



Diseño e Implementación de un Sistema de Alerta Temprana para la Detección de Somnolencia Aplicado en Distintos Entornos Ocupacionales Basado en Inteligencia Artificial y Visión por Computadora

2014119012 - Sebastián Manuel Charris Castrillón

2014219084 - Samuel Andrés Sabogal Peralta (Q.E.P.D)

Universidad del Magdalena

Facultad de Ingeniería

Programa de Ingeniería Electrónica

Santa Marta, Colombia

Año 2022



Diseño e Implementación de un Sistema de Alerta Temprana para la Detección de Somnolencia Aplicado en Distintos Entornos Ocupacionales Basado en Inteligencia Artificial y Visión por Computadora

2014119012 - Sebastián Manuel Charris Castrillón
2014219084 - Samuel Andrés Sabogal Peralta (Q.E.P.D)

Trabajo presentado como requisito parcial para optar al título de:
Ingeniero Electrónico

Director:
ing. Omar Francisco Rodríguez Álvarez
Codirectora:
Ph.D. (c), Aura Margarita Polo Llanos

Línea de Investigación:
Bioingeniería
Grupo de Investigación:
MAGMA Ingeniería

Universidad del Magdalena
Facultad de Ingeniería
Programa de Ingeniería Electrónica
Santa Marta, Colombia
2022

Nota de aceptación:

Aprobado por el Consejo de Programa en cumplimiento de los requisitos exigidos por el Acuerdo Superior N° 11 de 2017 y Acuerdo Académico N° 41 de 2017 para optar al título de Ingeniero Electrónico.

Jurado 1

Jurado 2

Santa Marta, ____ de _____ de 2022.

Dedicatoria

A mi querido amigo y compañero de tesis, pensar en tu inesperada partida me llena de tristeza el alma. Siempre te llevare en mi memoria y mi corazón. Estoy seguro de que todas las personas que te conocimos jamás olvidaremos la gran persona que fuiste. Pediré que me sigas cuidando y guiando desde el cielo.

Descansa en paz, Sammy.

Agradecimientos

Me gustaría agradecer en primer lugar a mis padres Damaris Castrillón y Julio Charris, por todo el esfuerzo, la paciencia y el amor que me han brindado para que yo haya llegado hasta aquí. A mi hermana Andrea Charris que ha sido un ejemplo para seguir y aconsejarme en todo lo posible. A toda mi familia y a mis amigos más cercanos que han sido un apoyo a lo largo de estos años de carrera universitaria.

También quiero agradecer a mi tutora Julie Vilorio Porto, por su constante ayuda y asesoramiento en la elaboración de este trabajo. A todos mis profesores que me han formado en esta etapa y a mis compañeros de carrera que he tenido el placer de encontrármelos en este camino.

Gracias a todos.

Resumen

En este trabajo de investigación se presenta la implementación de un sistema inteligente de visión por computadora para detectar rasgos de somnolencia en una población específica dentro de un contexto de productividad de una organización pública o privada. Este prototipo cuenta con la ventaja de no ser invasivo al cuerpo y estructuralmente está construido con componentes de fácil implementación, como una placa raspberry, una cámara digital y una alarma audible, pero con un gran poder de procesamiento. Durante el desarrollo del dispositivo, se optó por comparar dos métodos para determinar la existencia de la fatiga ocular, el primer método parte de un dataset llamado "yawn_eye_dataset_new" tomado de la base de datos de imágenes de kaggle, este contiene 726 imágenes para la clase 'Ojo Abierto' y 726 imágenes para la clase 'Ojo Cerrado', del cual se dividió el 70% de datos para entrenamiento y el 30% para validación. Mediante estos datos se construyó un modelo de red neuronal convolucional de la mano de la metodología de transferencia de aprendizaje para clasificar el estado del ojo y obtener un índice de somnolencia, para evaluar el rendimiento del modelo entrenado se optó por elaborar un set de prueba que contiene 18 imágenes para cada clase. El segundo método, utiliza el modelo de malla facial para obtener la posición del ojo y a su vez usa la relación de aspecto del ojo (Eye Aspect Rate) para medir la distancia que hay en la apertura del globo ocular y de este modo generar un valor medible de somnolencia. Finalmente, el primer método arroja una exactitud del 69% en la predicción y un error del 31%, un porcentaje muy alto para la predicción de un modelo. En cambio, el segundo método presenta un 76% de exactitud en medición del EAR y un error considerablemente bajo del 24%, en comparación con el método uno. En conclusión, el método dos demuestra una mejora en el nivel exactitud y en la disminución del porcentaje de error con respecto al primer modelo desarrollado, porque el EAR acompañado del modelo de malla facial 3D es una manera eficiente de estimar la existencia de la fatiga ocular.

Palabras claves: Somnolencia, salud ocupacional, redes neuronales convolucionales, transferencia de aprendizaje, visión por computador.

Abstract

This research work presents the implementation of an intelligent computer vision system to detect drowsiness traits in a specific population within a productivity context of a public or private organization. This prototype has the advantage of being non-invasive to the body and structurally it is built with easy to implement components, such as a Raspberry board, a digital camera and an audible alarm, but with a great processing power. During the development of the device, it was chosen to compare two methods to determine the existence of eye fatigue, the first method is taken from a dataset called “yawn_eye_dataset_new” from a Kaggle image database, it contains 726 images for the 'Open Eye' class and 726 images for the 'Closed Eye' class, which divided 70% of the data for training and 30% for validation. Using these data, a convolutional neural network model was built with the transfer learning methodology to classify the state of the eye and obtain an index of drowsiness, to evaluate the performance of the trained model, it was decided to develop a test set containing 18 images for each class. The second method uses the facial mesh model to obtain the position of the eye and uses the aspect ratio of the eye (Eye Aspect Rate) to measure the distance between the opening of the eyeball and thus generate a measurable value of drowsiness. Finally, the first method yields an accuracy of 69% in the prediction and an error of 31%, a very high percentage for the prediction of a model. On the contrary, the second method presents a 76% accuracy in EAR measurement and a considerably low error of 24%, compared to method one. In conclusion, method two demonstrates an improvement in the accuracy level and in the reduction of the error percentage with respect to the first model developed, because the EAR accompanied by the 3D facial mesh model is an efficient way to estimate the existence of eye fatigue.

Keywords: Drowsiness, occupational health, convolutional neural network, transfer learning, computer vision.

Contenido

	Pág.
Lista de figuras.....	X
Lista de tablas	XII
1. Introducción.....	13
1.1 Motivación y justificación	15
1.2 Planteamiento del problema.....	16
1.3 Estado del arte	18
2. Fundamentación teórica	20
2.1 Somnolencia	20
2.2 Inteligencia Artificial.....	20
2.3 Machine Learning.....	21
2.3.1 Aprendizaje supervisado.....	22
2.3.2 Aprendizaje no supervisado.....	22
2.3.3 Aprendizaje por refuerzo.....	22
2.4 Deep Learning.....	23
2.5 Estimación de la posición de los globos oculares	23
2.5.1 Método 1: Viola & Jones	23
2.5.2 Método 2: Malla Facial 3D	30
2.6 Estimación del estado de apertura del globo ocular	39
2.6.1 Método 1: Redes Neuronales Convolucionales	39
2.6.2 Método 2: Ratio del ojo	48
3. Metodología.....	49
3.1 Objetivos	49
3.1.1 Objetivo General.....	49
3.1.2 Objetivos específicos.....	49
3.2 Tecnologías y herramientas	49
3.2.1 Dataset	50

3.2.2	Entorno de programación y librerías	50
3.2.3	Google Colab.....	52
3.2.4	OpenCV.....	52
3.2.5	MediaPipe.....	52
3.3	Desarrollo.....	53
3.3.1	Método 1	53
3.3.2	Método 2	60
4.	Resultados y discusión	65
4.1	Método 1	65
4.2	Método 2	67
5.	Conclusiones.....	68
6.	Trabajos futuros.....	69
7.	Presupuesto	70
8.	Bibliografía	71
ANEXOS.....		74
A.	Algoritmo de entrenamiento de una CNN mediante el modelo VGG16.	74
B.	Algoritmo final para detección de somnolencia mediante MediaPipe.	81

Lista de figuras

	Pág.
Fig. 2.1. Subdivisión del concepto de IA.....	21
Fig. 2.2. Tipos de ML.....	22
Fig. 2.3. Harr feature que se ve similar en la región de los ojos en comparación con la región de las mejillas.....	24
Fig. 2.4. Features de dos, tres y cuatro rectángulos.	25
Fig. 2.5. Suma del área rectangular indicada.	26
Fig. 2.6. Proceso del algoritmo adaptative Boostin.	27
Fig. 2.7. Descripción esquemática del clasificador en cascadas.....	29
Fig. 2.8. Malla facial genérica 3D.	30
Fig. 2.9. Método de extracción de características, descrito en a), b) y c) respetivamente.....	31
Fig. 2.10. Modelo de malla 3D.....	32
Fig. 2.11. Vista 2D de los vértices del modelo 3D alineados con los datos de alcance.....	33
Fig. 2.12. Vista 3D del modelo a los datos de alcance.....	33
Fig. 2.13. Vértices del triángulo con respecto a un punto de coordenadas baricéntricas.	34
Fig. 2.14. Ventana rectangular encerrando los puntos de datos de alcance en 3D.....	35
Fig. 2.15. Ejemplo de segmentación de puntos de datos 3D dentro de un triángulo usando coordenadas baricéntricas.	35
Fig. 2.16. Ejemplo conceptual de un triángulo con puntos de datos 3D en un espacio tridimensional.	36
Fig. 2.17. Deformación de los triángulos del modelo de malla 3D, (a) ajuste y segmentación de la malla de datos 3D en el plano, (b) deformación y cambio de postura del triángulo de la malla 3D.	37
Fig. 2.18. Modelo superpuesto sobre la información 2D y 3D, respetivamente.....	37
Fig. 2.19. Ejemplo visual de la subdivisión de 1 a 4 triángulos del modelo.	38
Fig. 2.20. Resultado final después de aplicar subdivisiones triangulares de modelo deformado.	38
Fig. 2.21. Estructura de una red neuronal convolucional.	40
Fig. 2.22. Proceso de convolución en una imagen de entrada.....	41
Fig. 2.23. Max Pooling.....	42
Fig. 2.24. Arquitectura CNN LeNet.	43
Fig. 2.25. Arquitectura AlexNet.....	44
Fig. 2.26. Arquitectura VGGNet.....	44
Fig. 2.27. Coordenadas de referencias faciales para la fórmula de EAR, a) Ojo abierto y b) Ojo cerrado.....	48
Fig. 3.1. Dataset del estado del ojo, a) Ojo cerrado y b) Ojo abierto.....	50
Fig. 3.2. Diagrama de flujo del método 1.	54
Fig. 3.3. Cuadro delimitador para el RoI, a) Dibujo del cuadro delimitador en la imagen de entrada y b) Recorte del rostro de la imagen de entrada.	56
Fig. 3.4. Recorte de secciones cuadradas de las características y posición de un ojo.	57

Fig. 3.5. Arquitectura del modelo VGG16.	57
Fig. 3.6. Arquitectura de la red VGG16 modificada en las últimas 3 capas.	59
Fig. 3.7. Diagrama de flujo del método 2.	61
Fig. 3.8. 468 puntos de referencia mediante la librería de MediaPipe.....	62
Fig. 3.9. Landmarks que describen la posición de los ojos en la imagen de entrada.	63
Fig. 3.10. Numeración de los puntos de interés en zona del ojo.	63
Fig. 4.1. Matriz de confusión del modelo.	65
Fig. 4.2. Métricas asociadas a la matriz de confusión.	66

Lista de tablas

	Pág.
Tabla 2.1. Ejemplo de cómo está conformada una matriz de confusión.	46
Tabla 3.1. Capas densas y de salida del modelo VGG16 original.	58
Tabla 3.2. Tabla de prueba para medir el número de parpadeos en video.	64
Tabla 4.1. Parpadeos calculados con diferentes valores de EAR.	67
Tabla 7.1. Presupuesto general del proyecto.	70
Tabla 7.2. Aportes realizados por cada entidad participante en el proyecto.	70

1. Introducción

La somnolencia sin limitarse a un tipo de enfermedad en específico puede llegar a ser el principal problema en el comportamiento humano relacionado con los trastornos del sueño, ya sea por el consumo de algunos medicamentos, incluso por la ingesta excesiva de alimentos. En esta investigación no es importante llegar a conocer las causas agravantes para poder identificar los verdaderos motivos por los cuales una persona presenta somnolencia durante el día mientras realiza sus actividades de trabajo, esa es la labor de estudio del personal médico general.

Pero tampoco se puede ignorar el hecho de que la somnolencia constante en un individuo se puede manifestar como resultado, además de la falta de sueño, por el padecimiento de otras enfermedades correlacionadas con la depresión, la ansiedad, el estrés o quizás el aburrimiento. La detección de este fenómeno completamente fisiológico mediante un sistema electrónico inteligente externo no invasivo para la identificación de rostros mediante visión por computadora, puede ser una herramienta muy útil para observar los efectos sobre el ser humano ocasionados por la somnolencia.

Sobre todo, en personas que hacen parte de un centro u organización. La idea es que pueda ser aplicado justo antes de que ocurra algún accidente, precisamente para eliminar o minimizar lo más posible los riesgos en un entorno laboral. Por esta razón, el proyecto es muy atractivo desde el punto de vista del bienestar psicológico, y de la salud y seguridad en el trabajo, ya que se presenta como un mecanismo de prevención novedoso. Las variables involucradas en este estudio serían la fatiga física visible, más que todo en el área de los ojos.

Según diversos trabajos consultados, las personas que sufren de somnolencia tienen en común que experimentan una serie de síntomas, tales como pesadez en el cuerpo, cansancio, poca concentración y hasta confusión, es más, al ser algo constante, esta puede escalar en un dolor crónico difícil de tratar. A largo plazo, el comportamiento errático o movimientos en falso por motivos psicosociales durante la realización de una tarea representa el 17,8% de las causas posibles de un accidente ocupacional [1].

También se entiende que este comportamiento, se presentan con mayor frecuencia cuando se aumenta la carga laboral otorgada por los empleadores, y que no corresponde con las capacidades físicas y mentales de los empleados. Dado lo anterior, no es de extrañar que los movimientos sociales se enfoquen en mejorar las condiciones laborales muy por fuera de la infraestructura física. Bajar las horas de la jornada laboral es la nueva propuesta a nivel mundial, ya que una excesiva carga de actividades repercute de manera negativa en la calidad de vida de los trabajadores.

Como consecuencia, lo anterior puede desencadenarse en enfermedades física y psicológicas, deteriorando el reloj biológico. Por otro lado, se pierden habilidades significativas para el desarrollo de una vida adulta normal con la disminución de los niveles de atención, pérdida de la capacidad de memoria a corto y largo plazo, retraso en el aprendizaje, y por supuesto mala coordinación motora. Todos estos factores tienen relación entre sí con la somnolencia, y como se ha mencionado líneas arriba, son el principal detonante de los niveles altos de estrés.

La preocupación ante todo es que, las consultas a los especialistas en salud por el intenso agotamiento físico y mental se realizan cuando el paciente ha terminado con un diagnóstico del síndrome de Burnout [2], lo cual afecta también su autoestima y personalidad. Todo este fenómeno aparentemente se manifiesta de forma cíclica, en el sentido que, a su vez el agotamiento emocional genera problemas en la calidad del sueño y, por consiguiente, inconvenientes en el desarrollo de las actividades cotidianas.

Esta es una situación bastante desfavorable tanto para el trabajador como para la organización, puesto que como causa afecta la productividad. La problemática ya tratada, se explica más a fondo en la sección de planteamiento del problema. La necesidad de continuar con una investigación de este calibre es expuesta en el estado del arte, pero desde un punto de vista más actualizado, aprovechando las mejoras y las nuevas tecnologías para desarrollar sistemas inteligentes completamente autónomos para la toma de decisiones.

El apartado de fundamentación teórica detalla todos los conceptos que fueron necesarios absorber para el entendimiento de la información relacionada con la detección de somnolencia y con el uso de las herramientas computacionales existentes, además mediante este ejercicio fue en donde se descubrió que, con los años, las personas que se encuentran en la tercera edad

reportan que por medio de la terapia y las horas recomendadas de sueño se podían reducir los síntomas asociados, pero que aun así, siguen padeciendo de somnolencia.

Más adelante, se describen las herramientas tecnológicas utilizadas para la implementación y desarrollo del software y construcción del hardware. Finalmente se exponen los resultados y las conclusiones, junto con un presupuesto aproximado de los gastos generados por la compra de los diversos materiales para efectos de continuar trabajando, si se requiere en esta misma problemática, pero con distinto enfoque, o quizás de manera interdisciplinar con otras áreas del saber.

1.1 Motivación y justificación

El desarrollo de un sistema de detección de somnolencia en entornos laborales de bajo riesgo, como aquellas actividades realizadas en oficinas o escuelas es el trabajo de investigación que se pretende describir en este documento, dado que el principal interés es el de aumentar la productividad, siempre y cuando se asegure el bienestar del personal humano o comunidad estudiantil. La somnolencia o fatiga, desde la perspectiva de la salud ocupacional es la pérdida de concentración en el lugar de trabajo o de estudio, en muchos casos debido a un agotamiento físico y mental, el cual puede ocasionar una leve pérdida de la lucidez y a cometer errores que pueden poner en riesgo la integridad física.

Generalmente, este comportamiento es asociado con la falta de sueño [3], es decir, por insomnio, que en otras palabras advierte que, la falta de descanso físico trae consigo que la persona tome un aspecto desorientado y sin control de sus facultades cognitivas ni motoras, esto por supuesto que, afecta drásticamente la toma de decisiones. La preocupación del personal encargado de la salud ocupacional aumenta conforme se acelera el ritmo de trabajo y la carga de estudio, en estos momentos se cree que es por la reactivación de las actividades humanas después de pasar por una pandemia que cambió por completo el movimiento del mundo a nivel económico, social, educativo y laboral. La idea es que conforme avanza la tecnología, de la misma forma se traten de cambiar los métodos para detectar la somnolencia en pro de la seguridad y la salud en el trabajo.

Si bien es notable que, durante la revisión bibliográfica se encontró que este fenómeno afecta en gran proporción la actividad de conducción de automóviles y significativamente pone en riesgo la vida de los conductores en un tiempo previsible, dado que se estima que del 15% al 30% de los accidentes de tránsito son causados por la falta de sueño [4]. La hipótesis de que la somnolencia puede prolongarse en el tiempo hasta causar un efecto psicosocial con lesiones mentales de leves a graves completamente duraderas en trabajos de oficina proyecta ser un enemigo silencioso que en la inmediatez no parece ser preocupante ante los ojos de la población objeto de estudio. Sin embargo, sí se estima que visiblemente pueden experimentar secuelas a nivel cognitivo-conductual.

Aunque en ningún momento fue descartada la premisa encontrada en la bibliografía, la cual le da validez a los efectos causados por la somnolencia en actividades de alto riesgo, se precisa enfocar esta investigación en entornos de oficina por la afinidad con la población a tratar y por el interés de mejorar las condiciones labores y de estudio en función del bienestar de cada persona, además de abarcar el acondicionamiento de la infraestructura física. Por consiguiente, la solución planteada sí se presenta como un prototipo de mejoramiento en la planta física, el cual funciona como un indicador para establecer pausas activas y de herramienta de aprovechamiento en la gestión del personal que se hace cargo de la salud ocupacional.

El aporte de este estudio es el de identificar patrones faciales asociados con la somnolencia en una población delimitada en distintos entornos ocupacionales, ya sean laborales o académicos, escalable a industrias, hospitales, deportes, conducción, entre otras actividades humanas, con el propósito de integrar nuevas tecnologías desde áreas como la IA y la visión por computadora con la seguridad y la salud en el entorno laboral para la prevención y atención contra accidentes o errores que puedan afectar el normal desarrollo de las tareas a realizar, mejorando así la calidad de vida y aumentando el rendimiento y la productividad.

1.2 Planteamiento del problema

La somnolencia como el acto repentino de quedarse dormido y de manifestar visiblemente un cansancio físico, debido a la privatización voluntaria o involuntaria del sueño es un tema que ha tomado una gran importancia en el área de la salud laboral, dado que si se presenta de manera constante puede llegar a generar problemas de seguridad. Tal es el caso que, este trastorno se

encuentra fuertemente documentado más que todo desde la puesta en marcha del primer medio de transporte masivo, debido al inmenso riesgo de accidentes con consecuencias fatales en el mundo por la actividad de conducir, puesto que la tarea de manejar requiere de un alto nivel de atención visual y motor.

Otras ocupaciones no tan riesgosas, pero que necesitan de toda la concentración posible, como el trabajo en oficina para tareas poco repetitivas o las horas de dedicación al estudio, también pueden verse seriamente afectadas por la presencia regular de la somnolencia, la cual en otras investigaciones es tomada como sinónimo de fatiga mental. La diferencia es que se puede manifestar a largo plazo, dejando secuelas psicosociales con una recuperación muy lenta y posiblemente deteriorando el proceso de aprendizaje del individuo quien la pueda padecer, incluso puede llegar a ser el detonante de otras patologías mucho más complejas y graves a nivel cognitivo.

En este mismo contexto, el enfoque sobre la falta de un periodo sano de sueño, además de un plan de acción de concientización, se propone un producto como medida preventiva, de acompañamiento y de caso de estudio para el personal de los departamentos de psicología, salud y seguridad en el entorno laboral. La idea es abordar y monitorear de una forma no invasiva, el comportamiento, en primera instancia de la población que integra la Universidad de Magdalena, aprovechando la reactivación en las actividades administrativas, académicas, de extensión y proyección social, teniendo en cuenta la gran presión por continuar con la dinámica de funcionamiento que se tenía antes de la sufrida pandemia global.

El afán de mejorar las condiciones de bienestar y privacidad hoy en día es más grande que nunca, y se entiende que para poder acceder a la información personal en entidades tanto públicas como privadas se requiere de la concesión de ciertos permisos y de la firma de actas de consentimiento informado. Esto supone la gestión de una base de datos que, si dado el caso no es posible adquirirla de la misma comunidad académica y administrativa objeto de estudio, como alternativa se prevé el uso de datos externos que modelen las características más relevantes que se puedan observar durante el proceso de desarrollo metodológico, lo que a su vez genera el desarrollo de actividades de pruebas experimentales dentro del laboratorio y no en un ambiente real de trabajo.

1.3 Estado del arte

Durante la fase de revisión bibliográfica fueron encontradas varias implementaciones con diferentes métodos que involucraban tecnologías como la Inteligencia Artificial (IA) y la visión por computadora para resolver una problemática en concreto. Algunos de estos documentos cuentan con información académica detallada y fueron tomados como antecedentes, los cuales proporcionaron información bastante útil para esta investigación. Las aplicaciones más estudiadas fueron las siguientes:

- Uso de visión artificial.
- Implementación de redes neuronales convolucionales.

Algunos de los trabajos mencionados se limitan a la solución de problemas similares a los que se presentaron en el transcurso del desarrollo de este proyecto.

Sistema avanzado de asistencia a la conducción para la detección de la somnolencia (Marco J. Florensa, Jose M. Armingol, Arturo de la Escalera): En este artículo se presenta un sistema avanzado de asistencia a la conducción (SAAC) diseñado para la de detección de somnolencia y distracción del conductor. Los principales objetivos fueron localizar el rostro y los ojos del conductor para analizarlos a través del tiempo y generar un índice de somnolencia en el conductor. El sistema fue diseñado utilizando técnicas de visión por computadora e IA [5].

Control del cursor de un computador mediante Eye-tracking (Daniel Alejandro Rodríguez Morillo): Este documento propone la realización de un programa que permita facilitar la interacción de los equipos de cómputo con personas que tengan parálisis cerebral. Para el desarrollo se entrenó y empleó una red neuronal convolucional con el fin de predecir la posición de enfoque visual en la pantalla de un computador para contratar el curso del ratón solamente con la mirada del usuario [6].

Desarrollo de un sistema inteligente de detección de fatiga en conductores (Pedro Javier García Paterna): El objetivo de este trabajo consistió en el desarrollo de un Sistema ADAS (Sistema Avanzado de Asistencia a la Conducción) orientado a la detección de fatiga en conductores, con el fin de mejorar la seguridad del automovilista y minimizar el riesgo de un accidente de tráfico. En el proyecto, uno de los métodos usados es utilizar técnicas no invasivas

con el fin de obtener distintos parámetros, como la apertura de ojos o un bostezo del conductor, a través del procesamiento de imágenes obtenidas en tiempo real desde una única cámara enfocada hacia el rostro de este [7].

Diseño e implementación de un brazo robótico para jugar ajedrez utilizando un sistema de visión artificial (Cristian Miguel Del Toro Rondón): En este documento fue implementado un brazo autómatas capaz de jugar ajedrez con una persona. La estructura mecánica consiste en un brazo robótico con una cámara acoplada a una pinza, además el programa instalado procesa las imágenes de la cámara para evaluar las configuraciones del tablero y así estimar la posición de las fichas controlando los posibles movimientos de las piezas de ajedrez [8].

2. Fundamentación teórica

En este capítulo se detalla toda la información teórica relacionada con la actualidad del tema de investigación, la cual fue recopilada no sólo para la elaboración de este proyecto, sino que juega un papel muy importante en la apropiación del conocimiento. Entre los conceptos adquiridos se destacan términos tales como, IA, aprendizaje automático (AA) o Machine Learning (ML), aprendizaje profundo o Deep Learning (DL) y Malla Facial 3D.

2.1 Somnolencia

La somnolencia es estudiada en el campo de la medicina y la salud ocupacional como un estado fisiológico que invade todo el cuerpo humano, regularmente por la falta de un descanso adecuado durante las horas destinadas para el sueño, o después de consumir grandes cantidades de alimentos, y que suele manifestarse como un decaimiento narcoleptico o cansancio físico [9].

Por consiguiente, el consenso internacional acerca de lo que se define como somnolencia, en muchas ocasiones es tratada como una clase de fatiga asociada a un trastorno del sueño, generalmente ocasionado por el insomnio [10]. Una definición operacional es la tendencia a quedarse dormido en momentos inoportunos durante el día bajo la habilidad de pasar de la vigilia a la del sueño [11].

2.2 Inteligencia Artificial

La Inteligencia Artificial o por sus siglas IA puede llegar a ser un término muy complejo difícil de explicar. Podemos encontrar diferentes definiciones de esta, pero la más acertada es “la habilidad de dotar a la máquina de conocimiento para la toma de decisiones tal y como lo haría el ser humano”. Sin embargo, a diferencia de las personas, los dispositivos basados en IA no necesitan descansar para seguir analizando información o cumpliendo una labor específica.

La IA se puede aplicar en casi todas las situaciones, estas son algunas de las aplicaciones técnicas de esta tecnología: reconocimiento, clasificación y etiquetado de imágenes, detección y clasificación de objetos, procesamiento eficiente y escalable de datos, entre otros.

En consecuencia, la IA también será capaz de ofrecernos sugerencias y predicciones en asuntos importantes en la vida cotidiana, lo que genera un mayor impacto en la salud, el bienestar, la educación, el trabajo y hasta las relaciones interpersonales [12]. El desarrollo de la IA ha llegado a tal punto que se subdividen en dos ramas importantes, como se muestra en la Fig. 2.1.

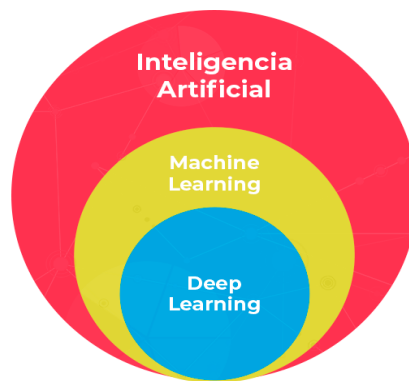


Fig. 2.1. Subdivisión del concepto de IA.

2.3 Machine Learning

El AA o aprendizaje de máquina (en inglés, Machine Learning) es una de las ramas principales de la IA. Este se centra en el uso de algoritmos para analizar, aprender e identificar patrones de datos [12]. En la Fig. 2.2 se muestran los tres subconjuntos en los que se divide el ML: aprendizaje supervisado, no supervisado y por refuerzo.



Fig. 2.2. Tipos de ML.

Tomado de: [12].

2.3.1 Aprendizaje supervisado

En esta subrama el aprendizaje supervisado usa datos que han sido etiquetados y organizados previamente para que el algoritmo pueda categorizar la nueva información. Este método requiere de la intervención humana para proporcionar retroalimentación [12].

2.3.2 Aprendizaje no supervisado

En el aprendizaje no supervisado los datos usados no han sido organizados ni etiquetados previamente, el algoritmo tendrá que buscar la forma de aprender a reconocer y categorizar la nueva información [12].

2.3.3 Aprendizaje por refuerzo

Por último, en el aprendizaje por refuerzo se tienden a aprender por “recompensa”, es decir, el algoritmo recibe un estímulo positivo cada vez que acierta en el resultado de una clasificación [12].

2.4 Deep Learning

El DL es un subconjunto del ML, el cual como se pudo observar en las anteriores imágenes es un subconjunto de la IA. El término de DL aparece en diversos campos de estudio de las ciencias y al parecer es posible aplicarlo en cualquier problemática que requiera ser automatizada. Sin embargo, los expertos opinan que su aplicación varía en la profundidad del análisis y del tipo de información que se necesita automatizar, por ejemplo, es ideal para complementar paradigmas diferentes (tipos de técnicas de aprendizaje que imitan el funcionamiento del cerebro humano), arquitecturas flexibles (ajuste de los hiperparámetros de acuerdo con las necesidades) y definición de características autónomas (no requiere de la intervención humana). En pocas palabras la estructura computacional del DL es la de una red neuronal con muchas neuronas en su arquitectura [13].

2.5 Estimación de la posición de los globos oculares

La presente sección puntualiza los métodos elegidos para la extracción de la posición del rostro y los ojos en una imagen o en un video.

2.5.1 Método 1: Viola & Jones

Detección de la posición del rostro y los ojos, mediante el algoritmo de Viola & Jones

El algoritmo de Viola y Jones (V&J) es un método muy eficaz en la detección de objetos y se destaca por su bajo coste computacional, lo cual permite ser empleado en tiempo real. Puede aplicarse en la identificación de otra clase de objetos como la detección de rostros u otros rasgos faciales (ojos, nariz, boca, etc.).

Viola y Jones se basan en una sucesión de clasificadores débiles llamados Haar-Like Features que pueden ser calculados de manera eficiente a partir de una imagen integral. Dichos

clasificadores tienen una pequeña posibilidad de acertar en su detección, se agrupan en grupos denominados cascadas, empleando un algoritmo de aprendizaje basado en AdaBoost para aumentar el rendimiento y la probabilidad de acierto en la detección.

3.3.1.1. Haar-Like features

Los Harr-Like features (o características de tipo Harr) son elementos básicos usados en el área del reconocimiento de objetos, reciben su nombre por la similitud a la wavelet de Harr [14]. Paul Viola y Michelle Jones consideraron regiones rectangulares adyacentes en una ubicación específica de una ventana de detección para una imagen, sumar las intensidades lumínicas de cada píxel de la región y calcular la diferencia entre esas sumas. Por lo tanto, este resultado se utiliza para categorizar secciones de una imagen.

Una observación referente al rostro humano es acerca de la poca iluminación que tienen algunas regiones con respecto a otras, por ejemplo: la sección de los ojos es más oscura en comparación con las zonas de mejillas, o la región del puente de la nariz es más brillante que la zona de los ojos. De modo que, una característica de tipo Harr para la detección del rostro se basa en un conjunto de dos rectángulos adyacentes que se encuentran sobre dichas regiones respectivamente, y la posición de estos rectángulos define la relación mediante una ventana que actúa como un cuadro delimitador para el objeto destino, el rostro en este caso [15].



Fig. 2.3. Harr feature que se ve similar en la región de los ojos en comparación con la región de las mejillas.

En la literatura presentada por V&J, proponen tres tipos de características tipo Haar básicas que fácilmente se pueden apreciar en la siguiente Fig. 2.4.

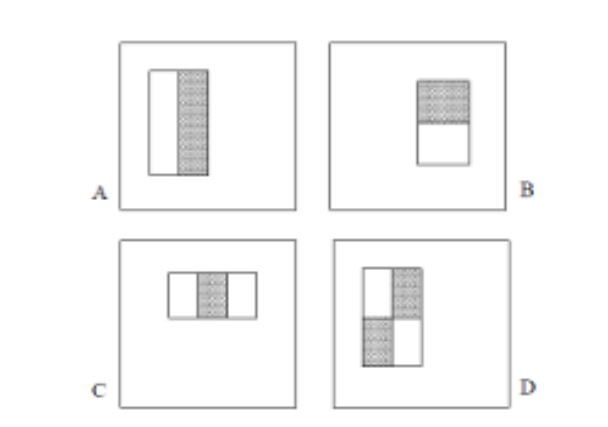


Fig. 2.4. Features de dos, tres y cuatro rectángulos.

Tomado de: [14], [16].

- Para A y B, existen dos rectángulos cuyo valor es la diferencia entre las sumas de los píxeles contenidos en la zona de ambos rectángulos.
- En el caso de C, el feature corresponde a tres rectángulos cuyo valor se calcula con la diferencia de los rectángulos exteriores y el interior, multiplicando el resultado por un valor para compensar las diferencias de áreas.
- En última posición tenemos a D, es un feature de 4 posiciones que calcula la diferencia entre pares diagonales del rectángulo.

Por último, estos mismos autores anexan una ventana estándar de búsqueda en 24x24 píxeles por feature, lo que representa un valor de 180000 features posibles por región seleccionada [14].

3.3.1.2. Imagen integral

Para calcular de manera eficiente la suma de los píxeles de un rectángulo, se emplea una representación denominada imagen integral. Esta imagen integral es una sección de la imagen original, la cual posee coordenadas (x, y) y además contiene la suma de todos los píxeles que se encuentran ubicados en la parte superior y hacia la izquierda desde ese punto [16].

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y') \quad (1)$$

Donde $ii(x, y)$ es la imagen integral y $i(x, y)$ es la imagen original. La imagen integral se puede calcular en un solo barrido por la imagen, empleando un par de sentencias recurrentes:

$$s(x, y) = s(x, y - 1) + i(x, y) \quad (2)$$

$$ii(x, y) = ii(x - 1, y) + s(x, y) \quad (3)$$

Donde $s(x, y)$ representa la suma acumulada de la fila x con $s(x, -1) = 0$ y $ii(-1, y) = 0$.

Además, con la imagen integral podemos calcular la suma de áreas rectangulares de una imagen en cualquier lugar y posición mediante 4 referencias de búsqueda como en la siguiente figura:

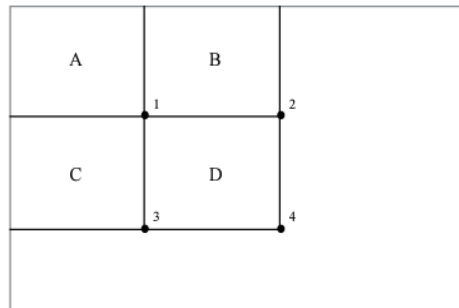


Fig. 2.5. Suma del área rectangular indicada.

Tomado de: [14].

Cada característica del tipo Harr puede necesitar más de 4 búsquedas, dependiendo de cuántos rectángulos fueron definidos con anterioridad. Las características de 2 rectángulos de V&J necesitan 6 búsquedas, las de 3 rectángulos necesitan 8 búsquedas y las de 4 rectángulos requieren de 9 búsquedas [14].

3.3.1.3. Entrenamiento mediante el algoritmo de Adaboost

Para crear las cascadas clasificadoras se necesita realizar un entrenamiento supervisado. Este proceso es realizado a través de un meta algoritmo adaptativo de ML con el nombre de Adaptive Boosting o para abreviar AdaBoost.

El boosting toma una serie de clasificadores débiles para mezclarlos y construir un clasificador robusto con buena precisión. Adaboost fue introducido por Freund y Shapire en 1995 [14], con lo cual lograron resolver varios problemas del proceso de boosting.

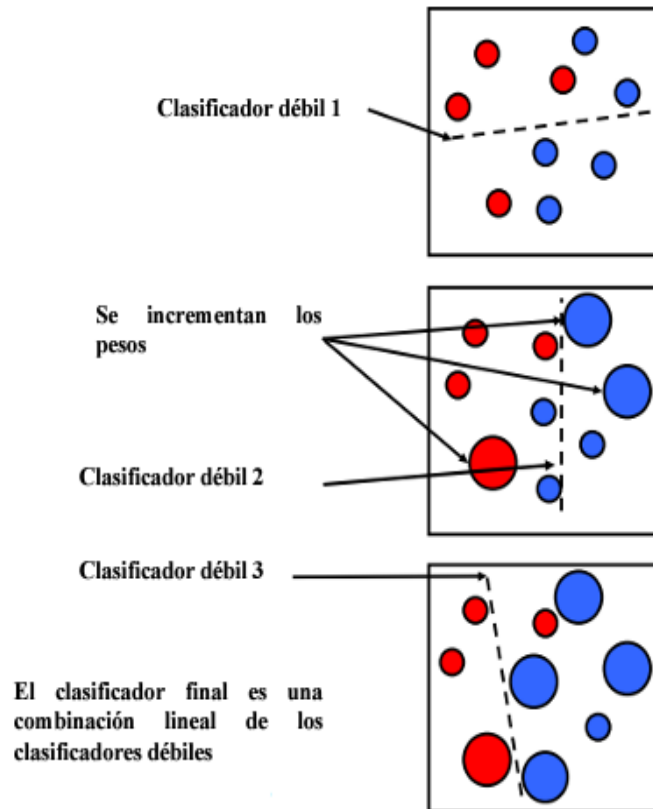


Fig. 2.6. Proceso del algoritmo adaptative Boostin.

Tomado de: [14].

El método de V&J utiliza AdaBoost para seleccionar un pequeño set de features de 180000 posibles para entrenar un clasificador débil $h_j(x)$. Para cada feature f_j el clasificador débil determina un valor umbral θ_j que minimiza los features mal clasificados y un coeficiente p_j que indica el signo de desigualdad [14].

$$h_j(x) = \begin{cases} 1 & \text{si } p_j f_j(x) < p_j \theta_j \\ 0 & \text{e. o. c} \end{cases} \quad (4)$$

A continuación, se describe el funcionamiento del algoritmo AdaBost empleado [16]:

- Se toma un conjunto de imágenes $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ donde $y_i = 0, 1$ para imágenes negativas y positivas respectivamente.
- Se inicializan los pesos $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ para $y_i = 0, 1$ respectivamente, donde m es el número de negativos y l el número de positivos.
- Para cada vuelta, $t = 1, \dots, T$:

1. Normalizar los pesos:

$$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^n w_{t,j}} \quad (5)$$

2. Para cada característica j , entrenar un clasificador h_j que solo use una característica. El error se evalúa con respecto a los pesos:

$$w_{t,j}, \epsilon_j = \sum_i w_i |h_j(x_i) - y_i| \quad (6)$$

3. Se selecciona el clasificador h_t con menor error ϵ_t .

4. Se actualizan los pesos:

$$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i} \quad (7)$$

Donde $e_i = 0$ si el ejemplo x_i se clasifica correctamente y 1 en caso contrario para $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.

5. El clasificador fuerte en su forma final:

$$h(x) = \begin{cases} 1 & \text{si } \sum_{t=1}^T \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^T \alpha_t \\ 0 & \text{e. o. c} \end{cases} \quad (8)$$

Donde $\alpha_t = \frac{1}{\beta_t}$.

3.3.1.4. Clasificadores en cascada

Para que el algoritmo de V&J detectaste de manera más eficiente, los autores trabajaron en un método para combinar varios clasificadores fuertes en forma sucesivas llamadas cascada. Este procedimiento incrementa la velocidad de detección solo centrando su atención en las ventanas más prometedoras de una imagen. La idea es determinar velozmente donde puede aparecer un objeto (el rostro para nuestro ejemplo) y disminuir la probabilidad de no detección por cada etapa.

El clasificador en cascada entrenado por AdaBoost propuesto por V&J contiene 48 etapas y más de 6000 feautres. Por reducción de coste de procesamiento se evalúan 10 features por ventana de búsqueda [14].

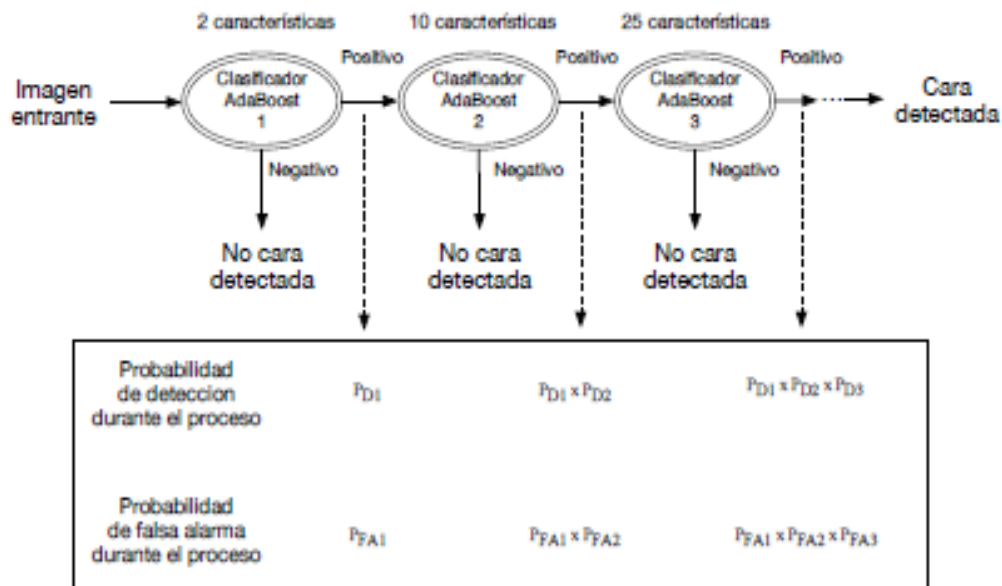


Fig. 2.7. Descripción esquemática del clasificador en cascadas.

Tomado de: [15].

En el proceso de entrenamiento, las imágenes usadas por el algoritmo fueron normalizadas para reducir los efectos de las diferentes condiciones de iluminación. Normalizar la imagen implica cambiar los valores de todos los píxeles y facilitar la corrección del valor de los features a medida que se van calculando.

La varianza es la forma de normalizar dichas imágenes:

$$\sigma^2 = m^2 - \frac{1}{N} \sum x^2 \quad (9)$$

Donde m es la media del valor de los píxeles, que pueden calcularse partiendo de la imagen integral. La suma de los píxeles al cuadrado se puede obtener apartando de la imagen integral el cuadrado [14].

Puesto a que, el detector en cascadas es sensible a pequeños movimientos y diferentes escalas, se pueden producir múltiples detecciones alrededor de rostro. Como solución se exige que las detecciones tengan un determinado número de detecciones vecinas, fusionar esas detecciones acompañado de un valor umbral y el recuadro resultante se calcula con la media de las detecciones anteriores [14].

2.5.2 Método 2: Malla Facial 3D

Detección de la posición del rostro y los ojos, mediante el algoritmo de Malla Facial 3D

En las últimas décadas se han publicado una gran cantidad de algoritmos basado en imágenes 2D (como es el caso del algoritmo de V&J). Debido a las numerosas limitaciones del enfoque 2D surgen algoritmos basados en imágenes de rango 3D que proporcionan más información sobre la superficie de un rostro, en consecuencia, trabajar con este tipo de datos resulta potencialmente ventajoso para un sistema de reconocimiento facial.

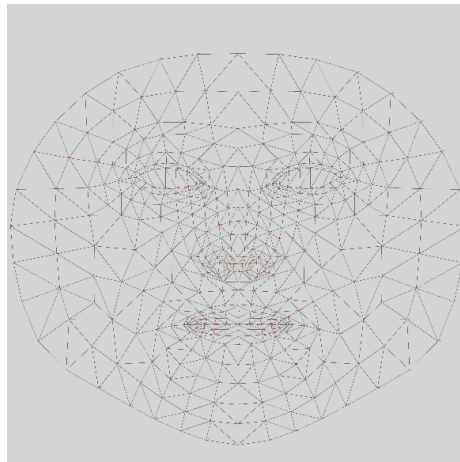


Fig. 2.8. Malla facial genérica 3D.

Un rostro bajo un modelo de malla facial presenta un tejido triangular, que es como una red de pesca que cubre la superficie del rostro y describe con mayor precisión las características geométricas de un individuo.

Para extraer y procesar los datos del modelo de malla facial, el algoritmo presenta diferentes etapas:

2.5.2.1 Preprocesamiento de datos y extracción de rasgos faciales

En el paso de preprocesamiento se aplica un filtro mediano para eliminar picos agudos y el ruido, esto se producen durante el escaneo de un rostro en una imagen, seguido de una interpolación para rellenar huecos y luego con un filtro pasa bajo para suavizar la superficie final. Para la localización del rostro y otras características faciales, se aplica una plantilla de la malla facial (mostrada en la Fig. 2.8), para descartar zonas, tales como: el fondo de la imagen, el cuello y el cabello de un individuo [17].

Después de seccionado el rostro, se procede a detectar en la imagen la ubicación de la punta de la nariz y luego en la plantilla facial situarse sobre la punta de la misma nariz. Finalmente, el área debajo de dicha plantilla con la máxima correlación se considera la región facial detectada. Además, se utiliza una curvatura gaussiana para extraer las 2 esquinas interiores de los ojos y la punta de la nariz.

La superficie detectada para las esquinas de los ojos presenta una forma de fosa y de punta para la zona de la nariz, estos puntos de curvatura gaussiana tienen valores positivos ($K > 0$) entre los puntos de la superficie del rostro [17]. El resultado del cálculo de curvatura gaussiana se muestra en la siguiente figura:

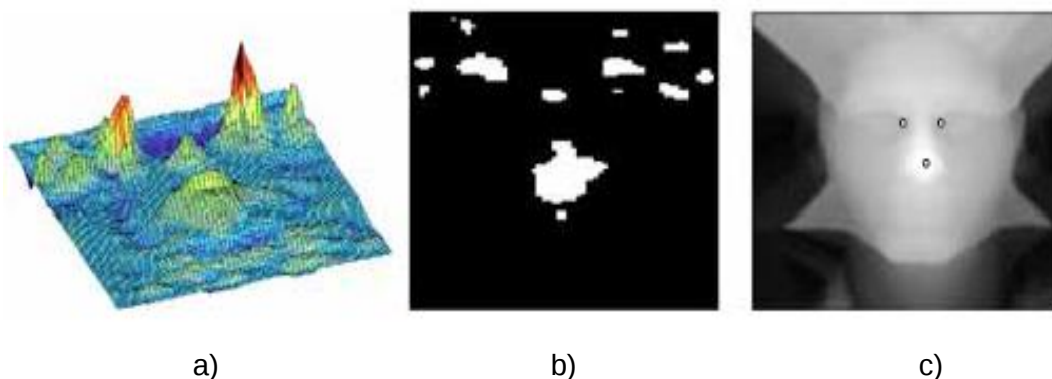


Fig. 2.9. Método de extracción de características, descrito en a), b) y c) respectivamente.

Tomado de: [17].

La Fig. 2.9.a proporciona los puntos positivos más altos produciendo una imagen binaria que describe la forma de fosa y/o pico. Para Fig. 2.9.b el umbral se calcula partir de un conjunto de datos de entrenamiento de diferentes imágenes de rostros utilizados en el experimento. Finalmente, en la Fig. 2.9.c las 3 regiones con mayor valor medio de curvatura gaussiana son las regiones candidatas a ser etiquetados como puntos característicos.

2.5.2.2 Alineación por modelado facial 3D

La idea es alinear un rostro utilizando características 3D, el algoritmo procede a nivelar los triángulos alineados de la malla a los datos de rango de la imagen, mediante un ajuste de planos por cuadrados mínimos. Los triángulos alineados del modelo se subdividen en triángulos de mayor resolución, el algoritmo se asegura de obtener una superficie más realista y suave, semejante a la superficie del rostro antes de aplicar otro ajuste de planos. En la Fig. 2.10 se muestra el modelo neutro del modelo 3D con un total de 109 vértices de características etiquetadas y 188 mallas poligonales definidas [17].

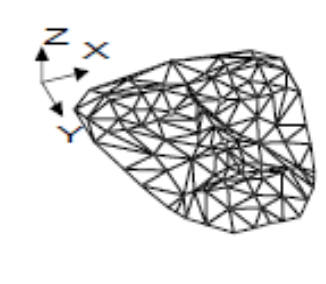


Fig. 2.10. Modelo de malla 3D.

Tomado de: [17].

Además, el modelo está diseñado para que los lados derecho e izquierdo de la mandíbula estén dentro de los parámetros de la malla, pero no sus bordes. Este encuadre evita incorporar datos inexactos e innecesarios a los bordes de la cara de las imágenes de rango capturadas.

2.5.2.3 Alineación global

En la alineación global el modelo es ubicado rígidamente mediante tres puntos de características 3D.

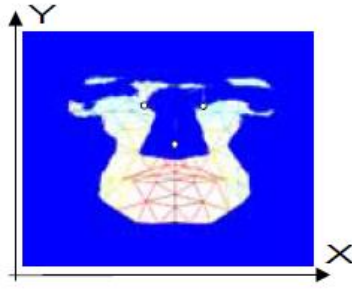


Fig. 2.11. Vista 2D de los vértices del modelo 3D alineados con los datos de alcance.

Tomado de: [17].

El primer punto P_I es obtenido a través del rango de la imagen y sus correspondientes vértices de características P_M en el modelo. Los subíndices I y M indican respectivamente las características que posee la imagen y los vértices del modelo. Para lograr este objetivo el modelo es rotado, trasladado y escalado. La ecuación $\text{Min } E(S, R, T)$ es la suma cuadrada entre P_I y P_M en términos S , rotación R y traslación T para $n = 3$ puntos [17].

$$\text{Min } E(S, R, T) = \sum_{j=1}^n \|P_{I_j} - P_{M_j}\|^2 \quad (10)$$

En la siguiente figura, los triángulos del modelo están encerrados total o parcialmente por encima o por debajo de los datos 3D, debido a la segmentación de los puntos mediante el alineado 3D.

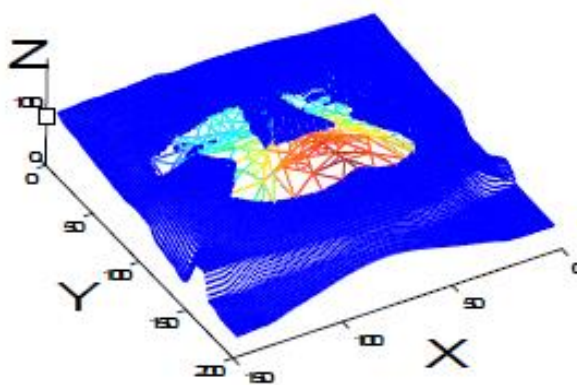


Fig. 2.12. Vista 3D del modelo a los datos de alcance.

Tomado de: [17].

2.5.2.4 Segmentación de puntos faciales en 3D

El paso previo a la deformación del modelo es segmentar y extraer los puntos de datos orientados en 3D (por encima, por debajo o por dentro) de cada triángulo de la malla usando una técnica de gráficos por ordenador de nombre coordenadas baricéntricas [17]. Una coordenada baricéntrica es un arreglo de tres puntos de vértices P_1 , P_2 y P_3 que forman un plano triangular, como se muestra en la siguiente figura:

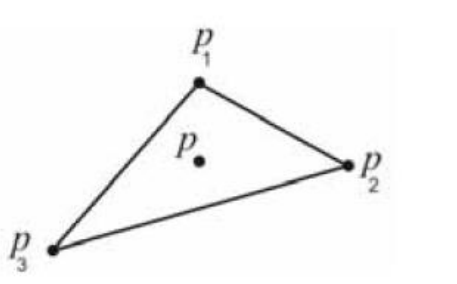


Fig. 2.13. Vértices del triángulo con respecto a un punto de coordenadas baricéntricas.

Tomado de: [17].

El punto P dentro del triángulo, se define como el centro de masa o baricentro de los pesos de los vértices que lo conforman y están definidas por la siguiente ecuación:

$$P = uP_1 + vP_2 + wP_3, \text{ donde } u + v + w = 1 \quad (11)$$

De este modo, P está dentro del triángulo y $[u, v, w]$ son coordenadas baricéntricas de P con respecto a P_1 , P_2 y P_3 respectivamente. De forma equivalente se modifica la ecuación de la siguiente manera:

$$P = uP_1 + vP_2 + (1 - u - v)P_3 \quad (12)$$

La ecuación anterior, representa 3 ecuaciones y de este modo se pueden formar un sistema lineal mediante las incógnitas $[u, v, w]$ y está dada por:

$$\begin{bmatrix} P_1 & P_2 & P_3 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} = P \quad (13)$$

Los puntos u , v y w dentro del triángulo tienen valores positivos. Por otro lado, los puntos por fuera del triángulo tienen por lo menos una coordenada negativa. La ecuación anterior tiene un coste computación bastante alto debido a que uno de los 188 triángulos del modelo de malla 3D tiene que comprobar todas las coordenadas de los datos de rango, para determinar si las coordenadas sí se encuentran dentro su interior [17].

Una implementación practica para aplicar la ecuación anterior consiste en crear una ventana con los datos, encerrando las coordenadas del triángulo, como se aprecia en la figura:

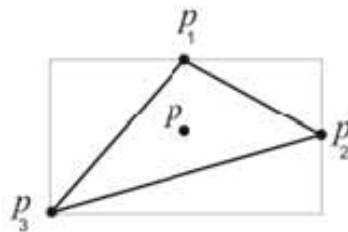


Fig. 2.14. Ventana rectangular encerrando los puntos de datos de alcance en 3D.

Tomado de: [17].

En la Fig. 2.15 se observa una vista 2D del modelo de malla 3D real superpuesto a los puntos de datos de alcance. La imagen muestra un ejemplo de puntos de datos 3D segmentados dentro de un triángulo de la malla ubicado en parte superior de malla. No obstante, la figura muestra como ajustar y deformar los triángulos del modelo que sean lo más cercano de los datos 3D recopilados.

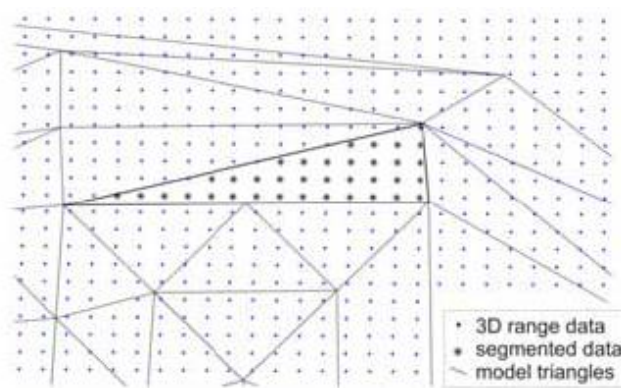


Fig. 2.15. Ejemplo de segmentación de puntos de datos 3D dentro de un triángulo usando coordenadas baricéntricas.

Tomado de: [17].

2.5.2.5 Deformaciones del modelo facial 3D

Después de segmentar la nube de puntos de datos 3D mediante las coordenadas baricéntricas, se representa a través de un plano utilizando el método de ajustes por cuadrados mínimos. La ecuación que representa el plano con un vector normal N en tercera dimensión es:

$$aX + bY + vZ + d = 0, \text{ donde } N = (a, b, c) \quad (14)$$

Para un numero n de puntos, la ecuación puede escribirse en forma de mínimos cuadrados como:

$$\begin{bmatrix} X_1 & Y_1 & Z_1 & 1 \\ X_2 & Y_2 & Z_2 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ X_n & Y_n & Z_n & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = AB = 0 \quad (15)$$

Las coordenadas (X_i, Y_i, Z_i) representan todos los puntos de datos segmentados por la coordenada baricéntrica. La ecuación anterior puede resolverse para los parámetros de la ecuación del plano $B = [a, b, c, d]$, sustituyendo luego en la ecuación n , dando lugar a un plano que representa a todos los puntos de datos 3D.

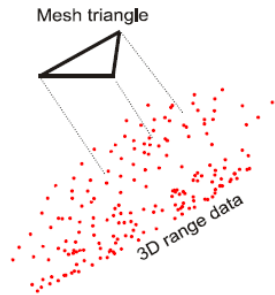


Fig. 2.16. Ejemplo conceptual de un triángulo con puntos de datos 3D en un espacio tridimensional.

Tomado de: [17].

En la Fig. 2.17.a se muestran los datos segmentados que se representan mediante un plano usando la ecuación de representación de un plano. Con la información matemática desprendida de la geometría del plano, teniendo en cuenta los parámetros B , cualquier punto del plano puede

ser evaluado. El algoritmo al deformar cada triángulo de la malla correspondiente a los puntos de datos 3D, puede descartar las coordenadas Z de los tres vértices del triángulo, evaluar las coordenadas restantes (X y Y) y resolver los parámetros B , que a su vez se transforman en la nueva coordenada Z [17].

Esta transformación genera un triángulo de malla con nuevas coordenadas de profundidad en el plano calculado, siendo así el más aproximado a los datos densos en 3D de la imagen. La Fig. 2.17.b muestra el concepto de deformación del triángulo de la malla al plano que representa los datos y la postura del triángulo se cambia para que coincida con la del plano.

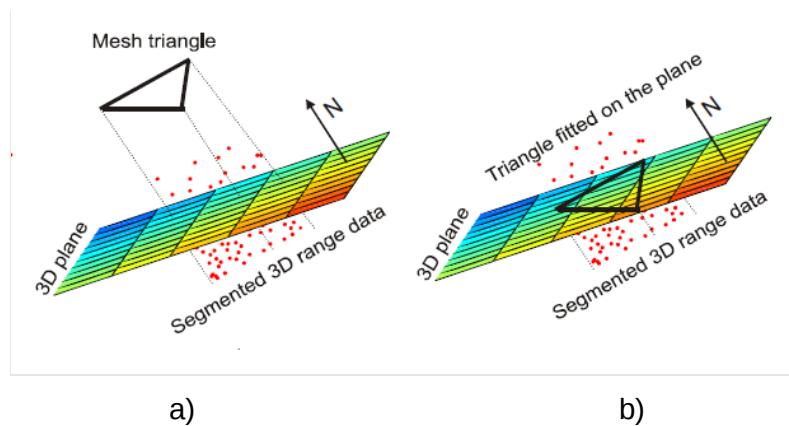


Fig. 2.17. Deformación de los triángulos del modelo de malla 3D, (a) ajuste y segmentación de la malla de datos 3D en el plano, (b) deformación y cambio de postura del triángulo de la malla 3D.

Posteriormente, el modelo de deformación se aplica a cada triángulo de la malla 3D. En la Fig. 2.18 se observa un ejemplo del modelo de deformado superpuesto a los datos de visuales de un rostro en 2D y 3D respectivamente.

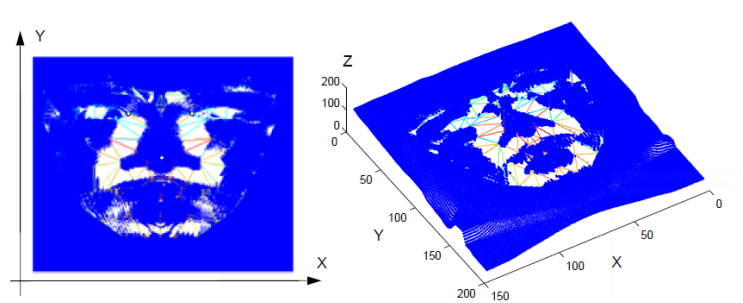


Fig. 2.18. Modelo superpuesto sobre la información 2D y 3D, respetivamente.

El modelo de deformado realiza una buena representación de los datos, pero no es suficientemente suave para trabajar datos de alta resolución y curvaturas de datos 3D. La solución a esta problemática planteada por los autores fue subdividir los triángulos del modelo para alcanzar una mayor escala de resolución [17].

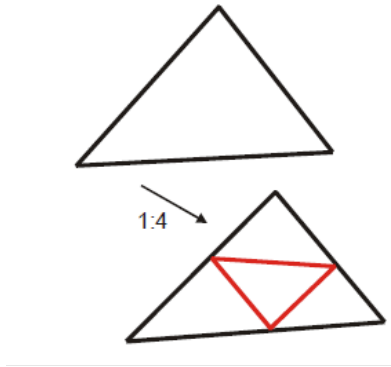


Fig. 2.19. Ejemplo visual de la subdivisión de 1 a 4 triángulos del modelo.

Tomado de: [17].

Con la aparición de nuevos triángulos, estos generan nuevos vértices, estas incógnitas se calculan basándose en las ubicaciones de los vértices deformados de los triángulo anteriores. La Fig. 2.20 se muestra el resultado de la subdivisión del modelo deformado de la Fig. 2.19. Este procedimiento incrementa el número de vértices y triángulos (malla) del modelo original de 109 y 188, respectivamente, a 401 vértices y 752 mallas poligonales. Por último, los nuevos triángulos no reflejan la deformación real de los datos, se vuelven a deformar usando el mismo proceso de deformación [17]. La aparición de triángulos más pequeños proporciona más eficacia en las zonas de alta curvatura o relieve.

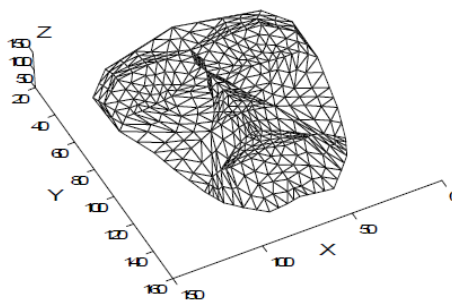


Fig. 2.20. Resultado final después de aplicar subdivisiones triangulares de modelo deformado.

Tomado de: [17].

2.6 Estimación del estado de apertura del globo ocular

La presente sección puntualiza los métodos seleccionados para estimar el estado de apertura de los ojos y de este modo generar un indicador de somnolencia.

2.6.1 Método 1: Redes Neuronales Convolucionales

Redes Neuronales Convolucionales para el entrenamiento del clasificador del estado del ojo

Cuando se trabaja con imágenes es momento de pasar a métodos más robustos como los son la Redes Neuronales Convolucionales (Convolutional Neural Networks o CNN por sus siglas en inglés).

Las CNN son un tipo algoritmo de DL, que fueron diseñadas para el procesamiento de datos de dos dimensiones como entrada, se usan en gran medida para clasificar cualquier tipo de imagen de tamaño de $m \times m \times r$, donde m es tanto el ancho como la altura, y r es el número de canales de la imagen. Por lo tanto, son especialmente útiles en problemas de visión por computadora y para detectar o categorizar objetos [18].

Las CNN toman las imágenes como entradas o inputs, asignándole un tipo de importancia (pesos) a ciertos elementos para diferenciar unos píxeles de otros. Este algoritmo contiene diferentes capas o layers, donde las capas pueden entrenarse para extraer características como los bordes, colores, líneas o curvas y así especializarlas hasta poder detectar formas más complejas como el rostro humano, siluetas, entre otras.

Las CNN realizan dos procesos en forma paralela para facilitar su entrenamiento. La fase número uno extrae características de la imagen y la siguiente fase, toma esos datos de características para buscar la relación con el objeto y encontrar su correcta clasificación [18].

Estas redes pueden ser entrenadas desde cero, esto requiere de una gran cantidad de imágenes el cual aprenderá de eso datos previamente etiquetados. Este tipo de entrenamiento suele tener un alto coste de cómputo y requiere una gran cantidad de tiempo. No obstante, existe el proceso de transferencia de aprendizaje que reduce en gran medida el tiempo y el coste del entrenamiento de una CNN [18].

En la Fig. 2.21 se observan los 3 componentes de una CNN: Capa de convolución, capa de reducción o Pooling y la capa totalmente conectada o full Connection.

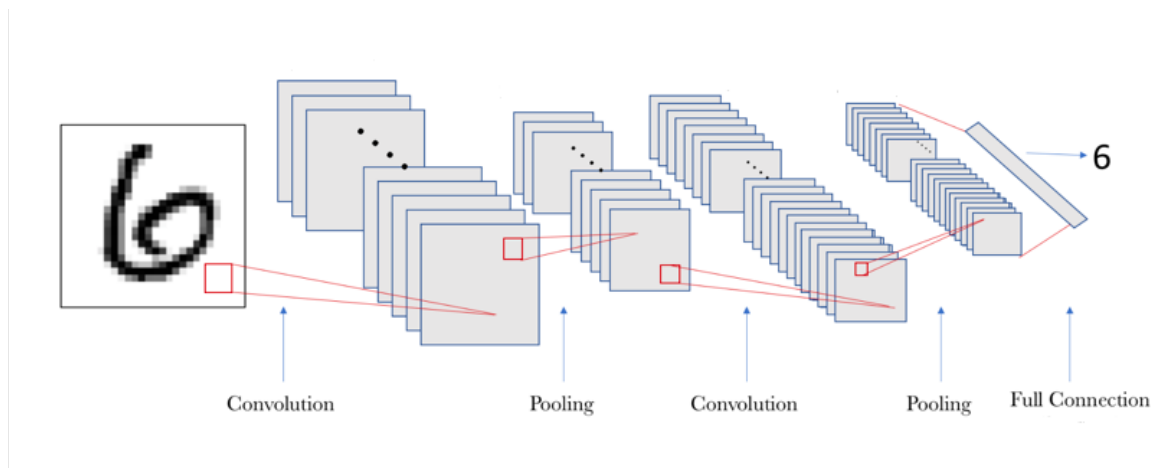


Fig. 2.21. Estructura de una red neuronal convolucional.

2.6.1.1 Capa de convolución

La convolución es una operación de productos y sumas de una imagen de entrada, finalmente tratada como una matriz de diferentes dimensiones, la cual contiene números de píxeles por número de píxeles por número de canales de color. Si dicha imagen es a color, necesitamos 3 canales para representar las variaciones de intensidades de Red, Green y Blue (RGB) [18].

Para realizar la convolución es necesario aplicar un filtro o también llamado kernel, en el caso de que una imagen tenga los 3 canales de colores, el kernel debe tener 3 componentes del mismo tamaño de píxeles para obtener una imagen de salida.

A grandes rasgos, el proceso de convolución consiste en desplazar el kernel de $n \times n$ dimensiones a lo largo de una imagen, de izquierda a derecha, como de arriba a abajo y la longitud de este

desplazamiento es conocida como salto o stride. Por cada uno de los elementos de la imagen original, se realiza un producto escalar entre la matriz de la imagen y el filtro para obtener un mapa de características (de dimensiones similares a la entrada) [18].

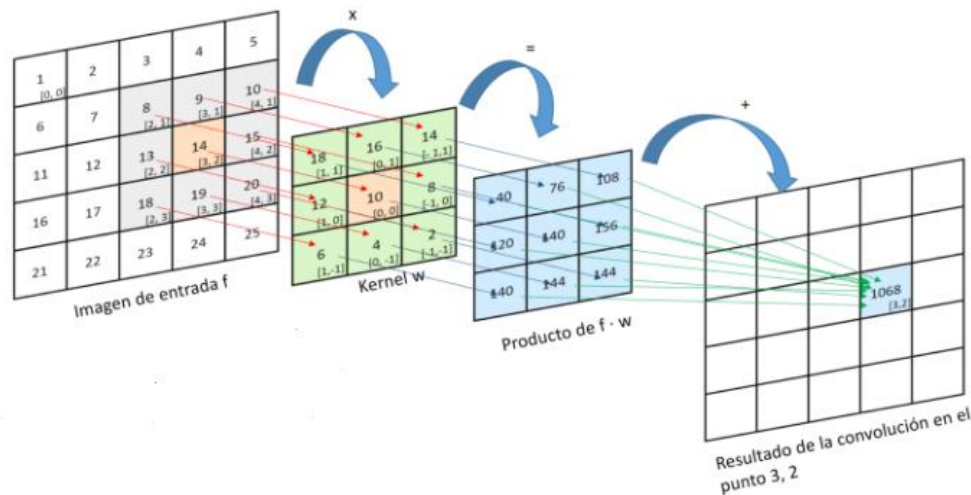


Fig. 2.22. Proceso de convolución en una imagen de entrada.

Un problema muy recurrente en la convolución es la reducción de dimensiones de la imagen de entrada, es decir, si una imagen pasa por diferentes capas de convolución, el tamaño se reduce a tal grado que puede ser demasiado pequeña y no tener utilidad. Por otro lado, la operación de relleno o padding consiste en agregar un marco y así ampliar el área de la imagen, esta operación permite un análisis más preciso de los bordes de la imagen y evita en gran medida la reducción del tamaño de la imagen, si no se usa el relleno puede provocar pérdida de información en la imagen original [18].

Las capas convolucionales son el punto más interesante en el reconocimiento de imágenes porque al tener un filtro fijo que se desplaza por la imagen hace que puedan detectar patrones, independientemente de la posición en la que se encuentre dentro de la imagen de entrada.

2.6.1.2 Capa de Pooling

La capa de Pooling o reducción se aplica entre dos capas de convolución y consiste en disminuir las dimensiones espaciales del mapa de características previamente obtenido por la convolución.

El objetivo es preservar las características más esenciales de una imagen, la forma más común es usar el proceso de Max Pooling [18].

En el Max Pooling se define un tamaño de ventana $n \times n$ dimensiones que va recorriendo el mapa de características, se calcula un producto escalar para obtener el valor medio de cada ventana y así lograr reducir el tamaño de los datos al tamaño de la venta de muestra.

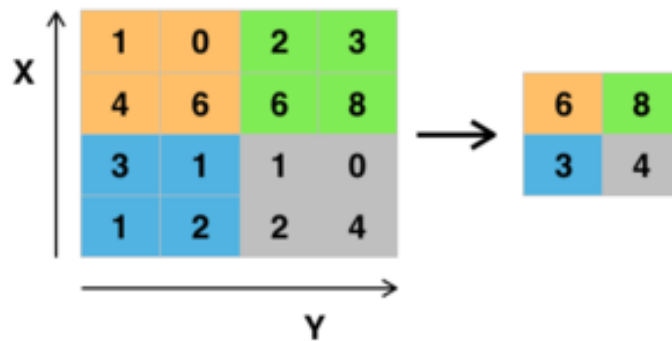


Fig. 2.23. Max Pooling.

En la Fig. 2.23 se observa un ejemplo del proceso. Un mapa características de 16 datos, después de pasarlo por un filtro de Max Pooling de 2×2 proporcionando como resultado que los datos se reduzcan de 16 a 4 datos [18].

2.6.1.3 Capa Full Connection

Esta capa es la última del esquema de las CNN y se trata de un clasificador que determina la clase a la que pertenece la imagen de entrada, generalmente se compone por un número de neuronas iguales al número de clases que se desea detectar en el modelo [19].

Las CNN se entrenan con el algoritmo de backpropagation, este calcula el error que se ha cometido a lo largo de las operaciones correspondientes (convolución, Pooling, ReLU entre otras), toda esa información regresa hacia atrás para pasar nuevamente por las neuronas e ir calculando el error por cada una de ellas. De manera que, al volver al inicio, se pueden calcular y reajustar nuevos pesos para cada neurona y que el error vaya disminuyendo. Si el error no se reduce y se acumula, la red entrenada se vuelve inestable y deja de aprender.

2.6.1.4 Arquitecturas de las CNN

Dentro de la literatura de CNN existen arquitecturas diseñadas para reducir la tasa de error que pueden presentar a lo largo de su desarrollo, las relevantes son las siguientes:

- **LeNet:** Este modelo de arquitectura fue usado para el desarrollo de un problema de reconocimiento de caracteres escritos a mano en una imagen. Fue una de las primeras CNN publicadas por su alto desempeño en tareas de visión por computadora, este modelo fue presentado por el entonces investigador Yann LeCun de AT&T Bell Labs [20].

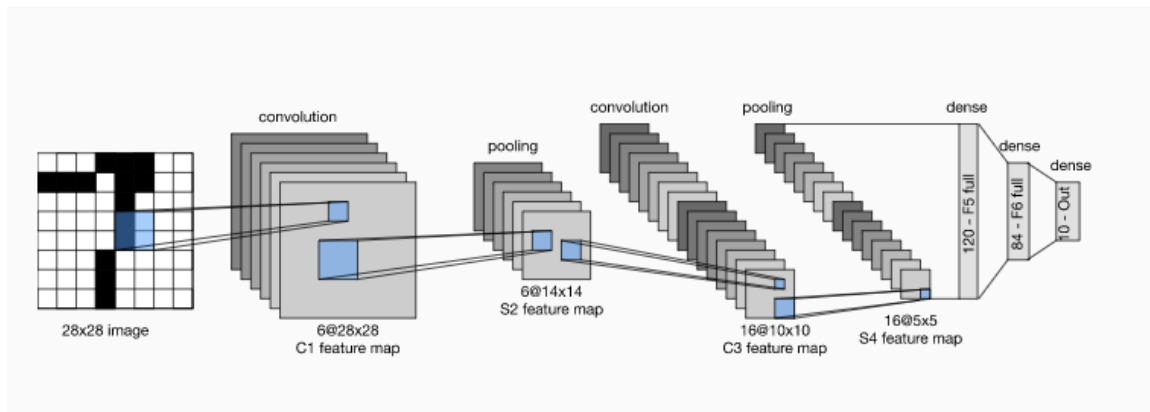


Fig. 2.24. Arquitectura CNN LeNet.

Tomado de: [21].

- **AlexNet:** Fue un modelo de red neuronal convolucional creado por Alex Krizhevsky que tiene como objetivo clasificar fotogramas de objetos. Esta arquitectura fue la primera ganadora del concurso ILSVRC, la cual consiguió bajar tasa de error al 17%, en comparación con otras arquitecturas como LeNet. Entre sus descubrimientos más importantes, se encuentra el uso de la función de activación ReLu en su entrenamiento, así aceleró el proceso 6 veces más rápido sin perder la precisión. Esta arquitectura marcó un hito en las tareas de la visión artificial, debido a que las tareas de clasificación de imágenes mejoraron sus resultados, llevando así a la creación de CNN más complejas y profundas. A pesar de ser una arquitectura antigua se sigue usando por su disminución en los tiempos de entrenamiento [22].

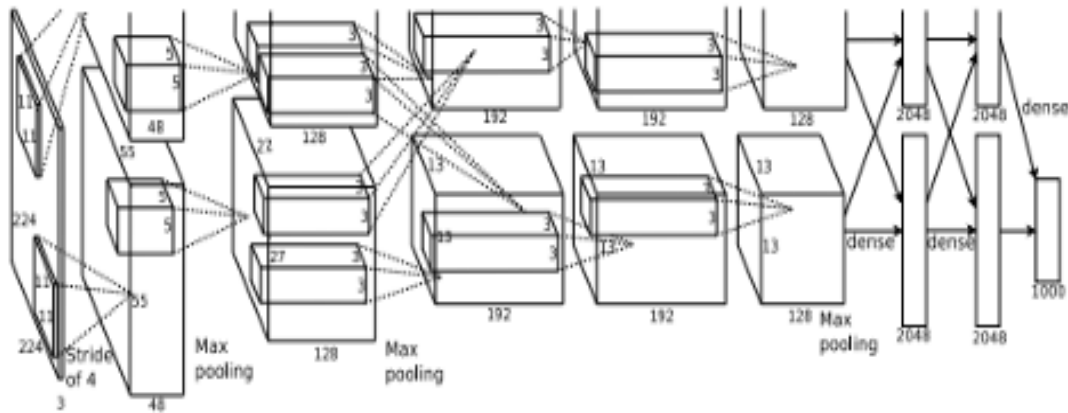


Fig. 2.25. Arquitectura AlexNet.

Tomado de: [22].

- VGGNet:** Es una arquitectura influyente desde su aparición, este reforzó la construcción intuitiva detrás de las CNN para que una red logre una estructuración jerarquía de datos lo suficientemente buena debe contener una arquitectura de capas profundas parecidas a su antecesora AlexNet. Esta red destacó por su reducción de tasa de error del 7.2% y a su simplicidad. Está formada por 19 capas de convolución, Kernel del tamaño de 3x3 con saltos y pad de 1, y las capas de Max Pooling de tamaños 2x2 con stride de 1. En la Fig. 2.26 se aprecia la forma de arquitectura de VGGNet [22].

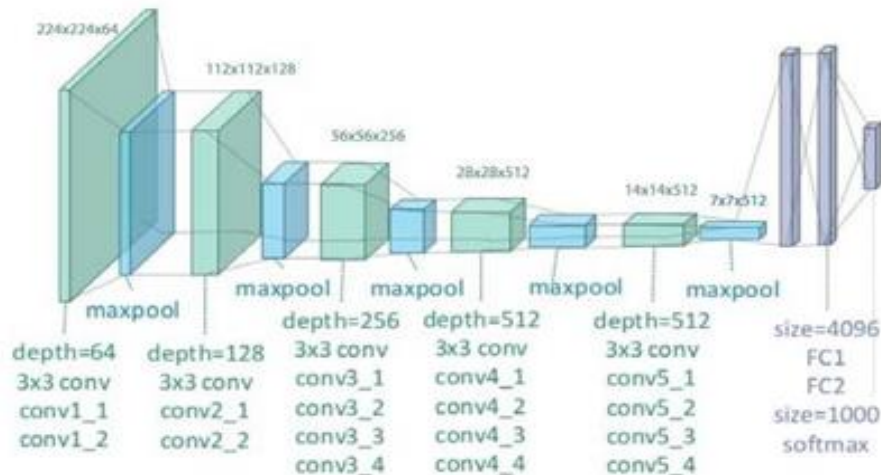


Fig. 2.26. Arquitectura VGGNet.

Tomado de: [22].

2.6.1.5 Modelo de entrenamiento por arquitectura de Transfer Learning

La tecnología de transferencia de aprendizaje (Transfer Learning, TL) es un área dentro del DL en la que un modelo o una arquitectura se utiliza como punto de partida para entrenar un nuevo modelo para realizar una tarea similar, requiriendo menos cantidad y calidad de datos, también mejora el rendimiento final del nuevo modelo entrenado. Este tipo de metodología se implementa frecuentemente en aplicaciones de detección de objetos, reconocimiento de imágenes, reconocimiento de voz y más [23].

Existen diferentes estrategias y técnicas de TL que pueden aplicarse en función del tamaño de los datos, la precisión del modelo y la velocidad de predicción. Se debe tener cuidado al elegir el modelo pre entrenado que se va a usar en cada caso, debido a que, si su problema es muy diferente, la predicción que se obtiene será inexacta [23]. Existen muchas arquitecturas pre entrenadas, entre las más populares tenemos: AlexNet, VGG16, VGG19, GoogLeNet, SqueezeNet, MobileNet-v2, ShuffleNet, entre muchas otras.

2.6.1.6 Métricas de evaluación

Un paso esencial en todo proyecto que involucre redes neuronales es el de evaluar el desempeño de predicción seleccionada. Las métricas de evaluación se utilizan primordialmente para evaluar el ajuste entre la salida del modelo y los datos [24]. Para las tareas de medición se encuentran las siguientes métricas:

- **Matriz de confusión**

Esta métrica se usa para evaluar el rendimiento completo de un clasificador en su salida, puede ir de 2 o más clases. La matriz crea una “tabla” donde almacena los tipos de errores que se están cometiendo en el modelo, básicamente, compara las predicciones obtenidas con los datos reales. Además, tiene dimensiones de $N \times N$, donde N es el número de clases a clasificar.

La matriz se compone de: verdaderos negativos (True Negative, TN), verdaderos positivos (True Positive, TP), falsos negativos (False Negative, FN) y falsos positivos (False Positive, FP). A partir de este modelo de evaluación se pueden calcular o construir otras métricas como: Exactitud (Accuracy), Precisión (Precision), Sensibilidad (Recall) y F1- Score.

Tabla 2.1. Ejemplo de cómo está conformada una matriz de confusión.

		Actual	
		Positivo	Negativo
Predicción	Positivo	Verdadero Positivo (TP)	Falso Positivo (FP)
	Negativo	Falso Negativo (FN)	Verdadero Negativo (TN)

Tomado de: [25].

- **Exactitud (Accuracy)**

Mide el desempeño del modelo. Indica el número de elementos clasificados correctamente en comparación con el número total de elementos. Se calcula mediante la siguiente formula:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (16)$$

Esta métrica presenta una limitación, ya que no tiene un buen desempeño para evaluar conjuntos desequilibrado [24].

- **Precisión (Precision)**

Este representa la proporción de verdaderos positivos que son correctamente clasificados en comparación con el número total de valores positivos que el modelo predijo. La precisión se obtiene como:

$$Precision = \frac{TP}{TP + FP} \quad (17)$$

Presenta una limitación en comparación con la anterior métrica, ambos son susceptibles a errores de clasificación con conjuntos de datos desequilibrados [24].

- **Sensibilidad (Recall)**

Indica la tasa de clasificación positiva correcta de un modelo, es decir, la proporción de positivos que el modelo ha calificado como correcto en comparación de número total de muestras positivas. Esta métrica se obtiene de la siguiente manera:

$$Recall = \frac{TP}{TP + FN} \quad (18)$$

Este modelo de evaluación tiene como ventaja que no es susceptible a presentar errores con conjuntos de datos desequilibrados [24].

- **F1-Score**

Consiste en una amalgama entre las métricas de precisión y sensibilidad. Esta métrica funciona como una forma de balance entre ellas, y como resultado, estas métricas solo reciben valores altos, si la precisión y la sensibilidad presentan valores altos. El valor F1 se obtiene mediante la siguiente fórmula:

$$F1 - Score = 2 * \frac{Presicion * Recall}{Presicion + Recall} \quad (19)$$

Al ser independiente de la precisión tiende a tener una alta sensibilidad con conjuntos desequilibrados entre las clases por predecir [24].

2.6.1.7 Error Cuadrático Medio (MSE)

Esta métrica calcula el valor medio de la diferencia al cuadrado del valor real (y) y el predicho (y') para todos los datos. Todos los valores presentes se elevan a la segunda potencia, por lo tanto, los valores que son negativos no se compensan con los positivos [24]. El MSE entre menor sea, más precisa serán las predicciones. La forma de calcularlo es mediante la siguiente fórmula:

$$MSE = \frac{1}{2} \sum_{i=1}^n (y'^{(i)} - y^{(i)})^2 \quad (20)$$

2.6.2 Método 2: Ratio del ojo

Ratio del aspecto del ojo para clasificar el nivel de apertura del ojo

La relación en el aspecto del ojo (Eye Aspect Ratio, EAR) es una propuesta de la Universidad Técnica Checa que responde al nivel de apertura y cierre de los ojos convirtiéndose en una cantidad escalar [26]. Pandey y Muppalaneni implementaron un sistema de detección de la somnolencia para evitar accidentes, basado en la duración del parpadeo en ambos ojos y su sistema de trabajo ha demostrado una buena precisión.

El EAR calcula sobre un rostro detectado en una imagen, la distancia entre las coordenadas oculares verticales y como denominador la distancia que hay entre las coordenadas oculares horizontales, donde los punto $P_1, P_2 \dots P_6$ son las seis coordenadas oculares. La organización de estos valores se observar en la Fig. 2.27, además estos puntos se remplazan directamente en la siguiente fórmula para obtener un valor escalar [27]:

$$EAR = \frac{\|P_2 - P_6\| + \|P_3 - P_5\|}{2\|P_1 - P_4\|} \quad (21)$$

Teniendo presente el resultado de investigaciones anteriores, un umbral EAR recomendado va de 0,2 hasta 0,3. La fórmula de EAR es insensible a la dirección y la distancia de la cara, por lo que ofrece la ventaja de identificar caras a distintas [26].

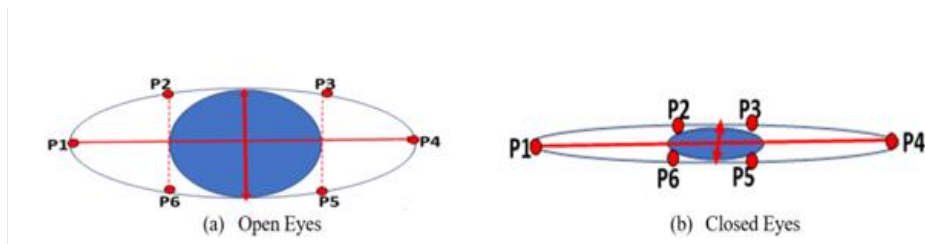


Fig. 2.27. Coordenadas de referencias faciales para la fórmula de EAR, a) Ojo abierto y b) Ojo cerrado.

Tomado de: [26].

3. Metodología

3.1 Objetivos

3.1.1 Objetivo General

Implementar un prototipo electrónico para la detección de somnolencia basado en visión por computadora.

3.1.2 Objetivos específicos

- Aplicar un método de extracción de características para la detección de la apertura ocular.
- Aplicar un modelo de clasificación de aprendizaje de máquina para la detección de ojos abiertos y ojos cerrados.
- Obtener un indicador de estado de somnolencia a partir del nivel de apertura y estado del ojo para generar una alerta.

3.2 Tecnologías y herramientas

En esta sección se presentan las herramientas tecnológicas que se han empleado a lo largo del desarrollo de este proyecto de investigación.

3.2.1 Dataset

En el desarrollo del proyecto se ha usado Kaggle, esta base de datos contiene más de 50.000 conjuntos de datos públicos. Dentro de Kaggle encontramos un dataset llamado “yawn_eye_dataset_new”, contiene 1452 imágenes del estado de apertura del ojo. Este set de datos nos ayudará en el entrenamiento de la red para detectar somnolencia, mediante el uso de dos estados de clasificación: ojo abierto y ojo cerrado [28].

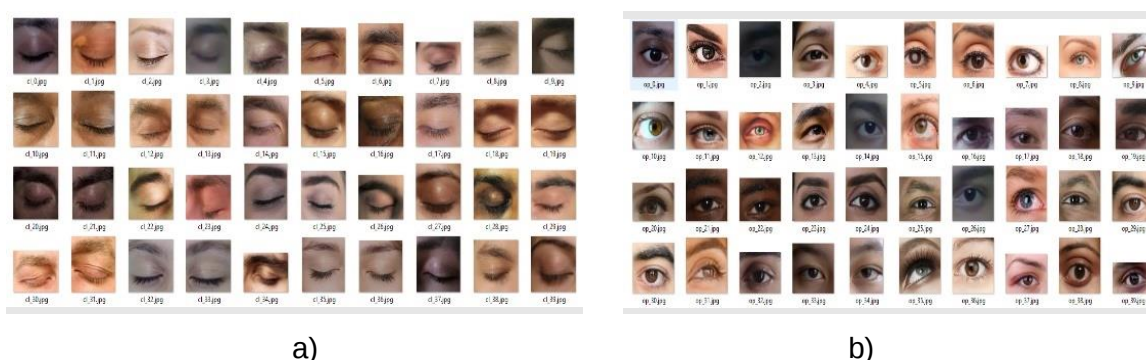


Fig. 3.1. Dataset del estado del ojo, a) Ojo cerrado y b) Ojo abierto.

3.2.2 Entorno de programación y librerías

El lenguaje de programación seleccionado ha sido Python, es un lenguaje de programación de alto nivel más utilizado a nivel mundial debido a su rápida escritura, escalable y de código abierto. Ventajas que lo hacen un aliado perfecto para el desarrollo en el campo del AA y la visión artificial.

Python cuenta con una amplia variedad de librerías desarrolladas para este tipo de tecnologías. Algunas de las empleadas en este proyecto son:

3.2.2.1 TensorFlow

Es una biblioteca de código abierto creada por Google especializada en el AA. Esta cuenta con un ecosistema integral y flexible de herramientas que sirven para la creación y entrenamiento de modelos de IA también cuenta con algoritmos de búsqueda, análisis y procesamiento de imágenes y muchas aplicaciones [29].

Algunas alternativas a TensorFlow son PyTorch, Keras, Theano, Caffee, entre otras.

3.2.2.2 Keras

Es una API de DL escrita en Python, que se ejecuta sobre la plataforma de TensorFlow. Esta librería fue concebida bajo la idea de minimizar la cantidad de acciones que debe realizar un usuario para crear modelos de aprendizaje automático.

Keras cuenta con una estructura simple, flexible y potente para proporcionar rendimiento y escalabilidad a la resolución de problemas modernos de AA [30].

3.2.2.3 Numpy

Es una librería de Python usada para el cálculo numérico y análisis de datos, proporciona potentes estructuras de datos implementando vectores y matrices multidimensionales [31].

3.2.2.4 PIL

Python Imaging Library o biblioteca de imágenes de Python: Esta librería proporciona un acceso rápido a los datos almacenados en pocos píxeles de información, además de herramientas sólidas para el procesamiento de imágenes [32].

3.2.2.5 OS

Es una librería de Python que provee funcionalidades dependientes de un sistema operativo. Su principal uso es el manejo de archivos (creación, lectura y sobreescritura) y directorios de alto y bajo nivel [33].

3.2.2.6 Pandas

Es una librería de Python que proporciona herramientas de manipulación y análisis de datos de manera flexible en cualquier tipo de lenguaje [34].

3.2.3 Google Colab

Para el desarrollo del proyecto se ha utilizado el entorno de programación Google Colab. Este entorno gratuito desarrollado por Google permite desarrollar código de Python de manera dinámica, aprovecha toda la potencia de las bibliotecas más populares en ciencia de datos y AA, no requiere de configuración previa y además cuenta con acceso a GPUs basada en la nube que reducen el proceso de cálculo y entrenamiento de redes neuronales artificiales [35].

3.2.4 OpenCV

Es una biblioteca de código abierto especializada en visión por computadora, AA y el procesamiento de imágenes y videos. La principal característica de OpenCV es identificar patrones en las imágenes, usando espacios vectoriales para realizar operaciones matemáticas y así poder extraer diferente información o características de la entrada.

Esta librería juega un papel muy importante debido a su alta compatibilidad con diferentes librerías y a su gran adaptabilidad en aplicaciones en tiempo real [36].

3.2.5 MediaPipe

Es un marco para crear canalizaciones de ML, procesar datos de series temporales como imágenes, videos, audio, etc. Media Pipe ofrece soluciones de ML personalizables y multiplataformas de código abierto.

Entre las soluciones que ofrece tenemos: segmentación de selfies, malla facial 3D, seguimiento de imágenes, detección y seguimiento de poses, detección de rostros, detección y seguimiento de objetos, entre muchos otros [37].

3.3 Desarrollo

Para el desarrollo del proyecto, se contrastaron dos métodos para detectar la somnolencia, comparando sus resultados y seleccionando el más adecuado para generar un indicador de somnolencia y a su vez construir un prototipo que integre esta funcionalidad.

A nivel de hardware, la estructura general del prototipo consta de un módulo de cámara Raspberry de 5Mpx conectada directamente a una placa Raspberry Pi 3 Modelo B+ y un adaptador de carga de 5V a 3A. Además, cuenta con una conexión SSH a un dispositivo de cómputo Lenovo G40-70 con procesador Intel Core i7-4510U a 2GHz y 6 GB. Por último, el software de control estará a cargo de uno de los métodos que se describirán a continuación:

3.3.1 Método 1

Para detectar la somnolencia, se implementó el algoritmo de V&J para segmentar la posición del rostro y ambos ojos en conjunto de una red neuronal convolucional para clasificar el estado de los ojos. En la siguiente Fig. 3.2 se muestra la lógica que toma el algoritmo diseñado para detectar la somnolencia en un usuario.

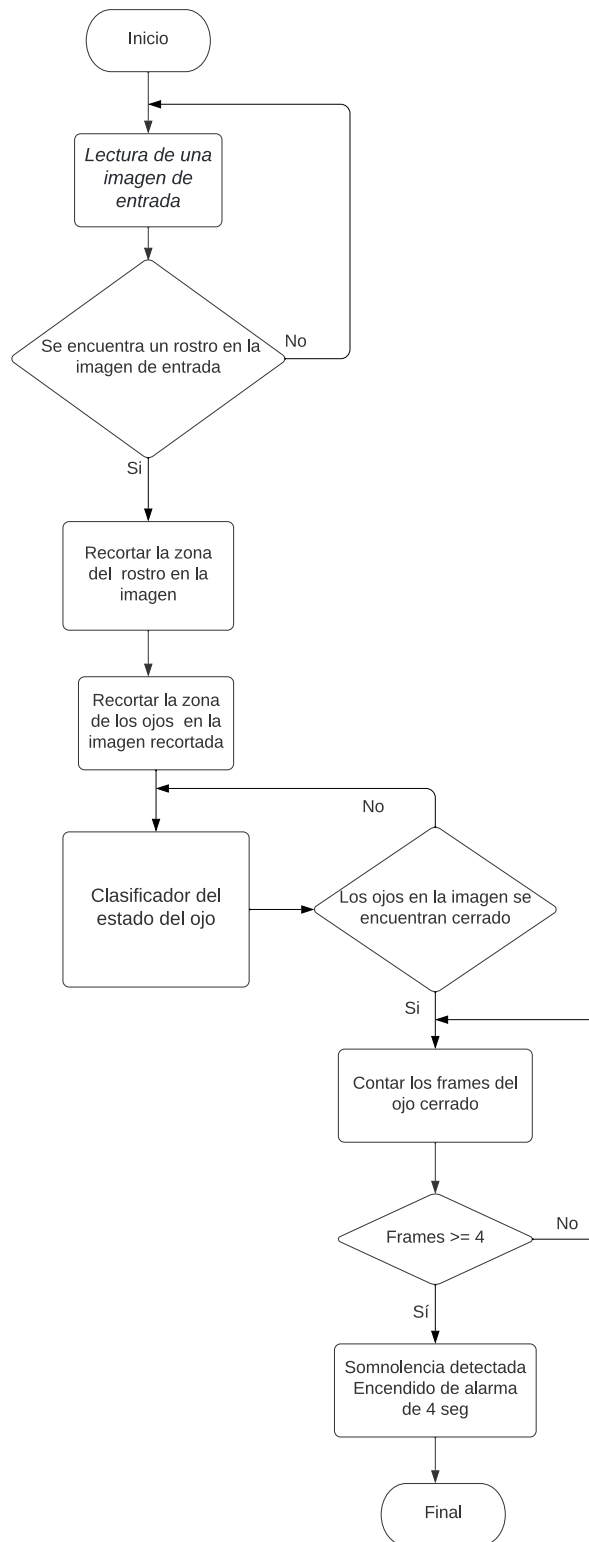


Fig. 3.2. Diagrama de flujo del método 1.

El método presente puede resumirse en 4 funciones principales:

1. Captura y preprocesamiento de la imagen de entrada.
2. Detección y recorte de la posición del rostro detectado en la imagen de entrada.
3. Detección y recorte de la posición de los ojos en la imagen del rostro recortado.
4. Modelo de CNN para la clasificación del estado de ambos ojos.

3.3.1.1 Captura y preprocesamiento de la imagen de entrada

Para la toma de imágenes, la cámara integrada al equipo de cómputo Lenovo G40-70 soporta la toma de imágenes con una resolución de 720p (1280 x 720 píxeles). El preprocesamiento se hizo a través de OpenCv. Esta librería tiene alojada la función `cv2.cvtColor()`, este método cambia el espacio de color de la imagen de entrada a escalas grises necesaria para que el algoritmo de V&J trabaje correctamente. Otra función importante es `cv2.resize()`, esta modifica el tamaño de la imagen original para reducir el número de píxeles a procesar por el detector.

3.3.1.2 Detección y recorte de la posición del rostro detectado en la imagen de entrada

En la literatura de detección de objetos, el algoritmo de V&J ofrece una serie de soluciones basadas en las cascadas clasificadoras de tipo Harr. Estos archivos .xml son entrenados para detectar un objeto, se puede considerar el rostro y los ojos como un objeto dentro de una imagen o video.

La cascada clasificadora "haarcascade_frontalface_alt2.xml" calcula sobre una imagen auxiliar en escala de grises una serie de coordenadas (x, y, w, h). En donde "x" representa la coordenada en el eje x, "y" es el valor de la coordenada en y, w es el ancho y h es la altura, respectivamente. Mediante estos valores, la cascada tipo Harr dibuja un cuadro delimitador sobre una región de interés (RoI) en la imagen de entrada original como se observa en la Fig. 3.3.a, cabe mencionar, que entre más píxeles contiene la imagen de entrada mayor tiempo de cálculo requiere el clasificador.

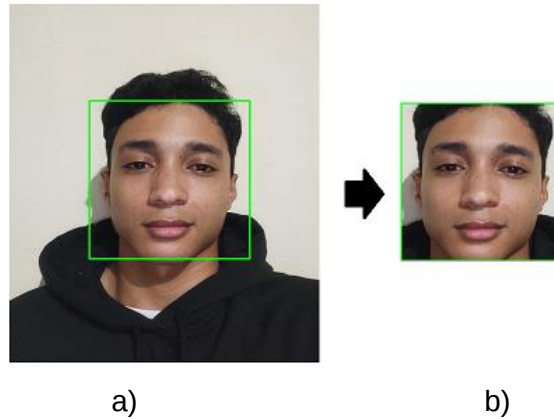


Fig. 3.3. Cuadro delimitador para el RoI, a) Dibujo del cuadro delimitador en la imagen de entrada y b) Recorte del rostro de la imagen de entrada.

La imagen de entrada contiene 1280 x 720 píxeles, para ayudar a su rendimiento se redimensionada la imagen de entrada a 450x550 pixeles. Como último paso, se procede a recortar el RoI en dimensiones de 259x259 píxeles, este último tramo contiene la información del rostro detectado en la imagen de entrada.

3.3.1.3 Detección y recorte de la posición de los ojos en la imagen del rostro recortado

Sobre la imagen anterior mostrada en la Fig. 3.3.b, se aplica la cascada clasificadora "haarcascade_eye.xml", este archivo calcula un nuevo RoI sobre la posición de ambos ojos, obteniendo coordenadas de diferentes valores (ex, ey, ew, eh) para cada objeto detectado. En la Fig. 3.4 se observa el proceso de recorte del cuadro delimitador, con esto se obtiene una imagen del estado del ojo con dimensiones de 224x224 pixeles, este nuevo RoI será la entrada del clasificador entrenado.

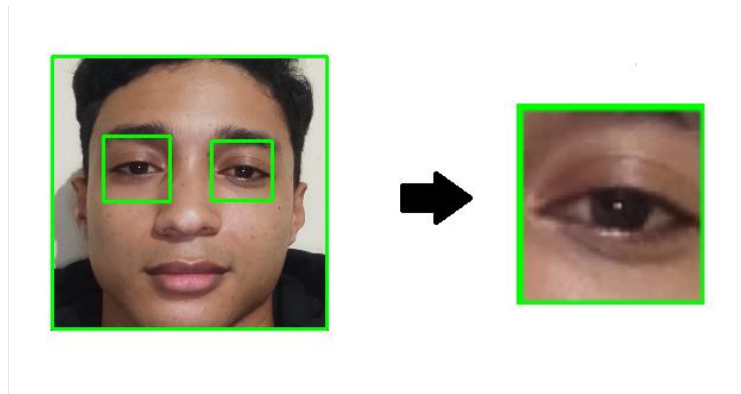


Fig. 3.4. Recorte de secciones cuadradas de las características y posición de un ojo.

3.3.1.4 Modelo de clasificación del estado de ambos ojos

La construcción de un clasificador desde sus raíces es un proceso costoso a nivel de hardware, requiere de una gran cantidad de datos y además necesita de largos periodos de entrenamiento. Para acelerar el proceso de construcción del modelo de clasificación, se optó por usar la tecnología de Transfer Learning mediante una red previamente entrenada de nombre VGG16.

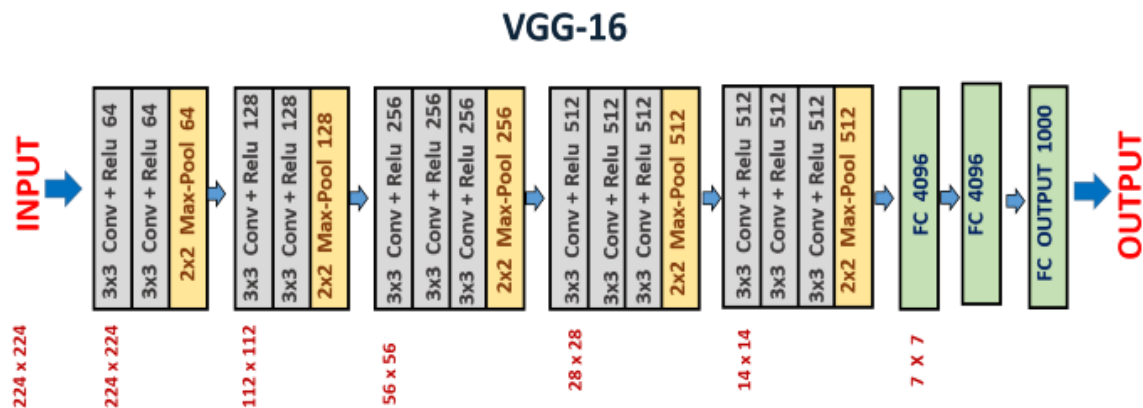


Fig. 3.5. Arquitectura del modelo VGG16.

Tomado de: [38]

VGG16 contiene múltiples capas de convolución, diversas capas de pooling y al final consta de dos capas densas y una capa de salida con activación softmax, que pueden ser reentrenadas para generar una red convolucional para detectar objetos basado en los requerimientos del diseñador.

Tabla 3.1. Capas densas y de salida del modelo VGG16 original.

fc1 (Dense)	(None, 4096)	102764544
fc2 (Dense)	(None, 4096)	16781312
predictions (Dense)	(None, 1000)	4097000

Entrenamiento del clasificador del estado del ojo

Los datos se separan en dos grupos: datos de entrenamiento y datos validación. En una proporción de 80% y 20% respectivamente; para sacar un mayor provecho al número de imágenes, se giran y se voltean para generar un mayor lote de imágenes, es decir, por cada imagen se puede generar hasta ocho imágenes nuevas para entrenamiento y validación en la base de datos. Después de la separación de los datos, se tienen 1202 imágenes para el entrenamiento y 250 para validación en dos respectivas categorías “Open Eye” y “Closed Eye”.

Para preparar los datos para el entrenamiento se importan las imágenes de la base de datos y además las respectivas etiquetas de los archivos en formato .zip. Las imágenes de entrada tienen un tamaño inicial de 150 x 150 píxeles con 3 canales de colores. La red VGG16 solo acepta imágenes de un tamaño específico, para ello, se escalan las imágenes del dataset a 224x224 píxeles para que sean de un valor aceptable para la entrada de la red.

Para comenzar con el entrenamiento, las últimas tres capas de la red VGG16 se sustituyen por otros parámetros como se muestra en la Fig. 3.6. Al realizar este cambio, la red será modificada para que el clasificador intente predecir las dos categorías asignadas. Para realizar la codificación del algoritmo de aprendizaje se usó un equipo con conexión a la GPU de TensorFlow y al entorno de programación Google Colab.

Antes del entrenamiento, se definen los hiperparámetros usados para ajustar el modelo y este pueda realizar mejores predicciones. Los que más destacan son:

- **Bach Size:** es el número de imágenes a procesar por lote de imágenes, es nuestro caso tiene un valor de 32.

- **Epoch:** es el número de ciclos o épocas que realiza el entrenamiento por cada lote de imágenes, actualmente es de 20.
- **Step per Epoch:** es el número de pasos por época que realiza el entrenamiento, para nuestra red se calcula dividiendo el número total de imágenes de entrenamiento entre el Batch Size. El valor calculado para nuestra red es de 37.
- **Validation Steps:** es el número de pasos por época que realiza la validación, se calcula tomando todas las imágenes del proceso de validación entre el Batch size. El valor calculado para la red es de 7.

Layer (type)	Output Shape	Param #
input_2 (InputLayer)	[(None, 224, 224, 3)]	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0
fc1 (Dense)	(None, 128)	3211392
fc2 (Dense)	(None, 128)	16512
output (Dense)	(None, 1)	129

Fig. 3.6. Arquitectura de la red VGG16 modificada en las últimas 3 capas.

Después del entrenamiento del modelo mediante Tranfers Learning, se preparó un nuevo set de datos llamada prueba (Test), el cual contine 34 imágenes divididas en dos grupos de 17 imágenes para su correcta clasificación. Este nuevo set es el encargado de generar las métricas (Ver Cap.3.6.1.6) de predicción escogidas para cuantificar la calidad de clasificación del modelo.

3.3.2 Método 2

Para el segundo método se aplicó la solución “Face Mesh” de Media Pipe para calcular el ratio de aspecto del ojo (Eye Aspect Ratio) y detectar somnolencia a través de un porcentaje de apertura en ambos ojos. Como en el método anterior, en la Fig. 3.7 se muestra la lógica del proceso que lleva a cabo el algoritmo presentado en esta sección del documento.

Teniendo como guía en el proceso anterior, podemos dividirlo en 3 partes claves:

1. Detección del rostro mediante la solución FaceMesh de MediaPipe.
2. Ubicar la zona de los ojos mediante los puntos de referencia faciales.
3. Cálculo de la relación de aspecto del ojo sobre la imagen de entrada.

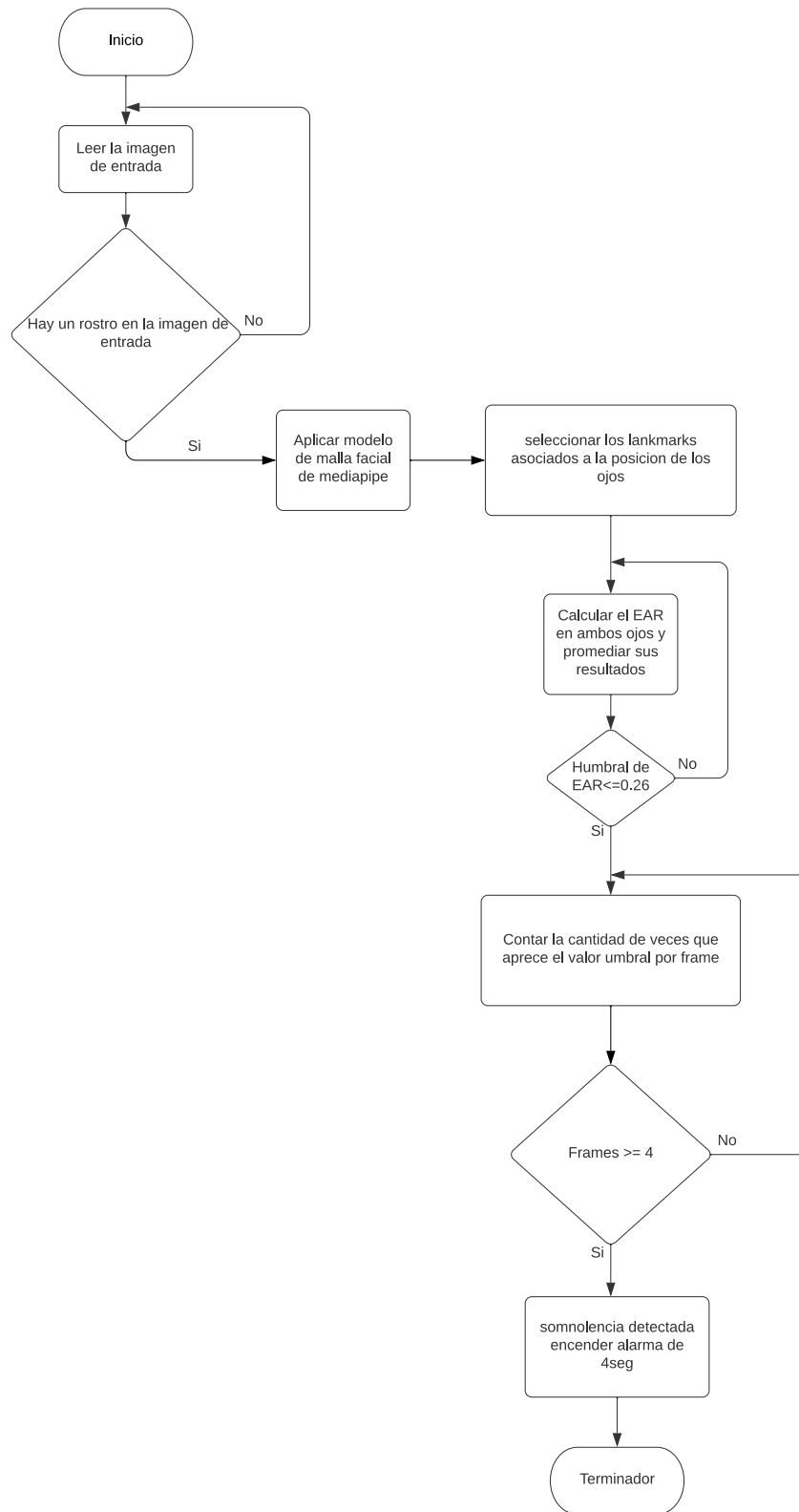


Fig. 3.7. Diagrama de flujo del método 2.

3.3.2.1 Detección del rostro mediante la solución FaceMesh de MediaPipe

La solución de Face Mesh (o Malla Facial) infiere sobre la imagen de entrada 468 puntos de referencias faciales (o Landmarks) como se muestra la Fig. 3.8. Cada punto de referencia tiene un valor estimado entre 1 y 468 respectivamente. Para el cálculo del EAR, solo necesitaremos los puntos relacionados con las zonas de los ojos, así descartamos otros puntos indeseados para aligerar el procesamiento de la librería MediaPipe.

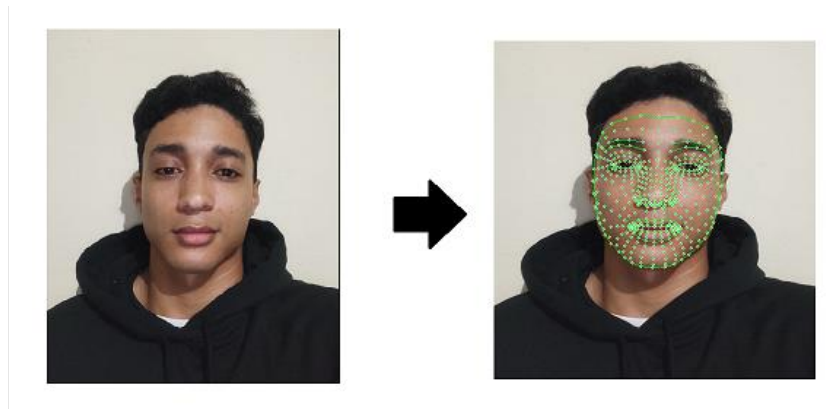


Fig. 3.8. 468 puntos de referencia mediante la librería de MediaPipe.

3.3.2.2 Localización de zona de los ojos con MediaPipe

La forma óptima para descartar zonas no requeridas de la imagen de entrada es a través de la creación de vectores de coordenadas mediante los puntos de referencia faciales. Para trabajar solo con las zonas de los ojos, se crean dos vectores, “Rigth_Eye” y “Left_Eye” estos representan la posición calculada del ojo derecho e izquierdo, respectivamente.

Al modificar los Landmarks de la malla facial, se usa la función `Cv2.circle()` para dibujar un círculo punteado sobre la zona de interés mediante los puntos de referencia almacenados en los vectores anteriores, en la Fig. 3.9 detalladamente se observa el proceso de inferencia sobre el RoI.



Fig. 3.9. Landmarks que describen la posición de los ojos en la imagen de entrada.

3.3.2.3 Cálculo de la relación de aspecto del ojo

Para el cálculo del EAR, se enumeran los puntos de referencia de P_1 hasta P_6 , en contra de las manecillas del reloj en el RoI. En la Fig.7.10 se observa la numeración mencionada anteriormente, ahora bien, la ecuación EAR calcula en la imagen del rostro detectado la distancia euclidiana de las coordenadas correspondiente de un ojo, esto puede aplicarse en ambos ojos.

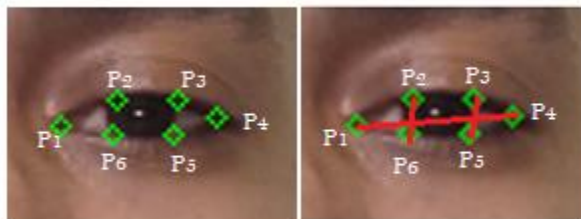


Fig. 3.10. Numeración de los puntos de interés en zona del ojo.

Cuando el ojo se encuentra abierto el valor EAR calculado es constante, a medida que el ojo se cierra este valor tiende a cero. El valor de EAR depende estrictamente del nivel de apertura en el que se encuentre el ojo, dado que se calcula el valor para ambos ojos, se promedia el valor calculado de ambos para obtener un resultado favorable.

La eficacia del modelo de EAR puede verse afectado por una serie de factores como la condición de iluminación, objetos que ocultan los ojos, movimientos abruptos del usuario, entre otros. Una forma efectiva de evaluar el rendimiento de EAR es mediante la frecuencia de parpadeo que

ocurre en los ojos, como objeto de validación seleccionamos dos videos de prueba como se muestra en la Fig 7.11, para medir el número de parpadeos (Blinks) que ocurren durante un tiempo determinado, con condiciones de buena iluminación y además con lentes para reducir el seguimiento del globo ocular.

Tabla 3.2. Tabla de prueba para medir el número de parpadeos en video.

Video	Tiempo de duración	FPS	Resolución	Total Frames
Rostro sin lentes	24 seg	30.1	720p	541
Rostro con lentes	20 seg	30.1	720p	513

4.Resultados y discusión

En el presente capítulo se muestran los resultados de los métodos empleados para la detección de la somnolencia y a su vez, presentar el método seleccionado para generar el indicador que controlara el proceso de detección de somnolencia en el prototipo.

4.1 Método 1

Para predecir el estado del ojo y generar una métrica para detectar somnolencia se estudiaron varios métodos. Se decidió usar red neuronal convolucional VGG16, modificar sus últimas capas para generar un modelo más robusto que clasifique entre dos clases, Closed_eye (o clase 0) que es la clase de mayor interés y la segunda clase Open_eye (o clases 1).

La primera métrica de evaluación aplicada al modelo final es la matriz de confusión, en la Fig. 4.1 se observa la predicción de colores de las clases y en ella se observan varios errores de predicción.

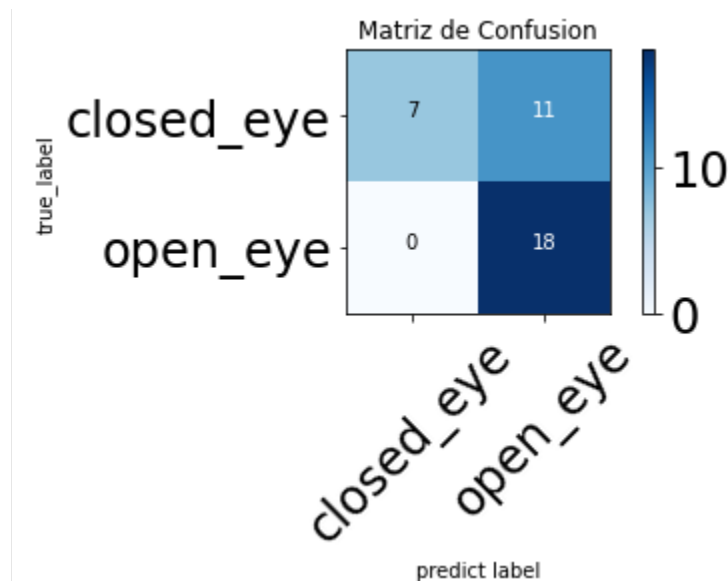


Fig. 4.1. Matriz de confusión del modelo.

Los valores de la diagonal principal $VP = 7$ y $VN = 18$ corresponde con los valores estimados de forma correcta del conjunto de prueba, en la otra diagonal informa los valores de error de predicción del modelo $FN = 0$ y $FP = 11$. El conjunto de datos está en equilibrio, es decir, ambas clases tiene el mismo número de datos, al calcular la exactitud del modelo nos entrega un valor aproximado del 69%, es resultado poco aceptable para el modelo. De este dato en particular podemos asumir que existe un error del 31%, es un porcentaje bastante considerable en a la predicción del modelo.

Ahora veamos, una interpretación con todas las métricas asociadas a la matriz de confusión:

	precision	recall	f1-score	support
closed_eye	1.00	0.39	0.56	18
open_eye	0.62	1.00	0.77	18
accuracy			0.69	36
macro avg	0.81	0.69	0.66	36
weighted avg	0.81	0.69	0.66	36

Fig. 4.2. Métricas asociadas a la matriz de confusión.

En la clase Closed_Eye, la Fig. 4.2 observa que tenemos una precisión de 100% y una sensibilidad 39%, por lo tanto, nuestro modelo tiene una probabilidad muy pequeña de predecir correctamente esta clase, y su vez está perdiendo predicciones correctas. Pero si el modelo logra predecir la etiqueta correctamente el resultado es altamente confiable.

Para la clase Open_Eye, nos encontramos con una precisión del 62% y una sensibilidad del 100%, esto hace que el modelo prediga la clase muy bien, pero es propenso a incluir falsos positivos dentro de sus verdaderos negativos, en otras palabras, el modelo entrenado selecciona predicciones erróneas de la otra clase como predicciones correctas para ella misma. Esto genera unas falsas alarmas para la clase Open_Eye.

Como diseñadores, nuestro principal objetivo era obtener un mayor porcentaje de aciertos de verdaderos positivos para clases Closed_Eye, como el resultado obtenido no fue el deseado optamos por descartar el modelo.

4.2 Método 2

Teniendo en cuenta la recomendación de investigaciones anteriores, inicialmente se consideró en usar el valor EAR de 0.20 para ser nuestro indicador de somnolencia, pero mediante algunas pruebas de experimentación se logró observar que, al aumentar este valor, podemos considerar otros niveles de apertura como un posible caso de parpadeo. En la Tabla 4.1 observamos los resultados del dataset seleccionado para probar valores entre 0.20, 0.23 y 0.26 es escoger más el más apto para asumir a para nuestro EAR umbral.

Tabla 4.1. Parpadeos calculados con diferentes valores de EAR.

	Parpadeos Reales	Parpadeos Calculados		
		EAR 0.20	EAR 0.23	EAR 0.26
Rostro sin Lentes	33	22	20	26
Rostro con Lentes	34	24	27	24

Para considerar que ocurrió un parpadeo, el EAR seleccionado debe repetirse al menos en dos ocasiones, es decir, este valor debe aparecer en mínimo 2 frames consecutivos del video para que la lectura del parpadeo sea exitosa. Ahora bien, de los resultados anteriores el valor de EAR 0.26 tiene un porcentaje de 76 % de aciertos con respecto a los otros valores de EAR. En otras palabras, este EAR predice mejor el número de parpadeos que de verdad ocurrieron.

El parpadeo está directamente ligado con la somnolencia y la fatiga, debido a ello, elegimos el valor de EAR 0.26 como el EAR umbral que debe superar el ojo para considerar que el ojo detectado se encuentre cerrado, el software de control toma esta consideración para medir el nivel de somnolencia. En otras palabras, el algoritmo está diseñado para que mida el tiempo (*tiempo = 3 seg*) en que el ojo se encuentre por debajo del EAR seleccionado y así estimar que la persona está teniendo un micro sueño, o en casos más extremos quedarse totalmente dormido.

5. Conclusiones

Durante todo el desarrollo de este documento, se presentó la idea de construir un prototipo que detecta la somnolencia en un entorno de trabajo. Mediante toda la investigación se observó que existen diferentes métodos de medir estas variables. Como propuesta se planteó contrastar dos tipos de tecnologías, comparar sus resultados y elegir el mejor método para detectar problemas de somnolencia. No obstante, los métodos sugeridos tomaron una posición no invasiva al cuerpo del usuario, es decir, no dependen de un sensor u otro dispositivo directamente ligado a la persona para medir la variable somnolencia.

Para el método 1, la visión artificial y las redes neuronales son dos grandes tecnologías que pueden servir para resolver diferentes problemas de clasificación en tiempo real o en datos específicos. Pero requieren de una buena infraestructura para trabajar, en el caso puntual de las redes neuronales convoluciones necesitan de grandes volúmenes de datos (imágenes) para lograr un resultado favorable en un diseño. En otros casos se puede optar por la tecnología de Transfer Learning para cumplir con dicho objetivo de clasificación, aunque es un método para aumentar la precisión y reducir el tiempo de entrenamiento de una red, también tienen su margen de error. En nuestro caso, la red entrenada mediante la arquitectura de VGG16 no cumplió con los resultados esperados, así que fue descartada para el prototipo final.

En el método 2, el modelo de malla facial 3D de la librería MediaPipe es una función que ha sido probada y reentrenada de diferentes maneras, para proveer una mayor exactitud y optimización. Al no depender de infraestructura de hardware demasiado alta, es compatible con diferentes dispositivos de bajo costo computacional como la Raspberry Pi y todos sus componentes. Al igual que otras tecnologías, no es 100% segura. El modelo “Face Mesh” tiende a ser limitado a las condiciones de iluminación, la calidad de píxeles de la imagen y la distancia a la que se encuentre el usuario de la cámara.

El EAR al depender de puntos de referencias faciales del rostro (véase la Fig. 3.10) es un complemento perfecto, que sirve para medir el nivel de apertura de los globos oculares y así poder generar un indicador que mida la fatiga ocular o somnolencia; al final este fue el método seleccionado para nuestro prototipo.

6.Trabajos futuros

Uno de los trabajos futuros propuesto en este documento, consiste en la optimización y reentrenamiento del modelo de clasificación, a través de diferentes metodologías de Transfer Learning, comparar sus resultados y verificar cual presentaría mejores métricas para la tarea de clasificación del estado del ojo.

Por otro lado, los modelos de malla facial 3D presentan muy buenos resultados en el seguimiento del rostro u otras características de este, pero tienden a presentar mediciones muy erráticas cuando un objeto (gafas en nuestro caso) se encuentra en su zona de muestreo, un trabajo futuro con esta tecnología consistiría en construir una red neuronal de apoyo para identificar el objeto que este superpuesto a la malla facial 3D.

Como recomendación, la mejor manera de realizar un entrenamiento es mediante un entorno de ejecución local con una tarjeta gráfica de alto rendimiento. Debido a que la web de Google Colaboratory presenta algunos límites al momento de entrenar modelos en la nube.

7.Presupuesto

A continuación, se presenta en la Tabla 7.1 la descripción generalizada del presupuesto junto con todos los elementos que fueron utilizados para el pleno desarrollo del proyecto de investigación que se encuentra consignado en este mismo documento. De este modo, se precisa dar a conocer los detalles de los materiales, mano de obra, cantidades requeridas, precios del mercado en la actualidad y demás herramientas tangibles e intangibles necesarias con las que se logró cumplir con los objetivos específicos planteados.

Tabla 7.1. Presupuesto general del proyecto.

Descripción	Entidad Aportante	Cant	Precio Unitario	Total
Kit Raspberry Pi 3B+ Esencial	Universidad del Magdalena	2	205.882	411.764
Cámara Raspberry Pi v2	Universidad del Magdalena	2	107.563	215.126
Adaptador de carga 5V a 3A.	Universidad del Magdalena	1	40.500	40.500
Cable HDMI	Universidad del Magdalena	1	10.924	10.924
Modulo Led Infrarrojo	Universidad del Magdalena	4	14.285	57.140
Cargador batería Li-Po 60" iMAX B6	Universidad del Magdalena	1	239,500	239,500
Batería Li-Po 3000mAh 14,8 v	Universidad del Magdalena	2	186.300	372.600
Filamento PLA para impresora 3D	Universidad del Magdalena	1	176.715	148.500
Camara DepthIntelk RealSense D435	Universidad del Magdalena	1	1.152.632	1.152.632
Total				2.648.686

La Tabla 7.2 posee la información resumida sobre el total de los aportes realizados por los actores involucrados y comprometidos con el desarrollo de este proyecto, relacionados con el tipo de aporte suministrado ya sea en dinero o en especie.

Tabla 7.2. Aportes realizados por cada entidad participante en el proyecto.

No.	Entidad Aportante	Tipo		Total
		Dinero	Especie	
1	Universidad del Magdalena	2.648.686	Equipos y Accesorios	2.647.186
Total				

8. Bibliografía

- [1] P. R. Gil-Monte, “Algunas razones para considerar los riesgos psicosociales en el trabajo y sus consecuencias en la salud pública”, *Rev Esp Salud Publica*, vol. 83, núm. 2, pp. 169–173, 2009, [En línea]. Available: https://scielo.isciii.es/scielo.php?script=sci_arttext&pid=S1135-57272009000200003
- [2] E. Miró, Á. Solanes, P. Martínez, A. I. Sánchez, y J. R. Marín, “Relación entre el burnout o «síndrome de quemarse por el trabajo», la tensión laboral y las características del sueño”, *Psicothema*, vol. 19, núm. 3, pp. 388–394, 2007, [En línea]. Available: <https://www.redalyc.org/pdf/727/72719305.pdf>
- [3] E. Rosales Mayor y J. R. De Castro Mujica, “Somnolencia: Qué es, qué la causa y cómo se mide”, *Acta Médica Peruana*, vol. 27, núm. 2, pp. 137–143, jun. 2010.
- [4] I. Radun y H. Summala, “Sleep-related Fatal Vehicle Accidents: Characteristics of Decisions Made by Multidisciplinary Investigation Teams”, Helsinki, 2004.
- [5] M. J. Flores, J. M. Armingol, y A. de la Escalera, “Sistema Avanzado de Asistencia a la Conducción para la Detección de la Somnolencia”, *Revista Iberoamericana de Automática e Informática Industrial RIAI*, vol. 8, núm. 3, pp. 216–228, jul. 2011, doi: 10.1016/j.riai.2011.06.009.
- [6] D. A. Rodríguez Morillo, “Control del cursor de un computador mediante eyetracking”, 2020.
- [7] P. J. García Paterna, “Desarrollo de un sistema inteligente de detección de fatiga en conductores”, 2019.
- [8] C. M. del Toro Rondón y C. A. Robles Algarín, “Diseño e implementación de un brazo robótico para jugar ajedrez utilizando un sistema de visión artificial”, Universidad del Magdalena, Santa Marta, 2019.
- [9] O. Buzón-García, C. Romero-García, y A. Verdú Vázquez, *Innovaciones metodológicas con TIC en educación*, Colección. Madrid: Dykinson S.L., 2021.
- [10] F. Escobar-Córdoba, *Somnolencia diurna excesiva e insomnio: males de los tiempos actuales*. Bogotá, D.C.: Universidad Nacional de Colombia, 2020. [En línea]. Available: https://books.google.com.co/books?id=TK_2DwAAQBAJ&printsec=frontcover&dq=que+es+somnolencia&hl=es-419&sa=X&ved=2ahUKEwjC-4iGvdb6AhXaRDABHegfC20Q6AF6BAgGEAI#v=onepage&q=que es somnolencia&f=false
- [11] E. Rosales Mayor y J. R. de Castro Mujica, “Somnolencia: Qué es, qué la causa y cómo se mide”, *Acta Médica Peruana*, vol. 27, núm. 2, pp. 137–143, jun. 2010, [En línea].

Available: http://www.scielo.org.pe/scielo.php?script=sci_arttext&pid=S1728-59172010000200010

- [12] L. Rouhiainen, *Inteligencia artificial 101 cosas que debes saber hoy sobre nuestro futuro*. Barcelona: Alienta Editorial, 2018.
- [13] J. P. Mueller y L. Massaron, *Deep Learning for dummies*. New Jersey: John Wiley & Sons, Inc., 2019. [En línea]. Available: [https://books.google.com.co/books?id=JjqRDwAAQBAJ&printsec=frontcover&dq=deep+learning&hl=es-419&sa=X&redir_esc=y#v=onepage&q=deep learning&f=false](https://books.google.com.co/books?id=JjqRDwAAQBAJ&printsec=frontcover&dq=deep+learning&hl=es-419&sa=X&redir_esc=y#v=onepage&q=deep+learning&f=false)
- [14] E. Parra Barrero, Á. Rodríguez Vázquez, y J. Fernández Berni, “Aceleración del algoritmo de Viola-Jones mediante rejillas de procesamiento masivamente paralelo en el plano focal”, Universidad de Sevilla, Sevilla, 2015.
- [15] C. Barroso Heredia y F. Díaz de María, “Implementación del algoritmo de detección de caras de Viola y Jones sobre una FPGA”, Universidad Carlos III de Madrid - Escuela Politécnica Superior, Madrid.
- [16] P. ; Viola y M. Jones, “Rapid Object Detection Using a Boosted Cascade of Simple Features”, 2004.
- [17] A. A-Nasser, M. Mohammad, y A.-M. Mohame, “3D Face Mesh Modeling for 3D Face Recognition”, en *State of the Art in Face Recognition*, I-Tech Education and Publishing, 2009. doi: 10.5772/6643.
- [18] E. del Hoyo Dorado y A. J. Sánchez Salmerón, “Diseño, implementación y evaluación de una red neuronal convolucional para la detección de puntos principales en naranjas”, Universitat Politècnica de València, Valencia, 2019.
- [19] J. Durán Suárez y A. Del Real Torres, “Redes Neuronales Convolucionales en R Reconocimiento de caracteres escritos a mano”, Universidad de Sevilla, Sevilla, 2017.
- [20] Y. LeCun y L. Bottou, “Comparison of learning algorithms for handwritten digit recognition.”, pp. 1–5, 1995.
- [21] “Redes neuronales convolucionales (LeNet)”, *Dive Into Deep Learning*, 2022. http://d2l.ai/chapter_convolutional-neural-networks/lenet.html#lenet (consultado oct. 08, 2022).
- [22] L. Moreno Díaz-Alejo y A. Ruiz Iniesta, “Análisis comparativo de arquitecturas de redes neuronales para la clasificación de imágenes”, Universidad Internacional de la Rioja, Madrid, 2020.
- [23] A. De Luca, M. Irigoitia, G. Pérez, y C. Pons, “Uso de la Técnica de Transfer Learning en Machine Learning para la Clasificación de Productos en el Banco Alimentario de La Plata”. Buenos Aires, pp. 1–16.

- [24] V. Daniel y M. Jaramillo, "Machine Learning Models for Crime Prediction in Medellin City", 2021.
- [25] E. del Hoyo Dorado y A. J. Sánchez Salmerón, "Diseño, implementación y evaluación de una red neuronal convolucional para la detección de puntos principales en naranjas", Tesis de Máster, Universitat Politècnica de València, Valencia, 2019.
- [26] C. Dewi, R. C. Chen, X. Jiang, y H. Yu, "Adjusting eye aspect ratio for strong eye blink detection based on facial landmarks", *PeerJ Comput Sci*, vol. 8, 2022, doi: 10.7717/peerj-cs.943.
- [27] A. Kuwahara, K. Nishikawa, R. Hirakawa, H. Kawano, y Y. Nakatoh, "Eye fatigue estimation using blink detection based on Eye Aspect Ratio Mapping(EARM)", *Cognitive Robotics*, vol. 2, pp. 50–59, 2022, doi: 10.1016/j.cogr.2022.01.003.
- [28] Kaggle, "Yawn Eye Dataset New", <https://www.kaggle.com/datasets/serenaraju/yawn-eye-dataset-new>, oct. 10, 2022. <https://www.kaggle.com/datasets/serenaraju/yawn-eye-dataset-new> (consultado oct. 09, 2022).
- [29] TensorFlow, "Why TensorFlow", <https://www.tensorflow.org/about>, oct. 10, 2022. <https://www.tensorflow.org/about> (consultado oct. 09, 2022).
- [30] Keras, "About Keras". <https://keras.io/about/> (consultado oct. 09, 2022).
- [31] NumPy, "What is NumPy?"
- [32] PyPI, "Python Imaging Library", <https://pypi.org/project/Pillow/>, oct. 10, 2022. <https://pypi.org/project/Pillow/> (consultado oct. 09, 2022).
- [33] Python, "Miscellaneous operating system interfaces (OS)". <https://docs.python.org/3.10/library/os.html> (consultado oct. 09, 2022).
- [34] Pandas, "About Pandas". https://pandas.pydata.org/docs/getting_started/overview.html (consultado oct. 09, 2022).
- [35] Google Colab, "What is Colaboratory?" https://colab.research.google.com/?hl=es#scrollTo=5fCEDCU_qrC0 (consultado oct. 09, 2022).
- [36] OpenCV, "Open Source Computer Vision Library". <https://forum.opencv.org/t/welcome-to-opencv-forum/7> (consultado oct. 09, 2022).
- [37] Google, "MediaPipe". <https://google.github.io/mediapipe/> (consultado oct. 09, 2022).
- [38] Vaibhav Khandelwal, "The Architecture and Implementation of VGG-16". <https://pub.towardsai.net/the-architecture-and-implementation-of-vgg-16-b050e5a5920b> (consultado nov. 28, 2022).

ANEXOS

A. Algoritmo de entrenamiento de una CNN mediante el modelo VGG16.

```
#Funcion para subir un archivo de pc a Google Colab
from google.colab import files
files.upload()
#Comando para descomprimir dentro de la carpeta /content de google colab
!unzip Data_s.zip -d /content

#Librerias usadas en el entrenamiento
from keras.models import Sequential, Model
from keras.layers import Conv2D, MaxPool2D, Dense, Flatten, Dropout, Batch
Normalization, Input
#from keras.optimizers import Adam
from keras.callbacks import TensorBoard, ModelCheckpoint
from keras.utils import np_utils
import os
import numpy as np
from keras.preprocessing import image
from keras.applications.imagenet_utils import preprocess_input, decode_pre
dictions
from keras.applications.vgg16 import VGG16
from keras.preprocessing.image import ImageDataGenerator
from sklearn.utils import shuffle
from sklearn.model_selection import train_test_split
import cv2
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import itertools
from itertools import product
import matplotlib.image as mpimg
import matplotlib.pyplot as plt

BATCH_SIZE = 32 #cantidad de imagenes a procesar por lotes de imágenes
IMG_SHAPE = 224 #dimensiones de las imágenes de entrada (224,224)

#Rutas de datos
dir_train='/content/data/train' #Imagenes de Entrenamiento
dir_val='/content/data/val' #Imagenes de Validación
```

```

#conteo del número de imágenes de entrenamiento
train_closed_dir = os.path.join(dir_train, 'closed_eye')
train_open_dir = os.path.join(dir_train, 'open_eye')

num_closed_tr = len(os.listdir(train_closed_dir))
num_open_tr = len(os.listdir(train_open_dir))
print('numero de imagenes de entrenamiento closed:', num_closed_tr)
print('numero de imagenes de entrenamiento open:', num_open_tr)
total_train = num_closed_tr + num_open_tr
print('número total de imágenes de entrenamiento:', total_train)

#conteo de imagenes de validacion
val_closed_dir = os.path.join(dir_val, 'closed_eye')
val_open_dir = os.path.join(dir_val, 'open_eye')

num_closed_val = len(os.listdir(val_closed_dir))
num_open_val = len(os.listdir(val_open_dir))
print('numero de imagenes de entrenamiento closed:', num_closed_val)
print('numero de imagenes de entrenamiento open:', num_open_val)
total_val = num_closed_val + num_open_val
print('número total de imagenes de entrenamiento:', total_val)

#
train_image_generator = ImageDataGenerator(rescale=1./255,
                                           shear_range=0.2,
                                           zoom_range=0.2,
                                           horizontal_flip = True)

val_image_generator = ImageDataGenerator(rescale=1./255)

#datos de entrenamiento
train_data_gen = train_image_generator.flow_from_directory(batch_size=BATCH_SIZE,
                                                           directory=dir_train,
                                                           shuffle=True,
                                                           target_size=(224,224),
                                                           class_mode='categorical')

val_data_gen = val_image_generator.flow_from_directory(batch_size=BATCH_SIZE,
                                                       directory=dir_val,
                                                       shuffle=True,
                                                       target_size=(224,244),
                                                       class_mode='categorical')

print(train_data_gen.class_indices)
#condiciones de las imagenes de entrada
height =224 #Alto de la imagen
width = 224 #Ancho de la imagen
num_class = 2 #numero de clases a dectectar
epoch = 20 #Epocas de entrenamiento
img_input= Input(shape=(width, height , 3))

```

```

#Descarga del modelo de tranferleaning VGG16
model_vgg = VGG16(input_tensor=img_input, include_top=True, weights='image
net')
model_vgg.summary()
#Modificacion de las ultimas 3 capas para el entrenamineto del nuevo model
o basado en VGG16

last_layer=model_vgg.get_layer('block5_pool').output
x = Flatten(name='flatten')(last_layer)
x = Dense(256, activation='relu', name='fc1')(x)
x = Dense(128, activation='relu', name='fc2')(x)
out=Dense(num_class, activation='softmax', name='output')(x)
custom_vggmodel = Model(img_input, out)
custom_vggmodel.summary()

#Apagar las ultimas 4 capas del modelo de vgg16
for layer in custom_vggmodel.layers[:-3]:
    layer.trainable = False

custom_vggmodel.summary()
custom_vggmodel.compile(loss='categorical_crossentropy', optimizer='adadel
ta', metrics=['accuracy'])

#calculo del numero de pasos por epocas de entrenaimento
batch_size=32
step_train=train_data_gen.samples//batch_size
step_val=val_data_gen.samples//batch_size

print("Numero de pasos por epoca de entrenamiento:", step_train)
print("Numero de pasos por epoca de validacion", step_val)

model_history = custom_vggmodel.fit(
    train_data_gen,
    batch_size=batch_size,
    steps_per_epoch = step_train,
    epochs=epoch,
    validation_data = val_data_gen,
    validation_steps = step_val)

#Conmando para guardad el modelo de red entrenado
custom_vggmodel.save('model_VGG16_catego_s.h5')
custom_vggmodel.save('/content/model_VGG16_catego_s.h5')
custom_vggmodel.save_weights('model_VGG16_catego_weight.h5_')

```

```

# funcion para graficar los valores:

def plotTraining(hist, epochs, typeData):

    if typeData=="loss":
        plt.figure(1,figsize=(10,5))
        yc=hist.history['loss']
        xc=range(epochs)
        plt.ylabel('Loss', fontsize=24)
        plt.plot(xc,yc, '-r',label='Loss Training')
    if typeData=="accuracy":
        plt.figure(2,figsize=(10,5))
        yc=hist.history['accuracy']
        for i in range(0, len(yc)):
            yc[i]=100*yc[i]
        xc=range(epochs)
        plt.ylabel('Accuracy (%)', fontsize=24)
        plt.plot(xc,yc, '-r',label='Accuracy Training')
    if typeData=="val_loss":
        plt.figure(1,figsize=(10,5))
        yc=hist.history['val_loss']
        xc=range(epochs)
        plt.ylabel('Loss', fontsize=24)
        plt.plot(xc,yc, '--b',label='Loss Validate')
    if typeData=="val_accuracy":
        plt.figure(2,figsize=(10,5))
        yc=hist.history['val_accuracy']
        for i in range(0, len(yc)):
            yc[i]=100*yc[i]
        xc=range(epochs)
        plt.ylabel('Accuracy (%)', fontsize=24)
        plt.plot(xc,yc, '--b',label='Training Validate')

    plt.rc('xtick',labelsize=24)
    plt.rc('ytick',labelsize=24)
    plt.rc('legend', fontsize=18)
    plt.legend()
    plt.xlabel('Number of Epochs',fontsize=24)
    plt.grid(True)

plotTraining(model_history,epoch, "loss")
plotTraining(model_history,epoch, "accuracy")
plotTraining(model_history,epoch, "val_loss")
plotTraining(model_history,epoch, "val_accuracy")

```

```

#prueba del modelo modificado de vgg16
import keras
from keras.applications.mobilenet import preprocess_input
from keras.models import load_model

dir_test='/content/data/test'

test_datagen = ImageDataGenerator(preprocessing_function=preprocess_input)

test_generator = test_datagen.flow_from_directory(
    directory=dir_test,
    target_size=(224,224),
    color_mode="rgb",
    batch_size=1,
    class_mode=None,
    shuffle=False,
    seed=42)

#cargar el modelo entrenado de vgg
model_new_vgg = load_model("model_VGG16_catego_s.h5")

step_size_test=test_generator.n//test_generator.batch_size
test_generator.reset()
prediction = model_new_vgg.predict(test_generator, steps=step_size_test, verbose=1)

y_pred = np.argmax(prediction, axis=1) #predicciones del model nuevo modelo de vgg16
print(y_pred)# vector de resultados

#funcion para generar un vector con informacion real de conjunto de testing
filenames = test_generator.filenames
print(filenames)
y_real=[]

for i in range(0, len(filenames)):
    your_path=filenames[i]
    path_list = your_path.split(os.sep)
    if ("cl" in path_list[1]):
        y_real.append(0)
    if ("op" in path_list[1]):

```

```

        y_real.append(1)

print("vector de valores reales del conjunto de testing:",y_real)

#funcion para dibujar la matrix de confusion en google colab
def plot_confusion_matrix(cm, classes, normalize=False,title='Confusion Ma
trix',cmap=plt.cm.Blues):
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks=np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)

    thresh=cm.max()/ 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j], horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('true_label')
    plt.xlabel('predict label')

#librerias de sklearn para calcular la matriz de confusion
from sklearn.metrics import confusion_matrix
from sklearn.metrics import mean_absolute_error
#-----#
cm = confusion_matrix(y_real, y_pred)
print(cm)

total_cm=sum(sum(cm) )

accuracy_1=(cm[0,0]+cm[1,1])/total_cm
print ('Accuracy :', accuracy_1)

sensitivity = cm[0,0]/(cm[0,0]+cm[0,1])
print('Sensitivity :', sensitivity )

```

```
specificity = cm[1,1]/(cm[1,0]+cm[1,1])
print('Specificity :', specificity)

print(mean_absolute_error(y_real, y_pred))

cm_plot_labels = (train_data_gen.class_indices)
plot_confusion_matrix(cm, cm_plot_labels, title='Matriz de Confusion')

from sklearn.metrics import classification_report

target_names = ['closed_eye', 'open_eye']

print(classification_report(y_real, y_pred, target_names=target_names))
```

B. Algoritmo final para detección de somnolencia mediante MediaPipe.

```
#Media pipe
import cv2
import mediapipe as mp
import numpy as np
import time
#from playsound import playsound

#Cargar as soluciones
mp_face_mesh = mp.solutions.face_mesh #modelo de malla facial 3D
mp_drawing = mp.solutions.drawing_utils #modelo para dibujar dentro de mediapipe

#funcion de cálculo de ear

def eye_aspect_ratio(coordinates):
    d_A = np.linalg.norm(np.array(coordinates[1]) - np.array(coordinates[5]))
    d_B = np.linalg.norm(np.array(coordinates[2]) - np.array(coordinates[4]))
    d_C = np.linalg.norm(np.array(coordinates[0]) - np.array(coordinates[3]))
    return (d_A + d_B)/(2 * d_C)

#variables auxiliares
sample = 0
cont_sleep = 0
t_time = 0
init = 0
final = 0
blink = False
ear_thresh = 0.26 #umbral
num_frames = 2 #numeros de frames consecutivos decir que es un parpadeo
count = 0 #variable auxiliar para contar
blink_counter = 0 #variable auxiliar para contar

#Carga del archivo .mp4 para prueba de libreria
cap = cv2.VideoCapture(0)

total_frames=int(cap.get(cv2.CAP_PROP_FRAME_COUNT))
print("total_frames:",total_frames)
#vectores de posicion de ojos mediante landmarks
```

```
index_left_eye=[33, 160, 158, 133, 153, 144]
index_right_eye=[362, 385, 387, 263, 373, 380]
```

```
#toma de datos por parte de mediapipe
with mp_face_mesh.FaceMesh(static_image_mode = False,
    max_num_faces=1,
    #model_selection=1,
    min_detection_confidence=0.5,
    min_tracking_confidence=0.5) as face_mesh:
```

```
#inicio del ciclo While
```

```
while True:
    ret, frame = cap.read() #lectura del video
    if ret == False:
        break
```

```
    frame = cv2.flip(frame, 1)#soluciona el probelma de el espejo
    #frame = cv2.resize(frame, (800,600))
    height, width, canal = frame.shape #extraccion de dimensiones de la entrada
    frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB) #cambio de olor a escalr grises para
el calulo de los landmarks
    results = face_mesh.process(frame_rgb)
```

```
#Almacena los valores de las coordenadas calculadas
coordinates_left_eye = []
coordinates_rigth_eye = []
```

```
if results.multi_face_landmarks:
    for face_landmarks in results.multi_face_landmarks:
        for index in index_left_eye:
            x = int(face_landmarks.landmark[index].x * width)
            y = int(face_landmarks.landmark[index].y * height)
            coordinates_left_eye.append([x,y])
            cv2.circle(frame, (x, y), 2, (124,252,0), 1)
            cv2.circle(frame, (x, y), 1, (124,252,0), 1)

        for index in index_right_eye:
            x = int(face_landmarks.landmark[index].x * width)
            y = int(face_landmarks.landmark[index].y * height)
            coordinates_rigth_eye.append([x,y])
            cv2.circle(frame, (x, y), 2, (124,252,0), 1)
```

```

cv2.circle(frame, (x, y), 1, (124,252,0), 1)

ear_left_eye = eye_aspect_ratio(coordinates_left_eye)
ear_rigth_eye = eye_aspect_ratio(coordinates_rigth_eye)
ear = (ear_left_eye + ear_rigth_eye)/2

"""
d_A = np.linalg.norm( np.array(coordinates_left_eye[1])-
np.array(coordinates_left_eye[5]) )
d_B = np.linalg.norm( np.array(coordinates_left_eye[2])-
np.array(coordinates_left_eye[4]) )
d_C = np.linalg.norm( np.array(coordinates_left_eye[0])-
np.array(coordinates_left_eye[3]) )
ear_left_eye = (d_A + d_B)/(2 * d_C)

d_A_1 = np.linalg.norm(
np.array(coordinates_rigth_eye[1])- np.array(coordinates_rigth_eye[5]) )
d_B_1 = np.linalg.norm( np.array(coordinates_rigth_eye[2])-
np.array(coordinates_rigth_eye[4]) )
d_C_1 = np.linalg.norm( np.array(coordinates_rigth_eye[0])-
np.array(coordinates_rigth_eye[3]) )

ear_rigth_eye = (d_A_1 + d_B_1)/(2 * d_C_1)

#ear = (ear_left_eye + ear_rigth_eye)/2
"""

cv2.putText(frame, f'blinks:{int(count)}',(50,20), cv2.FONT_HERSHEY_SIMPLEX, 1, (0,
255, 0), 1)
cv2.putText(frame, f'numero de microsueno:{int(cont_sleep)}',(50,80),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 1)
cv2.putText(frame, f'Tiempo de microsueno:{str(sample)}',(50, 50),
cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 1)

#ojo cerrado
if ear <= ear_thresh and blink == False:
    count += 1
    blink = True
    init = time.time()
else:
    if ear > ear_thresh and blink == True:
        blink = False
        final = time.time()

```

```
t_time = round(final - init, 0)

if t_time >= 3:
    cont_sleep += 1
    sample = t_time
    init=0
    final=0
    #playsound('C:/Users/Usuario/Desktop/Red_CNN/Alarma.mp3')

cv2.imshow("frame", frame)
k = cv2.waitKey(1) & 0xFF
if k==27:
    break

cap.release()
cv2.destroyAllWindows()
```