Sistema de detección de somnolencia en conductores de automóviles empleando técnicas de procesamiento de imágenes y machine learning

Eduard Antonio Suárez Buitrago

Universidad de Pamplona

167332: Trabajo de Grado

Hernando José Velandia Villamizar

M.Sc en Ingeniería Biomédica

Diciembre 2021

Dedicatoria

En primer lugar me gustaría dedicar este triunfo a mi Madre María de la Cruz Buitrago Hernandez y a mi Padre Angelmiro Suárez Gómez, siempre han sido mi apoyo y mi gran motivación para las cosas por las cuales he luchado. A mi abuela Aminta Gómez, que siempre me brindó su amor, apoyo y me motivó a que siempre debía estar firme a pesar de las adversidades.

Agradecimientos

Agradezco a Dios por permitirme llegar hasta acá, sé que siempre me ha ayudado y me ha guiado. Un agradecimiento enorme a mis padres por todo el apoyo brindado en todo este proceso y porque en todo momento han creído en mí. Quiero agradecer al programa de ingeniería en telecomunicaciones de la universidad de Pamplona, a mi director de tesis el ingeniero Hernando José Velandia por la enseñanza brindada como docente y por ser parte de este trabajo, sus palabras y consejos siempre me motivaron a realizar de la mejor manera posible este trabajo. De igual modo a todos los demás docentes que durante la carrera me dictaron clase y de alguna u otra forma aportaron a mi formación como profesional.

A mis hermanos y amigos, que también de algún modo hicieron parte fundamental en este proceso, a mi amigo Fredy Fernandez que en el momento en el que necesité de su ayuda o de un favor siempre estuvo ahí, a Juan José Gelves con quién he compartido muchos momentos y sé que en cualquier situación siempre va a estar ahí para apoyarme. Por ultimo a mis compañeros de carrera, en especial a Yeferson Sánchez y Miguel Barraza, con quienes compartí muchos momentos en el proceso de formación y con quienes siempre hubo apoyo mutuo. Solo me queda decir que sé que tengo un gran potencial y trabajaré en todo momento para demostrarlo.

Índice

Índice de figuras

Índice de tablas

Resumen

La inteligencia artificial en los últimos años ha tomado bastante fuerza en lo relacionado a la solución de problemas de la vida real y esto ha motivado a qué tanto, estudiantes, docentes, investigadores, entre otros, se vean interesados y hagan parte de este tema como es el caso de la presente propuesta, donde se emplean técnicas de machine learning. Estas técnicas son aplicadas en la clasificación de imágenes para determinar el grado de somnolencia en conductores de automóvil y según dicho resultado emitir una alerta de precaución, determinando de igual manera cuál de ellas presenta un mejor rendimiento. Entre las técnicas que se emplean tenemos: el Algoritmo k-Nearest Neighbor (KNN), el algoritmo Support vector machine (SVM) y una red neuronal convolucional (CNN). Además, se realiza un desarrollo matemático de algunas de las técnicas empleadas. El conjunto de datos a emplear es el denominado MRL EYE Dataset, este tiene recopilado los datos de 37 personas diferentes (33 hombres y 4 mujeres) y consta de 84.898 imágenes, pero solo se hace uso de un porcentaje de estas imágenes. Aparte de las imágenes utilizadas de la base de datos, se realizan clasificaciones con imágenes que no pertenecen a la base de datos, sino que estas imágenes son tomadas por una cámara web y también se realiza la clasificación de imágenes mediante la implementación de vídeo en tiempo real. Al culminar el presente proyecto se cuenta con una herramienta robusta y muy útil en el manejo del tema de somnolencia en conductores de automóvil.

Palabras clave: Alerta, Clasificación de imágenes, Machine learning, MRL EYE Dataset, Rendimiento, Somnolencia

Abstract

Artificial intelligence in recent years has gained considerable strength in relation to solving problems in real life and this has motivated how much, students, teachers, researchers, among others, are interested and become part of this topic as This is the case of the present proposal, where machine learning techniques are used. These techniques are applied in the classification of images to determine the degree of drowsiness in automobile drivers and, according to this result, issue a caution alert, determining in the same way which of them presents better performance. Among the techniques used we have: the k-Nearest Neighbor Algorithm (KNN), the Support vector machine algorithm (SVM) and a convolutional neural network (CNN). In addition, a mathematical development of some of the techniques used is carried out. The data set to be used is the so-called MRL EYE Dataset, this has collected the data of 37 different people (33 men and 4 women) and consists of 84,898 images, but only a percentage of these images is used. Apart from the images used from the database, classifications are made with images that do not belong to the database, but these images are taken by a webcam and the classification of images is also performed by implementing video in real time. At the end of this project, there is a robust and very useful tool in managing the issue of drowsiness in car drivers.

Keywords: Alert, Image Classification, Machine learning, MRL EYE Dataset, Performance, Sleepiness

Introducción

Muchos de los avances de la inteligencia artificial (IA) en los últimos años han hecho posible que las computadoras realicen tareas cada vez mas complejas e importantes. Gran parte del poder de la inteligencia artificial proviene a partir del uso que se le está dando en la actualidad a algoritmos de machine learning como de aprendizaje profundo en el que se basan en conjuntos de datos para entrenar modelos cada vez mas precisos.

Sin embargo, estas técnicas también pueden tener una debilidad, pues todos estos sistemas de inteligencia artificial aprenden a partir de lo que se les enseña. Por ello, si al momento del entrenamiento de estos sistemas no se usan conjuntos de datos robustos y diversos, la precisión y demás resultados que podamos obtener podrían generar inseguridad e incluso peligro.

La somnolencia en la conducción es un estado de fatiga y agotamiento que hacen que el cuerpo pierda capacidades de reaccionar; lo que puede causar una tragedia o accidente de tránsito. En Colombia, según el observatorio nacional de seguridad vial, en el 2019, 184 personas terminaron con lesiones como consecuencia de los micro sueños ya sea si eran los conductores o si oficiaban como pasajeros, mientras que alrededor de 50 personas perdieron la vida en 119 accidentes. Según lo anterior, en el presente estudio se trata este tema con el propósito de desarrollar un sistema de detección de somnolencia en conductores de automóviles empleando técnicas de procesamiento de imágenes y machine learning.

Para llevar acabo el estudio, el trabajo se ha estructurado en 5 capítulos. En el capitulo 1 "Análisis de la base de datos MRL Eye Datase" consta de un conjunto de datos a gran escala de imágenes del ojo humano adecuadas para probar varias características o clasificadores entrenables. En el capitulo 2 "Ajuste de las imágenes a procesar" se realiza el pre-procesamiento de las imágenes donde se analiza si es necesario algún ajuste a nivel de resolución, despliegue, ruido, etc. En el capitulo 3 "Aplicación de las técnicas de machine

learning" una vez se tienen las imágenes ya adecuadas para el reconocimiento se aplican las técnicas de machine learning; creando, compilando, entrenando y evaluando el modelo para el cumplimiento de los objetivos. En el capitulo 4 " Resultados y comparación de los mismos para cada técnica de machine learning" se realiza la comparación de las técnicas empleadas y se valora cuál de ellas presentó mejor rendimiento. En el capitulo 5 "Producto deseado " se implementa una interfaz gráfica amigable con el usuario.

1 Planteamiento del problema

La somnolencia en la conducción es un estado de fatiga y agotamiento que se genera por la falta de sueño, cansancio, ingesta de alcohol el día anterior, comer en exceso antes de conducir o por el exceso de trabajo. esta y otras causas hacen que el cuerpo pierda capacidades de reaccionar; lo que puede causar una tragedia o accidente de tránsito.

Según Castro, Rosales-Mayor, y Egoavil (2009) Un conductor fatigado o somnoliente disminuye progresivamente su capacidad de atención y concentración durante el manejo y pierde capacidad de respuesta cuando circula por carretera.

Un censo de la NHTSA, estima que en 2017, 91,000 de los choques informados por la policía involucraron a conductores somnolientos. Estos choques llevaron un estimado de 50,000 lesiones y casi 800 muertes. *Drowsy Driving | NHTSA* (s.f.)

En Colombia, según el observatorio nacional de seguridad vial, en el 2019, 184 personas terminaron con lesiones como consecuencia de los micro sueños ya sea si eran los conductores o si oficiaban como pasajeros, mientras que alrededor de 50 personas perdieron la vida en 119 accidentes. Javier (2019)

Debido a esta problemática está la necesidad de implementar un sistema de detección de somnolencia en conductores a partir del uso de técnicas de machine learning con el fin de que se pueda prevenir o haya una reducción de accidentes de tránsito.

1.1 Antecedentes

1.1.1 Internacionales

1. Ángel y Galindo (2019), “Algoritmos de detección de objetos para la detección y seguimiento de ojos” Sergioguillen.Com, no. 710, pp. 1–90, 2019
   efectuado en la “UNIVERSITAT POLITECNICA DE CATALUYA” donde El objetivo de este trabajo fue diseñar una solución de software capaz de identificar mediante el análisis de imágenes indicios de fatiga en el conductor.
   La mayoría de las técnicas de procesado de imágenes implican tratar la imagen como

una señal bidimensional y aplicarle técnicas de procesamiento de señal estándar, es por esto que hacen uso de la herramienta de software Matlab y específicamente su librería Image Processing Toolbox. En el capítulo 1 Algoritmos de detección de objetos se describe detalladamente el algoritmo de Viola-Jones y la transformada de Hough para la detección y rastreo de los ojos. En el capítulo 2 se hace la implementación en Matlab de estos 2 algoritmos en la cual se explica el desarrollo e implementación de los mismos y a través de una solución de software se pone en evidencia los aspectos positivos y negativos de cada uno, con ejemplos prácticos dentro de un entorno de prueba.

En cuanto a los resultados para los algoritmos (Viola-Jones + Transformada de Hough), de 450 imágenes, en 37 no se detectaron los ojos y en 16 no se detectó la cara, esto principalmente debido a que, si el conductor o persona a detectar estaba muy alejado de la cámara, el algoritmo no fue capaz de encontrar el objeto buscado.

A manera de conclusión, se señala que se ha dado con 2 soluciones de software que cumplen con los objetivos dentro de un escenario controlado. No obstante, los resultados arrojados fuera de dicho escenario no son muy prometedores. La representación de imágenes es un proceso muy costoso incluso mayor que el de detectar algún objeto, hecho que penalizó el rendimiento de la aplicación.

2. Fernández Villán, Fernández, y Casado (2017) “Sistema Automático Para la Detección de Distracción y Somnolencia en Conductores por Medio de Características Visuales Robustas” vol. 14, pp. 307–328, 2017.

   Llevado a cabo en Asturias, España. El objetivo de este artículo fue proponer, construir y validar una arquitectura especialmente diseñada para operar en entornos vehiculares basada en el análisis de características visuales mediante el empleo de técnicas de visión por computador y aprendizaje automático.

   Las pruebas en entornos reales las realizaron durante varios días recogiendo condiciones lumínicas muy diferentes durante las horas diurnas involucrando a 16

conductores. Los principales signos de somnolencia aparecen en los ojos, implementan algoritmo para el cálculo de perclos por ser el indicador visual más adecuado. Para la detección de la cara aplicaron un detector robusto de características faciales basado en "Modelos de partes Deformables" (DPM), una vez la región facial fue procesada y normalizada, aplicaron un framework que fue diseñado para la normalización de imágenes ante condiciones lumínicas adversas. Posteriormente aplicaron el operador LBP y el operador Center-Symmetric Local Binary Pattern.

En cuanto a los resultados menciona, para el cálculo del estado de los ojos, se hizo uso de la base de datos CEW. Los métodos basados en proyecciones, obtienen una tasa de reconocimiento del 70.1 %, para la aproximación basada tanto en LBP como en CS-LBP, obtienen unas tasas de acierto del 93.39 % y del 91.84 % respectivamente. En cuanto a los resultados con la detección facial, las pruebas arrojaron que la posición de la cámara juega un papel fundamental en la detección facial y por lo tanto en la detección posterior de la somnolencia.

A manera de conclusión manifiestan que, los resultados obtenidos tras la ejecución de las pruebas en entorno real controlado, deducen que detecta con bastante precisión los eventos de distracción y somnolencia, obteniendo los resultados más elevados para la detección del estado del ojo.

3. Emerson y Raúl (2018) "Sistema De Detección De Somnolencia Mediante Inteligencia Artificial En Conductores De Vehículos Para Alertar La Ocurrencia De Accidentes De Tránsito," Repos. Inst. - UNH, p. 80, 2019.

   Se efectuó en Huancavelica, Perú. El proyecto se desarrolló para un conductor de vehículo de transporte interprovincial, tramo lima hacia Huancayo (Perú). El sistema se desarrolló utilizando C# con emguCV para la detección de ojos abiertos o cerrados, esto mediante técnicas de visión artificial mientras que una red neuronal artificial para entrenar los eventos de ojos abiertos, cerrados y distracción.

En cuanto a los resultados indica, que el trabajo desarrollado e implementado presenta las bases suficientes para una operación eficiente, sin embargo, este sistema podría migrar a la tarjeta de raspberry PI 3.

En cuanto a las conclusiones manifiesta que, clasificar los ojos como "abiertos" o "cerrados" es posible solamente cuando se tiene una apertura de párpados amplia.

1.1.2 Nacionales

1. El trabajo de investigación, "fatiga y somnolencia en conductores Por: Villa y Castro (2020) Facultad de Ingenierías Ingeniería de Sistemas y Computación Noviembre del 2020," 2020.

   El objetivo fue Investigar y analizar algoritmos de reconocimiento de imágenes y vídeo para reconocer gestos faciales de fatiga y somnolencia en conductores.

   En cuanto a las conclusiones manifiestan que, el algoritmo de Viola-Jones compuesto es bueno en condiciones favorables, ya que en estas condiciones el umbral de error es bajo.

2. "Desarrollo de un sistema inteligente de detección de fatiga en conductores desarrollado por: Paterna (2019) Presentado en la Universidad Politécnica de Cartagena, en este proyecto utiliza técnicas no invasivas con las que obtuvo distintos parámetros del conductor, los parámetros que tiene en cuenta son:

   - Apertura de los ojos o PERCLOS
   - Detección de bostezos

   Lo siguiente fue calcular la estimación del estado de fatiga del conductor en función de las medidas anteriores. La imagen tomada como entrada en el sistema pasó por distintos subsistemas los cuales era:

   - Detección de rostro
   - Detección de ojos

- Seguimiento del ojo
- Estado del ojo detección de la boca
- Estado de la boca
- Estado de fatiga

Acerca de las conclusiones manifiesta que, se cumplieron los objetivos marcados, pero sin embargo quedó como tarea pendiente entrenar con situaciones reales este detector y así poder ajustar los umbrales que marcan el funcionamiento del sistema.

3. En el trabajo de investigación para obtener el título para ingeniero electrónico, Piazuelo (2017), "DETECCIÓN DE SUEÑO EN CONDUCTORES DE AUTOMÓVILES POR RECONOCIMIENTO DE IMAGEN," 2017.
   El objetivo fue diseñar e implementar un sistema de alarma para detección de sueño en conductores de automóviles, por medio de reconocimiento de imágenes y métodos de aprendizaje supervisado.
   El proceso de diseño fue realizado siguiendo las siguientes etapas: adquisición de datos, selección de la tecnología, diseño del primer aprendiz (detector de rostros), diseño del segundo aprendiz (detector de ojos) y diseño del tercer aprendiz (clasificador de ojos).
   El sistema consistía en una cascada de redes neuronales convolucionales que permitían que el sistema detectara la cara del conductor, encontrar la ubicación de los ojos en la cara detectada y, finalmente, clasificar el estado de los ojos como abiertos o cerrados.

1.1.3 Regionales

1. En el trabajo de investigación para optar por el título de ingeniero electrónico, BAYONA ACOSTA (2007) "IMPLEMENTACIÓN SOFTWARE DE ALGORITMOS PARA LA DETECCIÓN DE RASGOS FACIALES UTILIZADOS

COMO INDICADORES DE SOMNOLENCIA EN CONDUCTORES USANDO SECUENCIAS DE IMÁGENES BIOMÉTRICAS", Universidad Industrial de Santander, Facultad de ingenierías físico-mecánicas, Bucaramanga. Implementa un algoritmo que permite, a partir de secuencias de imágenes biométricas de conductores, determinar en tiempo real cambios en las aberturas de los ojos, los cuales pueden ser utilizados para predecir estados de cansancio y somnolencia. El algoritmo constaba de 3 etapas:

a) La primera etapa realizaba una segmentación del rostro

b) En la segunda etapa se detectaba la localización del iris de uno de los ojos para el caso de personas con los ojos abiertos y se extraía una sub-región de este.

c) En la tercera etapa utilizaba el coeficiente de correlación y una imagen patrón.

En cuanto a conclusiones manifiesta que, con conductores con anteojos, durante la segmentación del rostro el marco de los anteojos elimina la conectividad entre la parte superior e inferior del rostro por lo tanto se segmentan partes del rostro en las que no están incluidos los ojos.

2 Justificación

Los conductores de vehículos no se encuentran ajenos a situaciones adversas generadas como consecuencia de un estado de fatiga y agotamiento; por el contrario, estos se ven involucrados en gran medida ya que al estar fatigados o somnolientes su capacidad de atención y concentración disminuye; lo que puede causar un accidente de tránsito.

En un mundo en el que la tecnología está cada vez más presente, resulta relevante conocer la relación que existe entre sistemas que usan tecnología, inteligencia artificial y problemas del mundo real; Pues esto permite que se puedan diseñar e implementar sistemas avanzados para abordar problemáticas como la somnolencia y los micro sueños en conductores y de esta forma procurar que se eviten tragedias o accidentes de tránsito. Conocer cómo la tecnología y la inteligencia artificial influyen en esta relación puede suponer grandes cambios en el sector de la conducción en aras de que los conductores puedan realizar viajes sin problemas como los mencionados anteriormente y que con el paso del tiempo se traduzcan en beneficios tanto para los conductores como para muchos sectores como lo son el de la educación, salud, investigación, etc.

3 Objetivos

3.1 Objetivo general

Desarrollar un sistema de detección de somnolencia en conductores de automóviles empleando técnicas de procesamiento de imágenes y machine learning

3.2 Objetivos específicos

- Analizar la base de datos a emplear en el presente proyecto; la denominada MRL EYE Dataset
- Aplicar técnicas de pre-procesamiento y machine learning para detectar y clasificar el grado de somnolencia del conductor
- Validar el sistema implementado con un número considerable de imágenes comparando los resultados obtenidos por cada una de las técnicas de machine learning para determinar cuál es la de mejor rendimiento

4 Marco teórico

4.1 ¿Qué es una imagen?

Una imagen es el resultado de la combinación de tres factores que intervienen en el proceso de captura de la imagen

- El color de la luz
- El material de la superficie de los objetos
- La sensibilidad de la cámara

Cuando se captura una imagen se obtiene una matriz de puntos, estos son conocidos como los píxeles de la imagen.

Si se amplia cierta zona de una imagen se puede ver que el color en cada uno de los píxeles de la imagen tiene la combinación de 3 valores, que corresponden al canal rojo, al canal verde y al canal azul (RGB). Esto quiere decir que cada píxel está dado por 3 valores numéricos que están en el rango de 0 a 255.

4.1.1 Imagen Digital

Una imagen digital está definida como una matriz rectangular de números, se compone de una cuadricula rectangular de bloques llamados píxeles. Está compuesta de diferentes tonos de gris. Las imágenes digitales tienen valores de intensidad entre 0 y 255, por lo que se necesitan 256 valores de intensidad para representar una imagen. Los tonos de gris más oscuros tienen valores más bajos (0), mientras que los tonos de gris más claros tienen valores más altos (255).

4.2 Clasificación de imágenes

La clasificación de imágenes es el proceso de tomar una imagen o fotografía y hacer que una computadora la clasifique automáticamente o proporcionando la probabilidad de la clase de imagen.

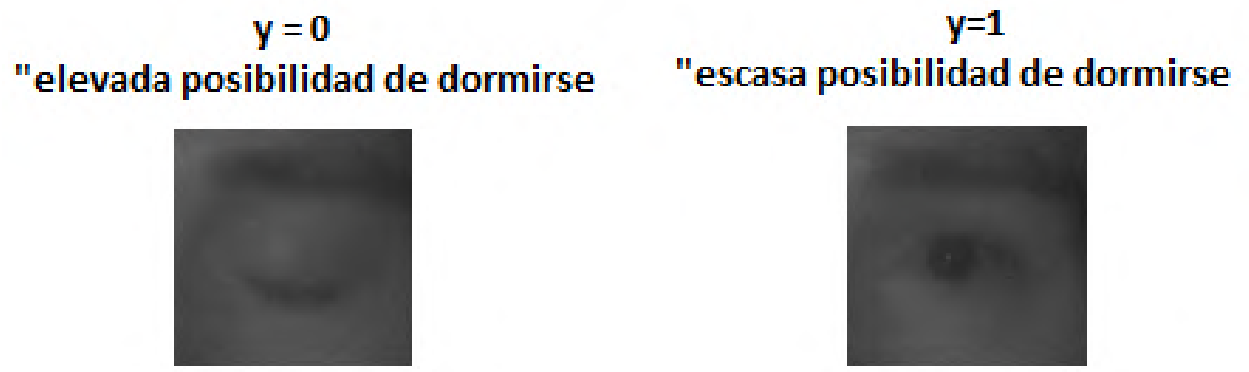
Para ello se basa en la extracción de ciertas características comunes en una categoría de imágenes. ”Para realizar de forma completa una clasificación de imágenes se siguen los siguientes pasos: adquisición y procesado, extracción de características, bloque de entrenamiento, etapa de clasificación y evaluación” Megías y Serrano (s.f.).

4.2.1 Etiquetas

Una clase es esencialmente una etiqueta, por lo general, comenzamos con un subconjunto de categorías o clases, por ejemplo ”alta probabilidad de dormirse” y ”escasa posibilidad de dormirse”

Figura 1

Etiqueta para una imagen



Usamos la etiqueta y para representar las clases a la que pertenece una imagen, $y = 0$ para elevada posibilidad de dormirse, $y = 1$ para escasa posibilidad de dormirse.

Figura 2

Valores de intensidad de una imagen

fuente: Dobernack (s.f.)

Las computadoras no pueden comprender las imágenes, pero pueden comprender los valores de intensidad de una imagen digital. Se usan los valores de intensidad para clasificar una imagen, estas intensidades también pueden ser en escala de grises, dejamos que P represente la imagen que queremos clasificar, todas las imágenes deben tener el mismo numero de filas y columnas.

4.2.2 Esquema general de la clasificación de imágenes

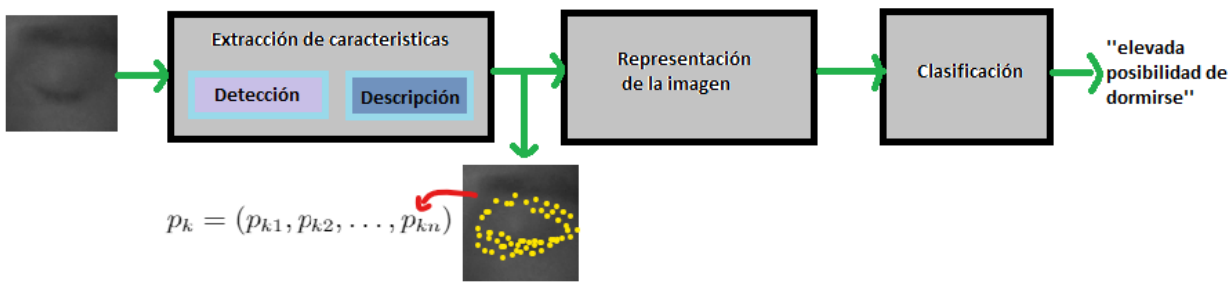
En la figura 3 tenemos un esquema general de clasificación de imágenes, la primera parte del esquema que hace referencia a la extracción de características relevantes de la imagen.

4.2.3 Detección de puntos de interés de una imagen

Los puntos de interés son aquellos puntos relevantes a partir de los cuales se puede describir el contenido de la imagen. En la clasificación de imágenes, el detector SIFT (Scale Invariant Feacture Transform) es uno de los mas utilizados para encontrar puntos de interés. Según Lowe (1999) este enfoque transforma una imagen en una gran colección

Figura 3

Esquema general de la clasificación de imágenes



de vectores de características locales, cada uno de los cuales es invariante a la traslación, escala y rotación de la imagen, y parcialmente invariante a cambios de iluminación y proyección afín o 3D.

Las características invariantes de escala se identifican de manera eficiente mediante el uso de un enfoque de filtrado por etapas. La primera etapa identifica ubicaciones clave en el espacio de escala buscando ubicaciones que sean máximas o mínimas de una función de diferencia de Gauss. Cada punto se utiliza para generar un vector de características que describe la región de la imagen local muestreada en relación con su marco de coordenadas del espacio de escala. Las características logran una invariancia parcial a las variaciones locales, como proyecciones afines o 3D, al difuminar las ubicaciones de degradado de la imagen. Este enfoque se basa en un modelo del comportamiento de células complejas en la corteza cerebral de la visión de los mamíferos. Los vectores de características resultantes se denominan teclas SIFT. En la implementación actual, cada imagen genera del orden de 1000 teclas SIFT, un proceso que requiere menos de 1 segundo de tiempo de cálculo. Lowe (1999)

Con el detector SIFT al aplicar una serie de filtros Gaussianos sobre una imagen teniendo en cuenta varias escalas y resoluciones, se detectan puntos de interés.

4.2.4 Filtro Gaussiano

Cuando se aplica un filtro a una imagen lo que se obtiene es que el valor de cada píxel en la nueva imagen filtrada, esta proviene de la combinación de sus vecinos ponderados según el valor que devuelve la función Gaussiana mostrada en la ecuación 1, esta función dependerá de la distancia al píxel original, esto provoca que los píxeles vecinos vayan perdiendo peso a medida que se aleja del píxel de referencia o píxel central.

$$g(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y2}{2\sigma^2}} \tag{1}$$

El valor de la desviación estándar (sigma de la función Gaussiana) controla cómo la imagen va obteniendo la suavización, si la desviación estándar es mayor, los valores en la función Gaussiana disminuyen, esto provoca que los píxeles mas lejanos obtengan mas peso, esto provoca que en las imágenes se obtenga un mayor suavizado y también una mayor perdida de detalles. En la diferencia de detalles entre 2 imágenes que se hayan suavizado con una diferente desviación estándar, es en lo que se basa la detección de puntos de interés en el SIFT.

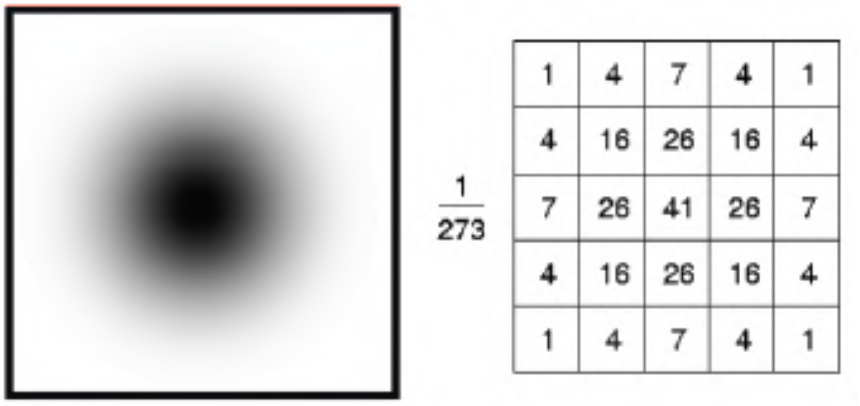
Los píxeles más cercanos al píxel actual tienen más peso que los exteriores. Los pesos de los píxeles se calculan con una campana de Gauss dependiente de la distancia al píxel actual, y su varianza indica el nivel de suavizado. Los filtros Gaussianos también mejoran la capacidad de suavizado, introduciendo un parámetro (la varianza) que es independiente del tamaño de la máscara. Megías y Serrano (s.f.)

4.2.5 Descripción de características locales

El descriptor se basa en el gradiente de la imagen. El **gradiente** es el cambio direccional en la intensidad de la imagen, este gradiente se compone por la dirección donde se presenta el máximo cambio de intensidad y por la magnitud de dicho cambio en la dirección donde se presenta la mayor variación. El gradiente de una imagen se calcula a partir de la diferencia de la intensidad de los píxeles en la dirección horizontal y en la

Figura 4

Mascara de un filtro Gaussiano $5x5$ *con varianza 1*



fuente: Megías y Serrano (s.f.)

dirección vertical como se indica en las ecuaciones 2 y 3

$$dx = I(x+1, y) - I(x-1, y) \tag{2}$$

$$dy = I(x, y+1) - I(x, y-1) \tag{3}$$

A partir de lo mencionado anteriormente, se puede construir u obtener un histograma de orientaciones del gradiente para una zona de la imagen, con ello se puede obtener el descriptor final combinando histogramas de orientaciones del gradiente con una interpolación lineal.

4.26 Representación de la imagen

Para clasificar imágenes que pertenezcan a una misma clase, se busca que la representación de cada imagen sean lo mas similar posible. Para ello se necesita que en imágenes que pertenezcan a la misma clase los puntos de interés detectados sean los mismos y en las mismas zonas. Esta correspondencia entre los puntos de interés detectados en las imágenes diferentes es la que se usa para poder representar las imágenes. Para calcular la distancia entre 2 imágenes, primero se busca cuales son los puntos de interés de

las imágenes que tienen correspondencia, para ello se toma cada punto de una imagen con el fin de encontrar el punto mas parecido entre todos los puntos de interés (punto que tiene la distancia mínima) en una segunda imagen. Esta distancia mínima se puede calcular con la ecuación 4

$$d(H_i^1, I_2) = \min_{j=1,\ldots,n_2} d(H_i^1, H_j^2) \tag{4}$$

Una vez se encuentren para cada uno de los puntos las distancias entre la primera y la segunda imagen y por medio de la media de todos los puntos entre estas dos imágenes se puede calcular la distancia de la primera a la segunda imagen por medio de la ecuación 5

$$d'(I_1, I_2) = \frac{1}{n_1} \sum_{i=1}^{n_1} d(H_i^1, I_2) \tag{5}$$

4.3 Algoritmo K-Nearest Neighbor (KNN)

KNN es un clasificador y es la abreviatura de K-vecino más cercano, es un clasificador multi-clase y no lineal perteneciente al aprendizaje supervisado, es decir, que a partir de un conjunto de datos inicial su objetivo será el de clasificar correctamente todas las instancias nuevas. Es uno de los algoritmos de clasificación más simples, el proceso que lleva a cabo se basa en que escoge el numero para k y la distancia. Encuentra el k vecino más cercano de la muestra o imagen que se quiere clasificar y por ultimo asigna la etiqueta de clase por votación mayoritaria.

Según Tapasco, Londoño, y Flórez (2014) el algoritmo KNN asume que todos los ejemplos corresponden a puntos en un espacio p-dimensional R_p los cuales tienen establecida una clase C. Los datos de entrenamiento se representan de la forma presentada en la ecuación 6

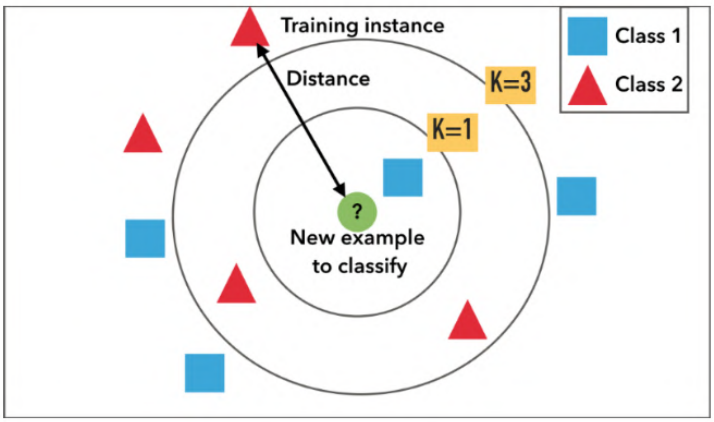
$$(x_i, c_i) = (x_{i1}, x_{i2}, \ldots, x_{ip}, c_i) \tag{6}$$

En la mayoría de los casos la representación de la imagen va a estar conformada por un vector numérico en un espacio determinado. El clasificador busca una frontera capaz de separar de la mejor manera a todas las clases posibles en el espacio de representación.

Una de las ventajas de este algoritmo es que se adapta a nuevos datos de entrenamiento ya que es un algoritmo basado en la memoria. No se requiere de un proceso de aprendizaje como tal sino que solamente se necesita seleccionar un buen conjunto de datos de entrenamiento, esto hace que este método sea bastante simple de utilizar pero siempre va a depender en gran medida en que el conjunto de datos para el aprendizaje sea bueno o que represente de la mejor manera las imágenes que se deseen clasificar. una desventaja que tiene es que a medida que el conjunto de datos de entrenamiento es mayor, el coste computacional que se requiere se incrementa linealmente respecto a los datos de entrenamiento.

Figura 5

K muestras que son más cercanas al punto que se quiere clasificar



fuente: Roman (2019)

Para el funcionamiento de este clasificador se parte de un conjunto de entrenamiento en el cual se tiene un conjunto de muestras para cada una de las clases, estos datos se representan de la forma presentada en la ecuación 6, y se tiene una etiqueta que va a identificar a las clases. ”El algoritmo encuentra las k muestras que son más cercanas al punto que se quiere clasificar, basando sus predicciones en la distancia métrica” Roman (2019).

4.3.1 Distancias

- **Distancia Euclidea:** "Esta es la distancia que se utiliza de manera predeterminada, la distancia entre 2 puntos es la linea recta que los une" Arbeloa (2018).

$$d_e(e_i, e') = \sqrt{\sum_{j=1}^{N} \left(e_i^j - e'^j\right)^2} \tag{7}$$

- **Distancia Manhattan:** La distancia entre dos puntos es la suma de las diferencias absolutas entre sus coordenadas.

$$d_m\left(e_i, e'\right) = \sum_{j=1}^{N} \left|e_i^j - e'^j\right| \tag{8}$$

- **Distancia Minkowski:**

$$d(x, y) = \left[\sum_{i=i}^{n} (x_i - y_i)^p\right]^{\frac{1}{p}} \quad con \quad n \geq 1 \tag{9}$$

Quezada (2017) es el índice de similitud mas usado, esta distancia puede considerarse una generalización de las distancias euclidea y Manhattan. para $p = 1$, la ecuación 9 coincide con la distancia Manhattan, y que para $p = 2$, coincide con la distancia Euclidea.

4.4 Maquinas de soporte vectorial (SVM)

Es un clasificador binario y lineal, la frontera de clasificación es una linea recta en un espacio bidimensional, mientras que si se trata de un espacio multidimensional es un hiperplano.

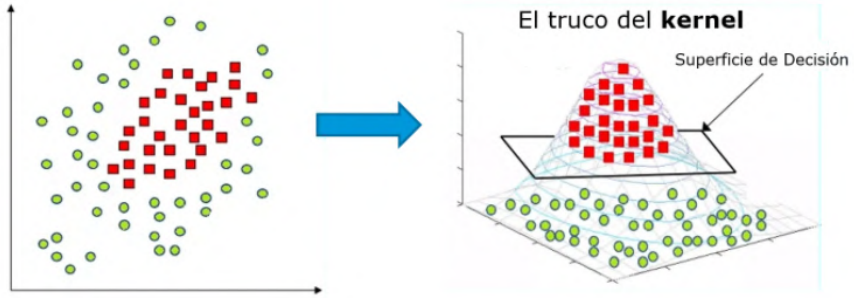
Es un algoritmo de aprendizaje automático supervisado que se puede utilizar para problemas de clasificación o regresión. Pero generalmente se usa para clasificar. Dadas 2 o más clases de datos etiquetadas, actúa como un clasificador discriminativo, definido formalmente por un hiperplano óptimo que separa todas las clases. Los nuevos ejemplos que luego se mapean en ese mismo espacio se pueden clasificar según el lado de la brecha en que se encuentran. Rodriguez (2020)

Según Alfaro (2010) la SVM mapea el conjunto de datos de entrada a un espacio de características de dimensión mayor, es decir, si los puntos de entrada se encuentran en R^2, son mapeados a R^3, posteriormente busca un hiperplano que los separe y que maximice el margen m entre las clases del espacio.

Un hiperplano es una superficie de decisión lineal que divide el espacio en dos partes como se indica en la figura 6. El hiperplano de un espacio n-dimensional es un subconjunto plano con dimensión $n - 1$.

Figura 6

Hiperplanos como decisión de superficies



fuente: Heras (2019)

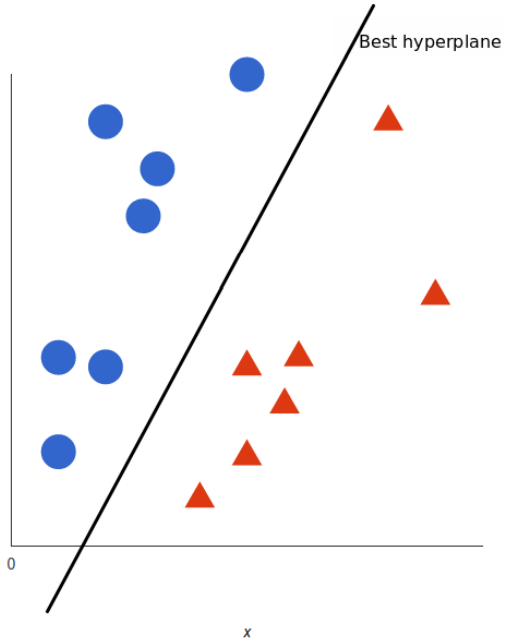
Si se tienen dos clases y se desea definir una frontera lineal entre ellas, no existe una única manera de de generar este hiperplano para la separación. Una máquina de vectores de soporte toma estos puntos de datos y genera el hiperplano (que en dos dimensiones es simplemente una línea) que separa mejor las etiquetas.

Una de las características de las maquinas de soporte vectorial es que la solución se basa en un margen máximo a partir de los vectores de soporte. Para calcular el hiperplano las SVM solo tienen en cuenta un numero limitado de las muestras del conjunto de datos de entrenamiento, a estas muestras se les denomina **vectores de soporte.**

La figura 8 muestra la representación para los vectores de soporte de un conjunto de datos de entrenamiento de dos clases, estos vectores de soporte se eligen de modo que la distancia entre los planos que los contienen (que se conoce como margen) sea máxima.

Figura 7

Ejemplo de datos etiquetados para dos clases



fuente: Rodriguez (2020)

Esta condición de margen máximo conlleva a que se encuentre la región mas amplia del espacio de características que separa a las dos clases y que no tiene ninguna muestra, esa región queda definida por dos hiperplanos (Hiperplano Negativo e Hiperplano Positivo) que son los que contienen a los vectores de soporte y el plano intermedio de esta región es la solución para las maquinas de soporte vectorial.

4.4.1 Caso linealmente separable

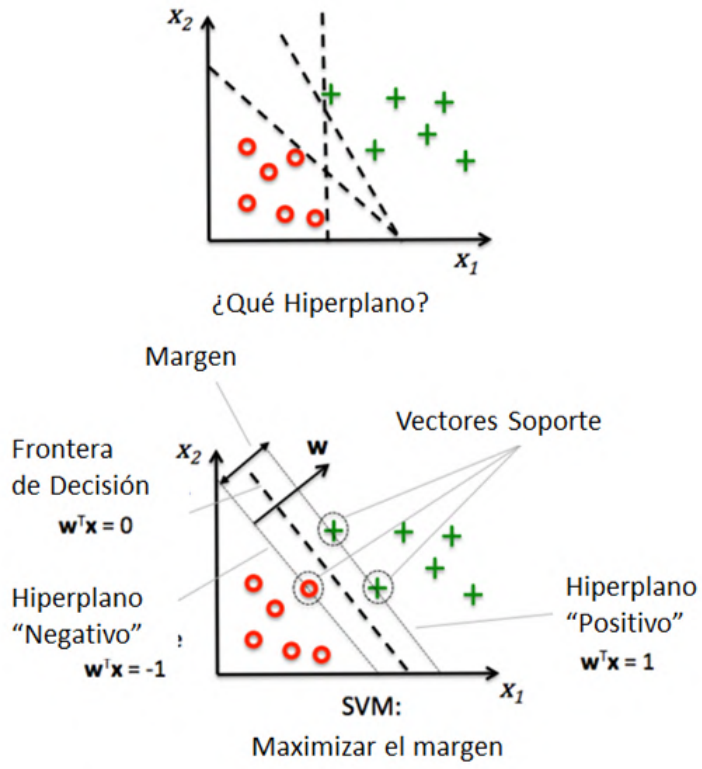
Dado un conjunto separable de ejemplos: $S = (x_1; y_1), \ldots, (x_n; y_n)$, donde $x_i \in R^d$ e $y_i \in \{-1, 1\}$, se define un **hiperplano de separación** como una función lineal que es capaz de separar dicho conjunto sin error. León (s.f.)

$$D(x_i) = (W_1 X_1 + \cdots + W_d X_d) + b = \langle W, X_i \rangle + b, \tag{10}$$

donde $w \in R^d$ y b es un coeficiente real. Las restricciones que debe satisfacer un hiperplano de separación para todo ejemplo del conjunto de entrenamientos son:

Figura 8

Vectores de soporte



fuente: Roman (2019)

$$\langle W, X_i \rangle + b \geq 0, \quad si\ y_i = +1,$$

$$\langle W, X_i \rangle + b \leq 0, \quad si\ y_i = -1,$$

Las ecuaciones mostradas anteriormente hacen referencia a dos hiperplanos, estos hiperplanos se pueden definir como hiperplanos paralelos al hiperplano solución.

Se tienen vectores de entrada $X_i \in R^d$ $(i = 1, 2, \ldots, n)$ correspondientes con las etiquetas $y_i \in \{-1, +1\}$. Existe un plano separador cuya función es $W.X + b = 0$. Acosta (2018)

donde W es el vector ortogonal al hiperplano y b es el coeficiente de intersección, mientras que la distancia del hiperplano al margen está dada por $d = \frac{1}{\|W\|}$.

Dos hiperplanos paralelos pueden ser representados como:

$$y_i(W.X_i + b) \geq 1 \tag{11}$$

Según Acosta (2018) las SVM trata de maximizar la distancia entre dos clases, donde la amplitud del margen entre dos hiperplanos paralelos es $d = \frac{2}{\|W\|}$, por lo tanto cuando se trata de un caso que es linealmente separable se puede encontrar el hiperplano solución a partir de un problema de optimización cuadrática.

$$\begin{aligned} &\text{Minimizar} \phi(W) = \frac{1}{2}\|W\|^2 \\ &\text{sujeto a: } y_i(W.X_i + b) \geq 1 \end{aligned} \tag{12}$$

Para la resolución de problemas como este, por lo general se plantea una función auxiliar que se conoce como el Lagrangiano.

$$L(X, \alpha) = f(x) + \sum_i \alpha g_i(x) \quad \forall \alpha_i \geq 0 \tag{13}$$

donde:

- α son los multiplicadores de Lagrange
- $f(x) \longrightarrow \phi(W)$ Función a optimizar
- $g_i(x) \longrightarrow y_i(W.X_i + b) - 1 \geq 0$ Restricciones

El Lagrangiano se crea a partir de la función $\phi(W)$ y de la condición de clasificación para todas las muestras del conjunto de entrenamiento multiplicado por los multiplicadores de Lagrange y con esto obtenemos:

$$L(W, b, \alpha_i) = \frac{1}{2}(\langle W, W \rangle) - \sum_{i=1}^{n} \alpha_i (y_i(\langle W, X_i \rangle + b) - 1), \quad i = 1, \ldots, n, \tag{14}$$

donde α_i son los multiplicadores de Lagrange. La minimización del Lagrangiano implica que las derivadas parciales con respecto a W y respecto a b, e igualando a cero los

productos por los multiplicadores de Lagrange tenemos:

$$\frac{\partial L(W^*, b^*, \alpha)}{\partial W} \equiv W^* - \sum_{i=1}^{n} \alpha_i y_i = 0, \quad i = 1, \ldots, n, \tag{15}$$

$$\frac{\partial L(W^*, b^*, \alpha)}{\partial b} \equiv \sum_{i=1}^{n} \alpha_i y_i = 0, \quad i = 1, \ldots, n.$$

A partir de la primera derivada parcial respecto a W se obtiene que el valor de W viene dado como una combinación lineal de los vectores de entrenamiento.

De 15 obtenemos la expresión de W^* en términos de los multiplicadores de Lagrange y sus restricciones,

$$W^* = \sum_{i=1}^{n} \alpha_i^* y_i X_i, \quad i = 1, \ldots, n, \tag{16}$$

$$\sum_{i=1}^{n} \alpha_i^* y_i = 0, \quad i = 1, \ldots, n, \tag{17}$$

Haciendo uso de 16 y 17, reemplazando en 14 y resolviendo tenemos,

$$L(W, b, \alpha_i) = \frac{1}{2}\left(\left\langle \sum_{i=1}^{n} \alpha_i^* y_i X_i, \sum_{i=1}^{n} \alpha_i^* y_i X_i \right\rangle\right) - \sum_{i=1}^{n} \alpha_i \left(y_i \left(\left\langle \sum_{i=1}^{n} \alpha_i^* y_i X_i, X_i \right\rangle + b\right) - 1\right),$$

obtenemos la función a maximizar en el problema dual que es el siguiente:

$$\text{Maximizar} \quad L(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1} \alpha_i \alpha_j y_i y_j \left(\langle X_i, x_j \rangle\right) \tag{18}$$

$$\text{Sujeta a:} \quad \sum_{i=1}^{n} \alpha_i y_i \geq 0, \quad \alpha_i \geq 0, \quad i = 1. \ldots, n.$$

Resolviendo las condiciones de Karush-Tucker (KKT), se tiene:

$$\alpha_i [y_i (W.X_i + b) - 1] = 0 \tag{19}$$

si $\alpha_i > 0$, los puntos correspondientes a los datos son los **vectores de soporte**, en donde la solución óptima para el vector normal está dada por la ecuación 16.

Si elegimos de la ecuación 19 cualquier vector de soporte (X_k, Y_k), se obtiene $b^* = Y_k - W^*.X_k$. A continuación (W^*, b^*) es determinado, y a partir de la ecuación 16 el hiperplano separador óptimo puede escribirse como:

$$f(x) = sgn\left(\sum_{i=1}^{N} \alpha_i Y_i (X.X_i) + b^*\right) \tag{20}$$

$$\text{donde sgn es la función signo,} \quad y \;\; x \;\; \in \begin{cases} +1 & si \quad f(x) > 0 \\ -1 & si \quad f(x) < 0 \end{cases}$$

4.4.2 Caso no linealmente separable

Cuando tenemos un conjunto de datos que no son linealmente separables, la solución pasa por usar una función de mapeo $\varphi(X)$ que aumenta la dimensionalidad del espacio de características con el fin de que en este nuevo espacio de características las clases sean linealmente separables. Para ello se define una función kernel $K(X, Z) = \varphi(X)\varphi(Z)$, la cual representa el producto escalar en el espacio mapeado por φ

Algunas de las funciones kernel más usadas son:

- Kernel lineal:

$$K(X, Z) = \langle X, Z \rangle.$$

- Kernel polinomial

$$K(X, Z) = \langle X, Z \rangle^d$$

- Kernel de base radial:

$$K(X, Z) = e^{\|x-z\|\frac{2}{2\sigma}}$$

- Kernel sigmoide:

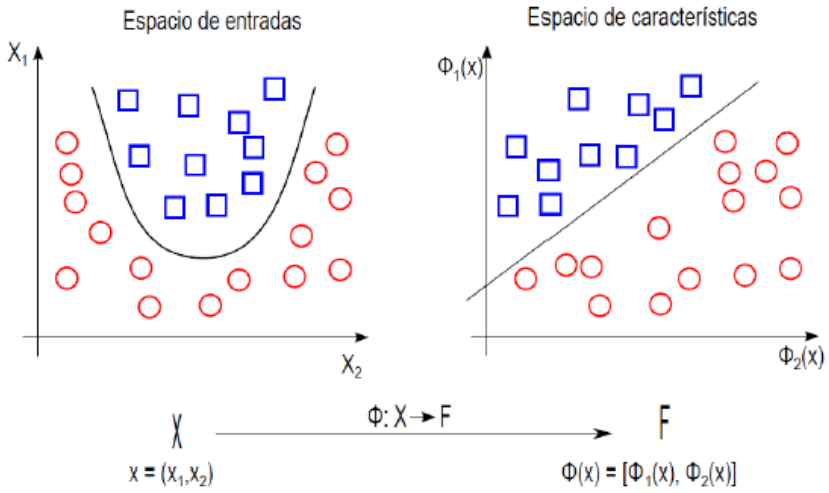
$$K(X, Z) = \tanh\left(k \langle X, Z \rangle - \delta\right)$$

- Kernel Gaussiano:

$$K(X, Z) = \exp\left(-\gamma \|x - z\|^2\right), \quad \gamma > 0$$

Figura 9

Búsqueda de una función de decisión lineal



fuente: León (s.f.)

4.5 Redes neuronales

Las redes neuronales hacen parte del aprendizaje profundo, estas redes neuronales están formadas por capas de neuronas en las que estas neuronas son la unidad central de procesamiento de la red. "Estas neuronas, están interconectadas con neuronas de otras capas mediante un enlace, el cual esta ponderado por unos pesos que multiplican la información que lo atraviesa" Arnal (2018). Generalmente una red neuronal está compuesta por la capa de entrada, las capas ocultas (mono capa si se trata de una sola capa) y la capa de salida que predice la salida de la red.

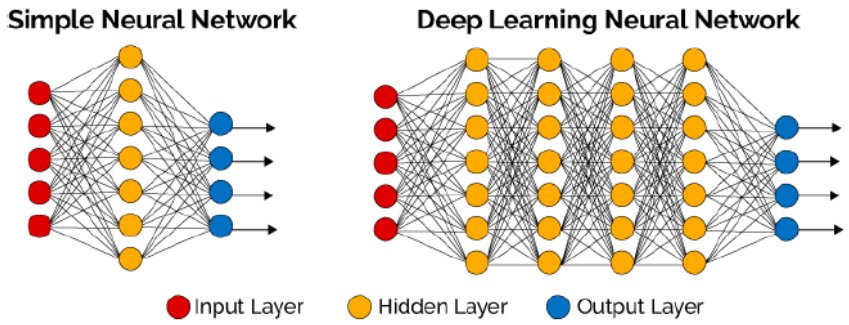
- Capa de entrada:

  Está compuesta por un conjunto de neuronas por las cuales las muestras de un conjunto de datos accede a la red. La entrada a una neurona es un número real multiplicado con un peso y va a indicar qué tan importante es esa entrada en relación al procedimiento o trabajo que se esté realizando.

- Capas ocultas:

Figura 10

Red Neuronal



l

fuente: Arroyo (2019)

Son las capas que reciben la información de las capas de entrada. Según Arnal (2018) las muestras o datos que reciben las neuronas de estas capas son sumadas mediante una función de transferencia en la cual a cada una de las neuronas se le puede aplicar un peso diferente a dicha información que reciben. A esta unión se le aplica una función de activación, que es una de las partes mas importantes de la red, se debe tener en cuenta que la función de activación no sea una función lineal, pues esto perjudicaría notablemente los resultados en el proceso.

- Capa de salida:

  "Es la capa encargada de recopilar la información y proveer una salida dependiendo del tipo de red y la clasificación que se quiera realizar" Arnal (2018). En problemas de clasificación, esta capa de salida puede tener 1 o 2 neuronas si se trata para un problema de clasificación binaria en el cual se tiene 1 o 2 clases como etiquetas de salida, si se trata de un problema de clasificación multi-clase, esta capa de salida puede tener mas de 2 neuronas que serán las encargadas de dar un resultado.

4.5.1 Funciones de activación

A lo largo largo del proceso de aprendizaje de una red neuronal se deben aplicar otros factores como las reglas de activación y desactivación de las neuronas, se trata de

una regla que da el efecto de las entradas en la activación de la unidad. Sánchez (2019) Algunas de las funciones de activación más comunes son las siguientes:

- Función ReLU (Rectified Lineal Unit):

$$y_k^{t+1} = F_k(s_k(t)) = F_k\left(\sum_i w_{jk}^t y_j^t + \theta_k^t\right) \tag{21}$$

  Esta función no tiene en cuenta valores negativos, sino que los toma como cero y para los valores positivos estos quedan igual.

- Función lineal:

$$y_k^{t+1} = a \cdot s_k(t), \quad \text{donde } a \text{ es una constante real} \tag{22}$$

  Esta función también es conocida como función identidad, los valores de entrada son igual a los de la salida, por ende, si se tiene una red neuronal y se aplica esta función de activación, se dice que es una regresión lineal.

- Función sigmoidal:

$$y_k^{t+1} = F(s_k(t)) = \frac{1}{1 + e^{-s_k(t)}} \tag{23}$$

  Esta función de activación también es conocida como función logística, comprende un rango de valores de salida entre 0 y 1. Si el valor de entrada es negativo, la función se hace igual a cero, si evaluamos la función en 0, el valor de salida es igual a 0.5, mientras que para los valores grandes o mayores a 1, el valor de salida es aproximado a 1. Con lo mencionado anteriormente, esta función de activación es usada en la capa de salida para cuando se tiene un problema de clasificación binaria, pues puede clasificar datos en dos categorías.

4.6 Redes neuronales convolucionales

Una red neuronal convolucional o CNN es un tipo de arquitectura de red neuronal profunda diseñada para tareas especificas como la clasificación de imágenes. Estas,

proporcionan algunas características muy interesantes que son útiles para procesar ciertos tipos de datos como imágenes, audio y vídeo con una CNN totalmente conectada.

Según Arroyo (2019) las redes neuronales convolucionales son full connected, lo anterior significa que todas las neuronas correspondientes a una capa oculta están conectadas a todas las neuronas de la capa que les precede. Esto significa que cada una de las muestras o características de un conjunto de datos de entrada están conectadas a cada una de las neuronas de la capa siguiente, pero si se trata de que la entrada es una imagen, cada píxel de la imagen estará conectado con cada píxel de la primera capa oculta. Lo anterior conduce a un problema que es que cuando se trabaja con imágenes se suele trabajar con imágenes de alta resolución, esto produce un problema y es que por la calidad de los pesos que se debería manejar se hace inviable el empleo de redes neuronales fully connected. A raíz de lo anterior, surgieron las redes neuronales convolucionales, estas redes constan de diversas capas de convolución y de poolling que al final terminan con capas convolucionales fully connected.

Una red neuronal convolucional está compuesta por una capa de entrada, una de salida y varias capas ocultas. Las capas ocultas de una CNN típicamente consisten en:

- Capas convolucionales
- Capas de agrupación (Maxpooling)
- Capas completamente conectadas
- Capas de normalización

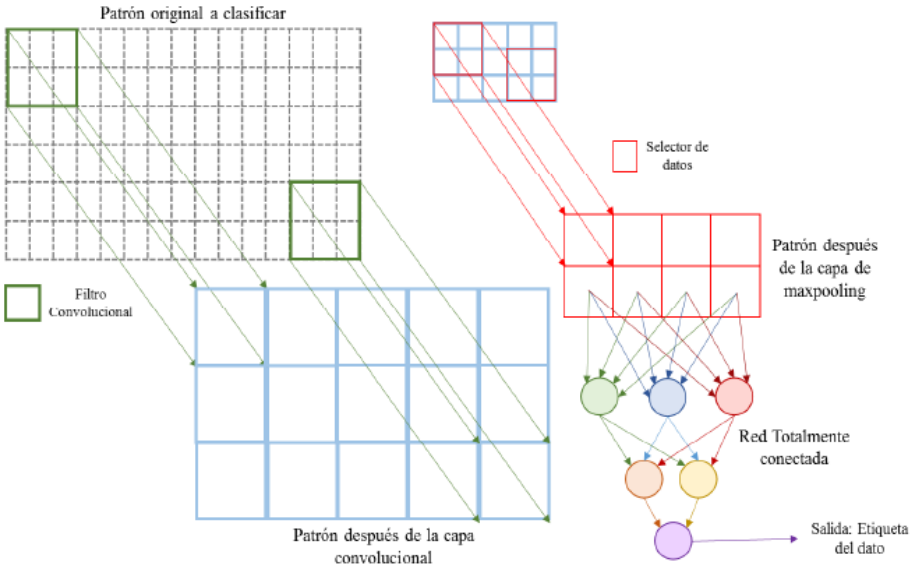
La entrada de una CNN suele ser una matriz bidimensional de neuronas que corresponden a los píxeles de una imagen, por ejemplo si se desea clasificar una imagen.

4.6.1 Capa convolucional

La capa convolucional de las CNN es el factor principal de estas, debido a que permite que se reciba el patrón puro a clasificar y no características extraídas del mismo.

Figura 11

Esquema general de una red neuronal convolucional



fuente: Lizarazo (2017)

Lizarazo (2017) Esta capa nos permite extraer características visuales en fragmentos a partir de una matriz 2D. Cada neurona en una capa de convolución es responsable de un pequeño grupo de neuronas de la capa anterior. El cuadro delimitador que determina el grupo de neuronas se llama filtro, también llamado kernel, este filtro se mueve a través de una imagen y realiza una operación matemática en regiones individuales de la imagen y luego envía su resultado a la neurona correspondiente de la capa de convolución.

Matemáticamente, una convolución de dos funciones se define como:

$$s(t) = x(t) * h(t) = \int_{-\infty}^{\infty} x(\tau)h(t-\tau)d\tau \tag{24}$$

La convolución es la operación del cálculo del área bajo la curva del producto entre dos funciones, una de estas rotada en torno al origen como se ve en la ecuación 24, y en función de un tiempo continuo τ.

$$s(t) = x(t) * h(t) = \sum_{\tau=-\infty}^{\infty} x(\tau)h(t-\tau)$$

la convolución se realiza en tiempo discreto y generalmente sobre arreglos

multidimensionales; para el caso de las CNN los arreglos son: segmentos de píxeles de la imagen de entrada $I(m,n)$ y el kernel $K(i-m,j-n)$ que es el parámetro adaptado por el algoritmo de aprendizaje. Lizarazo (2017)

Para el caso en el que las imágenes de entrada estén en blanco y negro (1 canal-bidimensión) la formula de convolución con entrada I y con núcleo K, es:

$$s(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m,j-n), \tag{25}$$

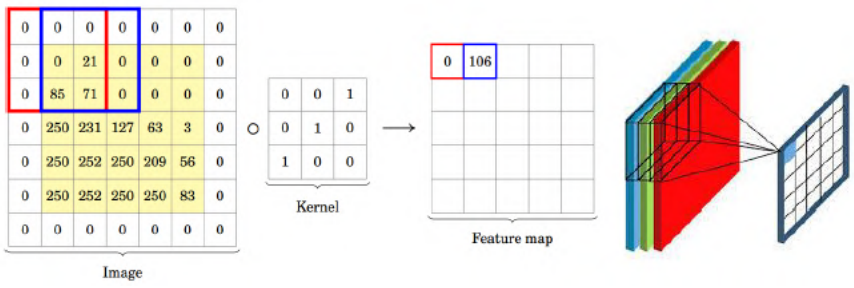
la cual es conmutativa:

$$s(i,j) = \sum_m \sum_n I(i-m,j-n)K(m,n)$$

donde m y n son las dimensiones de la imagen (filas y columnas respectivamente) y las variables i,j, indican el corrimiento del kernel en filas y columnas sobre la imagen. En la figura 12 se muestra un ejemplo de la operación de la convolución. en la parte izquierda de la imagen se muestra una convolución bidimensional sobre una matriz de entrada y a la derecha se muestra el resultado de esta operación sobre volúmenes de dimensión 3.

Figura 12

Operación de convolución



fuente: Arroyo (2019)

Lizarazo (2017) explica que en la operación de convolución en dos dimensiones no se hace la rotación de ninguna de las funciones (Imagen o Kernel) en torno al origen, es por ello que esta operación puede verse como una correlación entre las matrices operadas; esto conlleva a que en las CNN no todos los datos de entrada se conecten a cada una de

las neuronas (kernel) sino que genera un proceso de generación de imágenes nuevas por conjunto de datos aumentando la eficiencia del algoritmo.

4.6.2 Capa de Maxpooling

Es la siguiente capa en una red neuronal convolucional, su objetivo es reducir aún más la cantidad de neuronas necesarias en las capas posteriores de la red y al mismo tiempo conservar la información más importante. La agrupación máxima tiene varios métodos que se pueden usar para la decisión de la generación del nuevo dato, pero el que generalmente se usa se basa en recoger el valor máximo de la región seleccionada.

Dada una matriz A se puede definir el proceso de Maxpooling con una amplitud k y un stride p con la matriz $P(i, j)$ tal que:

$$\max_{n,m=1,\dots,k} A[(i-1)p+m, (j-1)p+n] \tag{26}$$

4.7 Matriz de confusión

Una matriz de confusión es una medida de desempeño para un problema de clasificación. Es una tabla con una combinación de valores predichos y reales. en el eje y tenemos la etiqueta Verdadero y en el eje x tenemos la etiqueta predicción.

Una matriz de confusión, también conocida como matriz de error, es una tabla resumida que se utiliza para evaluar el rendimiento de un modelo de clasificación. El número de predicciones correctas e incorrectas se resumen con los valores de conteo y se desglosan por cada clase. Shin (2020)

Cuadro 1

Ejemplo de una matriz de confusión

	Predicción: NO	Predicción: SI
Verdadero: NO	40	45
Verdadero: SI	18	65

En esta matriz, podemos ver que hay dos clases, por ejemplo si estuviéramos prediciendo que una imagen será ”elevada posibilidad de dormirse” o si por el contrario será ”elevada posibilidad de dormirse”. En este ejemplo tenemos 168 predicciones, en la que el clasificador predijo ”SI” 105 y ”NO” 63 veces, pero en realidad, hubo 83 ”SI” y 85 ”NO”.

Cuando se habla de una matriz de confusión se involucran los siguientes términos:

- Verdadero positivo (TP): nuestro modelo predijo ”SI” y en realidad era ”SI”
- Verdadero negativo (TN): nuestro modelo predijo ”NO” y en realidad fue ”NO”
- Falso positivo (FP): nuestro modelo predijo ”SI” pero en realidad fue ”NO”
- Falso negativo (FN): nuestro modelo predijo ”NO” pero en realidad fue ”SI”

4.8 Cascadas de Haar

”Un clasificador de Haar, o un clasificador en cascada de Haar, es un programa de detección de objetos de aprendizaje automático que identifica objetos en una imagen y un vídeo” Aditya (2020). El algoritmo cuenta con cuatro etapas:

- Calculo de las características de Haar
- Creación de imágenes integrales
- Usando Adaboost
- Implementación de clasificadores en cascada

Como se explica en Aditya (2020), este algoritmo requiere de muchas imágenes positivas de **rostros** e imágenes negativas de **no rostros** para entrenar al clasificador.

4.8.1 Cálculo de las características de Haar

Una característica de Haar se basa en cálculos que se realizan en regiones rectangulares adyacentes en una ubicación especifica en una ventana de detección. Este

cálculo suma las intensidades de píxeles en cada región y calcula las diferencias entre las sumas. Para imágenes grandes, estas características pueden ser difíciles de encontrar, por ello entra en juego las imágenes integrales porque el número de operaciones se reduce utilizando la imagen integral.

4.8.2 Imagen Integral

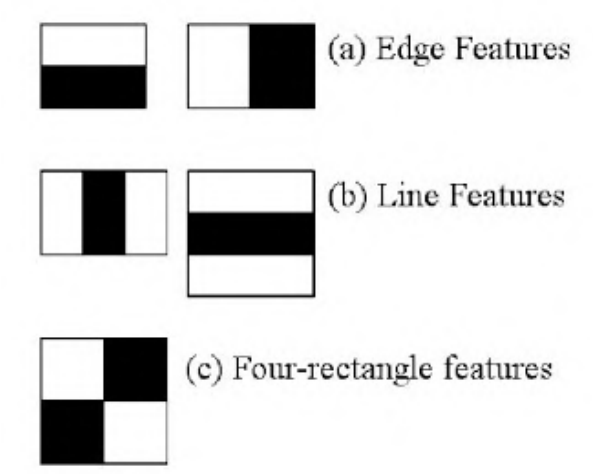
Las características rectangulares se pueden calcular muy rápidamente utilizando una representación intermedia de la imagen que llamamos imagen integral Viola y Jones (2001).

$$ii(x, y) = \sum_{x' \leq x, y \leq y} i(x', y') \tag{27}$$

La imagen integral en la ubicación x, y contiene la suma de los píxeles arriba y a la izquierda de x, y.

Figura 13

Ejemplos de las funciones de Haar

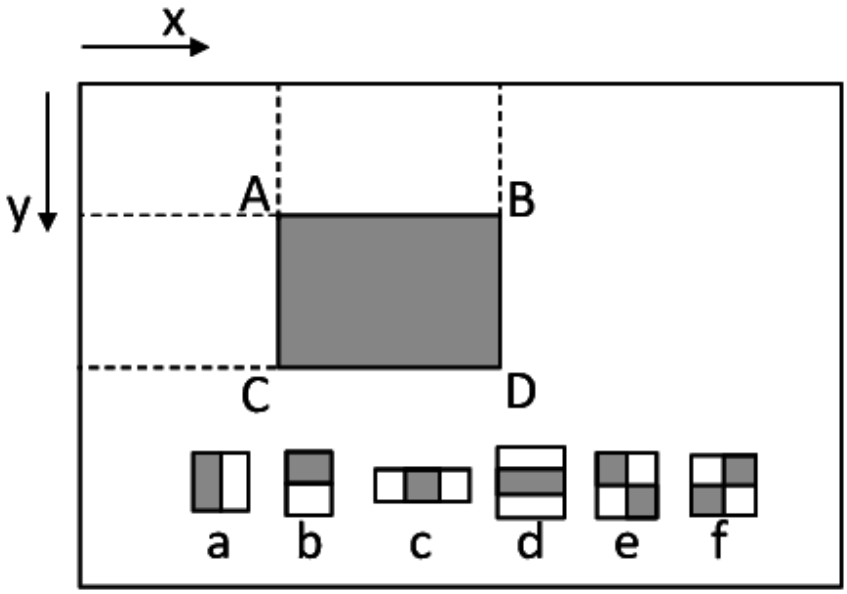


fuente: Aditya (2020)

Las imágenes integrales en lugar de calcular en cada píxel, crea sub-rectangulos y crea referencias de matriz para cada uno de esos sub-rectangulos.

Figura 14

Ilustración de cómo funciona una imagen integral



fuente: Aditya (2020)

4.8.3 Entrenamiento Adaboost

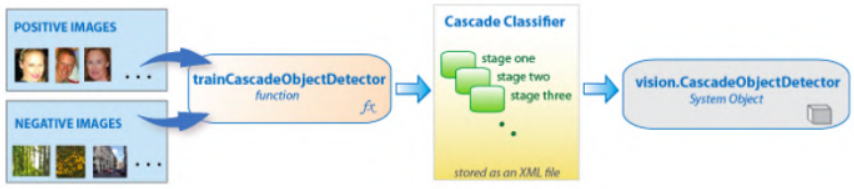
Adaboost esencialmente elige las mejores características y entrena a los clasificadores para que las utilicen. Utiliza una combinación de ”clasificadores débiles” para crear un ”clasificador fuerte” que el algoritmo puede usar para detectar objetos. Aditya (2020)

4.8.3 Implementación de clasificadores en cascada

El clasificador se compone por varias etapas, donde cada etapa es una colección de aprendices débiles, estos se entrenan mediante el refuerzo, lo que permite un clasificador de alta precisión a partir de la predicción de los aprendices débiles. Basado en esta predicción, el clasificador decide indicar que se encontró un objeto (positivo) o pasar a la siguiente región (negativo).

Figura 15

Proceso para crear un clasificador de rostros empleando Haar Cascade



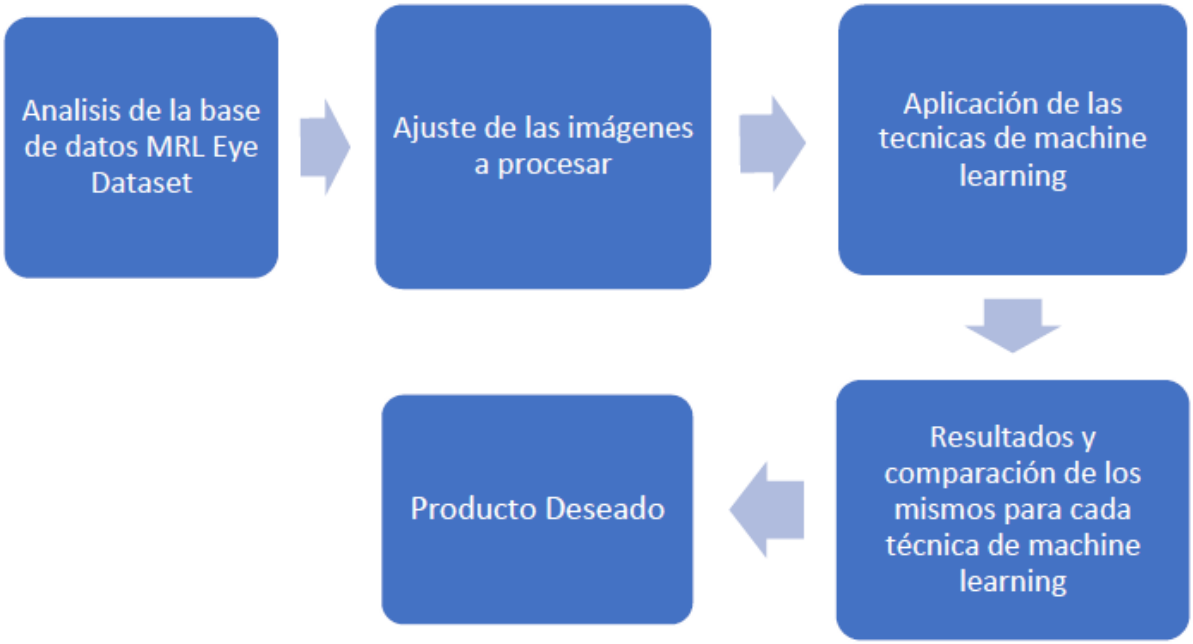
fuente: Aditya (2020)

5 Metodología

El objetivo general propuesto para este proyecto es desarrollar un sistema de detección de somnolencia en conductores de automóviles empleando técnicas de procesamiento de imágenes y machine learning.

Figura 16

Diagrama de bloques de la metodología



A continuación, se explicará el proceso realizado para cada objetivo especifico y el

desarrollo del mismo, con el fin de que al integrar cada una de las soluciones se haya cumplido con el objetivo general.

5.1 Análisis de la base de datos MRL Eye Dataset

La elección del conjunto de datos y la cantidad de los mismos que es necesaria en un proyecto depende en gran parte del problema que se está tratando de investigar o resolver, elegir el conjunto de datos correcto es fundamental, pues si los datos no son los correctos conlleva a que se tenga un modelo incorrecto.

Para este proyecto, era necesario contar con una base de datos, que a partir de incluir una gran cantidad de imágenes, incluyera ciertas propiedades que las describieran. Por ello, tener esta base de datos etiquetada o con las posibilidades de poderla etiquetar, con el fin de que se obtengan todos los datos a clasificar, se pueda determinar lo que cada dato indica y así poder obtener valores reales para posteriormente analizar sus resultados. Para proyectos como este donde se emplean varias técnicas de machine learning para la clasificación de imágenes, es importante contar con una cantidad considerable de datos o imágenes, pues esto ayuda a evitar que al momento del entrenamiento para cada uno de los modelos se llegue a un sobre-ajuste de los datos o un sobre-entrenado del sistema, además de que contar con una gran cantidad de datos o imágenes permite que en los entrenamiento se pueda realizar una validación cruzada en diversas iteraciones. La base de datos para este proyecto es la denominada **MRL Eye Dataset**, que es un conjunto de datos a gran escala de imágenes del ojo humano.

Como se indica en *MRL Eye Dataset | MRL* (s.f.), el conjunto de datos contiene imágenes en baja y alta resolución, todas ellas fueron capturadas en diversas condiciones de iluminación y por diferentes dispositivos. El conjunto de datos se divide en varias categorías, lo que hace que sea adecuado para probar varias características o clasificadores entrenables. Este, contiene anotadas diferentes propiedades y se indican en el siguiente orden:

- **ID del sujeto:** en el conjunto de datos se cuenta con datos de 37 personas diferentes (33 hombres y 4 mujeres)
- **ID de la imagen:** el conjunto de datos consta de alrededor de 84800 imágenes
- **género[0-hombre, 1-mujer]:** contiene la información del género por cada imagen
- **gafas[0-no, 1-si]:** se proporciona para cada imagen información si la imagen del ojo contiene gafas (con y sin gafas)
- **estado de los ojos [0-cerrado, 1-abierto]:** propiedad que indica los dos estados del ojo (abierto, cerrado)
- **reflexiones [0-ninguna, 1-pequeña, 2-grande]:** propiedad para para la reflexión basada en el tamaño de las reflexiones (ninguna, pequeña, grande)
- **condiciones de iluminación[0-malas, 1-buenas]:** Según la cantidad de luz durante la captura de las imágenes, se tiene para cada imagen dos estados (malo, bueno)
- **ID del sensor [01-RealSense, 02-IDS, 03-Aptina]:** el conjunto de datos contiene imágenes que fueron capturadas por tres sensores diferentes (sensor Intel RealSense RS 300 con resolución 640 $\times$ 480, sensor de imágenes IDS con resolución de 1280 $\times$ 1024 y sensor Aptina con resolución 752 $\times$ 480)

Cuadro 2

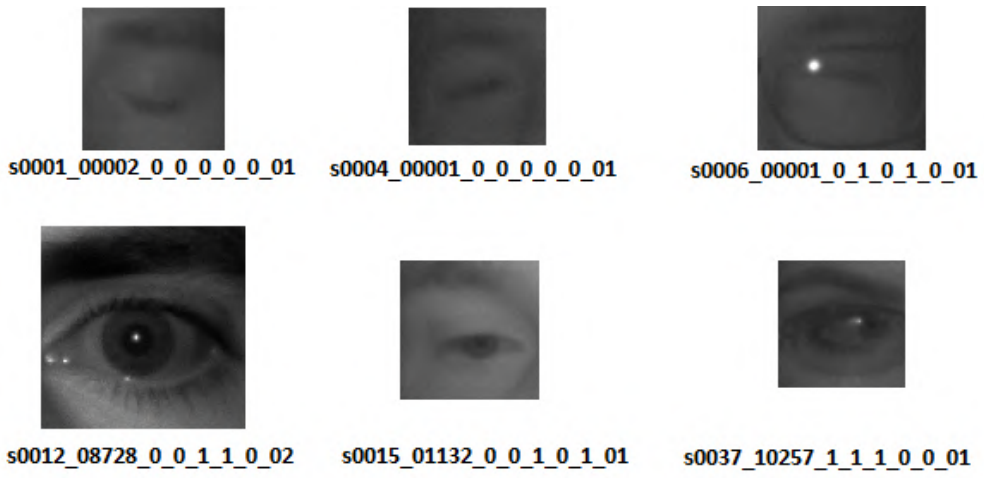
Resumen de las categorías del conjunto de datos

	Total
genero	masculino = 33, femenino =4
gafas	24001 imágenes
ojos abiertos	41945 imágenes
ojos cerrados	42953 imágenes
no reflexión	66060 imágenes
baja reflexión	6129 imágenes
alta reflexión	12709 imágenes
mala iluminación	53630 imágenes
buena iluminación	31268 imágenes

fuente: datos recopilados de *MRL Eye Dataset | MRL* (s.f.)

Figura 17

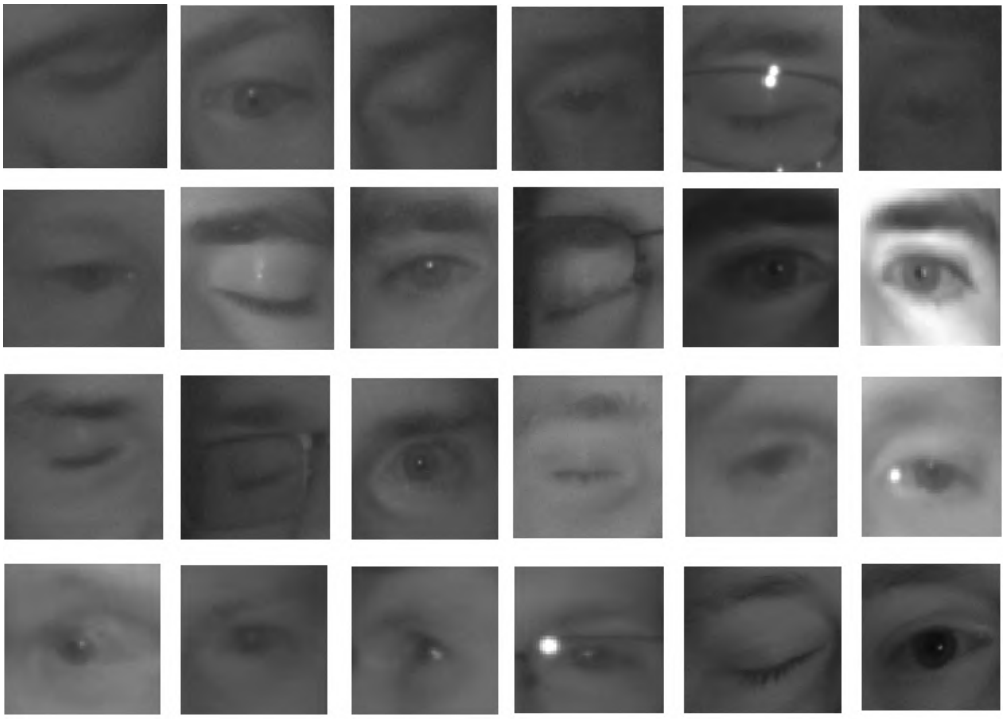
Ejemplo de anotaciones de imágenes del conjunto de datos



Una vez se describen cada una de las propiedades del conjunto de datos, se organizaron las imágenes a utilizar en carpetas, esto con el fin de dividir en conjunto de

Figura 18

Ejemplo de imágenes del conjunto de datos Mrl Eye DataSet

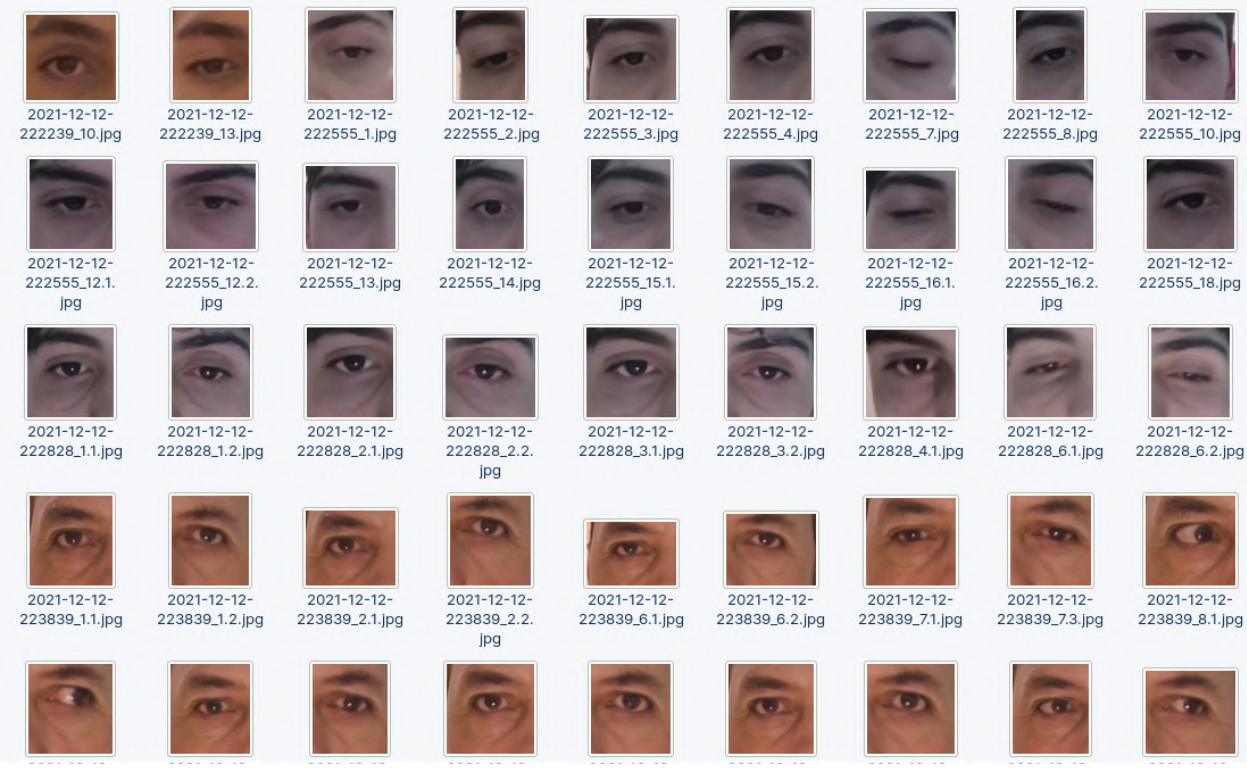


entrenamiento, prueba y validación. Para el conjunto de datos a usar en el entrenamiento y prueba se usaron 5 clases para la posterior clasificación de las imágenes, estas clases son: ”Aparición de Ojeras”, ”Elevada posibilidad de dormirse”, ”Enrojecimiento de los ojos”, ”Escasa posibilidad de dormirse” y ”Mediana posibilidad de dormirse”.

El tamaño de la base de datos usada consta de 2500 imágenes para la clase ”Elevada posibilidad de dormirse”, 2700 imágenes para la clase ”Escasa posibilidad de dormirse”, 326 imágenes para la clase ”Mediana posibilidad de dormirse”, la cantidad de estas imágenes mencionadas anteriormente pertenecen a la base de datos MRL Eye Dataset. Adicionalmente se generó una pequeña base de datos, esta con 100 imágenes para la clase ”Aparición de Ojeras” y alrededor de 20 imágenes para la clase ”Enrojecimiento de los ojos”. Para todas las imágenes mencionadas anteriormente, la longitud es diferente, no todas las imágenes que pertenecen a una clase están dadas con las mismas dimensiones, sino que cada imagen viene determinada por diferentes números de píxeles, por ello más

Figura 19

Ejemplo de imágenes de la clase "Aparición de Ojeras"



adelante se mencionará que dimensiones se establecen para todas las imágenes para cuando se realiza tanto el entrenamiento como las validaciones.

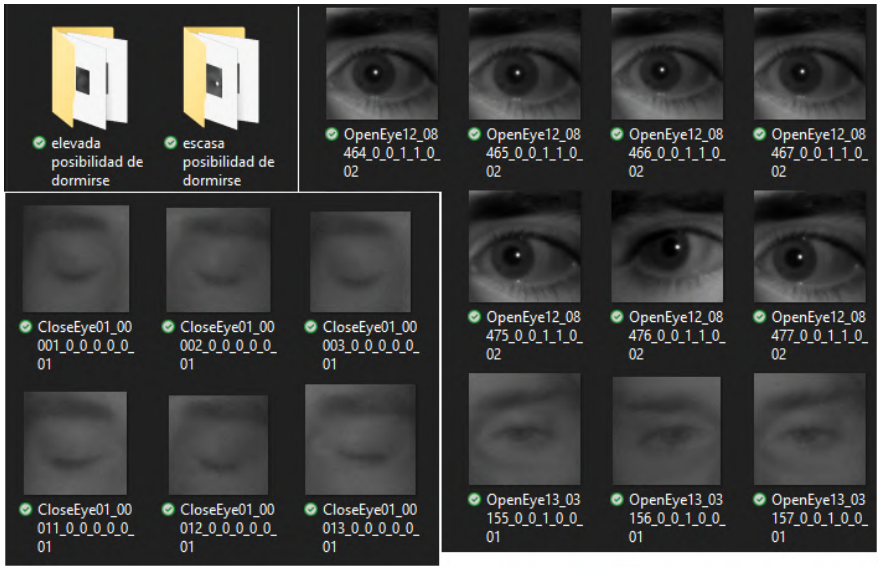
5.2 Ajuste de las imágenes a procesar

Una vez se ha seleccionado el conjunto de datos y se ha dividido en subconjuntos como los datos de entrenamiento, prueba y validación, se siguen algunos pasos de pre-procesamiento para las imágenes. Como se indica en la figura 17 para cada una de las imágenes del conjunto de datos su nombre está determinado por una anotación que es generada a partir de ciertas propiedades que se mencionaron anteriormente, y como se mencionó en el apartado anterior, para realizar el etiquetado de cada una de las imágenes, la etiqueta viene dada por el nombre de la carpeta. Por ello, para todas las imágenes de

cada clase se cambió o se estableció un nombre relacionado con su respectiva clase como se muestra en la figura 20 con el fin de hacer más fácil el etiquetado de las imágenes.

Figura 20

Cambio al nombre de las imágenes para realizar el etiquetado



Para las técnicas de clasificación de imágenes, se parte de cargar todas las imágenes de entrenamiento, estas imágenes son leídas y posteriormente se realiza a cada una de ellas algunos procedimientos entre los que se tienen:

- Una vez se han leído y cargado las imágenes, a cada una de ellas se le cambió el espacio de color a RGB, ya que al usar la biblioteca OpenCV y esta lee las imágenes como BGR, esto debido a que los primeros desarrolladores de OpenCV eligieron el formato BGR porque era el formato que estaba popular en ese entonces entre los fabricantes de cámaras y también de los diferentes proveedores de software.
- Se convierten las imágenes a escala de grises: las imágenes en escala de grises (1 canal), simplifican el algoritmo y reducen los requisitos computacionales.
- Se cambia el tamaño de la imagen: el cambio del tamaño de cada una de las imágenes permite que el algoritmo se entrene más rápido.

- Vectorización de la imagen: aplanar cada una de las imágenes, hace que las imágenes pasen a ser una matriz numerosa (arreglo unidimensional o vector), esto se realiza tanto en el algoritmo KNN como en el SVM, y permite que posteriormente el algoritmo las maneje y las reconozca.

El pre-procesamiento mencionado en los items anteriores se realizó a cada una de las imágenes para así pasar a cada una de las etapas de los algoritmos de machine learning usados para la clasificación de imágenes y de los cuales se va a hablar mas en detalle en la siguiente sección. El ultimo ítem que está relacionado con vectorizar la imagen, no se tuvo en cuenta para el pre-procesamiento de la red neuronal convolucional, pues la CNN para su entrenamiento espera una matriz o arreglo multidimensional.

5.3 Aplicación de las técnicas de machine learning

Como se mencionó anteriormente, el algoritmo KNN se usa como un clasificador que puede calcular la distancia euclidea entre las imágenes de entrada de un conjunto de datos. Haciendo una analogía con una CNN, estos algoritmos funcionan de manera competitiva pero las CNN producen una alta precisión superando a la que se puede obtener con el algoritmo KNN. Del mismo modo, una CNN puede llegar a obtener una mayor precisión que el algoritmo SVM, pero en algunos casos las SVM presentan un mejor rendimiento. Debido a esto se espera que la Red Neuronal Convolución sea la técnica que nos proporcione los mejores resultados, pero teniendo en cuenta de que sea la técnica SVM la que requiera de menor recursos computacionales.

5.3.1 KNN para la clasificación de imágenes

Una vez realizado el pre-procesamiento a todas las imágenes del conjunto de entrenamiento, se prosigue con el etiquetado a cada una de las imágenes, es decir, a cada imagen se le asignó su etiqueta correspondiente. La matriz correspondiente a todas las muestras de las imágenes de entrenamiento se le realizó un ajuste debido a que OpenCV

solo identifica matrices de tipo **float32**, además, para el arreglo correspondiente a las etiquetas de entrenamiento, este se lleva a la forma (tamaño de etiqueta, 1). Lo anterior se realizó especificando astype('float32') en la matriz numpy de las muestras de entrenamiento y para las etiquetas de entrenamiento se convirtieron a enteros y se remodeló la matriz a (tamaño de etiqueta, 1), entonces al imprimir en la consola la matriz de las etiquetas de entrenamiento se especifican de la siguiente forma: [[1], [0], ..., [0]].

```
imag_entrenamiento = np.array(imag_entrenamiento).
                                astype('float32')
etiquetas_entrenamiento = np.array(etiquetas_entrenamiento)
etiquetas_entrenamiento = etiquetas_entrenamiento.astype(int)
etiquetas_entrenamiento = etiquetas_entrenamiento.reshape(
                                    (etiquetas_entrenamiento.size,1))
```

Se dividió el conjunto de datos de entrenamiento y el conjunto de datos de prueba, en el cual se definió para los datos de prueba un 25 % y un 75 % para los datos de entrenamiento, por lo tanto de las 4648 imágenes, 1162 imágenes quedaron asignadas para los datos de prueba mientras que 3486 imágenes para el entrenamiento.

Para el entrenamiento del modelo KNN, de la biblioteca OpenCV se usó:

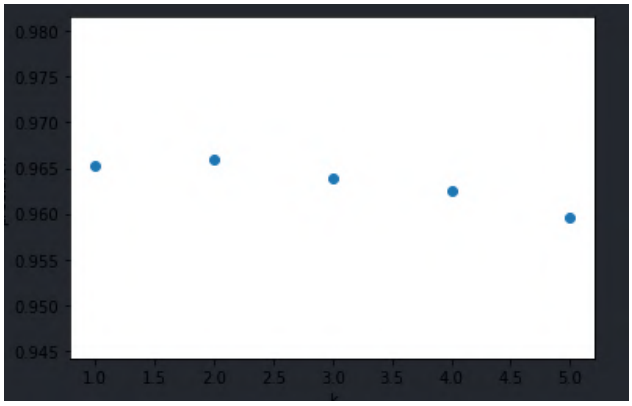
```
cv2.ml.KNearest_create()
```

Para ello se necesitó definir cuántos vecinos más cercanos se utilizarían para la clasificación como hiperparámetro k. Este parámetro k se ajustó con el proceso de validación del modelo o de entrenamiento. En este caso, se definió para varios valores de k (k_valores = [1, 2, 3, 4, 5]) para así encontrar el valor óptimo para el conjunto de datos con el que se contaba. Este valor de k se refiere al número de vecinos más cercanos que se incluyen en la mayoría del proceso de votación.

Como se puede observar en la figura 21 se eligieron 5 valores para el parámetro k, en los cuales para cuando $k = 1$ y $k = 2$ se tiene una mejor precisión en cuanto a el

Figura 21

Precisión respecto al valor de k



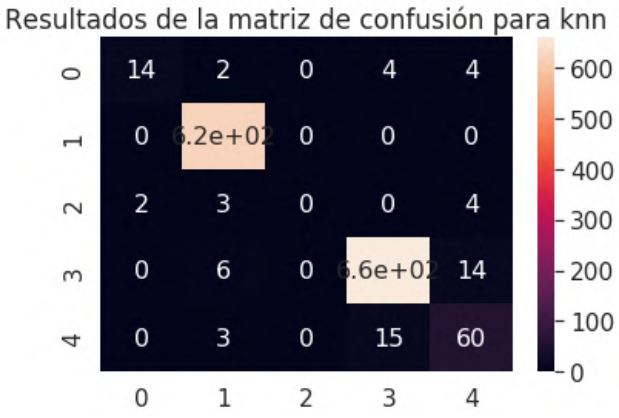
número de vecinos que pueden ser usados para encontrar la distancia mínima y así poder clasificar los datos o muestras que pertenezcan a una misma clase.

Una vez obtenido el valor de k, se realizó una predicción en el modelo para el porcentaje de muestras de prueba y se obtuvo el valor de precisión para el valor de k seleccionado, esto va relacionado con ¿Cuánto por ciento de las imágenes se clasificaron correctamente?

Para ello se hizo uso de una matriz de confusión para una evaluación más completa del modelo de clasificación.

Figura 22

Matriz de confusión para KNN en la predicción del modelo para el conjunto de muestras de prueba



En la figura 22 se muestra el resultado de la matriz de confusión en la predicción realizada sobre el conjunto de datos o muestras de prueba, tenemos 1413 predicciones de las cuales se especificarán a continuación solo alguno de sus resultados:

- verdadero positivo (TP): el modelo predijo ”escasa posibilidad de dormirse”en 661 ocasiones.
- Verdadero negativo (TN): el modelo predijo ”elevada posibilidad de dormirse” en 621 ocasiones.

En la siguiente sección, se hablará sobre la predicción del modelo KNN aplicado a el conjunto de datos o muestras correspondientes a las imágenes de la carpeta **validación**, para ello se cargó el modelo guardado y utilizando el mejor valor de k se cargó y clasificó cada una de las imágenes de validación en el modelo KNN.

5.3.2 SVM para la clasificación de imágenes

Al igual como se mencionó en el apartado anterior, una vez realizado el pre-procesamiento a todas las imágenes del conjunto de entrenamiento, se prosiguió con el etiquetado de cada una de las imágenes, en el cuál a cada imagen se le asigno una etiqueta correspondiente a la clase a la que pertenece. Además tanto a la matriz de las muestras de entrenamiento como a la matriz de las etiquetas de entrenamiento se les hizo el mismo ajuste que se realizó para el algoritmo KNN, pues tanto para las SVM como para KNN, al crear los modelos, estos esperan las matrices o arreglos de la misma forma, siendo estos arreglos de dos dimensiones.

Al igual que para el algoritmo KNN, se dividió el conjunto de datos de entrenamiento y el conjunto de datos de prueba, en el cual se definió para los datos de prueba un 25 % y un 75 % para los datos de entrenamiento, por lo tanto de las 4648 imágenes, 1162 imágenes quedaron asignadas para los datos de prueba mientras que 3486 imágenes para el entrenamiento.

```
tam_prueba = 0.25
muestras_entrenamiento,
muestras_test,
etiquetas_entrenamiento,
etiquetas_prueba = train_test_split(
                                    imag_entrenamiento,
                                    etiquetas_entrenamiento,
                                    test_size=tam_prueba,
                                    random_state=0)

etiquetas_entrenamiento = etiquetas_entrenamiento.ravel()
etiquetas_prueba = etiquetas_prueba.ravel()

modelo = SVC(C=1, kernel='rbf', gamma='scale')
```

Para la creación del modelo SVM, este requiere de varios hyperparámetros, entre estos tenemos:

- C que se comporta como un parámetro de regularización en la SVM. El parámetro C compensa la clasificación correcta de los ejemplos de entrenamiento con la maximizaron del margen de la función de decisión. Para valores mayores de C, se aceptará un margen menor si la función de decisión clasifica mejor todos los puntos de entrenamiento correctamente, mientras que para valores de C bajos, fomentará un margen mayor, esto quiere decir que, una función de decisión más simple a costa de la precisión. Para este modelo el valor se asignó $C = 1$

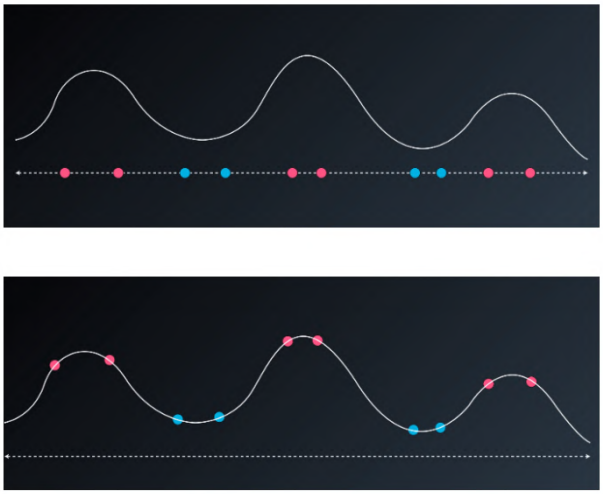
- *kernel* como se mencionó anteriormente, el objetivo principal de un kernel cuando se trata de un caso no linealmente separable, es proyectar las muestras de un conjunto de datos en un espacio con mas dimensiones a partir de una función, de forma que

los datos sean linealmente separables. Después se aplica una función de inversa que provoca la proyección para volver nuevamente a la distribución de las muestras a su versión original.

- *gamma* es un parámetro del kernel RBF y es considerada como la extensión del kernel, y por lo tanto la región de decisión. Los valores bajos significan ”lejos” mientras que los valores altos significan ”cerca”. El comportamiento del modelo SVM es muy sensible al parámetro *gamma*. Si la *gamma* es demasiado grande, el radio del área de influencia de los vectores de soporte solo incluye al vector de soporte en sí. Para este proyecto se eligió a $gamma =' scale'$

Figura 23

Kernel RBF para un caso con datos no linealmente separables



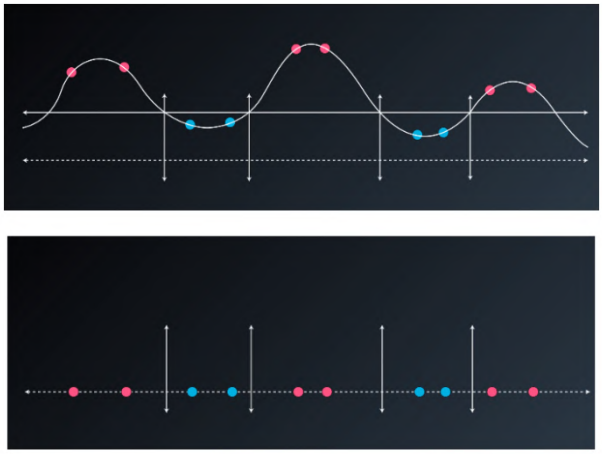
fuente: Roman (2019)

En la figura 23 se muestra la representación de datos para un caso no linealmente separable, Roman (2019) explica que la idea del kernel RBF es localizar las ”montañas” y ”valles” que coincidan con cada clase y proyectar a la linea original las lineas que separan las dos clases y que también cortan las linea de ”montaña-valle”.

Las proyecciones que se muestran en la primera parte de la figura 24 permiten que haya una separación efectiva de cada clase en el conjunto de datos original.

Figura 24

Proyecciones que separan cada clase en el conjunto de datos original

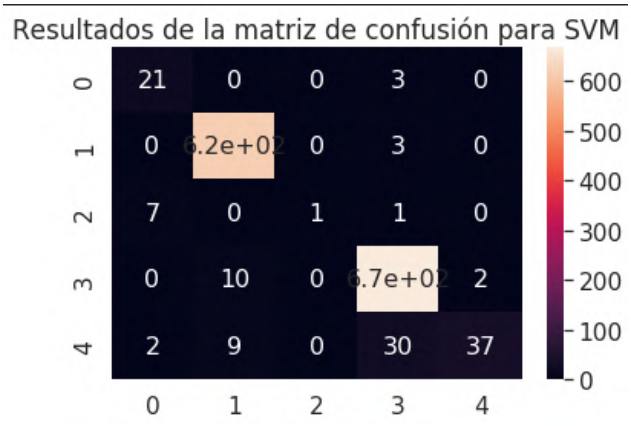


fuente: Roman (2019)

Una vez se definió el modelo SVM, se procedió a realizar su entrenamiento, para este entrenamiento se hizo uso del conjunto de datos de prueba correspondiente al 25 % del conjunto de datos original, y para saber qué cantidad de imágenes se clasificaron correctamente se hizo uso de la matriz de confusión para una evaluación mas completa del modelo.

Figura 25

Matriz de confusión para SVM en la predicción del modelo para el conjunto de muestreas de prueba



En la figura 25 se muestra el resultado de la matriz de confusión en la predicción realizada sobre el conjunto de datos o muestras de prueba, tenemos 1413 predicciones de las cuales se especificarán a continuación algunos de sus resultados:

- verdadero positivo (TP): el modelo predijo ”escasa posibilidad de dormirse” en 669 ocasiones.
- Verdadero negativo (TN): el modelo predijo ”elevada posibilidad de dormirse” en 618 ocasiones.

5.3.3 CNN para la clasificación de imágenes

Para la red neuronal convolucional, de igual forma se realizó un pre-procesamiento a todas las imágenes pertenecientes al conjunto de datos de datos. Este pre-procesamiento consistió en cargar y leer cada imagen, se le cambió el espacio de color a RGB, ya que al usar la biblioteca OpenCV, esta lee las imágenes como BGR. Se pasó cada imagen a escala de grises, el tamaño de las imágenes se redimensionó para tener las imágenes de $64 * 64$ píxeles. De igual manera que con las anteriores dos técnicas, se continuó con el etiquetado a cada una de las imágenes, es decir, a cada imagen se le asignó su etiqueta correspondiente. La matriz correspondiente a todas las muestras de las imágenes de entrenamiento se le realizó un ajuste debido a que OpenCV solo identifica matrices de tipo **float32**, además, para el arreglo correspondiente a las etiquetas de entrenamiento, este se lleva a la forma (tamaño de etiqueta, 1).

```
imag_entrenamiento = np.array(imag_entrenamiento).
                                    astype('float32')
etiquetas_entrenamiento = np.array(etiquetas_entrenamiento)


etiquetas_entrenamiento = etiquetas_entrenamiento.astype(int)


datosEntrenamiento,
```

```
datosPrueba,
etiquetas_entrenamiento,
etiquetasPrueba = train_test_split(
                                imag_entrenamiento,
                                etiquetas_entrenamiento,
                                test_size=0.25,
                                random_state=0)

datosEntrenamiento = datosEntrenamiento.reshape(
                                    datosEntrenamiento.shape[0],
                                    64,64,1).
                                    astype('float32')

datosPrueba = datosPrueba.reshape(
                            datosPrueba.shape[0],
                            64,64,1).astype('float32')
```

Se dividió el conjunto de datos de entrenamiento y el conjunto de datos de prueba, en el cual se definió para los datos de prueba un 25 % y un 75 % para los datos de entrenamiento, por lo tanto de las 4648 imágenes, 1162 imágenes quedaron asignadas para los datos de prueba mientras que 3486 imágenes para el entrenamiento. Tanto para los datos de entrenamiento como para los datos de prueba, se remoldearon con el fin de que estos arrays que contienen los datos pasaran a ser de 4 dimensiones, pues de esta forma es que el modelo para la red neuronal convolucional requiere los datos.

```
datosEntrenamiento = datosEntrenamiento / 255
datosPrueba = datosPrueba / 255

etiquetas_entrenamiento = to_categorical(etiquetas_entrenamiento)
```

```
etiquetasPrueba = to_categorical(etiquetasPrueba)
```

Adicionalmente, antes de establecer la arquitectura de la red neuronal convolucional, se normalizaron los valores de los píxeles de los datos de entrada para valores entre 0 y 255, lo que indica que se normalizan los vectores para datos entre 0 y 1 y finalmente para la clasificación se debe dividir la variable objetivo en categorías, esto se hizo con la función to_categorical del paquete Keras Utilities.

```
modelo = Sequential()
modelo.add(Conv2D(16, (5,5), strides=(1,1), activation='relu',
                  input_shape=(64,64,1)))
modelo.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))

modelo.add(Flatten())
modelo.add(Dense(100, activation="relu"))
modelo.add(Dense(units=5, activation='softmax'))
```

El anterior fragmento de código corresponde a la definición de la arquitectura para la red neuronal convolucional. Para esta se tiene la capa convolucional que cuenta con:

- el valor del filtro de 16: este número entero hace referencia a la dimensionalidad del espacio de salida.
- tamaño de kernel (5,5): esta tupla especifica la altura y el ancho de la ventana de la convolución 2D.
- strides de (1,1), esta tupla representa representa los pasos que toma la convolución respecto a lo largo y ancho.
- activation 'relu': la función de activación que se definió fue la 'relu'.
- input_shape de (64, 64, 1): la red neuronal debe recibir un tensor en 4D, para la

forma de la imagen de entrada se definió que todas la imágenes estarían formadas por $64 * 64$ píxeles y en escala de grises (filas=64, columnas=64, canales=1)

La capa convolucional está completamente conectada a la capa de agrupación (MaxPooling2D), en esta capa de agrupación se definieron los siguientes parámetros:

- pool_size (2,2), esta tupla indica el tamaño de la ventana sobre el cual se toma el valor máximo, (2,2) indica que teniendo una ventana de agrupación $2 * 2$ toma el valor máximo de dicha ventana.

- strides (2,2): como se mencionó para la capa de convolución, esta tupla indica el tamaño del paso que toma la ventana de agrupación para cada paso de agrupación, en este caso los pasos dados tanto para el alto como ancho de la ventana fueron de 2.

La siguiente capa oculta de la arquitectura de la red es una capa con 100 neuronas en la que al igual que en la capa de convolución se definió la función 'relu' como la función de activación. Finalmente se tiene la capa de salida que está conformada por 5 neuronas las cuales indican las 5 posibles salidas de la CNN, y la función de activación que se definió fue la función 'softmax'. La elección de esta función es debido a que con softmax los elementos del vector de salida están en el rango (0,1), se utiliza a menudo como función de activación para la ultima capa de una red de clasificación, por ende al ser un problema de clasificación binaria esta función adecuada porque el resultado podría interpretarse como una distribución de probabilidad.

```
modelo.compile(loss="binary_crossentropy",
               optimizer='adam',
               metrics=["accuracy"])
```

Posteriormente se compiló el modelo, para ello se establecieron los siguientes parámetros:

- loss="binary_crossentropy": calcula la perdida de entropía cruzada entre las etiquetas verdades y las etiquetas predichas. Esta perdida de entropía cruzada se utiliza para aplicaciones de clasificación binaria (0 o 1).

- optimizer='adam': la optimización con Adam es un método de descenso de gradiente estocástico que se basa en la estimación adaptativa de momentos de primer y segundo orden. *Adam* (s.f.)
- metrics=["accuracy"]: metrics es una función que se encarga de calificar el rendimiento del modelo. "Acurracy" calcula la frecuencia con que las predicciones son iguales a las etiquetas.

```
modelo.fit(datosEntrenamiento, etiquetas_entrenamiento,
                validation_data=(datosPrueba, etiquetasPrueba),
                epochs=10, batch_size=200, verbose=2)
```

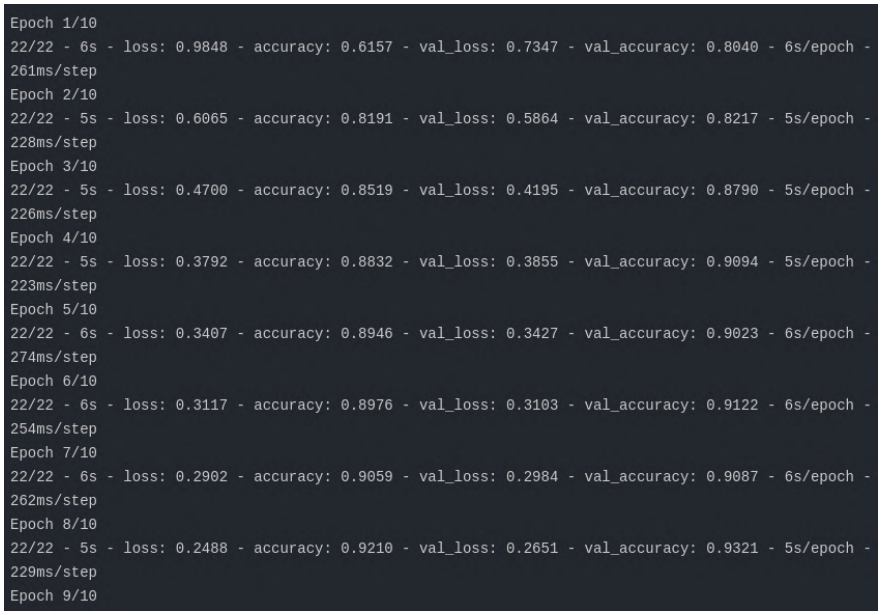
Finalmente se realizó el entrenamiento del modelo de la red neuronal, en donde se especificó los datos de entrenamiento, las etiquetas para el entrenamiento, los datos de validación (que corresponden tanto a los datos como etiquetas de prueba), se establecieron 10 épocas para el entrenamiento, el tamaño del lote y en verbose que nos muestra cierta cantidad de información mientras el modelo realiza cada época del entrenamiento.

5.4 Resultados y comparación de los mismos para cada técnica de machine learning

Una de las cosas a tener en cuenta en cada técnica era ver cuánto tiempo se tomaban en realizar el entrenamiento, en la tabla 3 se muestran los tiempos empleados por cada técnica en el entrenamiento así como el porcentaje de precisión del modelo.

Figura 26

Resultados del entrenamiento de la red neuronal convolucional (CNN)



```
Epoch 1/10
22/22 - 6s - loss: 0.9848 - accuracy: 0.6157 - val_loss: 0.7347 - val_accuracy: 0.8040 - 6s/epoch -
261ms/step
Epoch 2/10
22/22 - 5s - loss: 0.6065 - accuracy: 0.8191 - val_loss: 0.5864 - val_accuracy: 0.8217 - 5s/epoch -
228ms/step
Epoch 3/10
22/22 - 5s - loss: 0.4700 - accuracy: 0.8519 - val_loss: 0.4195 - val_accuracy: 0.8790 - 5s/epoch -
226ms/step
Epoch 4/10
22/22 - 5s - loss: 0.3792 - accuracy: 0.8832 - val_loss: 0.3855 - val_accuracy: 0.9094 - 5s/epoch -
223ms/step
Epoch 5/10
22/22 - 6s - loss: 0.3407 - accuracy: 0.8946 - val_loss: 0.3427 - val_accuracy: 0.9023 - 6s/epoch -
274ms/step
Epoch 6/10
22/22 - 6s - loss: 0.3117 - accuracy: 0.8976 - val_loss: 0.3103 - val_accuracy: 0.9122 - 6s/epoch -
254ms/step
Epoch 7/10
22/22 - 6s - loss: 0.2902 - accuracy: 0.9059 - val_loss: 0.2984 - val_accuracy: 0.9087 - 6s/epoch -
262ms/step
Epoch 8/10
22/22 - 5s - loss: 0.2488 - accuracy: 0.9210 - val_loss: 0.2651 - val_accuracy: 0.9321 - 5s/epoch -
229ms/step
Epoch 9/10
```

Cuadro 3

Tiempos y precisión empleados por cada técnica en el entrenamiento

Técnica	tiempo[s] empleado en el entrenamiento	Precisión
KNN	175.2156 segundos	95.966 %
SVM	29.7256 segundos	95.2583 %
CNN	60.12 segundos	93.5598 %

En cuanto a la predicción realizada con los con datos de prueba (pertenecientes al 25 % de los datos originales) en los modelos KNN y SVM en los cuales en las figuras 22 y 25 se muestra la matriz de confusión generada a partir de esta predicción, se hizo uso del visualizador de informes de clasificación (classification_report) de la librería sklearn para tener un informe más completo de estas predicciones, estos resultados se muestran en las tablas 4 y 5.

Cuadro 4

Resultados del visualizador de informes para KNN

	precision	recall	f1-score	support
Aparición de Ojeras	0.88	0.58	0.70	24
Elevada posibilidad de dormirse	0.98	1.00	0.99	621
Enrojecimiento de los ojos	0.00	0.00	0.00	9
Escasa posibilidad de dormirse	0.97	0.97	0.97	681
Mediana posibilidad de dormirse	0.73	0.77	0.75	78
accuracy			0.96	1413
macro avg	0.71	0.66	0.68	1413
weighted avg	0.95	0.96	0.96	1413

Cuadro 5

Resultados del visualizador de informes para SVM

	precision	recall	f1-score	support
Aparición de Ojeras	0.70	0.88	0.78	24
Elevada posibilidad de dormirse	0.97	1.00	0.98	621
Enrojecimiento de los ojos	0.00	0.00	0.00	9
Escasa posibilidad de dormirse	0.97	0.97	0.97	681
Mediana posibilidad de dormirse	0.95	0.47	0.63	78
accuracy			0.96	1413
macro avg	0.71	0.66	0.68	1413
weighted avg	0.95	0.96	0.96	1413

Para entender los resultados de esta tablas se describe que función cumple cada uno de sus encabezados. Como se explica en Kohli (2017) Hay cuatro formas de comprobar si las predicciones son correctas, estas están relacionadas con la descripción realizada a la tabla 1.

- **precision:** hace referencia a la capacidad que tiene el clasificador de no etiquetar a una instancia como positiva que en realidad es negativa.
  $precision = TP/(TP + FP)$ Se define como la relación entre verdaderos positivos y la suma de un verdadero positivo y un falso negativo.

- **recall:** hace referencia a recordar y es la capacidad que tiene el clasificador de encontrar todas las instancias positivas.
  $recall = TP/(TP + FN)$ Para cada clase se define como la relación entre los

verdaderos positivos y la suma de verdaderos positivos y falsos negativos.

- **f1 score:** es una media armónica poderada de precision y recall de manera que los puntajes f1 pueden ser mas bajos que las medidas de precision, ya que incorporan precision y recall en su cálculo.

  $f1 \quad score = 2 * (recall * precision)/(recall + precision)$

Para realizar una validación con cierto porcentaje de imágenes de la base de datos para cada una de las técnicas, se escogieron y organizaron 1200 imágenes en una carpeta nombrada **validación**, estas imágenes corresponden al conjunto de datos usados para validar cada técnica de machine learning. Cabe recalcar que estas imágenes **NO** fueron tenidas en cuenta en el conjunto de datos de entrenamiento y prueba. Para cada una de las técnicas, primeramente se guardó el modelo creado y entrenado que se explicó en la sección anterior, seguidamente en un nuevo archivo de python para cada técnica se cargó el modelo guardado previamente y se realizó un proceso similar al realizado en el entrenamiento de cada modelo, este proceso consistió en que a cada una de las imágenes pertenecientes a la carpeta **validación** se le hizo un pre-procesamiento y posteriormente unos ajustes a las matrices o arrays hasta tener la información de la misma forma en la que se realizó el entrenamiento, pues para realizar una validación para clasificación de datos en cualquiera de los modelos, el modelo espera recibir los datos de la misma manera en la que el modelo fue creado y/o entrenado.

Una vez realizado el procedimiento mencionado anteriormente se realizó la predicción en cada modelo de las 3 técnicas en la que se obtuvieron los siguientes resultados:

Cuadro 6

Resultados de la predicción con las tres técnicas para las imágenes del conjunto de datos de validación

Técnica	Aparición de Ojeras	Elevada posibilidad de dormirse	Enrojecimiento de los ojos	Escasa posibilidad de dormirse	Mediana posibilidad de dormirse
KNN	0	430	0	557	211
SVM	0	589	0	531	78
CNN	0	527	0	646	25

6 Resultados

En esta sección se exponen los resultados y evidencias obtenidas en el presente trabajo de grado.

En la primera etapa del desarrollo de este proyecto, se realizó un análisis a la base de datos MRL Eye Dataset, en el que se indica como están anotadas las diferentes propiedades de las imágenes y se realizó un resumen de las categorías de este conjunto de datos. Después se organizaron las imágenes a utilizar en carpetas como lo fueron para las imágenes de conjunto de entrenamiento y prueba, y para el conjunto de validación. Una vez organizados los conjuntos de datos, se le realizaron algunos ajustes a estos conjuntos de datos para así pasar a la implementación de la clasificación de imágenes con cada técnica. Tanto para la técnica KNN como para la SVM, cuando se realizó el entrenamiento y posterior validación con el conjunto de datos de prueba, los resultados se mostraron en la matriz de confusión en la que se indica que para la técnica KNN fue mayor la cantidad de imágenes clasificadas correctamente. Además, referente a los tiempos empleados en el entrenamiento por cada técnica, la técnica SVM es la que el entrenamiento de su modelo toma menos tiempo,pero contrario a esto, la técnica KNN el tiempo empleado en su entrenamiento fue mucho más grande. Para el porcentaje de las precisiones para las predicciones realizadas con el conjunto de datos de prueba, el valor de precisión más alto fue para la técnica KNN, seguido del porcentaje de la técnica SVM y por ultimo el porcentaje de la técnica CNN, cabe aclarar que la diferencia entre estos resultados fue pequeña.

Para la validación de cada modelo con imágenes pertenecientes a la base de datos, pero que no fueron utilizadas en el entrenamiento, se organizaron 1200 imágenes en una carpeta nombrada **Validación**, y se realizó la predicción en cada modelo de las tres técnicas en la que lo resultados de clasificación favorecieron a la técnica KNN, seguido por la técnica CNN y por último la técnica SVM.

Como **agregado ó adicional** a lo mencionado en su momento en el documento de

anteproyecto, se decidió realizar validaciones a cada una de las técnicas mediante el uso de imágenes que no pertenecieran a la base de datos **MRL Eye Dataset**, y también realizar validaciones mediante **vídeo en tiempo real** con el fin de ver cuál era el comportamiento de cada técnica. Para las imágenes a utilizar en las predicciones, estas fueron tomadas por la cámara web del computador portátil. Para poder realizar las predicciones y obtener una clasificación, tanto para las imágenes como para el vídeo en tiempo real fue indispensable el uso de las **Cascadas de Haar**, la implementación de las cascadas de Haar en código, se realizó haciendo uso de la función CascadeClassifier de OpenCV.

Para ello se generó una base de datos con imágenes de 6 personas las cuales fueron tomadas por la cámara web, de la misma manera para la implementación en tiempo real las imágenes iban a ser tomadas por medio de la cámara web, para ello se necesitaba de un clasificador de cascada de Haar que pudiera detectar un ojo, esto con el fin de poder obtener solo la parte del ojo de la imagen tomada por la cámara para posteriormente realizarle un pre-procesamiento y ajuste hasta llevarla a cada modelo de las técnicas empleadas para así obtener una clasificación. Algunos de estos clasificadores pueden detectar un ojo sin importar si está abierto o cerrado, pero algunos solo pueden detectar ojos abiertos.

Clasificadores que pueden detectar ojos abierto y cerrados:

- haarcascade_mcs_lefteye.xml
- haarcascade_mcs_righteye.xml
- haarcascade_lefteye_2splits.xml
- haarcascade_righteye_2splits.xml

Clasificadores que solo pueden detectar ojos abiertos:

- haarcascade_eye.xml
- haarcascade_eye_tree_eyeglasses.xml

Se hizo uso del clasificador **haarcascade_lefteye_2splits.xml** pues es un clasificador que puede detectar los dos estados del ojo. En este caso y dándole prioridad a la implementación de vídeo en tiempo real, poder saber cuando en la captura de una imagen ésta tenía el ojo cerrado para así generar un sonido de alarma, fue indispensable el uso de este clasificador.

Figura 27

Aplicación del clasificador de cascada de Haar para detectar el ojo abierto

(a) *Detección del ojo con el clasificador de las cascadas de Haar*

(b) *Recorte de la zona del ojo*

En las figuras 27 y 28 se muestra un ejemplo de cómo se obtuvieron las imágenes para realizar la predicción en cada una de las técnicas y así obtener un resultado de clasificación. En la figura 27a se tiene a la imagen capturada por la cámara web a la cuál se le aplicó el clasificador de cascada de Haar con el cual se obtuvieron los ojos de la imagen, en primera instancia, la imagen capturada es pasada a escala de grises para ahí si aplicar el clasificador de la cascada de Haar, este clasificador devuelve los 4 puntos de la imagen en el que se encuentra el ojo, una vez se tienen estos valores, se puede obtener el

Figura 28

Aplicación del clasificador de cascada de Haar para detectar el ojo cerrado

(a) *Detección del ojo con el clasificador de las cascadas de Haar*

(b) *Recorte de la zona del ojo*

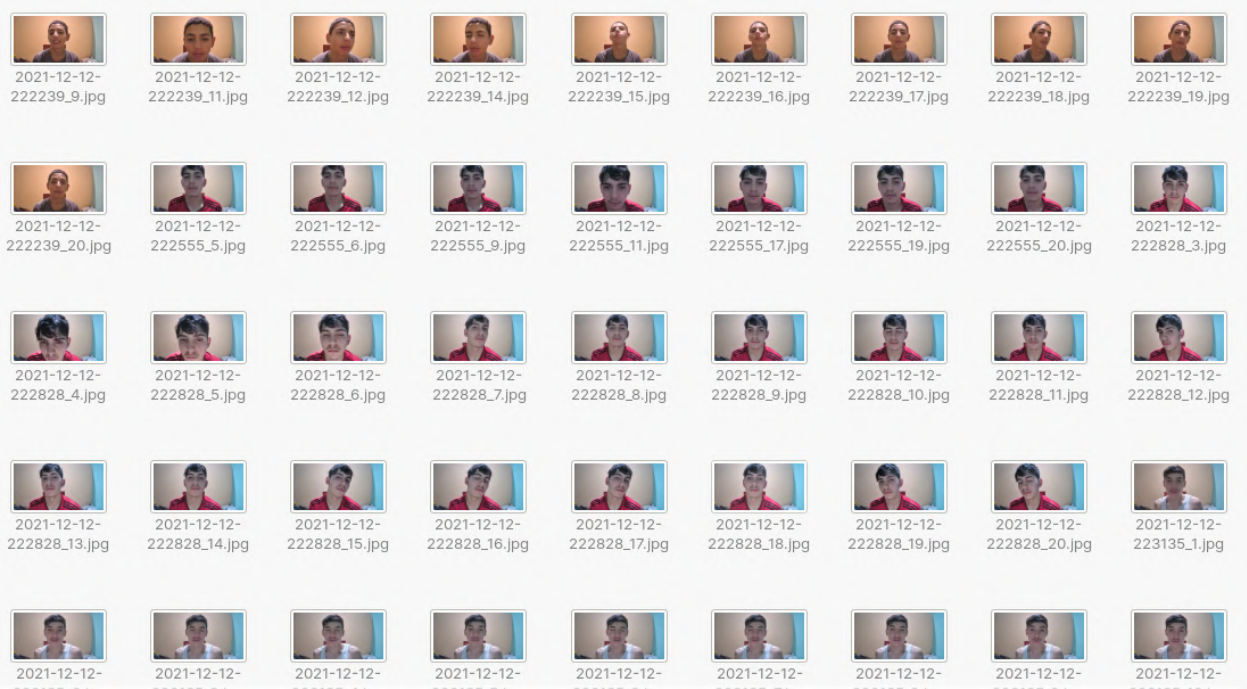
recorte de la zona de los ojos que se muestra en la figura 27b, esta imagen es con la que se va a realizar la predicción, para ello primeramente pasó por un pre-procesamiento y ajuste igual al que se le realizó a todas las imágenes tanto del conjunto de entrenamiento, prueba y predicción del cual se habló anteriormente, esto con el fin de que esta imagen tenga las mismas dimensiones o características y el modelo la pudiera clasificar.

Las imágenes 27b y 28b como se mencionó anteriormente se tomaron como ejemplo para la realización de una predicción en las tres técnicas de machine learning trabajadas en este trabajo, esto como un preámbulo para realizar la validación del sistema con la base de datos generada y para la posterior implementación de vídeo en tiempo real.

En la tabla 7 se muestra el resultado de la clasificación para las validaciones realizadas con las imágenes pertenecientes a la base de datos creada, en la que esta base cuenta con 330 imágenes de 6 personas diferentes.

Figura 29

Ejemplo de imágenes de la base de datos creada



Cuadro 7

Resultados de la predicción con las tres técnicas para las imágenes del conjunto de datos de validación

Técnica	Aparición de Ojeras	Elevada posibilidad de dormirse	Enrojecimiento de los ojos	Escasa posibilidad de dormirse	Mediana posibilidad de dormirse
KNN	0	193	0	117	20
SVM	5	72	0	253	0
CNN	0	161	0	134	34

Los resultados de la predicción para estas imágenes tuvieron un gran acierto, esto conllevó a poder implementar vídeo en tiempo real y así poder realizar predicciones para

muchas imágenes en poco tiempo, además de poder generar una alarma en la medida en que varias imágenes de forma seguida eran clasificadas como **elevada posibilidad de dormirse**.

6.1 Implementación de la clasificación de imágenes en tiempo real

Para la clasificación de imágenes en tiempo real, como se mencionó anteriormente, se hizo uso del clasificador de cascada de Haar, este se le aplicaba a cada frame que devolvía la cámara web, para así ir obteniendo solo las imágenes de los ojos y realizar la predicción, para estas imágenes obtenidas se realizó el mismo pre-procesamiento y ajustes realizados en los entrenamientos de cada técnica para así poder realizar dichas predicciones y obtener la clasificación. Para cada técnica, una vez realizada la predicción se tuvo en cuenta de generar un sonido de alarma para cuando el resultado era **elevada posibilidad de dormirse** para varias imágenes seguidas porque si de lo contrario el resultado era para una o dos imágenes seguidas solo se contaba como un parpadeo normal.

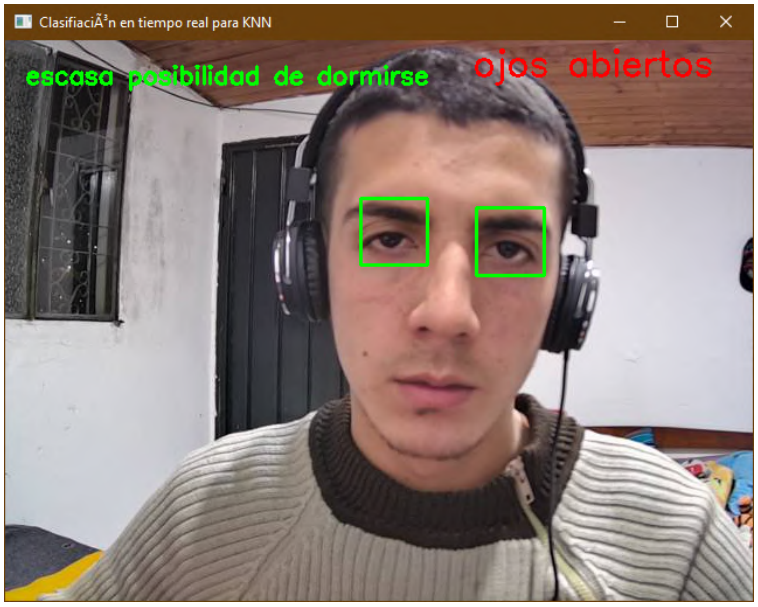
En la figura 30 se muestra la implementación en tiempo real en la cual cada una de las predicciones que se realizaban mediante la técnica KNN, en la figura 30a se muestra un ejemplo para cuando se tienen los ojos abiertos, en este caso se muestra la etiqueta **escasa posibilidad de dormirse**, que hace referencia a el resultado de la clasificación de la imagen pasada en ese preciso instante al modelo para su debida clasificación. En la figura 30b se una captura de ejemplo en la cual se tienen los ojos cerrados, como podemos ver una vez se hace la predicción en el clasificador y el resultado de la predicción para varios frames de forma consecutiva, se muestra de que se tienen los ojos cerrados y además se muestra el resultado de la clasificación de **elevada posibilidad de dormirse**, en ese preciso momento, por un par de segundos se reproduce el sonido de una alarma, para la cual en la implementación en un entorno real avisaría al conductor de un automóvil que se va a dormir.

De la misma manera se realizó para las técnicas SVM y CNN, el procedimiento

Figura 30

Clasificación en tiempo real para la técnica KNN

(a) *Clasificación para cuando los ojos están abiertos*



(b) *Clasificación para cuando los ojos están cerrados*

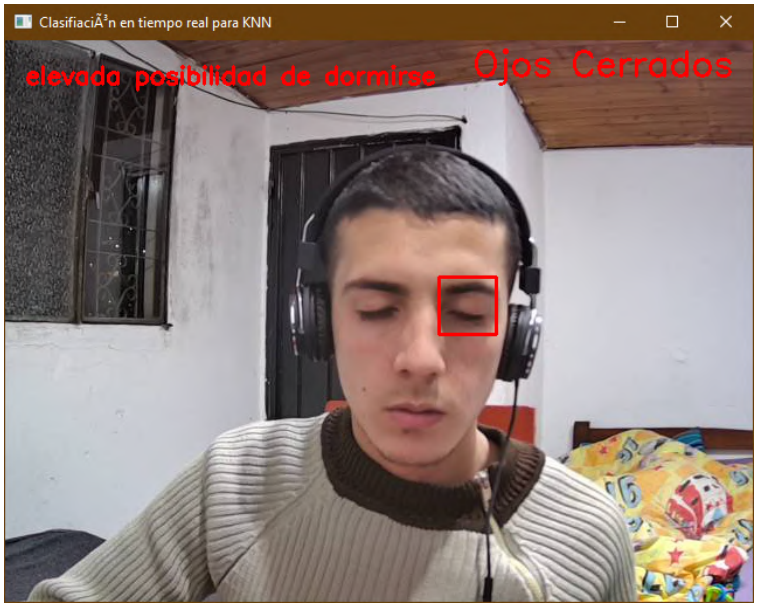


Figura 31

Clasificación en tiempo real para la técnica SVM

(a) *Clasificación para cuando los ojos están abiertos*

(b) *Clasificación para cuando los ojos están cerrados*

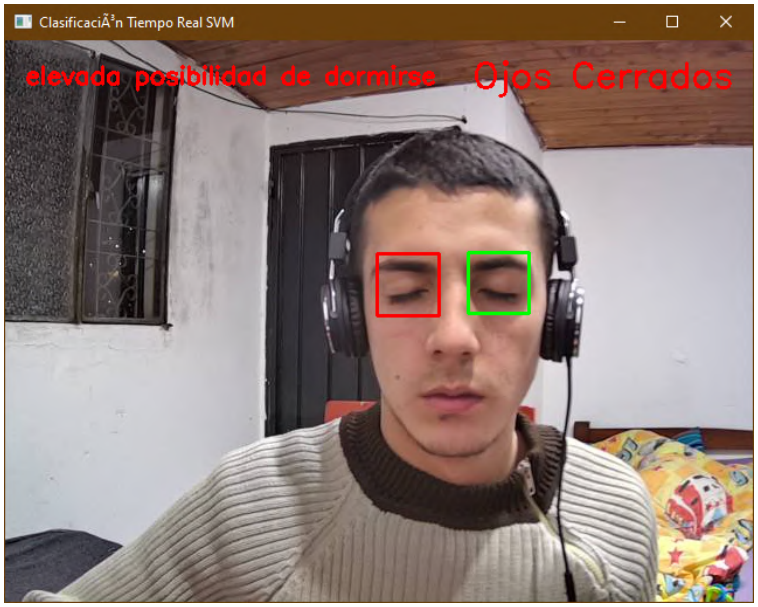
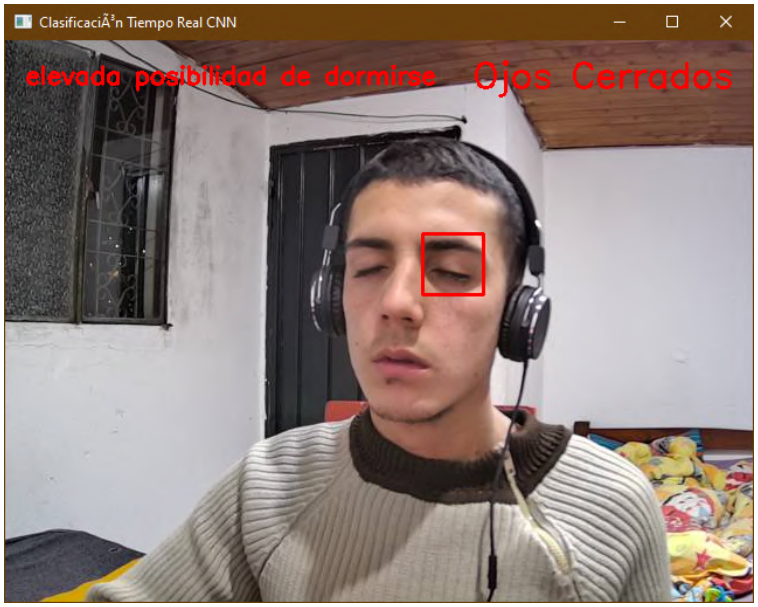


Figura 32

Clasificación en tiempo real para la técnica CNN

(a) *Clasificación para cuando los ojos están abiertos*

(b) *Clasificación para cuando los ojos están cerrados*



llevado a cabo sobre las imágenes que se iban a clasificar constantemente era prácticamente el mismo a como se ha venido comentando para cada técnica. La figura 31 muestra los ejemplos para cuando se realiza una clasificación de forma constante, en el que en la figura 31a se trata para el caso en el que se tenían los ojos abiertos mientras que en la figura 31b para cuando se tenían los ojos cerrados. La figura 32a muestra el resultado de la clasificación justo en el momento en el que se tienen los ojos abiertos, mientras que la figura 32b para el momento en el que se tenían los ojos cerrados y en el que además se generaba un sonido de alarma.

Después de probar cada una de las técnicas mediante la implementación de vídeo en tiempo real, la técnica KNN fue la más sensible, pues muchas veces así se tuvieran los ojos abiertos, el resultado de la clasificación era como si se tuvieran los ojos cerrados pues la etiqueta del resultado de la clasificación era **elevada posibilidad de dormirse**, por lo que de forma mas seguida se generaba el sonido de alarma. La técnica SVM realizaba predicciones en tiempo real de forma correcta, pero en la técnica CNN fue donde se presenció que se trabajaba con mejor robustez.

6.2 Producto Deseado

Como producto compacto de todo el trabajo realizado, se implementó una interfaz gráfica amigable con el usuario, con el fin de que los usuarios puedan comunicarse o relacionarse con la maquina de la forma mas sencilla e intuitiva posible.

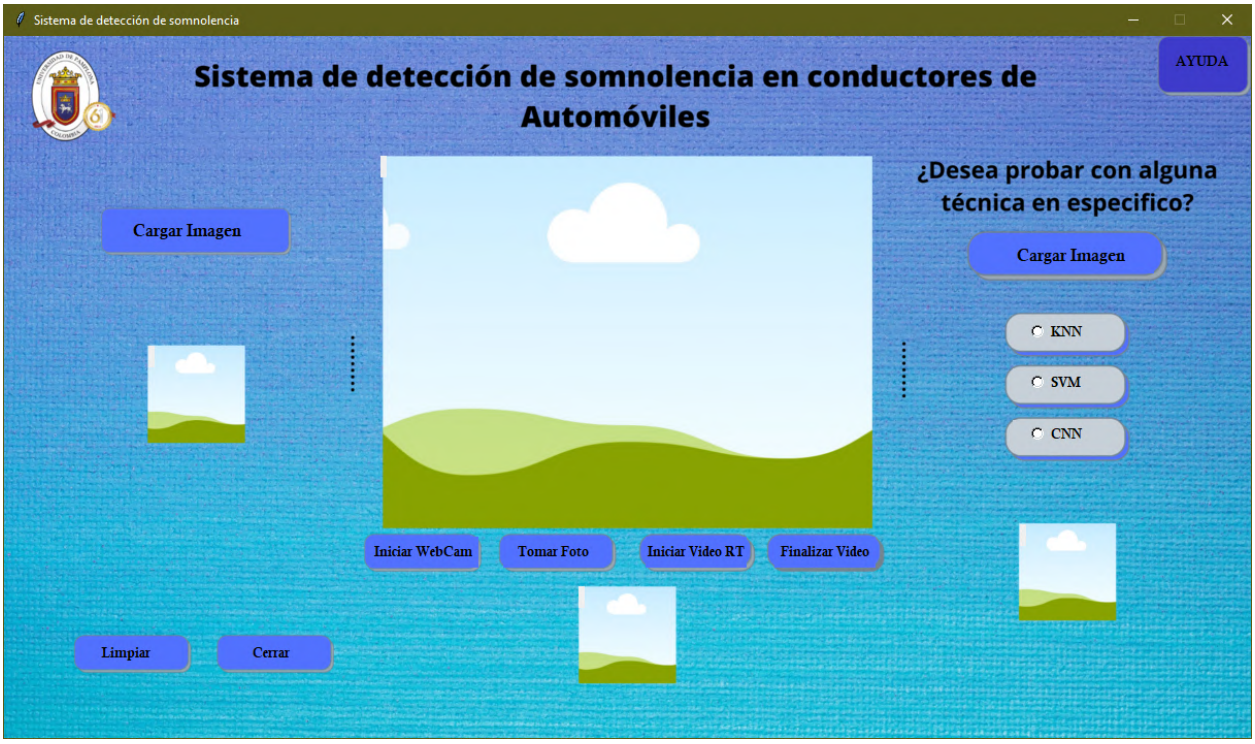
Para el presente trabajo esta interfaz gráfica de usuario se dividió en tres columnas o secciones en las que en cada una de ellas se presenta algo diferente que se puede realizar o con lo que se puede interactuar.

La figura 33 muestra la ventana de inicio de la interfaz una vez se ha ejecutado el código en python, esta contiene un diseño simple y la forma en que están organizados los componentes la hacen la interacción con ella sea fácil.

Como se mencionó anteriormente, para que la interfaz fuera amigable con el

Figura 33

Inicio de la interfaz gráfica

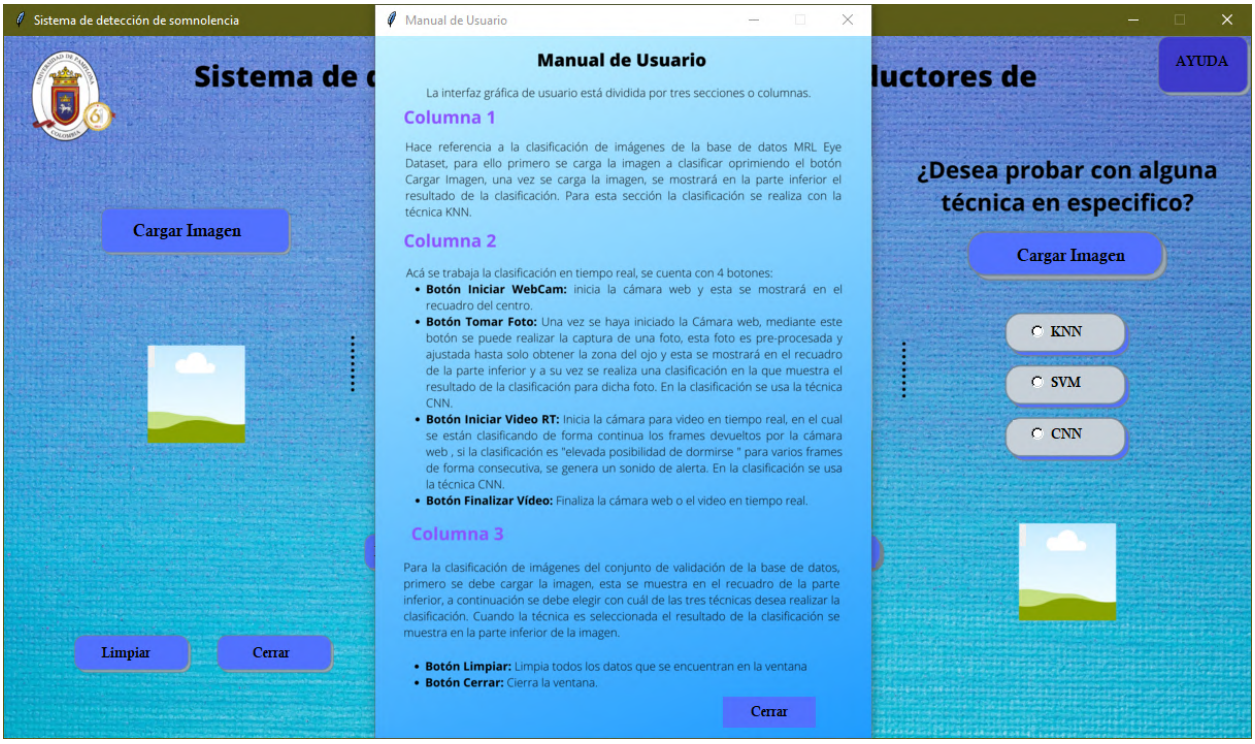


usuario, y para que pudieran relacionarse de forma mas sencilla e intuitiva, se agregó un botón de ayuda en la parte superior derecha de la ventana principal, el cual al ser presionado genera una sub-ventana con un manual de usuario en el que se especifica de forma mas detallada la manera en la que el usuario puede interactuar con ella, una vez el usuario haya leído e interpretado este manual, en la parte inferior derecha se encuentra un botón para cerrar esta sub-ventana y así volver a la ventana principal.

Ya interactuando con la interfaz, nos ubicamos en la primera sección o columna, y como se muestra en la figura 33, se tiene un botón para cargar una imagen, al presionar este botón se genera una sub-ventana en la que se tiene la posibilidad de elegir una imagen para ser clasificada como se indica en la figura 35. La imagen a elegir puede ser una imagen que pertenezca a la base de datos MRL Eye Dataset de las que están organizadas en la carpeta de validación, o esta imagen puede ser elegida de la base de datos creada,

Figura 34

Manual de Usuario



pues esta imagen a elegir no debió ser utilizada en el entrenamiento de las técnicas. Para este caso la imagen a elegir es una perteneciente a la carpeta **Validación** la cual contiene las imágenes que se seleccionaron para el conjunto de validación del cual se habló en secciones anteriores.

Una vez se ha seleccionado una imagen cualquiera, esta se mostrará o se ubicará en el recuadro presente en el centro de esta sección, y en la parte inferior de la imagen se muestra el resultado obtenido en el proceso de clasificación como se muestra en la figura 36.

La imagen que se cargó contiene el ojo cerrado, y el resultado de la clasificación es de **elevada posibilidad de dormirse**, por lo que para este caso se podría decir que el resultado de la predicción es correcto. La clasificación se está realizando en este caso con la técnica KNN, la elección de esta técnica se basa en que esta fue la técnica con la que se

Figura 35

Cargar la imagen que se desea clasificar

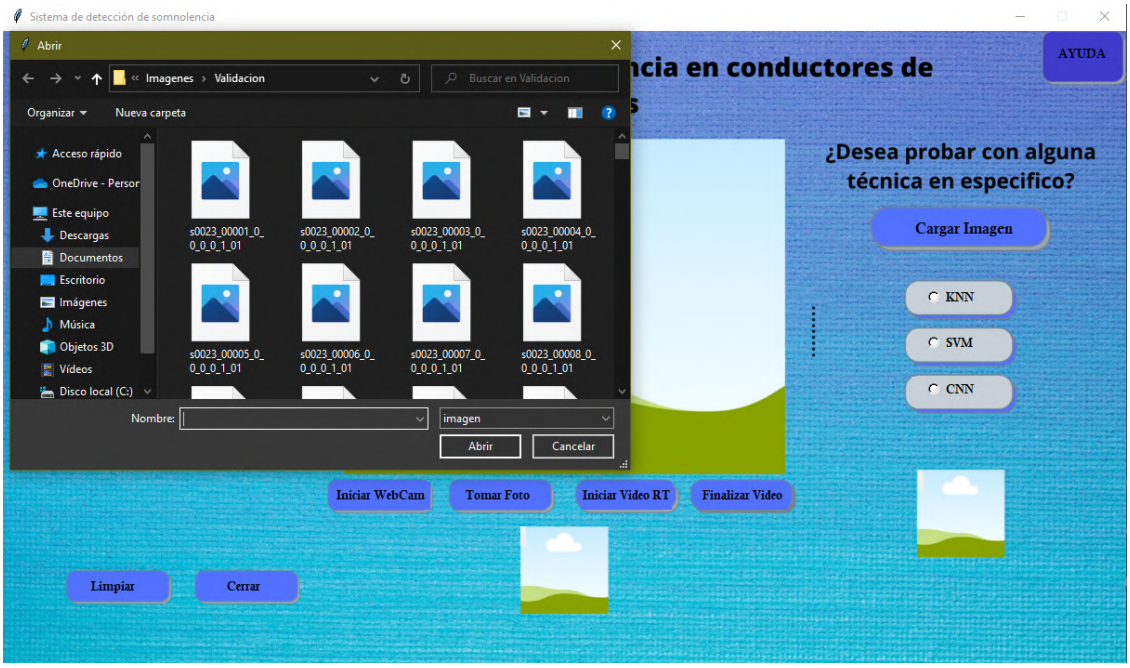
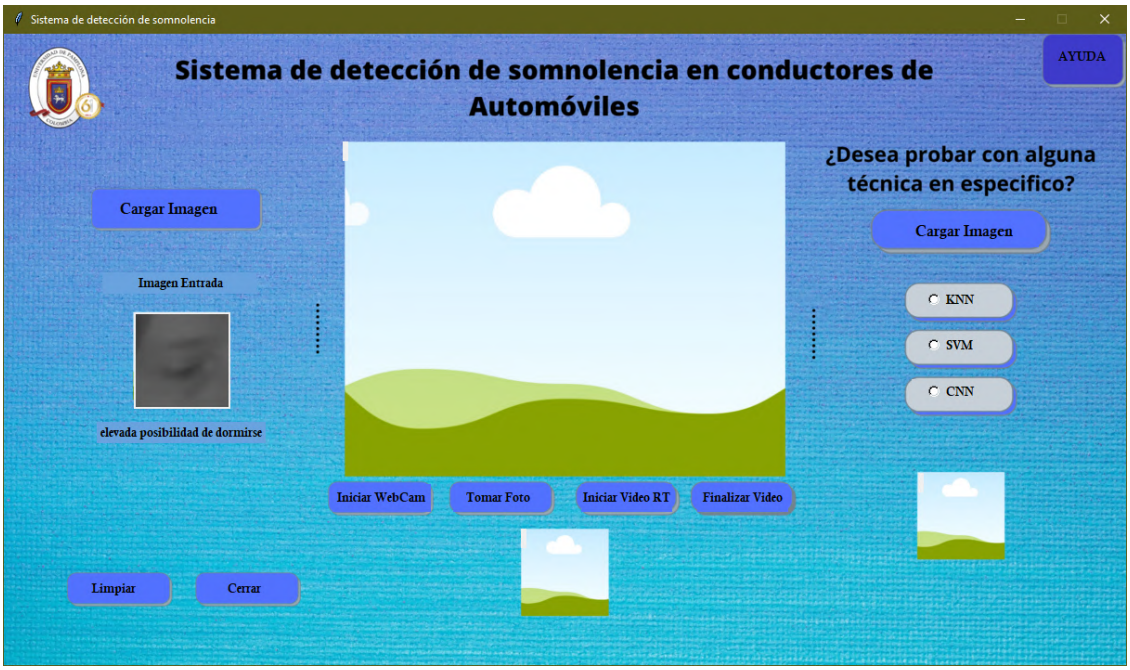


Figura 36

Clasificación de imagen con la técnica KNN



obtuvieron los mejores resultados cuando se realizó la clasificación de las 1200 imágenes contenidas en la carpeta **Validación**.

Además en esta sección, en la parte inferior se encuentran 2 botones, el botón **Limpiar** es útil para cuando en la ventana se ha cargado una imagen, en esta o en las demás columnas o secciones, pues si al ser oprimido va a limpiar todos los datos o información presentes en la pantalla para tenerla como en un inicio. El botón **Cerrar**, es útil para cuando ya se desea dar por terminado el uso o interacción con la interfaz, al ser oprimido finaliza el programa.

En la columna o sección del medio, es la parte donde se realizan clasificaciones pero con imágenes **NO** pertenecientes a la base de datos MRL Eye Dataset, sino que se hacen con fotos tomadas a partir del uso de la cámara web, para obtener imágenes en las que solo se obtengan la zona de los ojos se hizo uso de el clasificador de cascada de Haar, mediante el cual detectaba los ojos ya sea si estaban abiertos o cerrados y esta parte de la imagen era extraída para realizarle un pre-procesamiento y ajuste para poder llevarlas a los modelos de las técnicas empleadas y poder realizar la clasificación. El anterior procedimiento se llevaba a cabo tanto para realizar la toma de una foto, obtener la imagen y realizar la clasificación como para vídeo en tiempo real en el cual se realizaban validaciones de forma seguida y según el resultado generar una alarma de alerta.

En la figura 37 se muestra la forma en la que está organizada esta sección, que cuenta con una zona o recuadro grande en el que se inicia la cámara web tanto para tomar las fotos como para el vídeo en tiempo real, y un recuadro o zona en la parte inferior en la cual se muestran las fotos que son tomadas y son clasificadas por el modelo. Cuenta con cuatro botones, al ser oprimido el botón **Iniciar WebCam** se inicia la cámara web, lo que permite que al oprimir el botón **Tomar Foto** se realice una captura de la imagen presente a la cual se le realiza el procedimiento mencionado anteriormente hasta obtener la imagen lista para la clasificación, esta foto se muestra en el recuadro o zona inferior y a su vez debajo de esta, se muestra el resultado de la clasificación. En la figura 37a se muestra el

Figura 37

Captura de una foto a través de la cámara web para la clasificación

(a) *ejemplo de foto con los ojos abiertos*

(b) *ejemplo de foto con los ojos cerrados*

Figura 38

Clasificación mediante vídeo en tiempo real

(a) *ejemplo para cuando los ojos están abiertos*

(b) *ejemplo para cuando los ojos están cerrados*

Figura 39

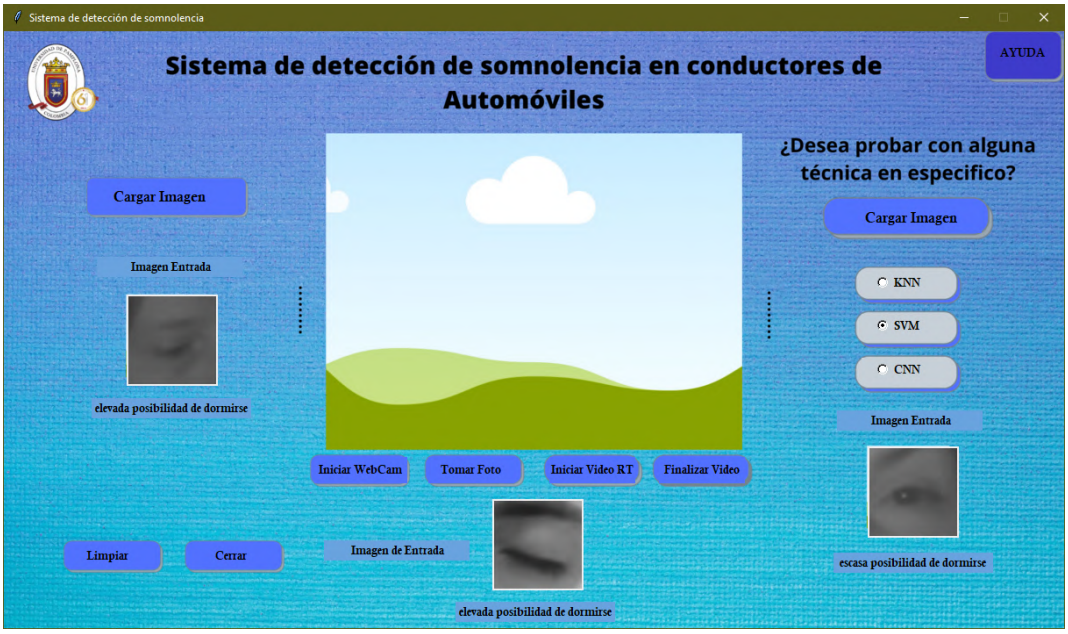
Ejemplo para cuando los ojos están entre abiertos

ejemplo para cuando se inicia la cámara web y se toma una foto en el momento en el que se tienen los ojos abiertos y que da como resultado de clasificación **escasa posibilidad de dormirse**, mientras que en la figura 37b se muestra el caso en el que una vez se inició la cámara web se tomó una foto justo en el instante en el que se tenían los ojos cerrados, en la que en el recuadro de la parte inferior se muestra la zona del ojo que contiene la foto con el ojo cerrado en el que el resultado de la clasificación es **elevada posibilidad de dormirse.** El botón **Iniciar Video RT** inicia la cámara para la clasificación mediante vídeo en tiempo real, en la figura 38a se muestra el ejemplo para en el que se indica tanto el estado de los ojos como **ojos abiertos** así como del resultado de las clasificación que es **escasa posibilidad de dormirse**.

Figura 40

Clasificación mediante la elección de alguna técnica de machine learning

(a) *Clasificación mediante la técnica SVM*



(b) *Clasificación mediante la técnica CNN*

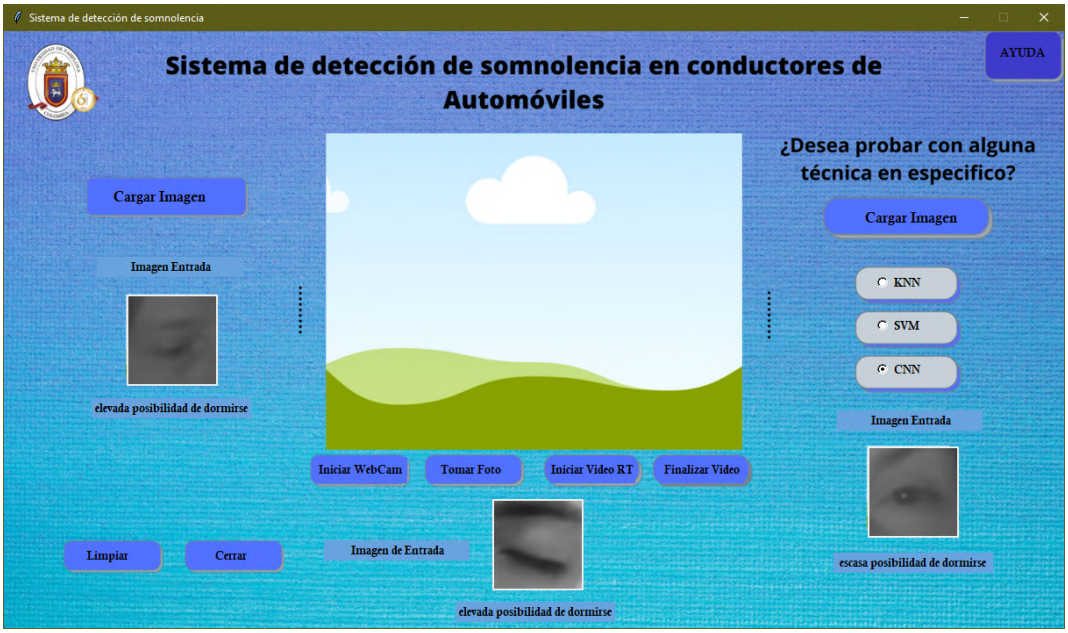
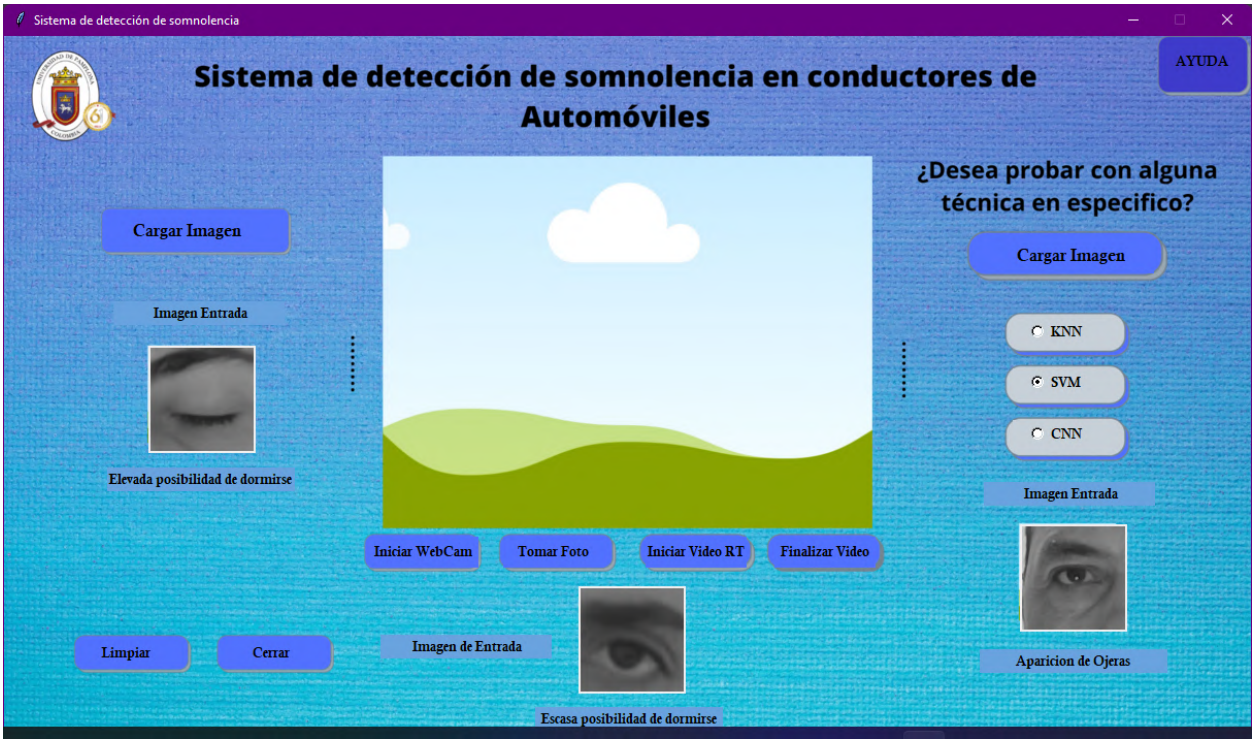


Figura 41

Ejemplo de clasificación donde el resultado es Aparición de Ojeras

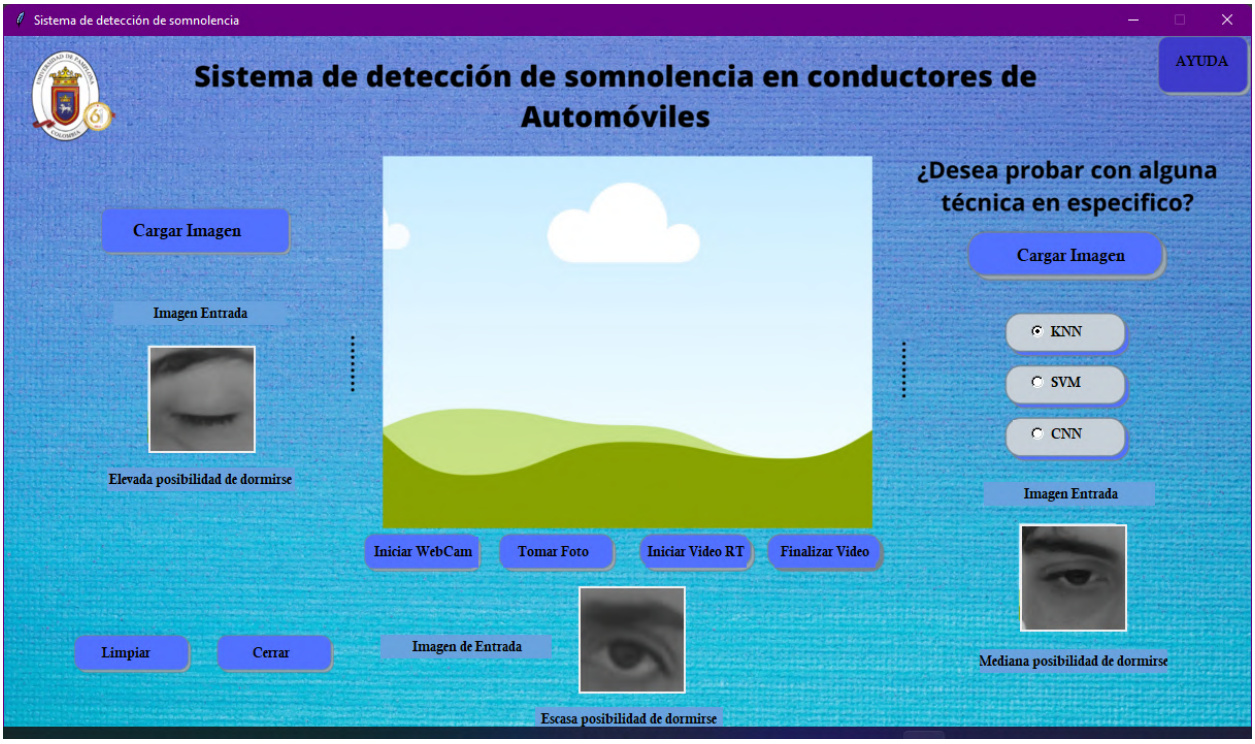


En la figura 38b se muestra el ejemplo de clasificación mediante vídeo en tiempo real en el que resultado para el estado de los ojos es **ojos cerrados** y su clasificación es **elevada posibilidad de dormirse**. Para la clasificación mediante vídeo en tiempo real la técnica utilizada es la CNN. El botón **Finalizar Vídeo** cumple la función de finalizar la cámara web, tanto para cuando se toman fotos como para cuando se trata de vídeo en tiempo real.

En la tercera sección o columna de la interfaz gráfica, se realiza la clasificación para imágenes pertenecientes a la base de datos y que estuvieran contenidas dentro de la carpeta **Validación**. Lo diferente de esta sección en comparación con la primera es que en esta se tienen la posibilidad de elegir con cuál de las tres técnicas se desea realizar la clasificación, mientras que en la sección uno, la clasificación solo se realizaba con la técnica KNN. En la figura 40a para la tercera sección, primero se cargó al azar una imagen para

Figura 42

Ejemplo para cuando el resultado de la clasificación es mediana posibilidad de dormirse



clasificar, a continuación se eligió la técnica SVM, obteniendo un resultado de clasificación como **escasa posibilidad de dormirse**. El mismo procedimiento se realizó para cuando se eligió la técnica CNN en la que el resultado de la clasificación fue el mismo como se indica en la figura 40b.

7 Conclusiones

- En el presente proyecto se efectuaron estudios para la clasificación de imágenes con tres técnicas de machine learning para la detección del grado de somnolencia de conductores de automóviles en el que este grado estaba relacionado con 5 clases, según los resultados obtenidos en las diferentes pruebas realizadas, este sistema de detección de somnolencia obtiene buenos resultados de clasificación tanto para las imágenes pertenecientes a la base de datos MRL Eye Dataset así como para imágenes no pertenecientes a esta base de datos, imágenes que fueron captadas por una cámara web.

- Una vez implementada la arquitectura para los modelos de las técnicas KNN, SVM y CNN, se percibe que tanto para el entrenamiento como para las validaciones el procesamiento de computo es rápido a pesar de tener casi 5000 imágenes usadas para el entrenamiento y las 1000 imágenes usadas para la validación, pues lo tiempos empleados en el entrenamiento fueron cercanos a los 30 segundos para las técnica SVM, mientras que para la red neuronal convolucional el tiempo empleado fue cercano a los 40 segundos para las 10 épocas realizadas. Para las validaciones con vídeo en tiempo real, el sistema siempre responde de forma rápida y eficaz, sin que haya algún tipo de congelamiento o algo similar, sino que las validaciones y sus respectivos resultados se muestran de manera instantánea.

- Para proyectos como este donde se emplean varias técnicas de machine learning para la clasificación de imágenes, es importante contar con una cantidad considerable de datos o imágenes, por lo que en este proyecto fue indispensable contar con la base de datos MRL Eye Dataset la cual cuenta con una gran cantidad de datos o imágenes y además con la posibilidad de poder etiquetar cada uno de ellos. Lo anterior fue importante para la etapa de entrenamiento, pues la gran cantidad de patrones que se obtienen de las imágenes escogidas de la base de datos tienen un valor fundamental

para el correcto aprendizaje de los modelos. Elegir el conjunto de datos correcto es fundamental, pues si los datos no son los correctos, esto conlleva a que se tenga un modelo incorrecto o un modelo en el que no se obtengan los resultados esperados.

- En la clasificación para imágenes capturadas por la cámara web, para una distancia igual o menor a un metro, en la gran mayoría de casos los ojos son detectados por el clasificador de cascadas de Haar y a partir de ahí se puede procesar la imagen para realizar la clasificación, de la misma manera para la clasificación de imágenes mediante la implementación de vídeo en tiempo real. Para distancias menores a un metro de la cámara web, el clasificador detecta los ojos y por ende se puede realizar la clasificación de forma correcta. Pero a medida que la distancia respecto a la cámara aumenta, en muchos casos no son detectados los ojos o en los casos en que se detectan, el resultado de la clasificación es **elevada posibilidad de dormirse** a pesar de tener los ojos abiertos. Lo anterior para vídeo en tiempo real genera un problema y es que al alejarse de la cámara, los resultados de las predicciones continuas son **elevada posibilidad de dormirse** por lo que el sonido de alarma se genera se forma constante a pesar de tener los ojos abiertos. Otro factor a tener en cuenta para las clasificaciones mencionadas anteriormente, es la calidad/resolución de las imágenes capturadas por la cámara web, pero lo mas importante es la cantidad de luz o la iluminación del entorno o sitio en el que se realizan las pruebas, pues si alguna de estas dos cosas no se cumplen, no se detectan los ojos de la imagen capturada o el resultado de la clasificación es **elevada posibilidad de dormirse** a pesar de tener los ojos abiertos. Por ello, es importante tener en cuenta la distancia respecto a la cámara y la iluminación del lugar en el que se realizan las pruebas o validaciones.

Referencias

Acosta, J. A. N. (2018, 1). *DetecciÓn de fallas y selecciÓn de variables en un proceso multivariado usando anÁlisis de seÑales y sistemas inteligentes.*

Adam. (s.f.). Descargado de `https://keras.io/api/optimizers/adam/`

Aditya, M. (2020, 12). *Haar cascades, explained. a brief introduction into haar... | by aditya mittal | analytics vidhya | medium.* Descargado de `https://medium.com/analytics-vidhya/haar-cascades-explained-38210e57970d`

Alfaro, E. A. C. (2010, 6). Máquinas de soporte vectorial con algoritmos basados en poblaciones para el pronóstico del precio de acciones lan chile.

Arbeloa, G. B. (2018, 3). *Implementación del algoritmo de los kvecinos más cercanos (k-nn) yestimación del mejor valor local de kpara su cálculo.* Descargado de `http://ieeexplore.org/document/7368188/`

Arnal, M. G. (2018, 10). *Estudio y aplicación de las redes neuronales convolucionales 3d.*

Arroyo, D. E. (2019, 6). *Visualizando neuronas en redes neuronales convolucionales.*

BAYONA ACOSTA, R. (2007). *Implementacion software de algoritmos para la deteccion de rasgos faciales utilizados como indicadores de somnolencia en conductores usando secuencias de imagenes biometricas [recurso electronico].* Bucaramanga: UIS.

Castro, J., Rosales-Mayor, E., y Egoavil, M. (2009, 01). Somnolencia y cansancio durante la conducción: accidentes de tránsito en las carreteras del perú. *Acta Med Per*, *26*, 48-54.

Dobernack, N. A. (s.f.). *Proyecto fin de carrera-nicolás aguirre dobernack.*

Drowsy driving | nhtsa. (s.f.). Descargado de `https://www.nhtsa.gov/risky-driving/drowsy-driving`

Emerson, M. S., y Raúl, L. M. (2018, 6). *Universidad nacional de huancavelica facultad de ingenierÍa electrÓnica-sistemas escuela profesional de ingenierÍa electrÓnica tesis lÍnea de investigaciÓn: Control y automatizaciÓn para optar el tÍtulo profesional de: Ingeniero electrÓnico presentado por: Sistema de detecciÓn de somnolencia mediante*

inteligencia artificial en conductores de vehÍculos para alertar la ocurrencia de accidentes de transito.

Fernández Villán, A., Fernández, R., y Casado, R. (2017, 07). Sistema automático para la detección de distracción y somnolencia en conductores por medio de características visuales robustas. *Revista Iberoamericana de Automática e Informática Industrial RIAI*, *14*, 307-328. doi: 10.1016/j.riai.2017.05.001

Heras, J. M. (2019, 5). *Máquinas de vectores de soporte (svm) - iartificial.net.* Descargado de `https://www.iartificial.net/maquinas-de-vectores-de-soporte-svm/`

Javier, J. (2019, 10). *Microsueños, una de las principales causas de accidentalidad en colombia | la fm.* Descargado de `https://www.lafm.com.co/colombia/que-son-los-microsuenos-una-de-las-principales-causas-de-accidentalidad-en-colombia`

Kohli, S. (2017). *Understanding a classification report for your machine learning model | by shivam kohli | medium.* Descargado de `https://medium.com/@kohlishivam5522/understanding-a-classification-report-for-your-machine-learning-model-88815e2ce397`

León, E. C. (s.f.). *Introducción a las máquinas de vector soporte (svm) en aprendizaje supervisado.*

Lizarazo, M. A. S. (2017, 6). *Sistema digital de reconocimiento de patrones mediante redes neuronales convolucionales.*

Lowe, D. G. (1999). *Object recognition from local scale-invariant features.*

Megías, C. J., y Serrano, J. (s.f.). *Adquisición y procesado extracción características bloque de entrenamiento clasificaciÓn de imÁgenes.*

Mrl eye dataset | mrl. (s.f.). Descargado de `http://mrl.cs.vsb.cz/eyedataset`

Paterna, P. J. G. (2019). Desarrollo de un sistema inteligente de detección de fatiga en conductores. Descargado de `http://hdl.handle.net/10317/7793`

Piazuelo, J. P. M. (2017). Detección de sueño en conductores de automóviles por

reconocimiento de imagen. *instname:Universidad de los Andes*. Descargado de http://hdl.handle.net/1992/15248

Quezada, L. N. (2017). *K - vecino más próximos en una aplicación de clasificación y predicción en el poder judicial del perú.*

Rodriguez, C. C. (2020). *Maquina de soporte vectorial (svm) | medium.* Descargado de https://medium.com/@csarchiquerodriguez/maquina-de-soporte-vectorial-svm-92e9f1b1b1ac

Roman, V. (2019). *Aprendizaje supervisado: Introducción a la clasificación y principales algoritmos | ciencia y datos | medium.* Descargado de https://medium.com/datos-y-ciencia/aprendizaje-supervisado-introducción-a-la-clasificación-y-principales-algoritmos-dadee99c9407

Shin, T. (2020). *Comprensión de la matriz de confusión y cómo implementarla en python.* Descargado de https://www.datasource.ai/es/data-science-articles/comprension-de-la-matriz-de-confusion-y-como-implementarla-en-python

Sánchez, E. G. (2019, 9). *Introducción a las redes neuronales de convolución. aplicación a la visión por ordenador.*

Tapasco, A. Z., Londoño, S. P., y Flórez, J. M. (2014, 3). Método basado en la clasificación k-nn parametrizados con algoritmos genéticos y la estimación de lareactancia para localización de fallas en sistemas de distribución. *Revista Facultad de Ingeniería Universidad de Antioquia, núm. 70, marzo-, 2014*, 220-232.

Villa, M. S. R., y Castro, J. S. V. (2020). Análisis de algoritmos de procesamiento de imágenes para el reconocimiento facial de fatiga y somnolencia en conductores. Descargado de http://hdl.handle.net/11059/12812

Viola, P., y Jones, M. (2001). *Rapid object detection using a boosted cascade of simple features.*

Ángel, M., y Galindo, A. (2019, 1). Título: Algoritmos de detección de objetos para el rastreo de ojos.