamiento, se evaluaron diversos indicadores de somnolencia, cuyos resultados se presentan de manera clara y resumida en la tabla 22.

A continuación, se detalla el análisis del estado en la figura 55 y 56.

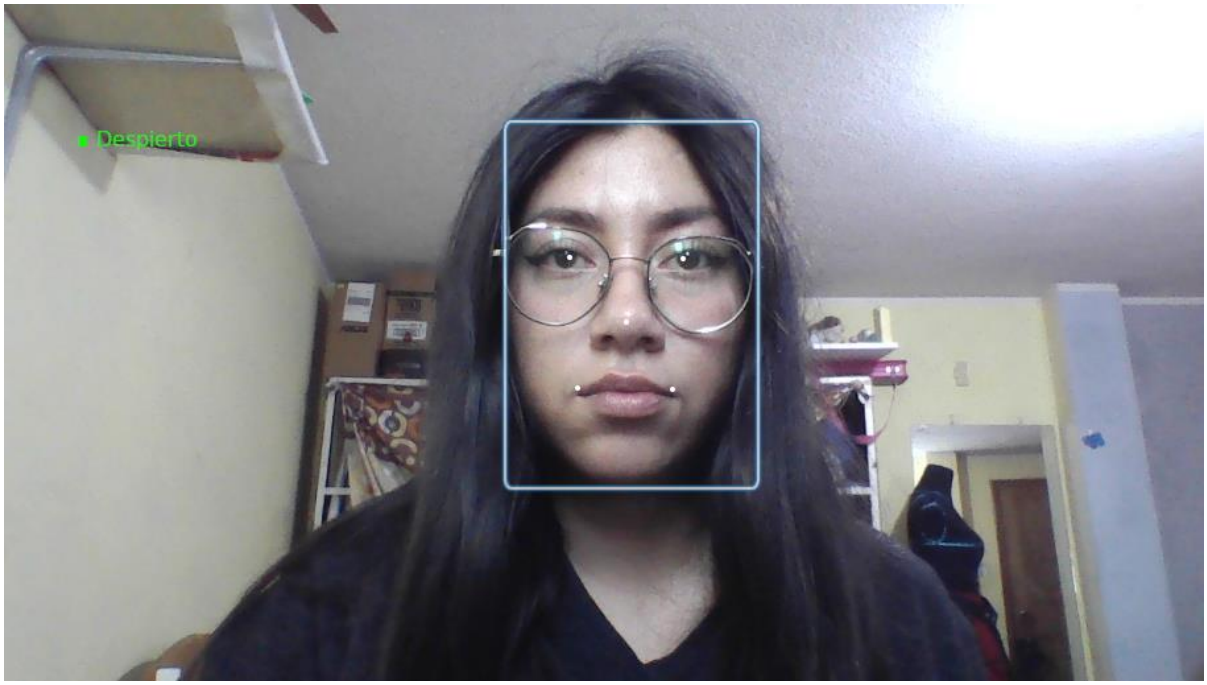


Figura 55. Análisis del estado primera prueba de funcionamiento primera toma

Elaborado por el investigador

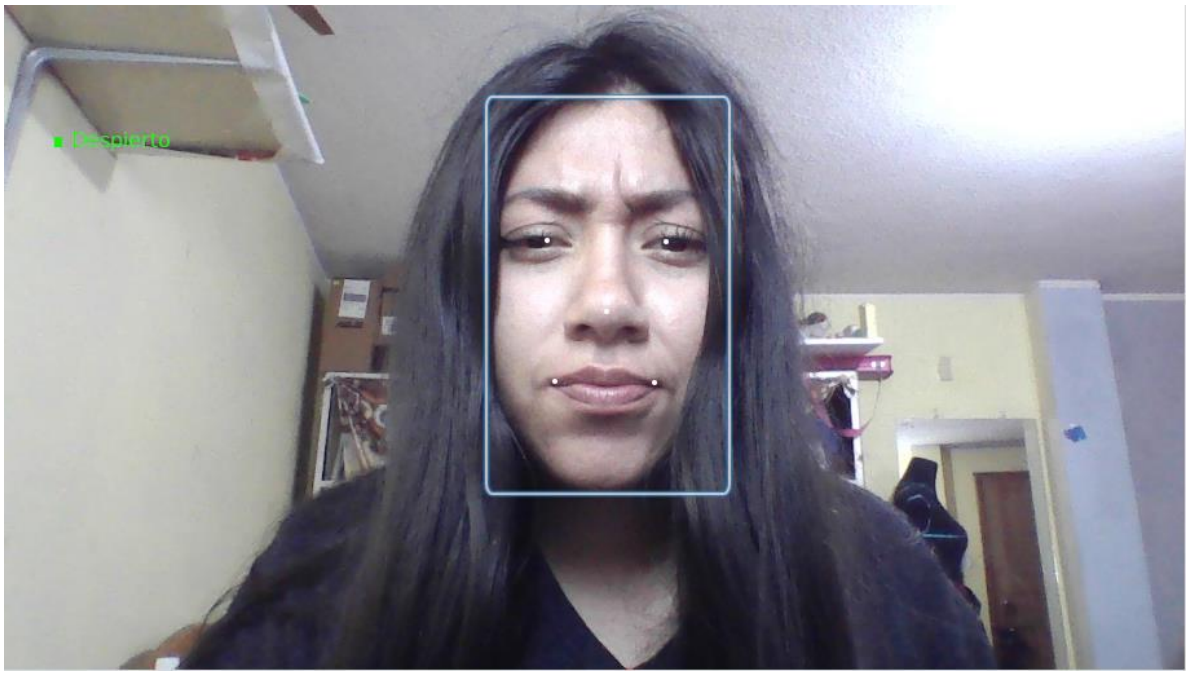


Figura 56. Análisis del estado primera prueba de funcionamiento segunda toma

Elaborado por el investigador

4. Resultados

Una vez completadas las tres etapas de prueba, en la tabla 22 exponen los resultados de manera resumida de del proceso de la primera prueba de funcionamiento, permitiendo una fácil interpretación.

Tabla 22. Resultados obtenidos durante la primera prueba de funcionamiento

Indicador	Resultado
Somnoliento	Negativo
No somnoliento	Positivo

Elaborado por el investigador

Prueba 2

El proceso de la segunda prueba de funcionalidad en el sistema electrónico de detección se lleva a cabo a través de cuatro etapas, que se desarrollan de manera cronológica y detallada a continuación:

1. Ingreso a la interfaz

Para acceder a la interfaz del sistema, el usuario simplemente necesita ingresar a la página en el navegador web. Esta le dará acceso a la interfaz al sistema, tal como se visualiza en la figura 52, donde podrá interactuar y utilizar el sistema de reconocimiento de indicadores de somnolencia mediante inteligencia artificial. La interfaz amigable y accesible facilita la experiencia del usuario y le permite realizar de manera intuitiva las acciones requeridas para la detección y análisis de indicadores de somnolencia en tiempo real.

2. Inicialización del proceso

Una vez que se selecciona la opción "Aplicación", el sistema activa automáticamente la cámara para iniciar la detección. Se observa a detalle en la figura 57 y 58.

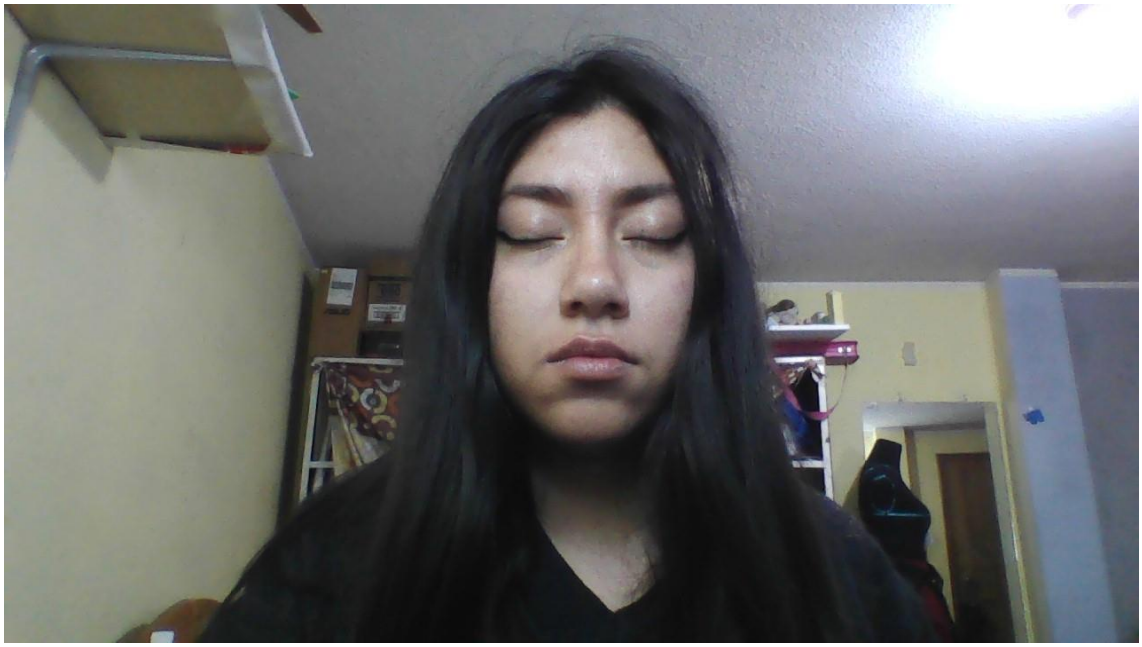


Figura 57. Activación de la cámara segunda prueba de funcionamiento en el sistema primera toma

Elaborado por el investigador

Figura 58. Activación de la cámara segunda prueba de funcionamiento en el sistema segunda toma

Elaborado por el investigador

3. Análisis del estado

La toma de datos en el sistema se realiza de manera periódica, capturando información cada 2 segundos. Esta frecuencia precisa y constante asegura que el sistema esté actualizado con datos relevantes en tiempo real, permitiendo un monitoreo efectivo y una detección oportuna de cualquier cambio o evento significativo.

En esta etapa, el sistema realiza la detección de los puntos del rostro necesarios para el análisis. Para el análisis exhaustivo de las pruebas de funcionamiento, se evaluaron diversos indicadores de somnolencia, cuyos resultados se presentan de manera clara y resumida en la tabla 23.

A continuación, se detalla el análisis del estado en la figura 59 y 60.

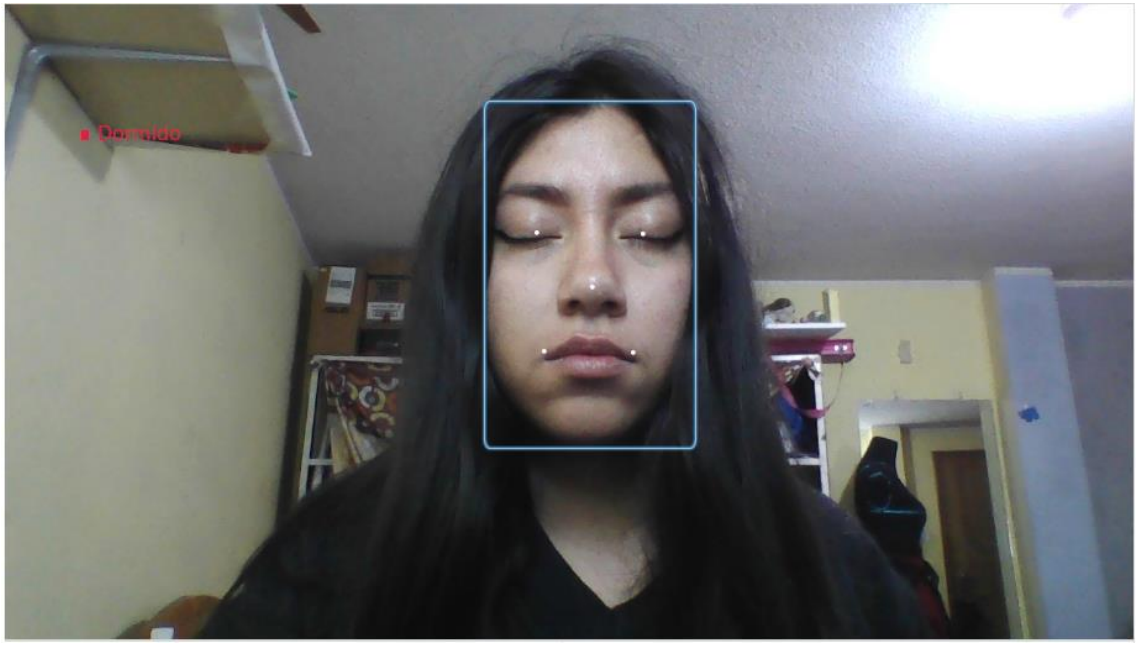


Figura 59. Análisis del estado segunda prueba de funcionamiento primera toma

Elaborado por el investigador

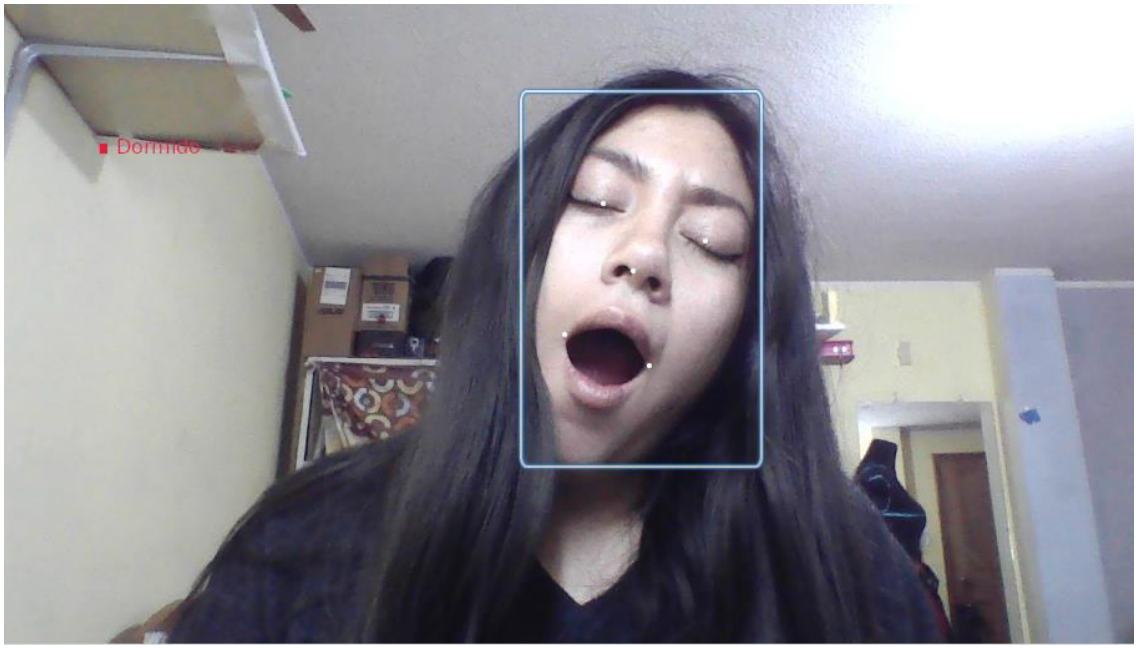


Figura 60. Análisis del estado segunda prueba de funcionamiento segunda toma

Elaborado por el investigador

4. Resultados

Una vez completadas las tres etapas de prueba, en la tabla 23 exponen los resultados de manera resumida de del proceso de la segunda prueba de funcionamiento, permitiendo una fácil interpretación.

Tabla 23. Resultados obtenidos durante la segunda prueba de funcionamiento

Indicador	Resultado
Somnoliento	Positivo
No somnoliento	Negativo

Elaborado por el investigador

Al detectar la somnolencia el sistema activa la alarma.

Después de completar con éxito las pruebas de funcionamiento del sistema de indicadores de somnolencia utilizando el microordenador Jetson Nano NVIDIA, se procedió a realizar la migración de la máquina virtual hacia la plataforma de nube de Microsoft Azure.

3.2.12. Resultados

Matriz de confusión

En el caso del sistema de reconocimiento de indicadores de somnolencia, la matriz de confusión se ha realizado con un conjunto de prueba de 100 datos los cuales se pueden resumir en la tabla 24:

Tabla 24. Matriz de confusión predicción del sistema

Predicción del sistema		
Predicción	**Predicción Positivos**	**Falsos positivos**
Valor Positivo	72 **(VP)**	3 **(FN)**
Valor negativo	1 **(FP)**	24 (**VN)**

Elaborado por el investigador

A partir de esta matriz, se calculó diferentes métricas como precisión, recall, F1-score y exactitud. Estas métricas nos brindaron información sobre la eficacia del sistema de reconocimiento de indicadores de somnolencia. Utilizando estos valores, podemos calcular las métricas de rendimiento.

En la matriz de confusión, tenemos:

- Verdaderos Positivos (TP): 72
- Falsos Positivos (FP): 1
- Verdaderos Negativos (TN): 24
- Falsos Negativos (FN): 3

Mediante los datos obtenido en las pruebas realizadas se puede observar lo siguiente:

Verdaderos Positivos (TP) 72: El sistema ha identificado correctamente a 72 casos como positivos, es decir, ha detectado adecuadamente a 72 personas que estaban somnolientas.

Falsos Positivos (FP) 1: Se obtuvo un caso en el que el sistema ha clasificado incorrectamente una persona como somnolienta cuando en realidad no lo estaba. Es un error de tipo I, llamado "falso positivo", que representa una falsa alarma.

Verdaderos Negativos (TN) 24: El sistema ha identificado correctamente a 24 casos como negativos, es decir, ha reconocido adecuadamente a 24 personas que no estaban somnolientas.

Falsos Negativos (FN) 3: En tres casos, el sistema no ha identificado correctamente que las personas estaban somnolientas cuando en realidad lo estaban. Estos son errores de tipo II, llamados "falsos negativos", y representan casos en los que el sistema no detectó una condición que estaba presente.

El nivel de peligro asociado con el fallo del sistema de detección de somnolencia depende de un falso positivo (FP), ocurre cuando el sistema indica que una persona está somnolienta cuando en realidad no lo está. Esto puede ser molesto para el usuario, ya que puede generar alarmas innecesarias o distracciones. Al igual que un falso negativo (FN), en el sistema de reconocimiento de indicadores de somnolencia, ocurre cuando el sistema no detecta que una persona está somnolienta cuando en realidad lo está. Este tipo de error es potencialmente más peligroso, ya que podría no alertar a una persona poniendo en riesgo su seguridad.

Validación de resultados

Mediante la ecuación 1, se calculó la precisión

$$Precisión = \frac{TP}{(TP + FP)}$$

Ecuación 1.

$$Precisión = \frac{72}{(72 + 1)}$$

$$Precisión = 0.986$$

Mediante la ecuación 2, se calculó la sensibilidad o tasa de verdaderos positivos llamado Recall.

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

Ecuación 2.

$$Recall = \frac{72}{(72 + 3)}$$

$$Recall = 0.96$$

Mediante la ecuación 3, se calculó el F1-score, la cual es una métrica que combina la precisión y el recall en una sola medida.

$$F1_{score} = 2 * \left(\frac{Presición * Recall}{Presición + Recall}\right)$$

Ecuación 3.

$$F1_{score} = 2 * \left(\frac{0.986 * 0.96}{0.986 + 0.96}\right)$$

$$F1_{score} = 0.973$$

Mediante la ecuación 4, se calculó la exactitud del sistema.

$$Exactitud = \left(\frac{TP + TN}{TP + FP + TN + FN}\right)$$

Ecuación 4.

$$Exactitud = \left(\frac{72 + 24}{72 + 1 + 24 + 3}\right)$$

$$Exactitud = 0.96$$

Basándonos en estas métricas, podemos concluir que el sistema tiene una buena precisión y recall, lo que indica que es capaz de clasificar correctamente la mayoría de los casos positivos y negativos. El valor de F1-score también es alto, lo que muestra un buen equilibrio entre precisión y recall. La exactitud también es aceptable, indicando que el sistema clasifica correctamente la mayoría de los casos.

Para evaluar el rendimiento del sistema en porcentaje, se utilizó el F1-score, que es una medida que combina tanto la precisión como el recall en una sola métrica. El F1-score varía entre 0 y 1, donde un valor de 1 indica un rendimiento perfecto y un valor de 0 indica un rendimiento nulo.

Para obtener el porcentaje equivalente, simplemente se aplicó la ecuación 5.

$$Porcentaje\ de\ rendimiento\ del\ sistema\ =\ F1_{score} * 100$$

Ecuación 5.

$$Porcentaje\ de\ rendimiento\ del\ sistema\ =\ 0.973\ * 100$$

$$\boldsymbol{Porcentaje\ de\ rendimiento\ del\ sistema = 97.3\%}$$

Este valor indica que el sistema ha demostrado un buen rendimiento en la detección y clasificación de los casos positivos y negativos evaluados en el conjunto de datos proporcionado. Sin embargo, es fundamental tener en cuenta que el rendimiento puede variar en diferentes conjuntos de datos y escenarios de prueba.

3.2.13. Presupuesto

El presupuesto total del proyecto denominado sistema de reconocimiento de indicadores de somnolencia mediante inteligencia artificial se divide en dos partes: el presupuesto de diseño y el presupuesto de construcción. Para determinar el presupuesto de diseño se consideró las horas dedicadas a su fabricación y el sueldo de un Ingeniero en

Telecomunicaciones, que se estipuló en 858 dólares según el Ministerio de Trabajo de Ecuador. Para obtener el salario diario, se consideró 5 días laborables por semana y un total de 21 días laborales por mes, mediante estos datos aplicando la ecuación 6 se puede se obtiene:

$$Salario_{diario} = \frac{Salario_{mensual}}{Días_{laborables}}$$

Ecuación 6.

$$Salario_{diario} = \frac{858\ [dólares]}{21\ [días]}$$

$$Salario_{diario} = 40.86\ [dólares]$$

Se tiene conocimiento de que una jornada laboral consta de 8 horas al día, y al aplicar la ecuación 7, se obtiene el cálculo de la remuneración por hora.

$$Salario_{hora} = \frac{Salario_{diario}}{Horas_{laborables}}$$

Ecuación 7.

$$Salario_{hora} = \frac{40.86\ [dólares]}{8\ [horas]}$$

$$Salario_{hora} = 5.11\ [dólares]$$

El presupuesto total del diseño del proyecto toma en cuenta 3 horas diarias de trabajo durante 4 meses, dando un total de 252 horas, que se distribuyen entre las etapas de investigación, desarrollo, implementación y pruebas de funcionamiento. Aplicando la ecuación 8 se obtiene:

$$Presupuesto_{diseño} = Horas_{investigación} * Salario_{hora}$$

Ecuación 8

$$Presupuesto_{diseño} = 252\ [horas] * 5.11\ [dólares]$$

$$Presupuesto_{diseño} = 1287.72\ [dólares]$$

En la Tabla 25, se presenta una descripción detallada del costo de los materiales utilizados en la construcción del sistema electrónico. Cada componente y su respectivo precio unitario se muestran de manera clara y organizada, lo que permite una evaluación transparente del presupuesto asignado a cada elemento. Se incluyen los costos de los microcontroladores, cámaras y otros dispositivos que fueron adquiridos para implementar el sistema de reconocimiento de indicadores de somnolencia.

Tabla 25. Costo de los materiales del sistema electrónico

Ítem	Detalle	Cantidad	Valor Unitario	Valor Total
1	Kit de desarrollo Jetson Nano	1	$ 180.00	$ 180.00
2	Case (Acrílico)	1	$ 16.00	$ 16.00
3	Cámara Pi Noir V2	1	$ 25.00	$ 25.70
4	Mouse y teclado	1	$ 15.00	$ 15.00
5	Fuente de 5V - 4A	1	$ 10.00	$ 10.00
6	Cable HDMI	1	$ 6.00	$ 6.00
7	PlanAmazon SageMaker	1	$ 80	$ 80
			Subtotal	$ 332,70
			Iva (12%)	$ 39,92
			Total	$ 372,62
			Imprevistos (5%)	$ 18,63
			TOTAL	**$ 391,26**

Elaborado por el investigador

Por último, el presupuesto total del proyecto de investigación se determina utilizando la ecuación 9.

$$Presupuesto_{total} = Presupuesto_{diseño} + Presupuesto_{construcción}$$

Ecuación 9.

$$Presupuesto_{total} = 1287.72\ [dólares] + 391.26\ [dólares]$$

$$\boldsymbol{Presupuesto_{total} = 1678.98\ [dólares]}$$

CAPÍTULO IV

CONCLUSIONES Y RECOMENDACIONES

4.1. Conclusiones

- La documentación de los métodos empleados para la adquisición de indicadores de somnolencia fue fundamental en el desarrollo del presente proyecto de investigación. El proceso de recopilación a través de diversas fuentes, como informes, artículos científicos, libros y publicaciones, ha proporcionado una base sólida y confiable para el análisis y selección de los indicadores de somnolencia considerados para el sistema de reconocimiento. Mediante esta documentación detallada, se pudo identificar y evaluar el método MAR (Mouth Aspect Ratio), el mapa de coordenadas utilizado por Dlib y la técnica "PERCICLOS", lo que ha permitido evaluar criterios visuales clave, como el nivel de apertura de los ojos, las expresiones faciales, el seguimiento de movimientos de la cabeza y los bostezos, como los indicadores más relevantes para el sistema.

- El algoritmo diseñado permitió la adquisición en tiempo real de imágenes o videos a través de una cámara incorporada, evaluando cada fotograma de manera independiente. Estos datos son procesados utilizando el microcomputador Jetson Nano, debido a que cuenta con una GPU NVIDIA Maxwell de 128 núcleos, CPU de cuatro núcleos ARM Cortex-A57 de 64 bits que brinda un rendimiento sólido para tareas de procesamiento, 4 GB de Memoria RAM y soporte para bibliotecas como TensorFlow, enfocadas a inteligencia artificial, permitiendo respuestas de 2.38 milisegundos en video en tiempo real y alertas oportunas en caso de detectar somnolencia en los individuos. Una de las principales fortalezas del sistema radicó en la aplicación de técnicas avanzadas de inteligencia artificial, como las redes neuronales. La arquitectura de la red neuronal fue diseñada para extraer y aprender características relevantes de imágenes mediante 3 capas convolucionales, 2 max pooling, una dense para clasificarlas y una flatten que convierte la información en forma lineal.

- El diseño del sistema de indicadores de somnolencia ha culminado con éxito, resultando en un sistema sólido y funcional que combina la adquisición de datos, el análisis de imágenes, el procesamiento de información, entrenamiento de la neurona y la interfaz gráfica. El sistema inicia en la adquisición en tiempo real de imágenes o videos mediante una cámara incorporada en donde los datos adquiridos son enviados al microordenador (Jetson Nano) para su procesamiento. Seguidamente se emplea la red neuronal entrenada para finalmente realizar la evaluación. La integración de la cámara, microcomputador, pantalla y periféricos de salida fue clave para la creación de un sistema de indicadores de somnolencia que permite detectar y obtener respuestas en tiempo real. Además, el diseño intuitivo de la interfaz de usuario ha optimizado la interacción con el sistema, brindando a los usuarios una experiencia agradable y facilitando la comprensión de los resultados obtenidos.

- Los resultados obtenidos a partir de la matriz de confusión, elaborada con un conjunto de prueba de 100 datos que respaldan la efectividad del sistema. Con una precisión de 98.6% y una sensibilidad del 96%, revelan que el sistema tiene la capacidad de identificar correctamente la gran mayoría de los casos de somnolencia. La exactitud general del sistema alcanza el 96%, y su alto rendimiento general, con un porcentaje del 97.3%, %, respalda su eficacia en la detección. Los resultados obtenidos en estas pruebas han demostrado de manera contundente que el sistema cumple con los objetivos establecidos y ha brindado la confianza necesaria para su implementación en entornos reales.

4.2. Recomendaciones

- Se recomienda llevar a cabo una investigación exhaustiva con el propósito de potenciar los algoritmos de procesamiento de imágenes para que adquieran la habilidad de reconstruir y procesar eficientemente imágenes capturadas en entornos con escasa iluminación. Esta iniciativa busca fortalecer la capacidad del sistema de reconocimiento de indicadores de somnolencia para operar con mayor precisión y eficacia en condiciones de luz limitada, garantizando así un rendimiento óptimo en diversas situaciones y ambientes con niveles de iluminación bajos.

- Asegurar una conexión a Internet estable es de vital importancia para lograr una comunicación efectiva con el servidor alojado en "Azure" y reducir la latencia al mínimo. La presencia de una conexión inestable podría ocasionar demoras en la transmisión de datos, impactando negativamente en el rendimiento del sistema. Por consiguiente, se recomienda una adecuada configuración de los firewalls y los puertos para facilitar una comunicación segura entre los microcontroladores y el servidor, garantizando así la privacidad de los datos transmitidos.

- Es aconsejable optar por una placa especializada en inteligencia artificial (IA) equipada con su sistema operativo correspondiente. Estos sistemas embebidos están diseñados para ofrecer un rendimiento óptimo en tareas de IA y tienen la capacidad de funcionar como miniordenadores debido a su robusta arquitectura. Utilizar este tipo de placa asegura contar con las características técnicas adecuadas para desplegar con eficiencia las funciones requeridas por el sistema de reconocimiento de indicadores de somnolencia. La elección de esta plataforma de aplicaciones proporcionará una mayor estabilidad y potencia, permitiendo un desempeño más sólido y confiable en el procesamiento de datos y el análisis de imágenes, aspectos importantes para el éxito y la precisión del sistema en su conjunto.

- Se recomienda realizar pruebas adicionales y evaluaciones exhaustivas para obtener una estimación más precisa y representativa del rendimiento general del sistema en diversos entornos y condiciones de uso. Esto permitirá verificar su robustez y capacidad para adaptarse a diferentes situaciones, lo que asegurará la confiabilidad y efectividad del sistema en aplicaciones prácticas del mundo real.

BIBLIOGRAFÍA

[1] «Organización Mundial de la Salud,» [En línea]. Available: https://www.who.int/es/. [Último acceso: 25 enero 2023].

[2] E. Rosales Mayor y J. Rey De Castro Mujica, «What it is, what causes it, and how measure it,» *SciELO.org*, pp. 137-143, 2010.

[3] PHILIPS, « Encuesta Anual del Sueño,» PHILIPS, 2018.

[4] I. N. d. E. y. C. (INEC), Ecuador, 2015.

[5] C. M. D. W. R. T. Roehrs T, «Daytime Sleepiness and alertness. In: Kryger MH, Roth T, Dement WC, eds. Principles and Practice of Sleep Medicine.,» *Saunders*, nº 4, pp. 39-49, 2005.

[6] C. P. Baiza Lovato, «SISTEMA DE DETECCIÓN Y ALERTA DEL ESTADO DE SOMNOLENCIA DE CONDUCTORES MEDIANTE VISIÓN ARTIFICIAL,» UNIVERSIDAD TECNOLÓGICA ISRAEL, Quito, 2020.

[7] E. A. Suárez Buitrago, «Sistema de detección de somnolencia en conductores de automóviles empleando técnicas de procesamiento de imágenes y machine learning,» Universidad de Pamplona, Colombia, 2021.

[8] Á. I. Velecela Santander, Implementación de un sistema de detección de Sueño, en el vehículo eléctrico de la Universidad católica de Cuenca, Azogues: UNIVERSIDAD CATÓLICA DE CUENCA, 2021.

[9] H. Fernández Solís, «Sistema de Detección de Somnolencia,» Universidad de la laguna, 2022.

[10] K. D. Delgado Egas y M. A. Yandún Velasteguí, «SISTEMA DE DETECCIÓN DE SOMNOLENCIA PARA CONDUCTORES DE TAXIS EN LA CIUDAD DE TULCÁN,» Horizontes de Enfermería, Tulcán, 2022.

[11] E. Mayon Sanchez y R. Limaquispe Miguel, Sistema de Deteccion de somnolencia mediante inteligencia artificial en conductores de vehículos para alertar la ocurrencia de accidentes de tránsito, Huancavelica: Universidad Nacional de Huancavelica, 2018.

[12] P. JF, «Excessive Daytime Sleepiness,» *Am Fam Physician*, pp. 391-396, 2009.

[13] J. L. Velayos Jorge, F. Moleres, Irujo, A, D. Yllanes y B. Paternain, «Bases anatómicas del sueño,» vol. 30.

[14] R. Aguirre, «Bases anatómicas y fisiológicas del sueño,» *Revista Ecuatoriana de Neurología*, vol. 15, nº 2-3, 2007.

[15] C. P. Baiza Lovato, «SISTEMA DE DETECCIÓN Y ALERTA DEL ESTADO DE SOMNOLENCIA DE CONDUCTORES MEDIANTE VISIÓN ARTIFICIAL,» UNIVERSIDAD TECNOLÓGICA ISRAEL, Quito, 2020.

[16] A. González Pinto, «Programa SOMNE. Terapia,» 2015.

[17] G. D. Ivan, «Detección de fatiga en conductores mediante fusión de sistemas ADAS,» Universidad de Alcalá, 2011.

[18] A. Torres, Reconocimiento Facial y detección de somnolencia en conductores, La Laguna: Univesidad de la Laguna, 2022.

[19] P. J. García Paterna, Desarrollo de un sistema inteligente de detección de fatiga en conductores, Cartagena: Universidad Politécnica de Cartagena.

[20] E. L. Becerril, A. Melendez Ramírez y E. T. Juárez Velázquez, Sistema para la detección del estado de somnolencia en seres humanos, con reconocimiento de patrones, Universidad Politécnica de Texcoco.

[21] C. G. S. G. F. J. Chaccere Rodríguez, Diseño y simulación de un sistema de detección de somnolencia y alerta basado en el procesamiento digital de imágenes con algoritmos de correlación en tiempo real, Perú: Escuela Profeional de Ingeniería Electrónica, 2015.

[22] A. S. Revelo Álava, Implementación de un algoritmo de detección de somnolencia humana, en tiempo real basado en visión artificial, Quito: Escuela Politécnica Nacional, 2019.

[23] A. Acıoğlu y A. Erçelebi, «Real time eye detection algorithm for PERCLOS calculation,» *Signal Processing and Communication Application Conference (SIU)*, pp. 1641-1644, 2016.

[24] C. A. Torres Varon, Detección temprana del sueño a partir del comportamiento de los ojos y boca, Bogotá: Universidad del Rosario , 2022.

[25] D. M. López Mena, Diseño e implementación e un sstema de visión artificial para determinar condiciones de fatiga en una persona mediante el índice PERCLOS, utilizando Open CV y una tarjeta Raspberry Pi3, Quito: Escuela Politécnica Nacional, 2017.

[26] R. Rodriguez Villalobos y K. Fajardo Suarez, «Estudio de los procesadores Digitales de señales para el desarrollo de aplicaciones en tiempo real,» Universidad Tecnológica de Bolivar, Cartagena de Indias, 2006.

[27] H. A. Hernández Rivera, «Sistema electrónico de adquisición y procesamiento de señales con comunicación USB,» Posgrado Interinstitucional en Ciencia y Tecnología, Santiago de Querétano, 2016.

[28] O. Reyes Ortiz, M. Mejia, Useche y J. Sebastián, «Técnicas de Intwligencia artificial utilizadas en el procesamiento de imágenes y su aplicación en el análisis de pavimentos,» *EIA*, vol. 16, nº 31, pp. 189-207, 2019.

[29] D. Delgado León, «Diseño de un sistema de tadquisición de imágenes basado en cámaras web USB y hardware reconfigurable,» *RIELAC*, vol. XXXVIII, pp. 1-11, 2017.

[30] J. I. Armijos Benalcázar, Sistema de alarma electrónico paea la detección del estado de somnolencia en conductores de vehículos de dos ejer mediante visión artificial, Ibarra: Universidad Técnica del Norte, 2018.

[31] R. F. Briceño Castillo y R. S. Pazmiño Pérez, Detección del estado de somnolencia para conductores mediante el análisis de ondas cerebrales, Quito: Universidad Politécnica Salesiana, 2020.

[32] C. Jacobé de Naurois, C. Bourdin y E. Diaz, «Detection and prediction of driver drowsiness using artificial neural network models,» *Accident Analysis and Prevention,* vol. 126, pp. 95-104, 2019.

[33] P. R. Lasse, Inteligencia Artificial, Barcelona: © Editorial Planeta, 2018.

[34] E. Manrique Rojas, «Machine Learning: análisis de lenguajes de programación y herramientas para desarrollo,» *Revista Ibérica de Sistemas e Tecnologias de Informação,* pp. 586-599, 2020.

[35] E. Vergara, J. Ordieres, M. Castejón, F. Alba, V. Pernía, F. Martínez de Pisón y A. Gonzáles, «Técnicas y Algoritmos básicos de visión artificial,» Universidad de la Rioja, España, 2006.

[36] J. P. Aimacaña Chuquimarca y A. R. Columba Guanoluisa, «Análisis comparativo de algoritmos de Machine Learning para la detección de plagas en los cultivos representativos de la sierra ecuatoriana,» Quito, 2021, 2021.

[37] W. Campos Wright y Y. Trujillo Casañola, «Redes Neuronales Artificiales en la estimación del esfuerzo,» *Revista Cubana de Ciencias Informáticas,* vol. 15, nº 2, pp. 183-198, 2021.

[38] F. A. Incio-Flores, D. L. Capuñay-Sanchez y R. O. Estela-Urbina, «Artificial Neural Network Model to Predict Academic Results in Mathematics II,» *Revista Electrónica Educare,* vol. 27, 2023.

[39] C. P. Baiza Lovato, «Sistema de detección y alerta del estado de somnolencia de conductores mediante visión artificial,» Universidad Tecnológica de Israel, Quito, 2020.

[40] R. P. Foundation, «Raspberry Pi,» Raspberry Pi, 2019.

[41] NVIDIA, Módulo y kit de desarrollo Jetson Nano, 2022.

[42] J. D. Briseño Sánchez, «Sistema electrónico para la detección de posición angular y ergonomía de ciclistas empleando visión artificial,» Universidad Técnica de Ambato, Ambato, 2023.

[43] N. CORPORATION, «NVIDIA Jetson Nano System-on-Module DATASHEET,» NVIDIA , San Tomas Expressway, 2022.

[44] GIGABYTE, «GB-BPCE-3455 (rev. 1.0),» GIGABYTE.

[45] E. D. Quimis Gómez, «Implementación de un hogar digital por medio de un kit de dispositivos inalámbricos,» Universidad de Guayaquil, Guayaquil, 2019.

[46] Logitech, «HD Webcam C920,» [En línea]. Available: https://www.logitech.com/es-es/products/webcams/c920-pro-hd-webcam.960-001055.html. [Último acceso: 12 Junio 2023].

[47] G. Fernández Fernández, «Elementos de sistemas operativos, de representación de la a información y de procesadores hardware y software,» Universidad Politécnica de Madrid, Madrid, 2015.

[48] S. Sánchez Prado, «Cloud Computing: Fundamentos y despliegue de un servicio en la nube,» Universidad Autónoma de Madrid, Madrid, 2021.

[49] P. Prantosh Kumar y M. K. G. , «Cloud Computing: Possibilities, Challenges and Opportunities with Special Reference to its Emerging Need in the Academic and Working Area of Information Science,» *ElServier*, vol. 38, pp. 2222-2227, 2012.

[50] C. D. M. R. M. A. Alvarado Follegatti, «Investigación de implementación de Cloud Computing en IT-EXPERT,» Universidad Peruana de Ciencias Aplicadas, Lima, 2018.

[51] E. Chen y A. Mark, «Using Jupyter Tools to Design an Interactive Textbook to Guide Undergraduate Research in Materials Informatics,» *Journal of Chemical Education*, vol. 99, nº 10, p. 3601–3606, 2022.

[52] W. R. Rodríguez Dueñas, «FREE SOFTWARE FOR ENGINEERING EDUCATION AND RESEARCH,» *Educación en Ingeniería,* vol. 9, nº 18, pp. 12-22, 2014.

[53] J. Gomez-Cornejo Barrena y V. S. Valverde, «Implementacion de redes neuronales en pltaformas hardware para su aplicación eléctrica,» Universidad del País Vasco, Vasco, 2021.

[54] A. S. Studio, «Amazon SageMaker Studio,» [En línea]. Available: https://aws.amazon.com/es/sagemaker/studio/. [Último acceso: 15 Julio 2023].

[55] Colaboratory. [En línea]. Available: https://research.google.com/colaboratory/faq.html. [Último acceso: 15 Julio 2023].

[56] E. Vural, M. Cetin, A. Ercil, G. Littlewort, M. Bartlell y J. Movellan, «Drowsy Driver Detection Through Facial Movement Analysis,» *International Workshop on Human-Computer Interaction,* pp. 6-18, 2007.

[57] J. Cano Bernet, «Diseñp de un sistema de bajo coste para la detección de la somnolencia en la conducción basado en reconocimiento de expresiones faciales,» Universidad Politécnica de Valencia, Valencia, 2020.

[58] L. F. León Segarra y C. A. López Pinargote, «Análisis del Movimiento de la Cabeza y Hombros de una persona como apoyo al diagnóstico en el tratamiento de traumatismo cervical utilizando técnicas de Visión por Computador,» ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL, Guayaquil, 2015.

[59] A. Peteiro Gándara, «Evaluación de las técncas de aprendizaje estadístico en el software R,» Universidad de Coruña, Coruña, 2023.

[60] T. Dwi Susanto, A. Ingesti Prasetyo y H. M. Astuti, «Web usability evaluation on BloobIS website by using hallway usability testing method and ISO 9241:11,» *Science.gov (United States),* 2018.

ANEXOS

ANEXO A. Categorización de Variables

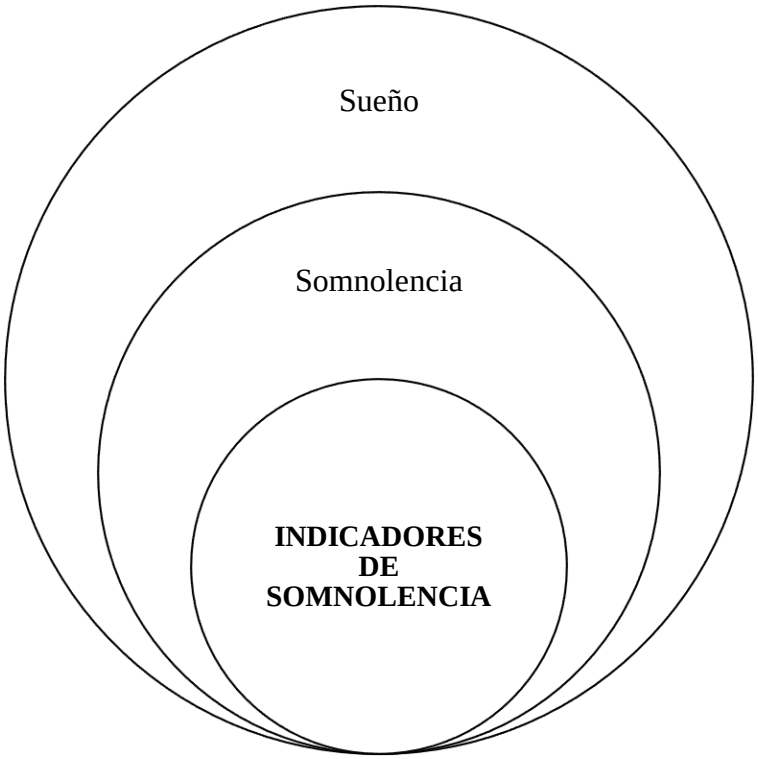


Figura 61. Categorización Variable 1

Elaborado por el investigador

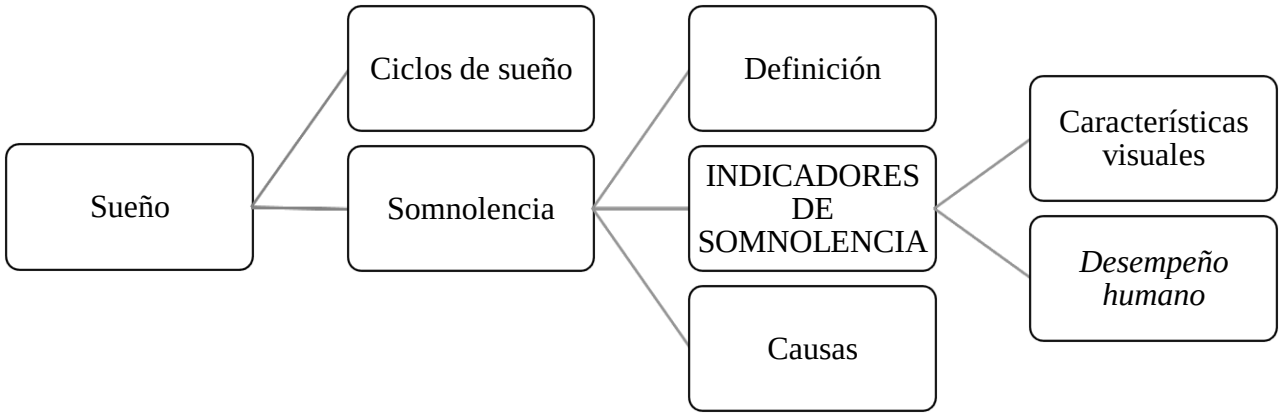
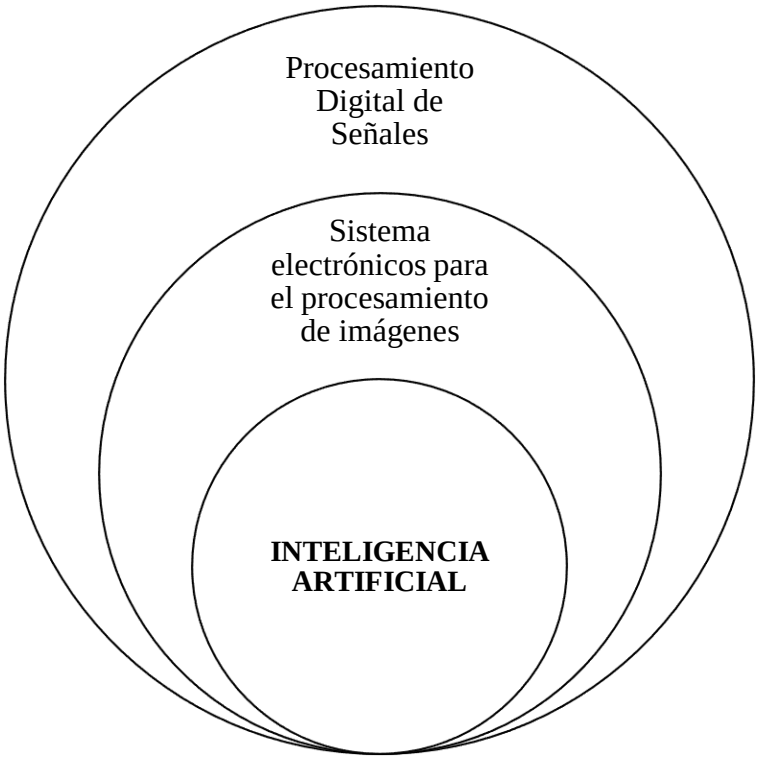


Figura 62. Constelación de Ideas Variable 1

Elaborado por el investigador



*Fi**gura 63.** Categorización Variable 2*

Elaborado por el investigador

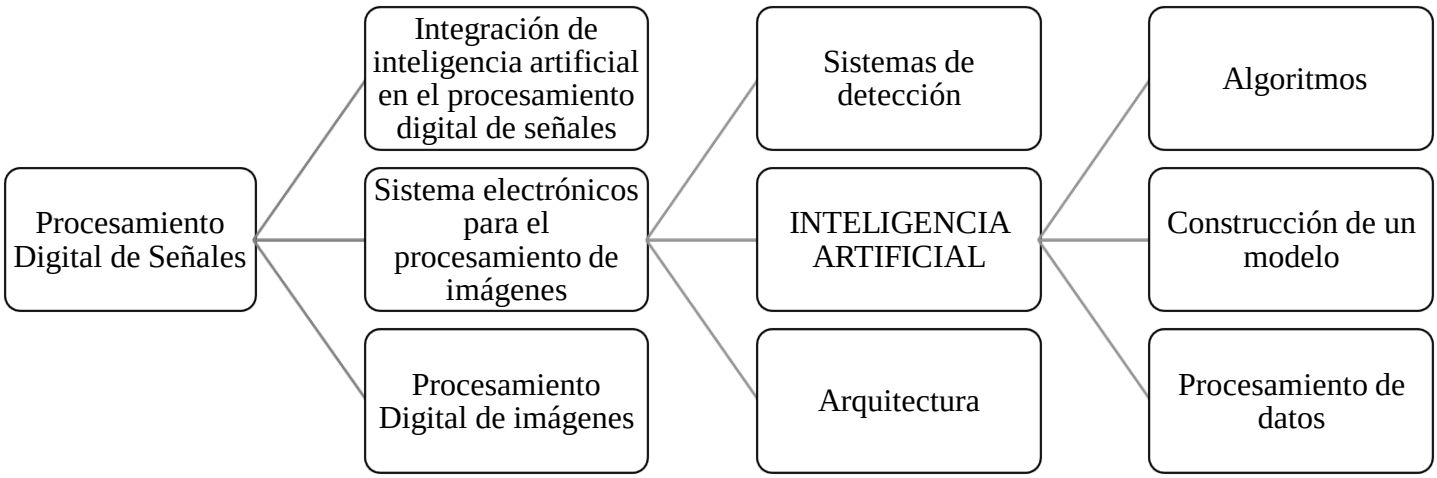


Figura 64. Constelación de Ideas Variable 2

Elaborado por el investigador

ANEXO B. Raspberry Pi Camera Module

Product Name	**Raspberry Pi Camera Module**
Product Description	High Definition camera module compatible with all Raspberry Pi models. Provides high sensitivity, low crosstalk and low noise image capture in an ultra small and lightweight design. The camera module connects to the Raspberry Pi board via the CSI connector designed specifically for interfacing to cameras. The CSI bus is capable of extremely high data rates, and it exclusively carries pixel data to the processor.
RS Part Numer	**913-2664**
Specifications	
Image Sensor	Sony IMX 219 PQ CMOS image sensor in a fixed-focus module.
Resolution	8-megapixel
Still picture resolution	3280 x 2464
Max image transfer rate	1080p: 30fps (encode and decode) 720p: 60fps
Connection to Raspberry Pi	15-pin ribbon cable, to the dedicated 15-pin MIPI Camera Serial Interface (CSI-2).
Image control functions	Automatic exposure control Automatic white balance Automatic band filter Automatic 50/60 Hz luminance detection Automatic black level calibration
Temp range	Operating: -20° to 60° Stable image: -20° to 60°
Lens size	1/4"
Dimensions	23.86 x 25 x 9mm
Weight	3g

Figura 65. Raspberry Pi Camera technical specifications *[40]*

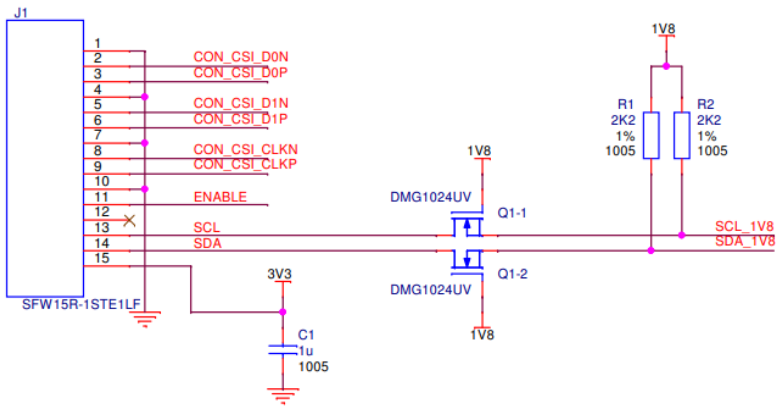


Figura 66. Schematic of the Raspberry Pi CSI camera connector *[40]*

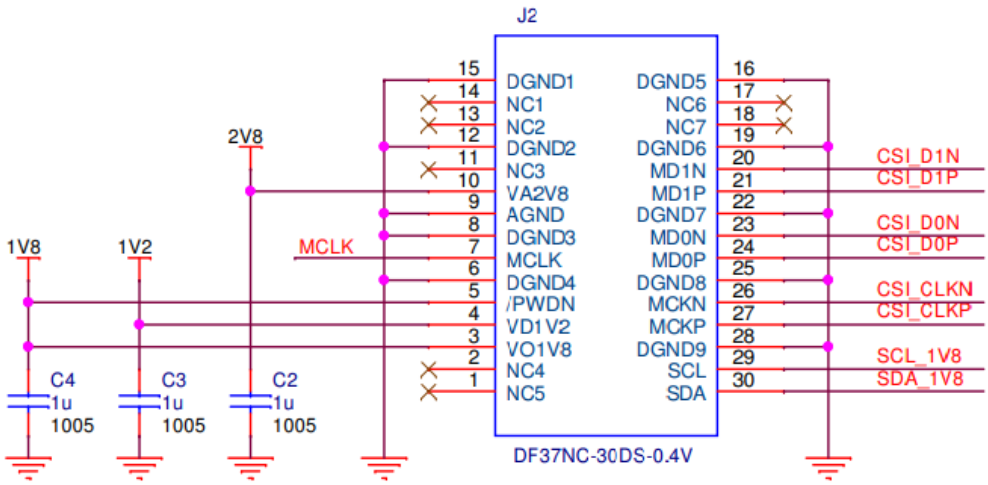


Figura 67. Schematic of the Raspberry Pi CSI *[40]*

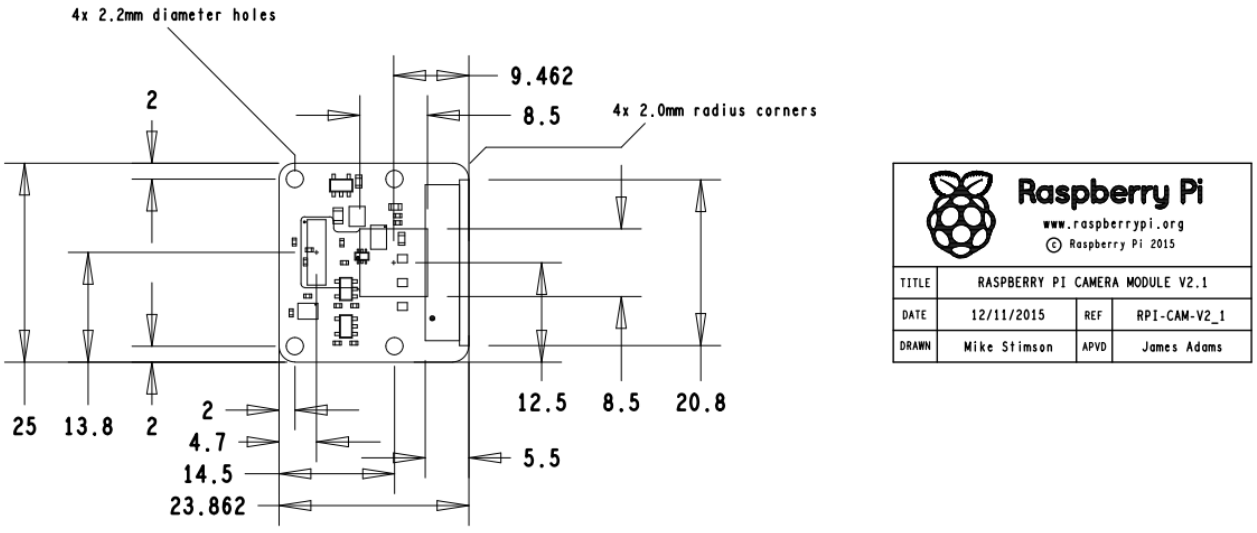


Figura 68. Raspberry Pi camera Module V2.1 Mechanical Drawings *[40]*

ANEXO C. Nvidia Jetson Nano Module

Jetson Nano module

- 128-core NVIDIA Maxwell GPU
- Quad-core ARM® A57 CPU
- 4 GB 64-bit LPDDR4
- 16 GB eMMC 5.1
- 10/100/1000BASE-T Ethernet

Power

- Voltage Input: 5 V
- Module Power: 5 W~10 W

Environment

- Operating Temperature: -25 C to 80 C*
- Storage Temperature: -25 C to 80 C
- Humidity: 85% RH, 85°C [non-operational]
- Vibration: Sinusoidal 5 G RMS 10 to 500 Hz, random 2.88 G RMS, 5 to 500 Hz [non-operational]
- Shock: 140 G, half sine 2 ms duration [non-operational]

NVIDIA JETSON NANO MODULE
TECHNICAL SPECIFICATIONS

GPU	**128-core Maxwell**
CPU	**Quad-core ARM A57 @ 1.43 GHz**
Memory	**4 GB 64-bit LPDDR4 25.6 GB/s**
Storage	**16 GB eMMC 5.1**
Video Encode	**4K @ 30 | 4x 1080p @ 30 | 9x 720p @ 30 (H.264/H.265)**
Video Decode	**4K @ 60 | 2x 4K @ 30 | 8x 1080p @ 30 | 18x 720p @ 30 (H.264/H.265)**
CSI	**12 (3x4 or 4x2) lanes MIPI CSI-2 D-PHY 1.1**
Connectivity	**Gigabit Ethernet**
Display	**HDMI 2.0, eDP 1.4, DP 1.2 (two simultaneously)**
PCIe	**1x1/2/4 PCIe Gen2**
USB	**1x USB 3.0, 3x USB 2.0**
Others	**I²C, I²S, SPI, UART, SD/SDIO, GPIO**
Mechanical	**69.6 mm x 45 mm 260-pin SODIMM Connector**

Figura 69. Nvidia Jetson Nano Module Technical Specifications *[43]*

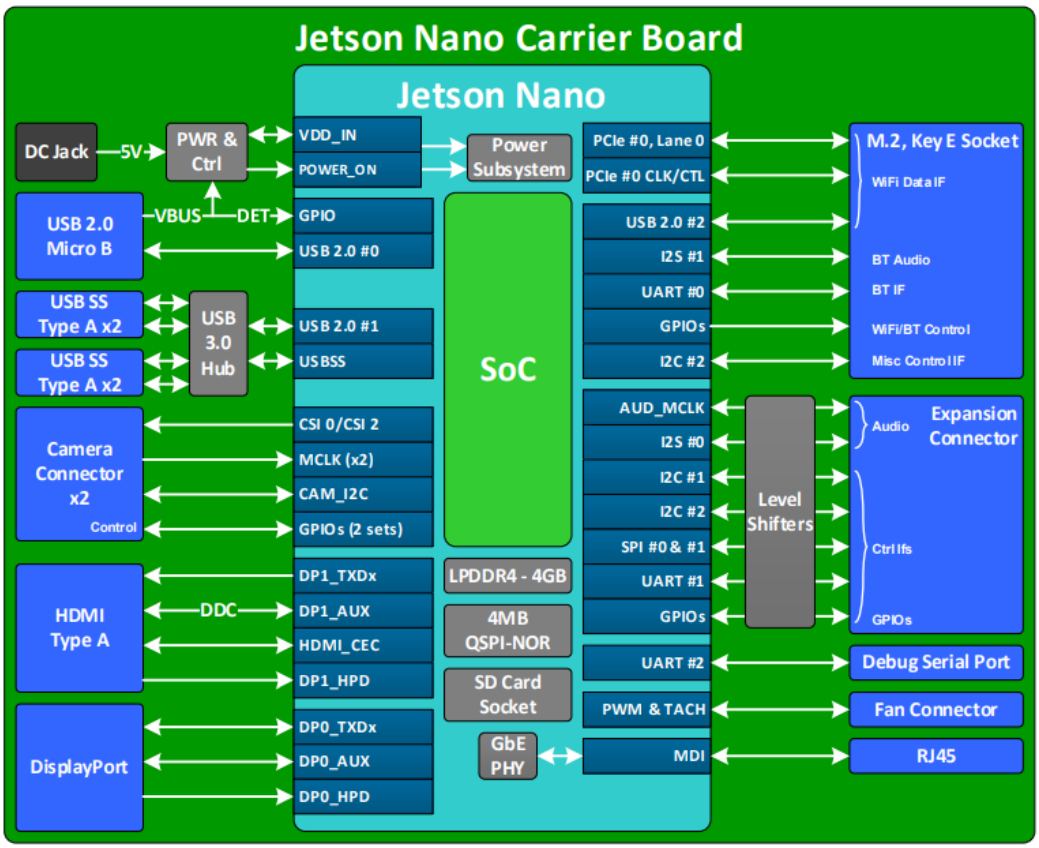


Figura 70. Jetson Nano Block Diagram *[43]*

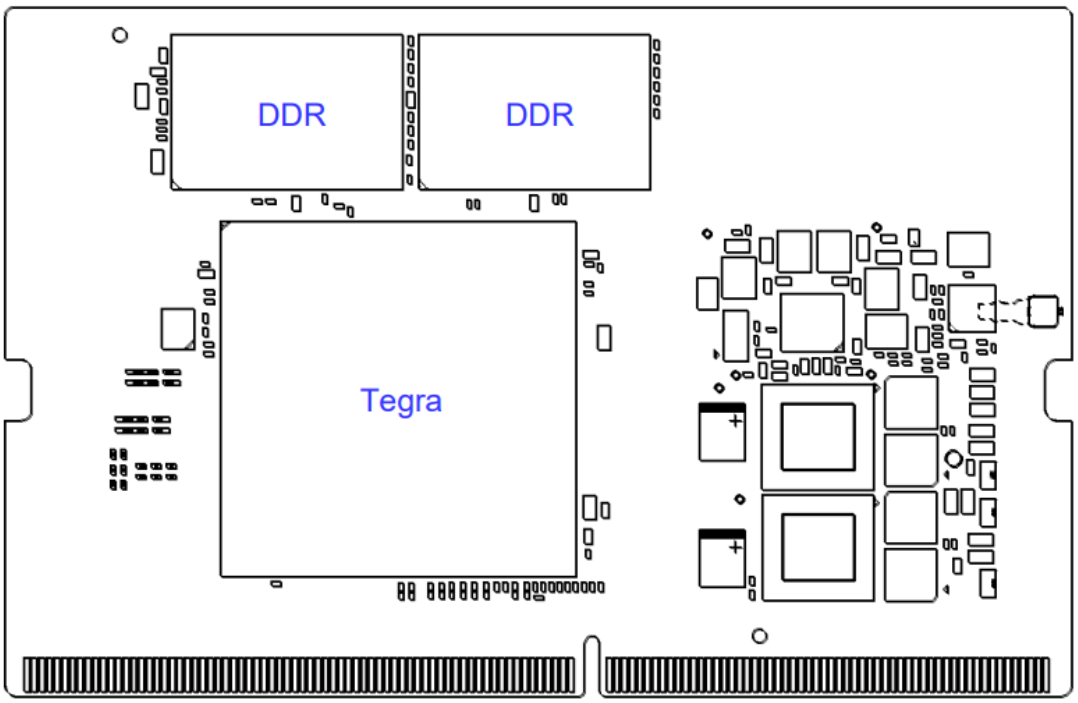


Figura 71. Jetson Nano Placement – Top View *[43]*

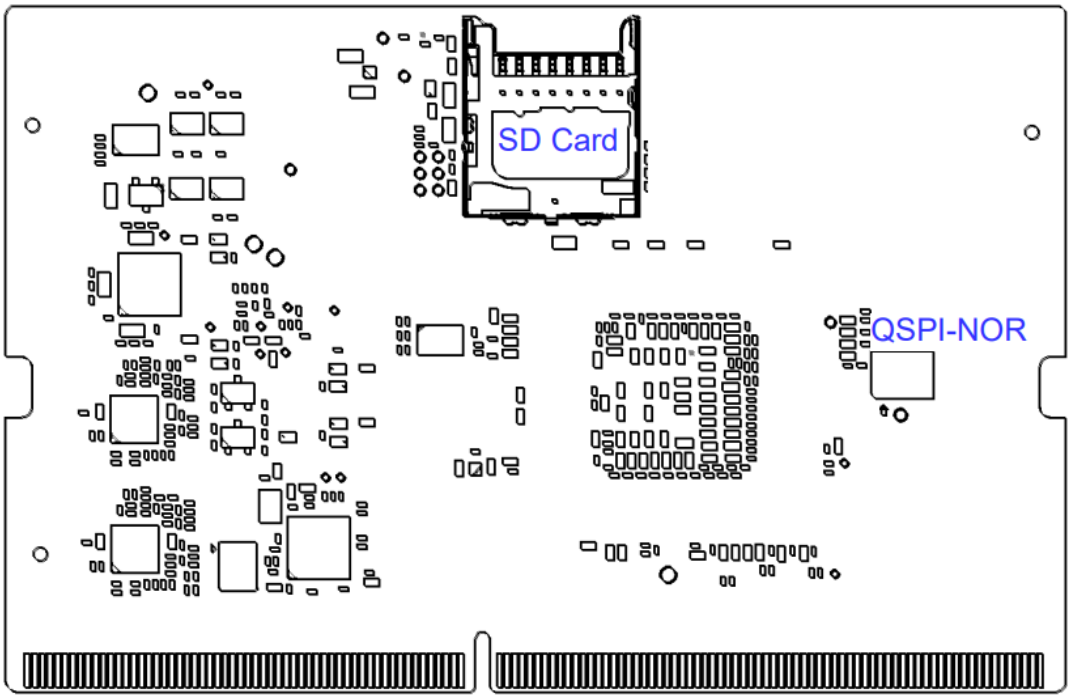


Figura 72. Jetson Nano Placement – Bottom View *[43]*

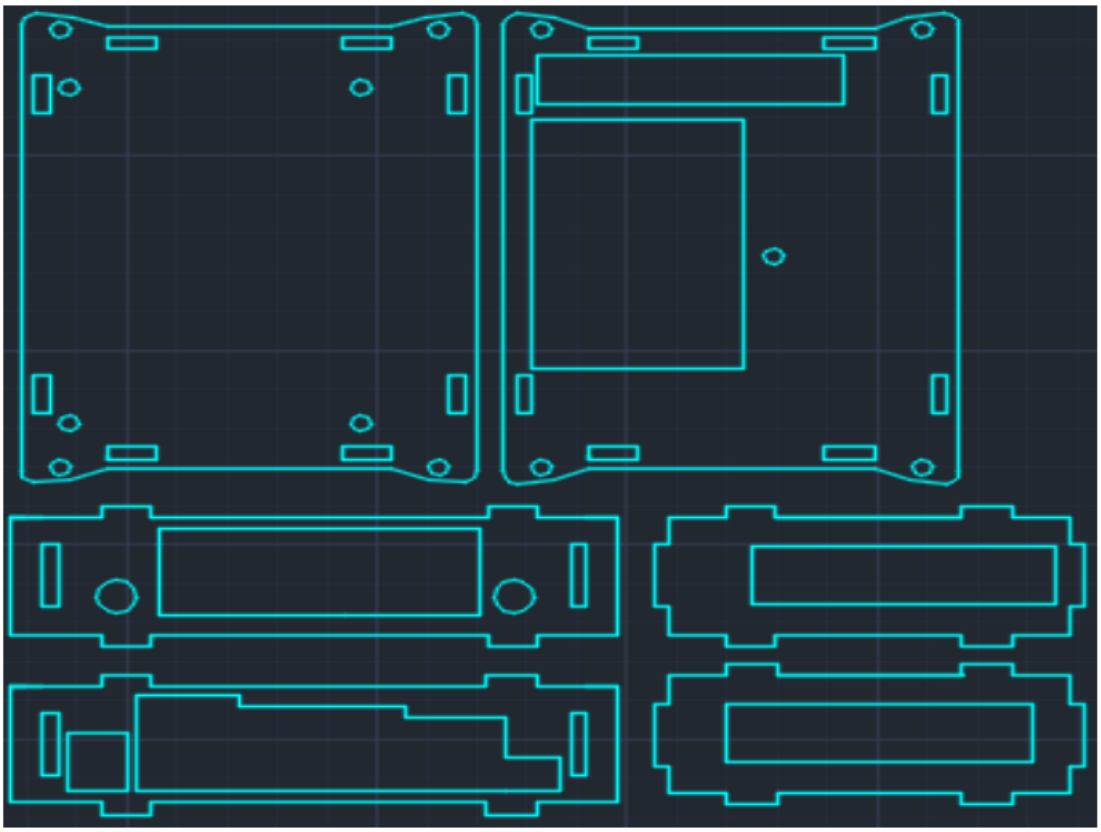


Figura 73. Diseño Case (Acrílico) de protección del kit de desarrollo Jetson Nano.

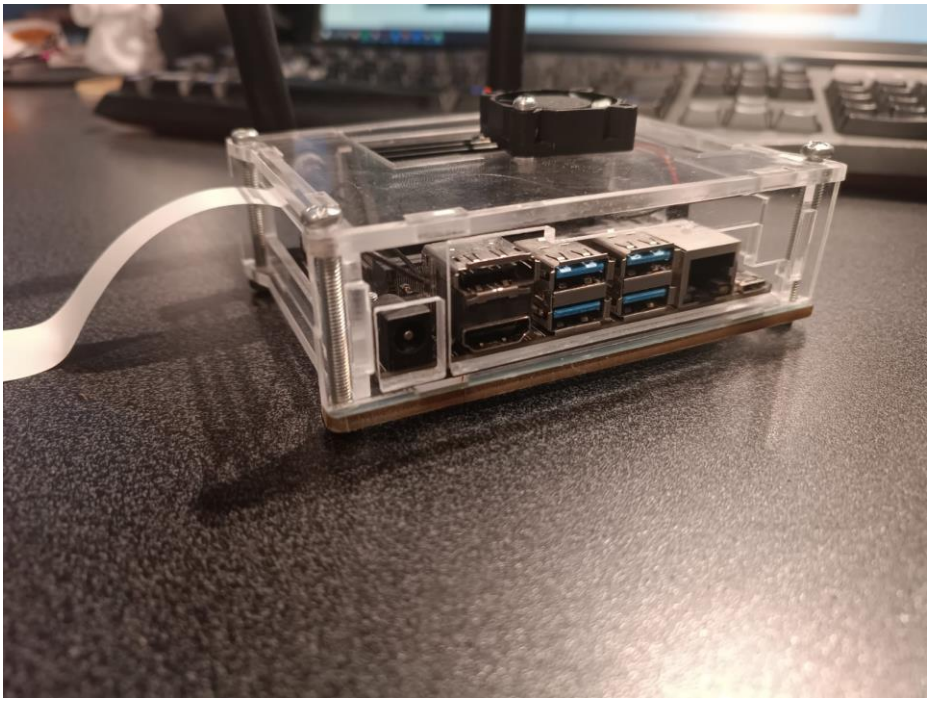


Figura 74. Ensamblado Jetson Nano Carcasa

Elaborado por el investigador

Figura 75. Protección Jetson Nano

Elaborado por el investigador

ANEXO D. Implementación del sistema

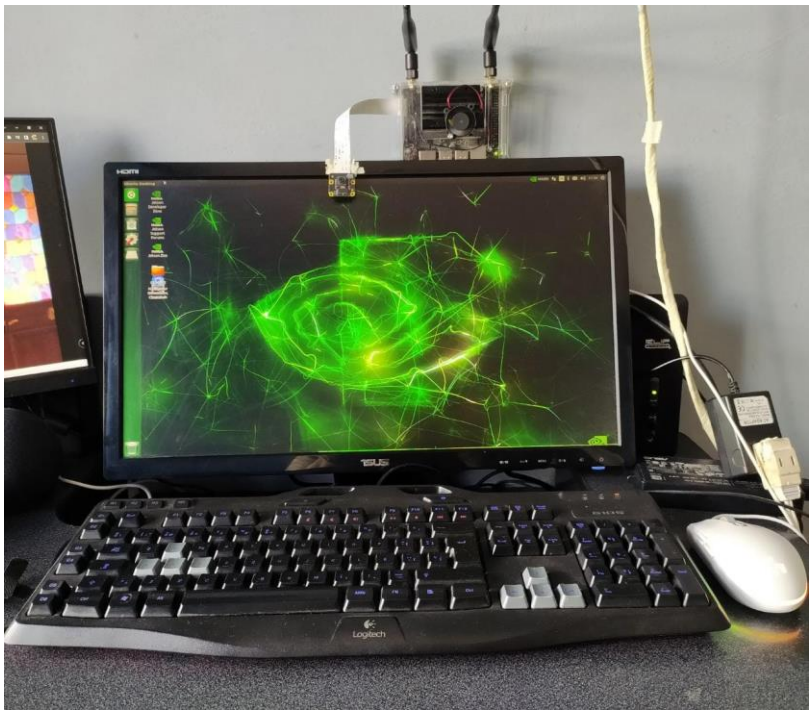
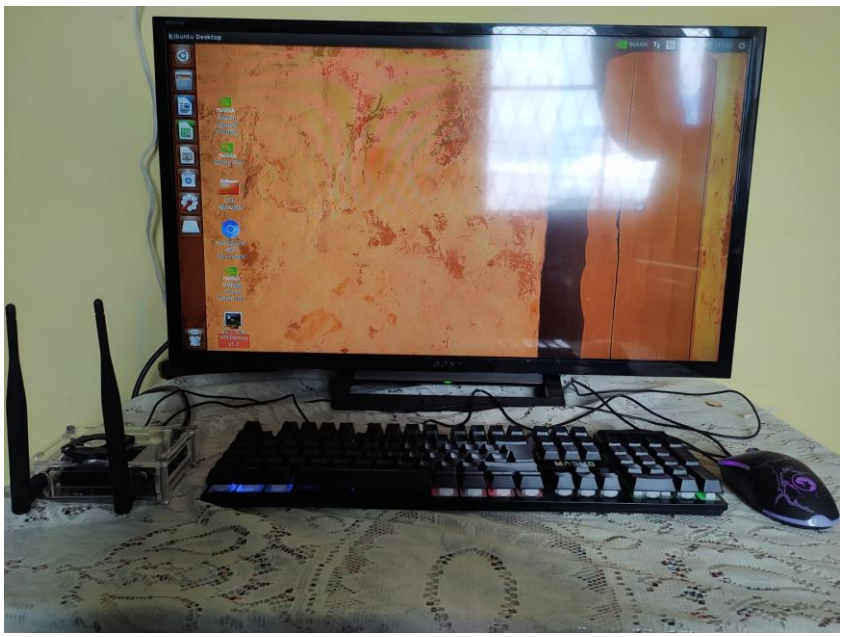


Figura 76. Implementación del Sistema de Reconocimiento de Indicadores de Somnolencia mediante Inteligencia Artificial

Elaborado por el investigador



ANEXO E. Algoritmo de detección en lenguaje Python

Detección mediante imagen

```
# construir el analizador de argumentos y analizar los argumentos
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", required=True,
    help="path to input image")
ap.add_argument("-f", "--face", type=str,
    default="face_detector",
    help="path to face detector model directory")
ap.add_argument("-m", "--model", type=str,
    default="mask_detector.model",
    help="path to trained face mask detector model")
ap.add_argument("-c", "--confidence", type=float, default=0.5,
    help="minimum probability to filter weak detections")
args = vars(ap.parse_args())

# cargar nuestro modelo serializado de detector de rostros desde el disco
print("[INFO] loading face detector model...")
prototxtPath = os.path.sep.join([args["face"], "deploy.prototxt"])
weightsPath = os.path.sep.join([args["face"],
    "res10_300x300_ssd_iter_140000.caffemodel"])
net = cv2.dnn.readNet(prototxtPath, weightsPath)

# cargar el modelo de detector de máscara facial desde el disco
print("[INFO] loading face mask detector model...")
model = load_model(args["model"])

# cargar la imagen de entrada desde el disco, clonarla y capturar la imagen espacial
# dimensiones
image = cv2.imread(args["image"])
orig = image.copy()
(h, w) = image.shape[:2]

# construir un objeto binario a partir de la imagen
blob = cv2.dnn.blobFromImage(image, 1.0, (300, 300),
    (104.0, 177.0, 123.0))

# pasar el objeto binario a través de la red y obtener las detecciones de rostros
print("[INFO] computing face detections...")
net.setInput(blob)
detections = net.forward()
```

```
# asegúrese de que los cuadros delimitadores estén dentro de las dimensiones de
# el marco
(startX, startY) = (max(0, startX), max(0, startY))
(endX, endY) = (min(w - 1, endX), min(h - 1, endY))

# extraiga el ROI de la cara, conviértalo de BGR a canal RGB
# ordenar, redimensionarlo a 224x224 y preprocesarlo
face = image[startY:endY, startX:endX]
face = cv2.cvtColor(face, cv2.COLOR_BGR2RGB)
face = cv2.resize(face, (224, 224))
face = img_to_array(face)
face = preprocess_input(face)
face = np.expand_dims(face, axis=0)

# pase la cara a través del modelo para determinar si la cara
# tiene una máscara o no
(mask, withoutMask) = model.predict(face)[0]

# determinar la etiqueta de clase y el color que usaremos para dibujar
# el cuadro delimitador y el texto
label = "Mask" if mask > withoutMask else "No Mask"
color = (0, 255, 0) if label == "Mask" else (0, 0, 255)

# incluir la probabilidad en la etiqueta
label = "{}: {:.2f}%".format(label, max(mask, withoutMask) * 100)

# mostrar la etiqueta y el rectángulo del cuadro delimitador en la salida
# marco
cv2.putText(image, label, (startX, startY - 10),
    cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)
cv2.rectangle(image, (startX, startY), (endX, endY), color, 2)
```

Detección por medio de video

```
def detect_and_predict_mask(frame, faceNet, maskNet):
    # tomar las dimensiones del marco y luego construir una gota
    # de eso
    (h, w) = frame.shape[:2]
    blob = cv2.dnn.blobFromImage(frame, 1.0, (300, 300),
        (104.0, 177.0, 123.0))
    # pasar el objeto binario a través de la red y obtener las detecciones de rostros
    faceNet.setInput(blob)
    detections = faceNet.forward()
    # inicializar nuestra lista de caras, sus ubicaciones correspondientes,
    # y la lista de predicciones de nuestra red de mascarillas
    faces = []
    locs = []
    preds = []
    # recorrer las detecciones
    for i in range(0, detections.shape[2]):
        # extraer la confianza (es decir, la probabilidad) asociada con
        # la detección
        confidence = detections[0, 0, i, 2]
        # filtrar las detecciones débiles asegurándose de que la confianza sea
        # mayor que la confianza mínima
        if confidence > args["confidence"]:
            # calcular las coordenadas (x, y) del cuadro delimitador para
            # el objeto
            box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
            (startX, startY, endX, endY) = box.astype("int")
            # asegúrese de que los cuadros delimitadores estén dentro de las dimensiones de
            # el marco
            (startX, startY) = (max(0, startX), max(0, startY))
            (endX, endY) = (min(w - 1, endX), min(h - 1, endY))
            # extraiga el ROI de la cara, conviértalo de BGR a canal RGB
            # ordenar, redimensionarlo a 224x224 y preprocesarlo
            face = frame[startY:endY, startX:endX]
            face = cv2.cvtColor(face, cv2.COLOR_BGR2RGB)
            face = cv2.resize(face, (224, 224))
            face = img_to_array(face)
            face = preprocess_input(face)
            face = np.expand_dims(face, axis=0)
```

```
# construir el analizador de argumentos y analizar los argumentos
ap = argparse.ArgumentParser()
ap.add_argument("-f", "--face", type=str,
    default="face_detector",
    help="path to face detector model directory")
ap.add_argument("-m", "--model", type=str,
    default="mask_detector.model",
    help="path to trained face mask detector model")
ap.add_argument("-c", "--confidence", type=float, default=0.5,
    help="minimum probability to filter weak detections")
args = vars(ap.parse_args())

# cargar nuestro modelo serializado de detector de rostros desde el disco
print("[INFO] loading face detector model...")
prototxtPath = os.path.sep.join([args["face"], "deploy.prototxt"])
weightsPath = os.path.sep.join([args["face"],
    "res10_300x300_ssd_iter_140000.caffemodel"])
faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)

# cargar el modelo de detector de máscara facial desde el disco
print("[INFO] loading face mask detector model...")
maskNet = load_model(args["model"])

# inicialice la transmisión de video y permita que el sensor de la cámara se active
print("[INFO] starting video stream...")
vs = VideoStream(src=0).start()
# vs = cv2.VideoCapture("http://192.168.1.37:8080/video")#cámara a través de servidor
time.sleep(2.0)

# recorrer los fotogramas de la transmisión de video
while True:
    # toma el marco de la secuencia de video encadenada y cambia su tamaño
    # tener un ancho máximo de 400 píxeles
    frame = vs.read()
#   valid, frame = vs.read()
    frame = imutils.resize(frame, height=600, width=800) #1536*864 ventana 1510*850

    # detectar rostros en el marco y determinar si llevan puesto un
    # mascarilla o no
    (locs, preds) = detect_and_predict_mask(frame, faceNet, maskNet)
```

```
for (box, pred) in zip(locs, preds):
    # unpack the bounding box and predictions
    (startX, startY, endX, endY) = box
    (mask, withoutMask) = pred

    # determinar la etiqueta de clase y el color que usaremos para dibujar
    # el cuadro delimitador y el texto
    label = "CON MASCARILLA" if mask > withoutMask else "SIN MASCARILLA"
    color = (0, 255, 0) if label == "CON MASCARILLA" else (0, 0, 255)

    # incluir la probabilidad en la etiqueta
    label = "{}: {:.2f}%".format(label, max(mask, withoutMask) * 100)

    # mostrar la etiqueta y el rectángulo del cuadro delimitador en la salida
    # marco
    cv2.putText(frame, label, (startX, startY - 10),
    cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)
    cv2.rectangle(frame, (startX, startY), (endX, endY), color, 2)
#crear interfaz
logenc=cv2.imread("portada.jpg")
loglat=cv2.imread("lat.jpg")
rell=cv2.imread("relleno.jpg")
conh=cv2.hconcat([loglat,frame,rell])
conv=cv2.vconcat([logenc,conh])
# print('Resolution: ' + str(frame.shape[0]) + ' x ' + str(frame.shape[1]))
cv2.namedWindow("FISEI", cv2.WINDOW_NORMAL)
cv2.setWindowProperty("FISEI",cv2.WND_PROP_FULLSCREEN,cv2.WINDOW_NORMAL)
cv2.imshow("FISEI", conv)
key = cv2.waitKey(1) & 0xFF
# if the `q` key was pressed, break from the loop
if key == ord("q"):
    break
```

Entrenamiento manual algoritmo

```
# construir el analizador de argumentos y analizar los argumentos
ap = argparse.ArgumentParser()
ap.add_argument("-d", "--dataset", required=True,
    help="path to input dataset")
ap.add_argument("-p", "--plot", type=str, default="plot.png",
    help="path to output loss/accuracy plot")
ap.add_argument("-m", "--model", type=str,
    default="mask_detector.model",
    help="path to output face mask detector model")
args = vars(ap.parse_args())

# inicializar la tasa de aprendizaje inicial, el número de épocas para entrenar,
# y tamaño del lote
INIT_LR = 1e-4
EPOCHS = 20
BS = 32

# tomar la lista de imágenes en nuestro directorio de conjunto de datos, luego inicializar
# la lista de datos (es decir, imágenes) e imágenes de clase
print("[INFO] loading images...")
imagePaths = list(paths.list_images(args["dataset"]))
data = []
labels = []

# recorrer las rutas de la imagen
for imagePath in imagePaths:
    # extrae la etiqueta de clase del nombre del archivo
    label = imagePath.split(os.path.sep)[-2]

    # cargar la imagen de entrada (224x224) y preprocesarla
    image = load_img(imagePath, target_size=(224, 224))
    image = img_to_array(image)
    image = preprocess_input(image)

    # actualizar las listas de datos y etiquetas, respectivamente
    data.append(image)
    labels.append(label)

# convertir los datos y las etiquetas en matrices NumPy
data = np.array(data, dtype="float32")
labels = np.array(labels)
```

```
# construye la cabeza del modelo que se colocará encima del modelo base
headModel = baseModel.output
headModel = AveragePooling2D(pool_size=(7, 7))(headModel)
headModel = Flatten(name="flatten")(headModel)
headModel = Dense(128, activation="relu")(headModel)
headModel = Dropout(0.5)(headModel)
headModel = Dense(2, activation="softmax")(headModel)

# coloque el modelo FC de la cabeza encima del modelo base (esto se convertirá
# el modelo real que entrenaremos)
model = Model(inputs=baseModel.input, outputs=headModel)

# recorrer todas las capas en el modelo base y congelarlas para que
# *no* ser actualizado durante el primer proceso de entrenamiento
for layer in baseModel.layers:
    layer.trainable = False

# compila nuestra modelo
print("[INFO] compiling model...")
opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
model.compile(loss="binary_crossentropy", optimizer=opt,
    metrics=["accuracy"])

# entrena nuestra cabecera de red
print("[INFO] training head...")
H = model.fit(
    aug.flow(trainX, trainY, batch_size=BS),
    steps_per_epoch=len(trainX) // BS,
    validation_data=(testX, testY),
    validation_steps=len(testX) // BS,
    epochs=EPOCHS)

# hacer predicciones sobre el conjunto de prueba
print("[INFO] evaluating network...")
predIdxs = model.predict(testX, batch_size=BS)

# para cada imagen en el conjunto de prueba, necesitamos encontrar el índice de la
# etiqueta con la mayor probabilidad predicha correspondiente
predIdxs = np.argmax(predIdxs, axis=1)

# mostrar un informe de clasificación bien formateado
print(classification_report(testY.argmax(axis=1), predIdxs,
    target_names=lb.classes_))
```

```
    "FaceDetails": [
            {
                "BoundingBox": {
                    "Width": 0.044981569051742554,
                    "Height": 0.0841514840722084,
                    "Left": 0.5064443945884705,
                    "Top": 0.24541109800338745
                },
                "AgeRange": {
                    "Low": 29,
                    "High": 39
                },
                "Smile": {
                    "Value": false,
                    "Confidence": 92.79318237304688
                },
                "Eyeglasses": {
                    "Value": false,
                    "Confidence": 97.64583587646484
                },
                "Sunglasses": {
                    "Value": false,
                "Confidence": 99.99668884277344
            },
                "Gender": {
                    "Value": "Female",
                    "Confidence": 99.99964904785156
                },
                "Beard": {
                    "Value": false,
                    "Confidence": 95.89328002929688
                },
                "Mustache": {
                    "Value": false,
                    "Confidence": 98.50775146484375
                },
                "EyesOpen": {
                    "Value": true,
                    "Confidence": 98.4903793334961
                },
                "MouthOpen": {
                    "Value": false,
                    "Confidence": 95.05741882324219
                },
                "Emotions": [
                    {
                        "Type": "CALM",
                        "Confidence": 78.67639923095703
                    },
                    {
                        "Type": "CONFUSED",
                        "Confidence": 10.154038429260254
            },
```

```
        {
            "Type": "SURPRISED",
            "Confidence": 6.941589832305908
        },
        {
            "Type": "FEAR",
            "Confidence": 6.1012187004089355
        },
        {
            "Type": "HAPPY",
            "Confidence": 3.671717643737793
        },
        {
            "Type": "SAD",
            "Confidence": 3.35422945022583
        },
        {
            "Type": "ANGRY",
            "Confidence": 1.3010478019714355
        },
        {
            "Type": "DISGUSTED",
            "Confidence": 1.0862025022506714
        }
    ],
    "Landmarks": [
        {
            "Type": "eyeLeft",
            "X": 0.5256866812705994,
         "Y": 0.277765154838562
              "Type": "eyeRight",
                  "X": 0.5426272749900818,
            "Y": 0.27649611234664917
        },
        {
            "Type": "mouthLeft",
            "X": 0.5279320478439331,
            "Y": 0.30769720673561096
        },
        {
            "Type": "mouthRight",
            "X": 0.5420536994934082,
            "Y": 0.30646422505378723
        },
        {
            "Type": "nose",
            "X": 0.5403344035148621,
            "Y": 0.2924666702747345
        },
        {
            "Type": "leftEyeBrowLeft",
            "X": 0.5169908404350281,
            "Y": 0.2717568278312683
        },
        {
            "
```

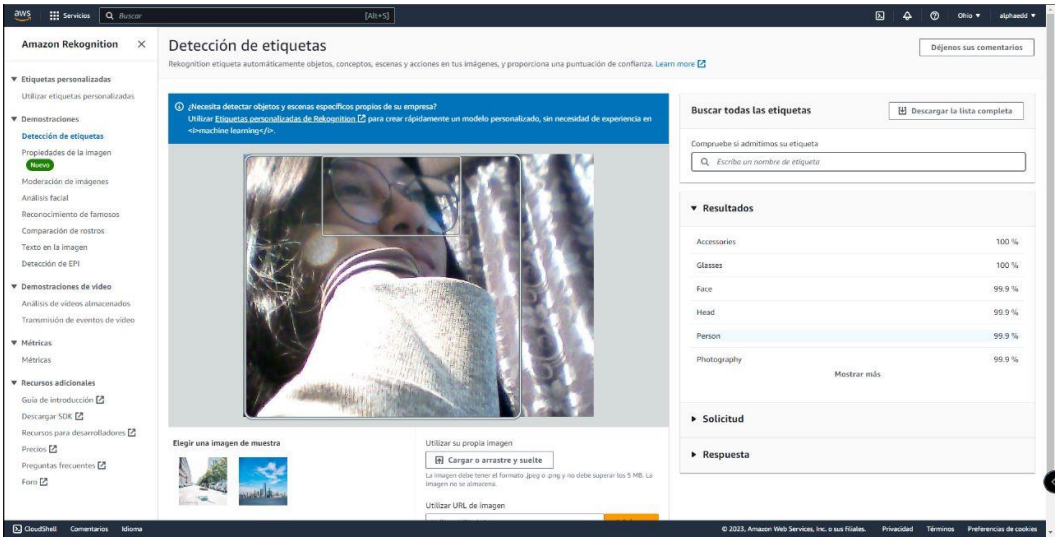


Figura 77. Entrenamiento por medio de AWS

Elaborado por el investigador

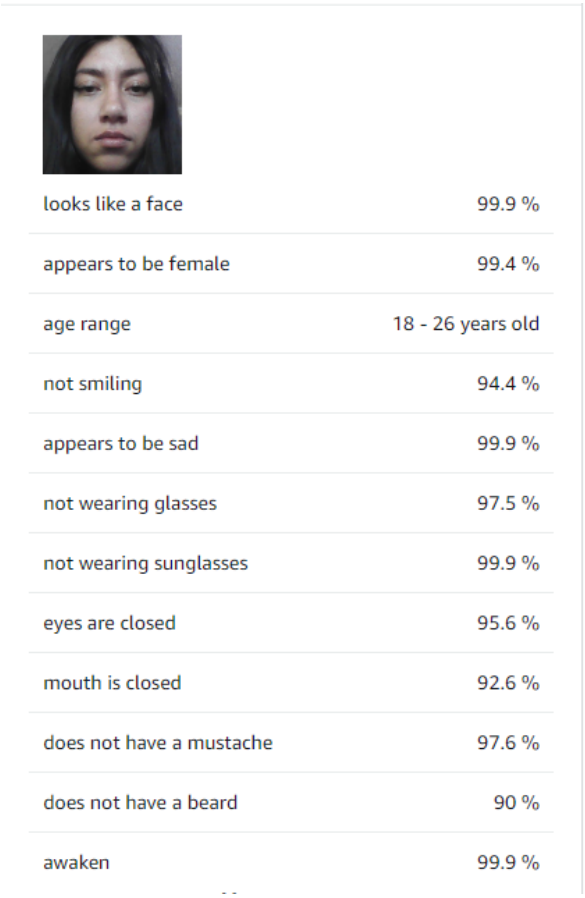


Figura 78. Parámetros detectados

Elaborado por el investigador

ANEXO G. Modelos preeentrenados

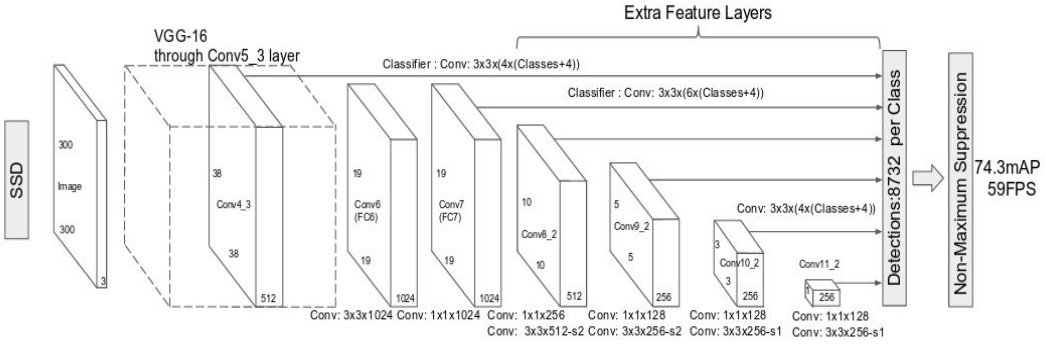


Figura 79. Res10_300x300_SSD_iter_140000

Res10_300x300_SSD_iter_140000	Deploy.prototxt	face_detector.model

input: "data"
input_shape {
dim: 1
dim: 3
dim: 300
dim: 300
}
 	
layer {
name: "data_bn"
type: "BatchNorm"
bottom: "data"
top: "data_bn"
param {
lr_mult: 0.0
}
param {
lr_mult: 0.0
}
param {
lr_mult: 0.0
}
}
layer {
name: "data_scale"
type: "Scale"
bottom: "data_bn"
top: "data_bn"
param {
lr_mult: 1.0
decay_mult: 1.0
}
param {
lr_mult: 2.0
decay_mult: 1.0
}
scale_param {
bias_term: true
}
}
layer {
name: "conv1_h"
type: "Convolution"
	name: "CaffeNet"
	layer {
	name: "data"
	type: "Input"
	top: "data"
	input_param { shape: { dim: 10 dim: 3 dim: 227 dim: 227 } }
	}
	layer {
	name: "conv1"
	type: "Convolution"
	bottom: "data"
	top: "conv1"
	convolution_param {
	num_output: 96
	kernel_size: 11
	stride: 4
	}
	}
	layer {
	name: "relu1"
	type: "ReLU"
	bottom: "conv1"
	top: "conv1"
	}
	layer {
	name: "pool1"
	type: "Pooling"
	bottom: "conv1"
	top: "pool1"
	pooling_param {
	pool: MAX
	kernel_size: 3
	stride: 2
	}
	}
	layer {
	name: "norm1"
	type: "LRN"
	bottom: "pool1"
	top: "norm1"
	lrn_param {
	local_size: 5
		{
		"FaceDetails": [
		{
		"BoundingBox": {
		"Width":
		0.044981569051742554,
		"Height":
		0.0841514840722084,
		"Left":
		0.5064443945884705,
		"Top":
		0.24541109800338745
		},
		"AgeRange": {
		"Low": 29,
		"High": 39
		},
		"Smile": {
		"Value": false,
		"Confidence":
		92.79318237304688
		},
		"Eyeglasses": {
		"Value": false,
		"Confidence":
		97.64583587646484
		},
		"Sunglasses": {
		"Value": false,
		"Confidence":
		99.99668884277344
		},
		"Gender": {
		"Value": "Female",
		"Confidence":
		99.99964904785156
		},
		"Beard": {
		"Value": false,
		"Confidence":
		95.89328002929688
		},
		"Mustache": {

```
  bottom: "data_bn"
  top: "conv1_h"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 1.0
  }
  convolution_param {
    num_output: 32
    pad: 3
    kernel_size: 7
    stride: 2
    weight_filler {
      type: "msra"
      variance_norm: FAN_OUT
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "conv1_bn_h"
  type: "BatchNorm"
  bottom: "conv1_h"
  top: "conv1_h"
  param {
    lr_mult: 0.0
  }
  param {
    lr_mult: 0.0
  }
  param {
    lr_mult: 0.0
  }
}
layer {
  name: "conv1_scale_h"
  type: "Scale"
  bottom: "conv1_h"
  top: "conv1_h"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 1.0
  }
  scale_param {
    bias_term: true
  }
}
layer {
  name: "conv1_relu"
  type: "ReLU"
  bottom: "conv1_h"
  top: "conv1_h"
}
layer {
  name: "conv1_pool"
  type: "Pooling"
  bottom: "conv1_h"
    alpha: 0.0001
    beta: 0.75
  }
}
layer {
  name: "conv2"
  type: "Convolution"
  bottom: "norm1"
  top: "conv2"
  convolution_param {
    num_output: 256
    pad: 2
    kernel_size: 5
    group: 2
  }
}
layer {
  name: "relu2"
  type: "ReLU"
  bottom: "conv2"
  top: "conv2"
}
layer {
  name: "pool2"
  type: "Pooling"
  bottom: "conv2"
  top: "pool2"
  pooling_param {
    pool: MAX
    kernel_size: 3
    stride: 2
  }
}
layer {
  name: "norm2"
  type: "LRN"
  bottom: "pool2"
  top: "norm2"
  lrn_param {
    local_size: 5
    alpha: 0.0001
    beta: 0.75
  }
}
layer {
  name: "conv3"
  type: "Convolution"
  bottom: "norm2"
  top: "conv3"
  convolution_param {
    num_output: 384
    pad: 1
    kernel_size: 3
  }
}
layer {
  name: "relu3"
  type: "ReLU"
  bottom: "conv3"
  top: "conv3"
}
layer {
  name: "conv4"
  type: "Convolution"
  bottom: "conv3"
  top: "conv4"
  convolution_param {
            "Value": false,
            "Confidence":
98.50775146484375
        },
        "EyesOpen": {
            "Value": true,
            "Confidence":
98.4903793334961
        },
        "MouthOpen": {
            "Value": false,
            "Confidence":
95.05741882324219
        },
        "Emotions": [
            {
                "Type": "CALM",
                "Confidence":
78.67639923095703
            },
            {
                "Type":
"CONFUSED",
                "Confidence":
10.154038429260254
            },
            {
                "Type":
"SURPRISED",
                "Confidence":
6.941589832305908
            },
            {
                "Type": "FEAR",
                "Confidence":
6.1012187004089355
            },
            {
                "Type": "HAPPY",
                "Confidence":
3.671717643737793
            },
            {
                "Type": "SAD",
                "Confidence":
3.35422945022583
            },
            {
                "Type": "ANGRY",
                "Confidence":
1.3010478019714355
            },
            {
                "Type":
"DISGUSTED",
                "Confidence":
1.0862025022506714
            }
        ],
        "Landmarks": [
            {
                "Type": "eyeLeft",
                "X":
0.5256866812705994,
                "Y":
0.277765154838562
            },
```

```
  top: "conv1_pool"
  pooling_param {
    kernel_size: 3
    stride: 2
  }
}
layer {
  name: "layer_64_1_conv1_h"
  type: "Convolution"
  bottom: "conv1_pool"
  top: "layer_64_1_conv1_h"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  convolution_param {
    num_output: 32
    bias_term: false
    pad: 1
    kernel_size: 3
    stride: 1
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "layer_64_1_bn2_h"
  type: "BatchNorm"
  bottom: "layer_64_1_conv1_h"
  top: "layer_64_1_conv1_h"
  param {
    lr_mult: 0.0
  }
  param {
    lr_mult: 0.0
  }
  param {
    lr_mult: 0.0
  }
}
layer {
  name: "layer_64_1_scale2_h"
  type: "Scale"
  bottom: "layer_64_1_conv1_h"
  top: "layer_64_1_conv1_h"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 1.0
  }
  scale_param {
    bias_term: true
  }
}
layer {
  name: "layer_64_1_relu2"
  type: "ReLU"
  bottom: "layer_64_1_conv1_h"
  top: "layer_64_1_conv1_h"
    num_output: 384
    pad: 1
    kernel_size: 3
    group: 2
  }
}
layer {
  name: "relu4"
  type: "ReLU"
  bottom: "conv4"
  top: "conv4"
}
layer {
  name: "conv5"
  type: "Convolution"
  bottom: "conv4"
  top: "conv5"
  convolution_param {
    num_output: 256
    pad: 1
    kernel_size: 3
    group: 2
  }
}
layer {
  name: "relu5"
  type: "ReLU"
  bottom: "conv5"
  top: "conv5"
}
layer {
  name: "pool5"
  type: "Pooling"
  bottom: "conv5"
  top: "pool5"
  pooling_param {
    pool: MAX
    kernel_size: 3
    stride: 2
  }
}
layer {
  name: "fc6"
  type: "InnerProduct"
  bottom: "pool5"
  top: "fc6"
  inner_product_param {
    num_output: 4096
  }
}
layer {
  name: "relu6"
  type: "ReLU"
  bottom: "fc6"
  top: "fc6"
}
layer {
  name: "drop6"
  type: "Dropout"
  bottom: "fc6"
  top: "fc6"
  dropout_param {
    dropout_ratio: 0.5
  }
}
layer {
  name: "fc7"
```

```
            {
              "Type": "eyeRight",
              "X":
0.5426272749900818,
              "Y":
0.27649611234664917
            },
            {
              "Type": "mouthLeft",
              "X":
0.5279320478439331,
              "Y":
0.30769720673561096
            },
            {
              "Type": "mouthRight",
              "X":
0.5420536994934082,
              "Y":
0.30646422505378723
            },
            {
              "Type": "nose",
              "X":
0.5403344035148621,
              "Y":
0.2924666702747345
            },
            {
              "Type":
"leftEyeBrowLeft",
              "X":
0.5169908404350281,
              "Y":
0.2717568278312683
            },
            {
              "Type":
"leftEyeBrowRight",
              "X":
0.5308933258056641,
              "Y":
0.26853516697883606
            },
            {
              "Type":
"leftEyeBrowUp",
              "X":
0.5248442888259888,
              "Y":
0.26744481921195984
            },
            {
              "Type":
"rightEyeBrowLeft",
              "X":
0.5406660437583923,
              "Y":
0.2679397761821747
            },
            {
              "Type":
"rightEyeBrowRight",
              "X":
0.5465580224990845,
              "Y":
0.26960599422454834
```

```
}
layer {
  name: "layer_64_1_conv2_h"
  type: "Convolution"
  bottom: "layer_64_1_conv1_h"
  top: "layer_64_1_conv2_h"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  convolution_param {
    num_output: 32
    bias_term: false
    pad: 1
    kernel_size: 3
    stride: 1
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "layer_64_1_sum"
  type: "Eltwise"
  bottom: "layer_64_1_conv2_h"
  bottom: "conv1_pool"
  top: "layer_64_1_sum"
}
layer {
  name: "layer_128_1_bn1_h"
  type: "BatchNorm"
  bottom: "layer_64_1_sum"
  top: "layer_128_1_bn1_h"
  }
```

```
  type: "InnerProduct"
  bottom: "fc6"
  top: "fc7"
  inner_product_param {
    num_output: 4096
  }
}
layer {
  name: "relu7"
  type: "ReLU"
  bottom: "fc7"
  top: "fc7"
}
layer {
  name: "drop7"
  type: "Dropout"
  bottom: "fc7"
  top: "fc7"
  dropout_param {
    dropout_ratio: 0.5
  }
}
layer {
  name: "fc8"
  type: "InnerProduct"
  bottom: "fc7"
  top: "fc8"
  inner_product_param {
    num_output: 1000
  }
}
layer {
  name: "prob"
  type: "Softmax"
  bottom: "fc8"
  top: "prob"
}
```

```
            },
            {
               "Type":
"rightEyeBrowUp",
               "X":
0.5445143580436707,
               "Y":
0.26617032289505005
            },
            {
               "Type": "leftEyeLeft",
               "X":
0.5218595266342163,
               "Y":
0.2779923975467682
            },
            {
               "Type": "leftEyeRight",
               "X":
0.5289638042449951,
               "Y":
0.2778269648551941
            },
            {
               "Type": "leftEyeUp",
               "X":
0.5258546471595764,
               "Y":
0.2762569189071655
            },
            {
               "Type":
"leftEyeDown",
               "X":
0.5257593989372253,
               "Y": 0.
 0.2778297960758209
            },
            {
               "Y": 0.              {
```