Desarrollo de un sistema autónomo de detección de somnolencia en conductores usando redes reuronales profundas.

Angie Daniela Thomas Bayona y David Gilberto Olarte Velasco

Trabajo de Grado para optar al título de Ingeniero Electrónico

Director

Jaime Guillermo Barrero Pérez

Magíster en potencia eléctrica

Codirector

Jeyson Arley Castillo Bohórquez

Ingeniero Electrónico

Universidad Industrial de Santander

Facultad de Ingenierías Fisicomecánicas

Escuela de Ingenierías Eléctrica, Electrónica y de Telecomunicaciones

Bucaramanga

2021

Agradecimientos

Agradecemos a todas las personas que apoyaron el desarrollo y la ejecución de este trabajo de grado, familiares, amigos, compañeros y profesores. En especial a nuestro director Jaime Barrero y nuestro codirector Jeyson Castillo, por su apoyo, confianza y constante guía para poder dar forma a este proyecto.

Este logro de finalizar nuestra carrera y obtener el título de ingenieros electrónicos pudo hacerse realidad gracias a Dios y a ustedes. A pesar de tantas dificultades en el camino, siempre contamos con su apoyo, consejos, acompañamiento y enseñanzas.

Dedicatoria

Dedico este proyecto de grado principalmente a mis padres, Olga Bayona y Arilson Thomas, por enseñarme los valores y principios necesarios para crecer cada día como persona y por ser siempre un apoyo incondicional. A mi hermana Vanessa Thomas, por ser mi mejor amiga, mi consejera y sobre todo mi compañera de vida, a mi hermano Andrés Thomas, por ser mi ejemplo de superación y permanecer en cada decisión que tomara a lo largo de mi vida. Y por último a mis amigos, por estar en cada parte de este proceso sin importar las dificultades presentadas.

Lo logré y todo gracias a ustedes.

Angie Daniela Thomas Bayona

Este proyecto de grado, que por cierto fue una meta que algunas veces vimos muy difícil de alcanzar, se lo dedico a mi madre Vilma Victoria por haber estado y estar siempre conmigo en todo este proceso, por su amor, orientación, apoyo incondicional y compañía durante toda mi vida y carrera universitaria, lo que me ha permitido convertirme en el hombre que soy. También a mi padre Gilberto Lelio, que siempre me ha brindado su amor, su apoyo y ha estado ahí para verme convertir en hombre e ingeniero. Sin ellos este logro no hubiese sido posible. Y finalmente a todos aquellos compañeros y amigos verdaderos que de una u otra forma me han brindado su ayuda y apoyo durante mi carrera universitaria.

David Gilberto Olarte Velasco

Tabla de Contenido

Lista de Figuras

Lista de Tablas

GLOSARIO

Archivo H5: archivo de datos guardado en el formato de datos jerárquicos (HDF). Este archivo contiene matrices multidimensionales de datos científicos.

Archivo XML: *eXtensible Markup Language*, estos archivos se componen de etiquetas que aportan datos e información la cual se desea procesar. Entre más grande sea un fichero, más información trae.

Base de datos: es un conjunto organizado de información estructurada o datos que son almacenados electrónicamente en un sistema.

Callbacks: serie de funciones que se utilizan para echar un vistazo a los estados internos del modelo entrenado y detener el proceso en la mejor época posible.

Clasificador en cascada: se entrenan con varias imágenes de entrenamiento positivas y negativas. Al entrenar el clasificador, se aplica una región de una imágen y se detecta la imágen en cuestión, generando un 1 si se muestra la región en la imágen o 0 en caso contrario.

Detección facial: tecnología del campo de la inteligencia artificial basada en técnicas, métodos o algoritmos capaces de verificar un rostro a través de una imágen, vídeo o elemento audiovisual.

Estrategia de datos aumentados: *"data augmentation"* técnica que permite aumentar el set de datos para mejorar la precisión, la generalización y controlar el *overfitting*.

Épocas: hiperpárametro que determina el número de veces que el algoritmo de aprendizaje funcionará a través del conjunto de datos de entrenamiento. Una época se compone de uno o más lotes.

Google Drive: servicio de almacenamiento de datos en internet que provee google en su versión gratuita. Tiene una capacidad de almacenamiento de 15 GB.

GPU: *"graphics processing unit"* es el centro de una tarjeta gráfica, su función principal es realizar cálculos complejos y mejorar la eficiencia y el rendimiento del computador.

Image Data Generator: clase de keras que acepta los datos originales los transoforma aleatoriamente y devuelve solo los datos nuevos transformados.

Kaggle: plataforma en línea de acceso público gratuito con recursos para solucionar problemas con temáticas como análisis predictivos, Machine Learning y bases de datos.

Keras: biblioteca de código abiero (con licencia MIT) escrita en python. El objetivo de esta biblioteca es acelerar la creación de redes neuronales, funcionando como una interfaz de uso intuitivo (API) que permite acceder a varios frameworks de aprendizaje automático y desarrollarlos.

Kernel: matriz encargada de extraer la misma característica en cualquier parte de la entrada, con el fin de reducir el número de conexiones y el número de parámetros a entrenar.

Oragnización mundial de la salud: *OMS* autoridad directiva y coordinadora de asuntos de sanidad internacional en el sistema de las Naciones Unidas. Es la organización responsable de desempeñar una función de liderazgo en los asuntos sanitarios mundiales, investigaciones en salud, establecer normas, articular opciones de política, prestar apoyo y vigilar las tendencias sanitarias.

OpenCV: librería de software abierto de visión artificial y machine learning. Estos algoritmos permiten identificar objetos, caras, clasificar acciones humanas en vídeo, hacer *tracking* de movimientos de objetos, extraer modelos 3D, encontrar imágenes similares, entre muchas más.

Pesos: son los coeficientes dentro de la red que determinan la intensidad de la señal de entrada

registrada por la neurona artificial.

Preprocesamiento: consiste en simplificar la imágen a la región que corresponde el objeto de interés, para reducir tiempo de procesamiento de las etapas posteriores y eliminar objetos de fondo que no aportan información al proceso.

Python: lenguaje de programación de código abierto, orientada a objetos. Tiene una sintaxis sencilla que cuenta con una amplia biblioteca de herramientas.

Matriz de confusión: la matriz de confusión es una herramienta que evalua el desempeño de un algoritmo de clasificación, a partir de un conteo de los aciertos y errores de cada una de las clases en la clasificación.

Red en cascada: la red en cascada se diseña bajo el esquema de crecimiento en el tamaño de la red o aprendizaje constructivo, es necesario conocer el número de neuronas ocultas requeridas, esto con el fin de que el aprendizaje sea más rápido.

Región de interés: área o región de una imágen definido por un contorno sobre la que se estudia o evalúa una característica determinada.

RGB: *(rojo, verde, azul / red, green, blue)* modelo crómatico formado a partir de los colores primarios. Empleando la luminosidad del rojo, verde y el azul, se produce el resto de colores.

Sobreajuste: se produce cuando un sistema de aprendizaje automático se entrena demasiado o con datos anómalos, generando que el algoritmo memorice y no aprenda.

Tamaño de paso de entrenamiento: *step size training*, número de datos en el conjunto de entrenamiento sobre el tamaño del lote.

TensorFlow: biblioteca de software de código abierto, utiliza gráficos de flujo de datos. Es una

gran plataforma para construir y entrenar redes neuronales, que permiten detectar y descifrar patrones y correlaciones, análogos al aprendizaje y razonamiento usados por los humanos.

Resumen

Título: Desarrollo de un sistema autónomo de detección de somnolencia en conductores usando redes neuronales profundas. [*]

Autores: Angie Daniela Thomas Bayona y David Gilberto Olarte Velasco. [**]

Palabras Clave: Accidentes de tránsito, Detección automática, Redes neuronales profundas (DNN), Somnolencia.

Descripción: Los trastornos por fatiga y sueño representan en muchos países la primera causa de mortalidad la primera causa de muerte por accidente de tráfico. En la actualidad, se están desarrollando sistemas electrónicos inteligentes de apoyo a la conducción con el fin de evitar este tipo de accidentes. Este proyecto se orienta al desarrollo de un sistema autónomo para la detección de la somnolencia en los conductores mediante el reconocimiento de signos o síntomas asociados a este fenómeno, usando Redes Neuronales Profundas (DNN). Para tal fin, se implementa un esquema de detección en cascada para dos patrones usuales en la somnolencia: los ojos cerrados y los bostezos. El sistema consta de tres partes, la primera es el preprocesamiento de la imágen de entrada que consta de un detector de rostros y de ojos en cascada para preparar la imágen para la siguiente etapa. La segunda es la acción en paralelo de dos DNN's para la detección de ojos cerrados y bostezos; cada red es distinta e independiente. Por último, un clasificador final que toma en cuenta la duración de estos patrones para determinar si hay índices fuertes de somnolencia. Las redes independientes arrojaron una exactitud del 99.59 % para la red de ojos cerrados y 99.01 % para la de los bostezos. El sistema final emitirá alertas en caso de que el discriminante detecte somnolencia para cumplir la función de prevenir accidentes de tránsito por somnolencia.

---

[*] Trabajo de grado

[**] Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingenierías Eléctrica, Electrónica y telecomunicaciones. Director: Jaime Guillermo Barrero Pérez, Msc. en potencia eléctrica. Codirector: Jeyson Arley Castillo Bohórquez, Ing. Electrónico

Abstract

Title: Development of an autonomous driver drowsiness detection system using deep neural networks. *

Authors: Angie Daniela Thomas Bayona and David Gilberto Olarte Velasco. **

Keywords: Traffic accidents, Automatic detection, Deep neural networks (DNN), drowsiness.

Description: Fatigue and sleep disorders represent the leading cause of death due to traffic accidents in many countries in the world. At present, intelligent electronic systems for driving support are useful to prevent these types of accidents. This project uses Deep Neural Networks (DNN) to develop an autonomous system for the detection of drowsiness in drivers by recognizing signs or symptoms associated with this phenomenon. To this end, a cascade detection scheme is implemented for two common patterns in sleepiness: closed eyes and yawning. The system consists of three parts, the first is the pre-processing of the input image that consists of a face detector and a cascade eye detector to prepare the image for the next stage. The second is the parallel action of two DNN's for the detection of closed eyes and yawns; each network is different and independent. finally, a final classifier that takes into account the duration of these patterns to determine if there are strong indices of drowsiness. The independent networks returned an accuracy of 99.59 % for the closed eyes network and 99.01 % for the yawning network. The final system will issue alerts if the discriminant detects drowsiness to fulfill the function of preventing traffic accidents due to drowsiness.

---

* Bachelor Thesis

** Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingenierías Eléctrica, Electrónica y telecomunicaciones. Director: Jaime Guillermo Barrero , Msc. en potencia eléctrica Co-director: Jeyson Arley Castillo, Ing. Electrónico

Introducción

A medida que la tecnología ha avanzado con el pasar de los años hasta los tiempos actuales, el hombre ha buscado automatizar las tareas de tal manera que su intervención en ellas sea la mínima posible; de esta búsqueda ha surgido la inteligencia artificial IA. La IA se está implementando actualmente en múltiples áreas de aplicación, lo que ha permitido que se desarrolle y avance más rápido. Entre las técnicas más destacadas que ofrece la IA, están las redes neuronales. Las redes neuronales son un tipo de algoritmo o modelo compuesto por neuronas artificiales interconectadas para transmitir información y obtener una respuesta ante un estímulo, lo cual es un comportamiento similar al del cerebro humano. Entre las aplicaciones de las redes neuronales, una de las más notorias y comúnmente utilizadas es la clasificación mediante imágenes.

En el presente trabajo de grado se evidenció como las redes neuronales, específicamente las de tipo convolucional, son aplicables y eficaces a la hora de clasificar imágenes, hecho que se comprobó en el caso específico de la detección de signos de somnolencia en conductores como método preventivo para evitar accidentes de tránsito e informarle al conductor sobre su estado.

El proceso de aprendizaje de una red neuronal necesita de una base de datos de imágenes, generalmente amplia, que le permita entender como diferenciar características de los datos con los cuales va a trabajar. Para este proyecto, el cual implementa dos redes neuronales, se seleccionan dos bases de datos con dos subconjuntos cada una, representando estos las clases de cada red. Las imágenes pasan por un proceso de detección de objetos con el fin de extraer porciones que

contienen características que optimizan las respuestas de las redes ante una entrada. Finalmente, un clasificador final, el cual toma en cuenta las clasificaciones individuales de las dos redes, hace la predicción de estado del conductor y se encarga de enviar una alerta de ser necesario. Las pruebas finales del sistema se llavaron a cabo con una base de datos propia, compuesta de imágenes de diferentes personas con previa autorización. Todo el trabajo del proyecto se realizó en el entorno de Google Colaboratory, que dispone de una capacidad de procesamiento robusta en la nube, lo que facilita todo el proceso de ejecución del sistema sin necesidad de consumir recursos propios del equipo de computo que se utiliza.

1. Objetivos

1.1. Objetivo general

Desarrollar un sistema autónomo de detección de somnolencia en conductores basado en imágenes del rostro, usando redes neuronales profundas.

1.2. Objetivos específicos

Acondicionar una base de datos con imágenes características para el entrenamiento y la validación de la red neuronal;

Seleccionar una arquitectura disponible para la red neuronal profunda de forma que sea lo más eficiente posible para detectar el estado de los ojos y bostezos;

Evaluar el desempeño de la red para determinar su confiabilidad, teniendo en cuenta su capacidad de respuesta.

2. Accidentes de Tránsito

Los accidentes de tránsito son un evento involuntario que ocurren cuando un vehículo colisiona con otro vehículo, peatón, animal o cualquier obstrucción estacionaria a causa de factores que reducen la capacidad para detectar y reaccionar ante situaciones de riesgo que provoquen lesiones, muertes, daños materiales, daños humanos y costos financieros para las personas involucradas y para la sociedad con una estimación entre el 1-3 % del producto interno bruto PIB, alcanzando la cifra de 500 billones de dólares (Terán-Santos, J., y Egea, C., 2006). Entre las principales causas y razones de los accidentes de tránsito esta conducir bajo los efectos del alcohol, el exceso de la velocidad y la ausencia de sistemas de protección para el factor humano, ya que presentan comportamientos variables al momento de conducir como somnolencia, estrés, consumo de alcohol y otras sustancias psicoactivas, la ira, la hostilidad, la percepción de la conducción como una tarea difícil o las actitudes negativas hacia la conducción (Touahmia M., 2018) (Zuluaga, J., 2012) (ARL Sura, 2018) (Terán-Santos, J., y Egea, C., 2006).

Los accidentes de tránsito son la principal causa de muerte de los jóvenes y la octava causa principal de todos los fallecimientos a nivel mundial con 1,24 y 50 millones de heridos (Mohhammmed, A., Kamarudin, A., Mosa, A., y Syamsunur, D., 2019)(Toroyan, T., Peden, M., y Laych, K., 2009) (Pede, M., Scurfield, R., y Sleet, D., 2004) (Touahmia M., 2018) y en Colombia, la organización mundial de la salud OMS, reporta que los accidente de tránsito son la segunda causa de muerte con un 21 % de los fallecimientos registrados (ARL Sura, 2018). También aporta que la pérdida de

años de vida por discapacidad AVAD causadas por el tránsito ha incrementado considerablemente, pasando de estar de novena causa esencial de AVAD en el año 1998 a estar en la tercera causa en el año 2020, como se muestra en la siguiente tabla 1 (Mohhammmed, A., Kamarudin, A., Mosa, A., y Syamsunur, D., 2019).

Tabla 1
Posición de las lesiones por accidente de tránsito según el AVAD.

S. No	1998 Enfermedad o lesión	2020 Enfermedad o lesión
1	Infecciones de las vías respiratorias inferiores	Enfermedad isquémica del corazón
2	VIH / SIDA	Depresión mayor unipolar
3	Condiciones perinateles	Lesiones por accidentes de tránsito
4	Enfermedades diarreicas	Enfermedad cerebrovascular
5	Depresión mayor unipolar	Enfermedad pulmonar obstructiva crónica
6	Enfermedad isquémica del corazón	Infecciones de las vías respiratorias inferiores
7	Enfermedad cerebrovascular	Tuberculosis
8	Malaria	Guerra
9	Lesiones por accidentes de tránsito	Enfermedades diarreicas
10	Enfermedad pulmonar obstructiva crónica	VIH / SIDA

2.1. Somnolencia y signos de detección

La somnolencia es la tendencia a quedarse dormido,también conocido como la propensión a dormirse o la habilidad de transición de la vigilia al sueño y está determinada por la calidad del sueño, cantidad de sueño y el ritmo circadiano (Rosales, E., y De Castro, J., 2010). El sueño y la vigilia están controlados por factores homeostáticos y circadianos, donde la duración de la vigilia previa determina la propensión a la somnolencia, mientras que los factores circadianos determinan la cronología, la duración y las características del sueño. Existen dos tipos de somnolencia,

la objetiva y la subjetiva. La somnolencia objetiva es la tendencia del cuerpo a la somnolencia, donde se determinan periodos de gran vulnerabilidad de 3:00 a 5:00 a.m. y 3:00 a 5:00 p.m., en estos periodos de tiempo se observa un mayor número de accidentes relacionados con el sueño. La somnolencia subjetiva es la percepción de la somnolencia por parte del individuo y puede deberse a desordenes del sueño (apenas del sueño), condiciones médicas (desordenes que promueven insomnio), medicamentos, cafeína, drogas estimulantes, horarios de trabajos extensos y estilos de vida. (Chokroverty, S., 2011), (Rosales, E., y De Castro, J., 2010)

La somnolencia, se presenta cuando hay una pausa de respiración o cuando la respiración se vuelve superficial (Fundación CEA, 2015), al presentar las pausas de respiración se detectan unos signos que dan aviso a la somnolencia como picor de ojos, pesadez, parpadeos de mayor duración, dolor de cabeza, sensación de brazos dormidos, bostezos, cuya duración está entre 5 y 10 segundos (Medrano, J., 2013), y movimientos de la cabeza que pueden ser detectados antes de que ocurra una situación crítica como un micro sueño o adormecimiento parcial.(Agencia Nacional de Seguridad Vial, 2020) La duración de un microsueño está determinada entre 3 y 30 segundos, cuando se presenta es peligroso dado que a pesar de que dura poco, la persona no está consciente y no reacciona de forma voluntaria con coordinación motora, lo que puede ocasionar accidentes fatales (Calderón, A., y Balcero, P., 2020).

2.1.1. *Accidentes de tránsito por somnolencia*.

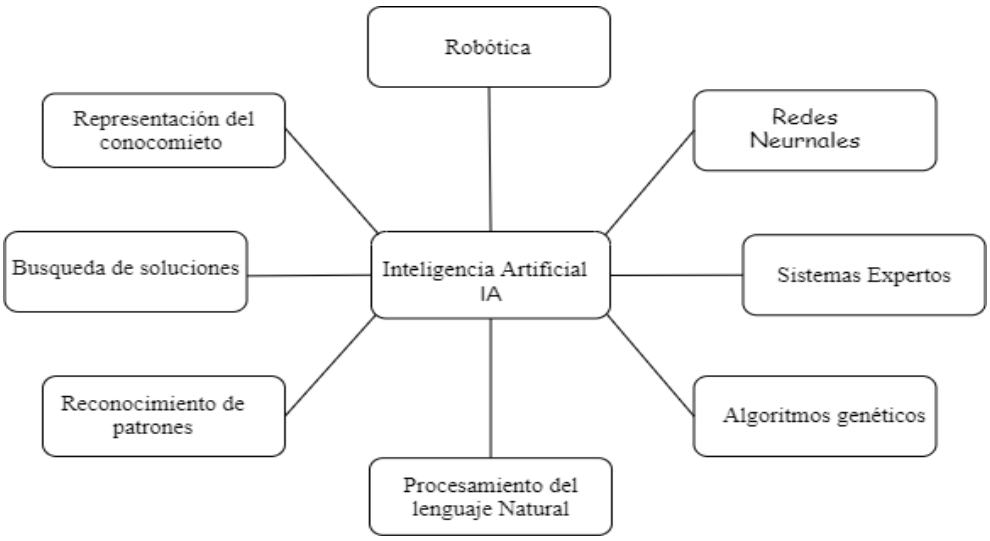
Los accidentes de tránsito por somnolencia están claramente relacionados con una de las principa-

les causas de mortalidad, la *National Highway Traffic Safety Administration* calcula que el 2,5 % de los accidentes fatales y el 2 % de los accidentes no fatales están relacionados con la presencia de somnolencia, debido a la disminución en la atención y el reconocimiento de riesgos de los conductores, lo que provoca una enorme cantidad de heridos, muertes y costos económicos. El rango de accidentes por somnolencia a nivel mundial está entre 12 y 25 % de los accidentes registrados (Ripoll, M., y Hernández, L., 2016) (Pede, M., Scurfield, R., y Sleet, D., 2004) (Terán-Santos, J., y Egea, C., 2006) (Garcés, A., Orosco, L., y Laciar, E., 2014).

3. Inteligencia Artificial

La IA es una disciplina enfocada en la computarización del comportamiento humano en los ámbitos de la comprensión, percepción, resolución de problemas, toma de decisiones, y procesos de reconocimiento de patrones, por lo cual las aplicaciones de la IA se enfocan principalmente en la simulación de actividades del hombre, es decir, imitan y hasta mejoran las actividades mentales del hombre por medio de máquinas electrónicas (Thomas, H., 2001) (SGMA, 2017). La IA utiliza diversas herramientas para aportar solución a los problemas, estas herramientas presentan varias técnicas (Figura 1) las cuales proveen elementos fundamentales en las áreas a utilizar la IA y tienen un método que utiliza el conocimiento de tal forma que, represente generalizaciones, sea comprendido por las personas que lo proporcionen, sea fácil de modificar y puede usarse en gran cantidad de situaciones (SGMA, 2017).

Figura 1.
Las técnicas de la IA.



Nota: Modificado de (SGMA, 2017)

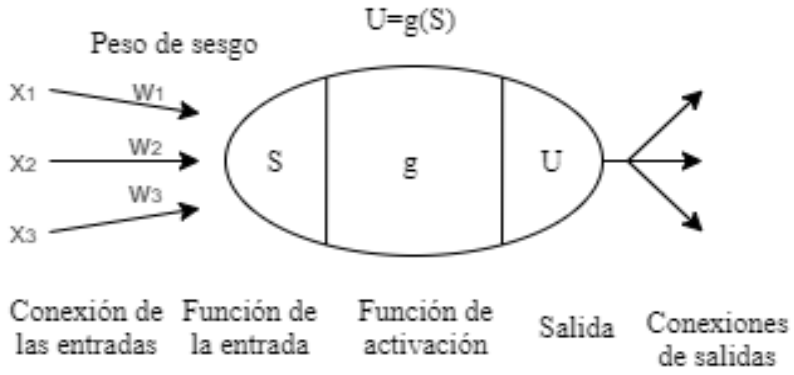
3.1. Redes neuronales artificiales

Las redes neuronales artificiales RNA simulan ciertas características propias del sistema nervioso central biológico como la capacidad de memorizar y de asociar hechos a través de neuronas artificiales de una manera efectiva y eficiente, por esta razón, una RNA se considera un modelo artificial y simplificado del cerebro humano, constituida por un conjunto de funciones matemáticas que se interconectan en una estructura en paralelo, siguiendo una organización jerárquica.(Flórez, R., y Fernández, J., 2008) (Basogain, X., 2008)

La neurona artificial mostrada en la Figura 2, fue la primera neurona creada por W. McCulloch y W. Pitts, de la universidad de Chicago en el año 1943. Se trata de un modelo binario cuyo estado es 1 activo o 0 inactivo (Gómez, F., Fernández, M., y López, M., 1994), este modelo

periódicamente actualiza su estado de la siguiente forma, primero cada unidad calcula la suma total de cada una de las entradas; luego se debe aplicar una función de activación g a esta suma para finalmente producir una salida de la unidad (ecuación 1). La función de activación debe ser no lineal y se utiliza para activar la unidad cuando se tengan entradas correctas y desactivadas si las entradas son erradas (Russel, S., 2004) (Gómez, F., Fernández, M., y López, M., 1994).

Figura 2.
Modelo de una neurona artificial.



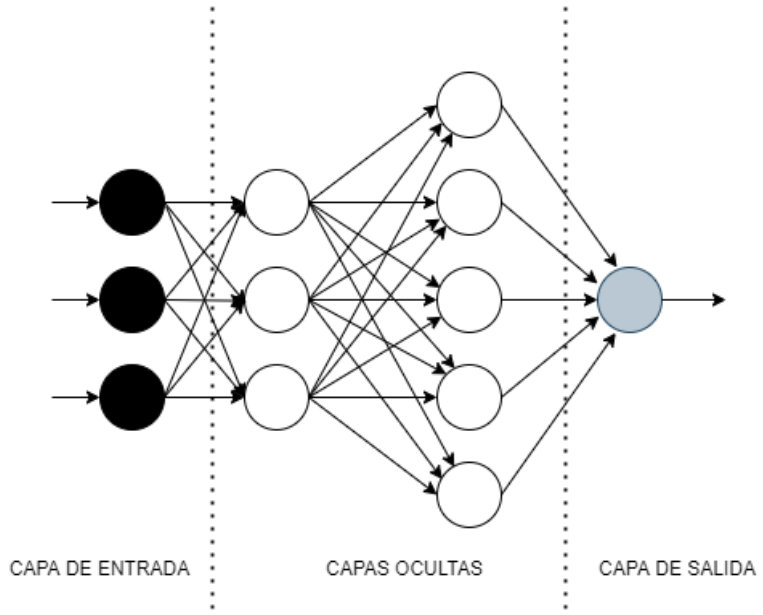
Nota: Adaptado de (Russel, S., 2004)

$$S = \sum_{j=0}^{n} = g(w_n X_n) \tag{1}$$

La distribución de neuronas dentro de la red se realiza formando capas o niveles. En general, su estructura se divide en tres niveles como se muestra en la figura 3; en primer nivel, se encuentra la capa de entrada, esta capa se encarga de recibir los datos externos que serán procesados; en el segundo nivel, se encuentra las capas ocultas, son internas a la red y no tienen relación directa con la información de entrada o con la salida y su número de capas ocultas pueden estar entre cero y un

número elevado (Bohorquez, P., y Villanueva, J., 2006); en el tercer nivel, se encuentra la capa de salida, en esta capa se procesa la información para finalmente dar respuesta al exterior.(Basogain, X., 2008)

Figura 3.
Esquema de una red neuronal multicapa.

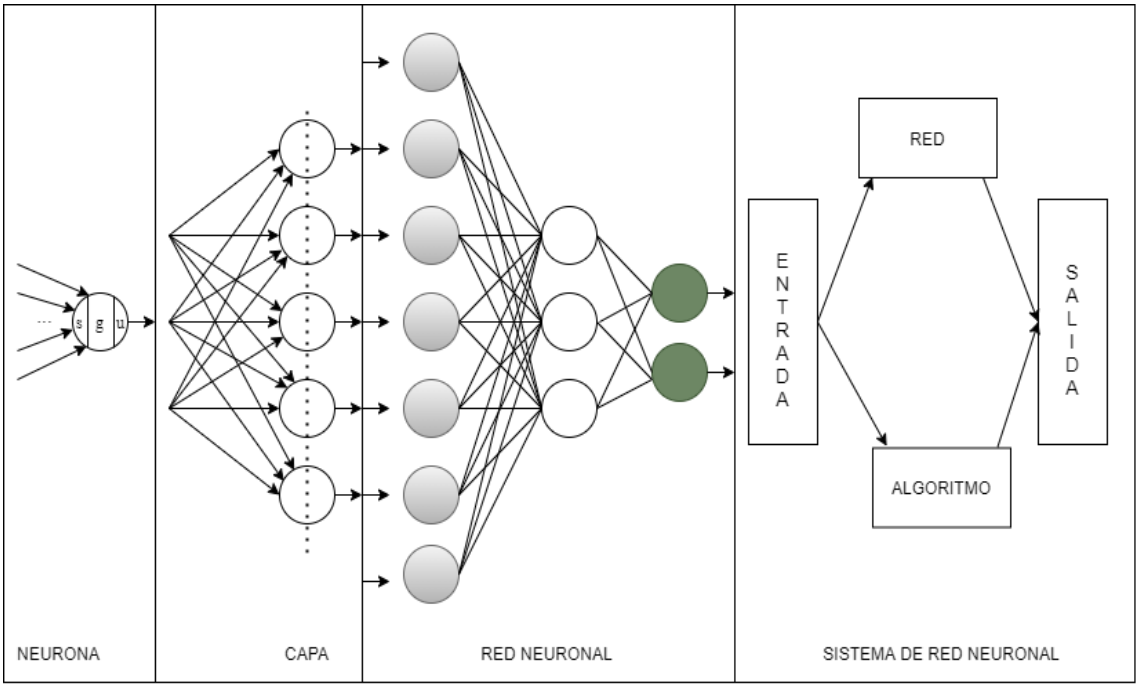


Nota: En esta figura se presenta un esquema de una red neuronal multicapa que tiene tres unidades en la capa de entrada, dos capas ocultas con tres y cinco unidades y por ultimo una unidad en la capa de salida. Modificado de (Bohorquez, P., y Villanueva, J., 2006)

El elemento básico de un sistema de red neuronal es la neurona artificial, la agrupación entre varias de ellas forman un conjunto de neuronas artificiales (cuyas entradas provienen de la misma fuente y cuyas salidas se dirigen al mismo destino) que constituyen las capas y estas capas interactuando entre ellas conforman lo que se denomina red neuronal. Esta agrupación más un algoritmo computacional forman el sistema de red neuronal como se muestra en la figura 4

(Flórez, R., y Fernández, J., 2008).

Figura 4.
Sistema de una red neuronal.



Nota: Modificado de (Flórez, R., y Fernández, J., 2008)

3.1.1. *Aprendizaje de una red neuronal.*

El aprendizaje en una red neuronal consiste en un proceso donde la red crea, modifica o destruye sus conexiones entre neuronas en respuesta a una información de entrada. Durante este proceso de aprendizaje, los pesos de las conexiones de la red sufren las respectivas modificaciones dejando los valores de sus pesos estables, indicando que la red neuronal ha terminado su proceso, es decir, ha aprendido (Flórez, R., y Fernández, J., 2008) (Basogain, X., 2008) (Hyttsten, M., Delgado, J., y Bailey P., 2021). Existen dos tipos de reglas de aprendizaje que se utilizan para la actualización de los pesos, el aprendizaje supervisado y el aprendizaje no supervisado, estos se caracterizan por el

campo de reconocimiento de patrones y se diferencian en la existencia o no de un agente externo que controla el aprendizaje de la red (Flórez, R., y Fernández, J., 2008) (Basogain, X., 2008).

3.1.1.1. Aprendizaje supervisado. El aprendizaje supervisado se caracteriza por tener la presencia de un agente externo que controla el proceso de entrenamiento, determinando la respuesta que debería generar la red a partir de una entrada determinada. De esta forma, el supervisor controla la salida de la red y en caso de que ésta no sea como la deseada, se modifican los pesos de las conexiones mediante iteraciones, con el fin de conseguir que la salida obtenida se aproxime a la deseada (Flórez, R., y Fernández, J., 2008) (Basogain, X., 2008). Dentro de este tipo de aprendizaje se consideran tres formas para llevarlo a cabo; aprendizaje por corrección de error, se basa en el ajuste por pesos de las conexiones de la red a partir de la diferencia entre los valores deseados y los obtenidos por el sistema (Flórez, R., y Fernández, J., 2008); aprendizaje por refuerzo, en este la función del supervisor se reduce a indicar mediante una señal de refuerzo si la salida obtenida de la red se ajusta a la deseada y en función a ello se ajustan los pesos (Flórez, R., y Fernández, J., 2008); aprendizaje estocástico, se basa en realizar cambios aleatorios en los valores de los pesos y evaluar su efecto a partir del objetivo deseado (Flórez, R., y Fernández, J., 2008).

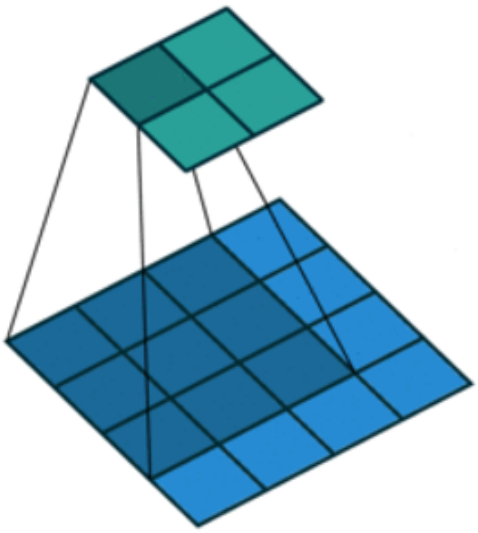
3.1.1.2. Aprendizaje no supervisado. El aprendizaje no supervisado no cuenta con una fuente externa para ajustar los pesos de las conexiones entre las neuronas. La red no recibe información externa que indique si la salida generada es correcta o no, por este motivo las redes deben encontrar las características, correlaciones o categorías que se establezcan entre los datos que se presenten a la entrada, por esto el algoritmo de entrenamiento es el encargado de modificar los pesos de la red de modo que produzca salidas consistentes (Pelaez, N., 2012) (Bibing.us, 2000).

Se encuentran dos tipos de algoritmos de aprendizaje no supervisado; aprendizaje hebbiano, mide la familiaridad o extrae características de los datos de entrada; aprendizaje competitivo y comparativo, se basa en la clasificación de los datos de entrada y cuya característica principal de este aprendizaje es que, si un patrón nuevo se determina que pertenece a una clase reconocida previamente, entonces la inclusión de este nuevo patrón a esta clase matizará la representación de la misma (Bibing.us, 2000).

3.1.2. *Redes neuronales convolucionales*.

Las redes neuronales convolucionales CNN son un tipo de red multicapa que consta de diversas capas convolucionales y capas de agrupación, con una topología capaz de trabajar con entradas de más de una dimensión y de submuestreo (*pooling*) alternadas. Las CNN hacen operaciones matemáticas de convolución entre matrices (Figura 5) y reducción de tamaño espacial mediante agrupación (Saha, S., 2018) (Shin, H., y Roth, H., 2016).

Figura 5.
Matrices para hacer operaciones matemáticas de convolución.



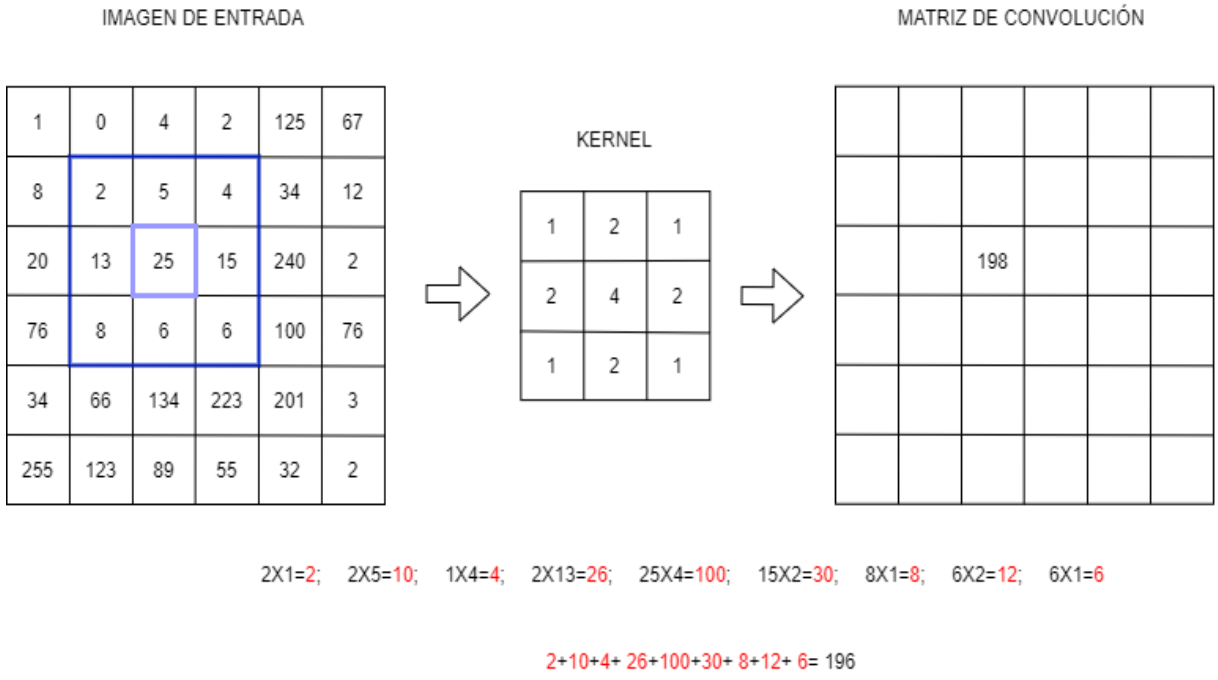
Nota: Se muestra una matriz azul representando la imagen de entrada, la matriz sombreada en la imagen de entrada corresponde al kernel y la matriz verde simboliza la matriz convolución.

Tomado de: (Saha, S., 2018)

La capa de convolución consiste en la detección de patrones de alto nivel de una imágen, esto mediante la toma de grupos de pixeles cercanos de la imágen de entrada los cuales se irán operando matemáticamente contra una pequeña matriz llamada kernel. El kernel es una matriz de menores dimensiones de la imágen que contiene la misma cantidad de canales, esta recorre la matriz de entrada en dirección izquierda-derecha con un determinado paso hasta que analiza el ancho de la misma y salta al principio (izquierda) y va de arriba-abajo con el mismo valor de paso realizando operaciones matemáticas en su recorrido hasta generar una nueva matriz llamada matriz

de convolución. A continuación en la Figura 6, se muestra un ejemplo del proceso de convolución, donde se puede observar la imágen de entrada, el kernel respectivo, el proceso matemático y la matriz de convolución generada (Saha, S., 2018) .

Figura 6.
Ejemplo del proceso de convolución.

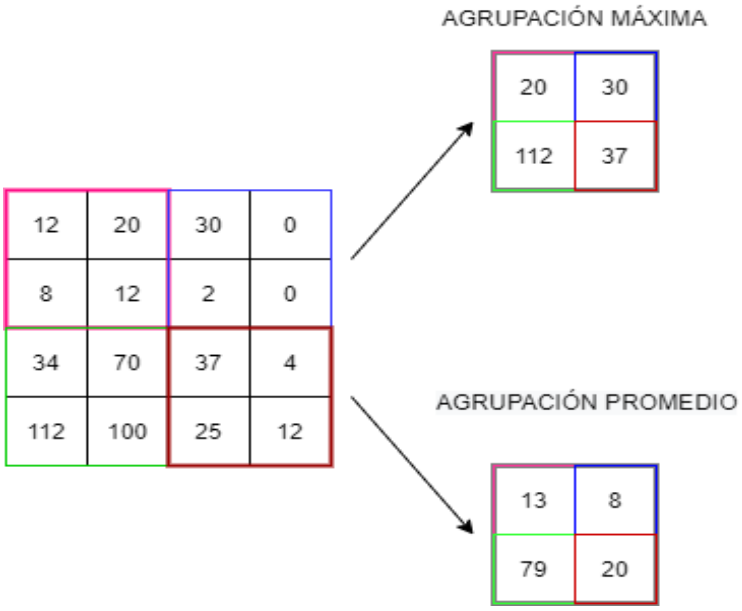


Nota: Adaptado de (Saha, S., 2018)

La capa de agrupación es la encargada de reducir el tamaño espacial de la matriz convolución, esto se hace con el fin de disminuir la potencia computacional requerida para procesar los datos a través de la reducción de dimensionalidad. Por otra parte, es útil para extraer características dominantes que son invariantes rotacionales y posicionales. En el proceso de agrupación existen dos tipos de agrupación la máxima y promedio (Figura 7); la agrupación máxima, devuelve el valor máximo de la parte de la imágen cubierta por el kernel, actúa como supresor de ruido, descarta

las activaciones ruidosas por completo y realiza la eliminación de ruido junto con la reducción de dimensiones; la agrupación promedio, devuelve el promedio de todos los valores de la parte de la imagen cubierta por kernel y realiza la reducción de dimensionalidad como un mecanismo de supresión de ruido. Por esta razón la agrupación que mejor funciona es la agrupación máxima (Saha, S., 2018).

Figura 7.
Ejemplo de agrupación máxima y promedio.



Nota: Adaptado de (Saha, S., 2018)

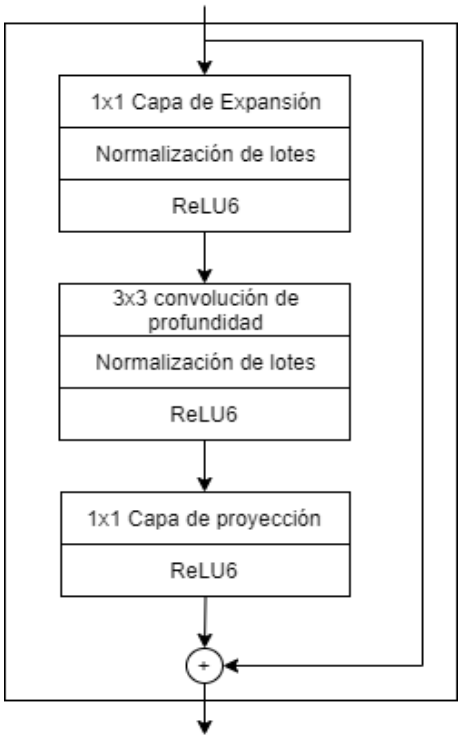
3.1.2.1. Arquitecturas CNN. Para el diseño del sistema autónomo para la detección de la somnolencia se utilizaron dos arquitecturas de CNN las cuales se describen a continuación.

MovbileNetV2

MobileNetV2 es una CNN, la cual se basa en una arquitectura optimizada que usa convoluciones separadas e introduce los conceptos de residuos invertidos (*inverted residuals*) y cuellos de

botella lineales (*linear bottlenecks*), permitiendo crear capas que toman entradas comprimidas en pocas dimensiones y las expande a más dimensiones, así permite la aplicación de filtros separables en profundidad para proyectar la salida a un número bajo de dimensiones mediante una convolución lineal (Velasco, J., 2019). Esta arquitectura utiliza tres capas convolucionales en serie como un bloque (Figura 8). Una capa es la de expansión, se le conoce como capa de cuello de botella (*bottleneck layer*), hace más pequeñas las imágenes que recorren la red e incorpora una convolución 1x1; la capa de convoluciones separables en profundidad (*depthwise separable convolutions*), formadas por una convolución puntual que se encarga de filtrar en la entrada, seguida de otra capa de convolución de 1x1 que combina los resultados filtrados para generar nuevas características (Velasco, J., 2019) (Montesdeoca, E., 2020) (García, N., 2019).

Figura 8.
Arquitectura de la red MobileNetV2.

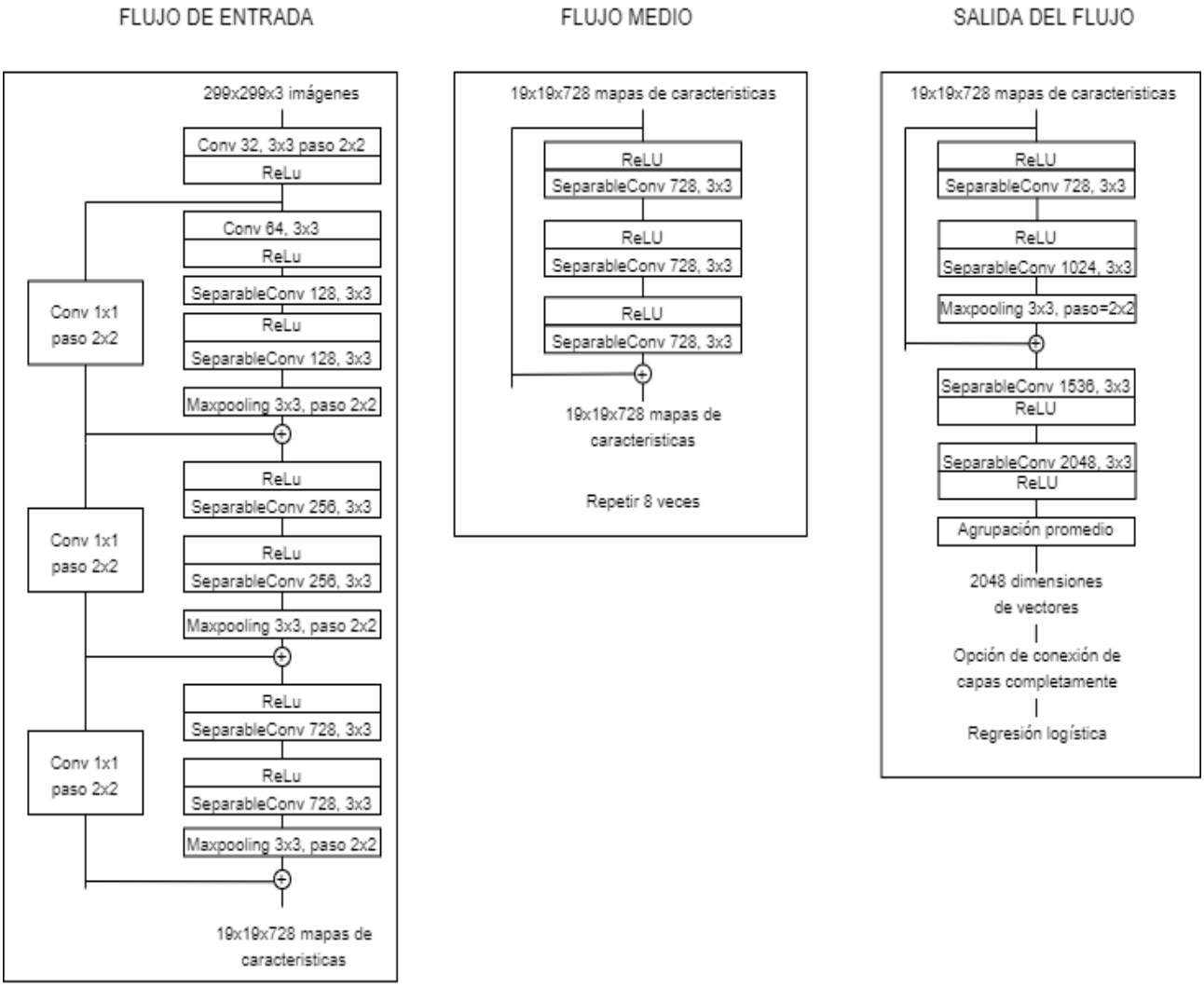


Nota: Adaptado de (Montesdeoca, E., 2020)

Xception

Xception es una arquitectura de CNN, la cual se basa en capas de convolución separables en profundidad y está compuesta por 36 capas convolucionales (Figura 9) que forman la base de extracción de características de la red, estas capas están estructuradas en 14 módulos, todos ellos con conexiones residuales lineales a su alrededor, a excepción del primer y último módulo. Esta arquitectura es sencilla y fácil de definir, solo necesita de 30 a 40 líneas de código utilizando una biblioteca de alto nivel (Chollet, F., 2017) como Keras o TensorFlow.

Figura 9.
Arquitectura de la red Xception.



Nota: Adaptado de (Chollet, F., 2017)

3.2. Transfer learning

El *transfer learning* es una técnica utilizada en *Deep learning* que ajusta y hace uso de alguna arquitectura de red neuronal con grandes datos y entrenada para solucionar un problema, esto con el fin de utilizar la información que esta red ya tiene y aplicarla a un nuevo problema.

El *transfer learning* en general primero entrena una red y reutiliza las características aprendidas o las transfiere a una segunda red para ser entrenadas en un conjunto de datos, este proceso solo funciona si las características son adecuadas para las tareas básicas en lugar de específicas para una tarea en particular (Brownlee, J., 2019).

4. Sistemas de Ejecución

4.1. Python

Python es un lenguaje de programación de alto nivel y multipropósito, utilizado en diversas plataformas y sistemas operativos, se pueden destacar los más populares, *Windows, -mac OS X y Linux* y también funciona en *smartphones*. Con este lenguaje es posible crear *software* para aplicaciones científicas, para comunicaciones de red y para aplicaciones de escritorio que utilicen una interfaz gráfica de usuario (GUI), con el fin de crear juegos y aplicaciones web. Entre las principales razones para elegir *Python* se encuentran que es un lenguaje potente, flexible y con una sintaxis clara y concisa, además no requiere dedicar tiempo a su compilación, otra razón importante a la hora de seleccionar *Python* como el lenguaje para programar es que es *open source*, cualquier persona puede contribuir a su desarrollo y divulgación, no es necesario pagar ninguna licencia para la distribución del *software* (Fernandez, A., 2013).

4.2. Google Colaboratory

Google Colaboratory también conocido Colab, es un entorno de servicio basado en *Jupyter Notebooks* de libre uso, este no requiere configuración y se ejecuta completamente en la nube. Con colab se puede escribir y ejecutar código, guardar y compartir análisis y acceder a potentes recursos informativos, todo gratis desde su navegador, sin necesidad de instalar nada y además los cuadernos se pueden crear, cargar, descargar y almacenar en Colab o guardar en Google Drive. Tiene como objetivo difundir la educación y la investigación sobre el aprendizaje profundo y tiene variedad de bibliotecas de aprendizaje profundo como PyTorch, Keras, TensorFlow y OpenCv (Carneiro, T., y

Medeiros, R., 2018) (Zhang, X., y Reveriano, F., 2019).

5. Metodología de Desarrollo

Para el desarrollo del sistema autónomo de detección de somnolencia en conductores usando DNN, se adquirió y estructuró una buena base de datos, con un número adecuado de imágenes divididas en varios subconjuntos, que son datos de entrenamiento, datos de validación y datos de prueba. Mediante estos subconjuntos de datos se llevarón a cabo los procesos de entrenamiento, validación y prueba de las redes. Estas redes se construyeron mediante el proceso de transfer learning a partir de modelos ya existentes, lo cual ayudó a conseguir una alta exactitud en el desempeño de la red.

Para obtener un desempeño lo más acertado posible en cuanto a las predicciones del modelo, se realizó una detección de regiones específicas del rostro con las cuales se ejecutó la predicción. Finalmente, con base en las predicciones y según la información sobre somnolencia analizada en capítulos anteriores, se proporcionó un aviso o alerta sobre el estado en el que se encuentra el conductor.

5.1. Adquisición y estructura de la base de datos.

Para detectar la somnolencia en conductores se seleccionó y acondicionó una base de datos amplia a partir de bases de datos de acceso libre y público proporcionadas por el repositorio Kaggle. Se analizaron dos características, ojos y bostezos, las cuales se dividieron en dos DNN's, una capaz de detectar si la persona tenia los ojos abiertos o cerrados y la otra red neuronal capaz de detectar si una persona estaba bostezando o no. Los datos en cada red son presentados en las

siguientes tablas; Tabla 2 y Tabla 3, muestran la estructura de los datos para el entrenamiento, validación y prueba de la red neuronal de los ojos y la red neuronal de los bostezos, respectivamente.

Tabla 2
Estructura de la base de datos para el entrenamiento, validación y prueba de la red neuronal de los ojos.

Subconjuntos	Categoría	Imágenes
Entrenamiento	Ojos abiertos	494
	Ojos cerrados	494
Validación	Ojos abiertos	123
	Ojos cerrados	123
Prueba	Ojos abiertos	120
	Ojos cerrados	120

Tabla 3
Estructura de la base de datos para el entrenamiento, validación y prueba de la red neuronal de los bostezos.

Subconjuntos	Categoría	Imágenes
Entrenamiento	Bostezos	448
	No bostezos	447
Validación	Bostezos	112
	No bostezos	112
Prueba	Bostezos	102
	No bostezos	102

5.2. Entrenamiento y validación de las redes neuronales

Para entrenar las redes, fue necesario identificar el tipo de red neuronal que se adapta mejor al objetivo del proyecto; como las entradas eran imágenes fue apropiado trabajar con las CNN. Los códigos fuente donde se realizó entrenamiento y validación de cada red están en (Thomas, A., y Olarte, D., 2021b).

Para empezar a programar se importaron las librerías y funciones necesarias (figura 10) y (figura 11).

Figura 10.
Librerías para la red de los ojos.

```
import tensorflow as tf
import glob
import shutil
import io
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, Flatten, Dropout, MaxPooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import layers
from keras.models import Model
from keras.optimizers import Adam
from keras.layers import Dense,GlobalAveragePooling2D
from keras.preprocessing.image import load_img, img_to_array
from keras.models import load_model
from sklearn.metrics import accuracy_score
import zipfile
import os
import numpy as np
import cv2
from google.colab.patches import cv2_imshow
import matplotlib.pyplot as plt
import sys
from os import listdir
from os.path import isfile, join
```

Figura 11.
Librerías para la red de los bostezos.

```
import tensorflow as tf
import glob
import shutil
import io
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, Flatten, Dropout, MaxPooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import layers
from keras.models import Model
from keras.optimizers import Adam
from keras.layers import Dense,GlobalAveragePooling2D
from keras.preprocessing.image import load_img, img_to_array
from keras.models import load_model
from sklearn.metrics import accuracy_score
import zipfile
import os
import numpy as np
import matplotlib.pyplot as plt
```

Para los procesos que realizan las redes neuronales en *Google Colaboratory*, fue indispensable que las bases de datos estuviesen organizadas por clases, lo cual se hizo mediante carpetas identificadas con el nombre de la clases. Estas clases mencionadas fueron las características que dieron la respuesta al planteamiento del problema.

Para la red neuronal de los ojos se contó con una base de datos especial con imágenes de ojos y para la red neuronal de los bostezos se contó con una base de datos especial con imágenes de bostezos . Las bases de datos fueron guardadas en *Google Drive* para mantenerlas enlazadas a Colab;

estos enlaces se realizaron mediante la asignación del directorio donde se encontraban los datos en Drive a una variable en Colab, como se muestra en las siguientes figuras.

Figura 12.
Manera de llamar la base de datos de la red de los ojos desde Drive a Colab.

```
from google.colab import drive
from google.colab import files
drive.mount('/content/drive/', force_remount=True)

base_direct='/content/drive/MyDrive/eyes'
classes = ['abiertos', 'cerrados']
```

Figura 13.
Manera de llamar la base de datos de la red de los bostezos desde Drive a Colab.

```
from google.colab import drive
from google.colab import files
drive.mount('/content/drive/', force_remount=True)

base_direct='/content/drive/MyDrive/bostezos'
classes = ['no_bostezo', 'si_bostezo']
```

La separación de los datos de entrenamiento y los de validación, se llevaron a cabo mediante códigos, los cuales distribuian los datos con un porcentaje del 80 % de los datos totales contenidos para entrenamiento y el 20 % restante para validación (figura 14). Esta separación se hizo igual

para las dos redes.

Figura 14.
Separación de datos de entrenamiento.

```
for c in classes:
  img_path = os.path.join(base_direct, c) # añade la carpeta de imagenes imgpath con la vairable cl que son las clases
  images = glob.glob(img_path + '/*.jpg') # glob busca todos los archivos jpg contenidos en imgpath
  print("{}: {} Images".format(c, len(images)))# imprime la clase con la cantidad de imagenes
  n_train = int(round(len(images)*0.8)) # saca el 80% de las imagenes para train
  train, valid = images[:n_train], images[n_train:]# acomoda el 80% en train y el 20 en val

  for t in train:
    if not os.path.exists(os.path.join(base_direct, 'train', c)):# si no existe el directorio train
      os.makedirs(os.path.join(base_direct, 'train', c)) # crearlo a partir de la carpeta basedir
    shutil.copy(t, os.path.join(base_direct, 'train', c))# shutil permite hacer operaciones con archivos como copiar y mover

  for v in valid:
    if not os.path.exists(os.path.join(base_direct, 'valid', c)):
      os.makedirs(os.path.join(base_direct, 'valid', c))
    shutil.copy(v, os.path.join(base_direct, 'valid', c))

train_direct = os.path.join(base_direct, 'train') #añade lo que hay en train a traindir
val_direct = os.path.join(base_direct, 'valid')
```

Se realizó un preprocesamiento de los datos ya que las imágenes de las bases de datos tenían tamaños diferentes y no eran compatibles con el modelo con el que se construyeron las redes. En las líneas de código se cambió el tamaño de las imágenes a 224 x 224 píxeles para la red de los ojos y a 480 x 480 píxeles para la red de los bostezos. Estos tamaños de imágen, en el caso de los ojos, se seleccionó ya que la arquitectura MobileNetV2 trabaja con ese tamaño de 224 píxeles, el cual no generó inconvenientes para el modelo; en el caso de la red de los bostezos, se ajustó el tamaño

de la imágen a un valor mayor a 299 píxeles, tamaño estandar para la arquitectura Xception, ya que esto proporcionaba al modelo una mejor resolución para encontrar los patrones que necesitaba la detección de bostezos y obtener una mayor excatitud. Se realizó un escalamiento a cada una de ellas (figura 15, figura 16), pues inicialmente cada píxel de la imágen tenía un valor entre 0 y 255, luego de escalarlas, los valores de los píxeles fueron entre 0 y 1, esto permitió más eficiencia en el entrenamiento.

Figura 15.
Redimensionamiento y escalamiento a las imágenes de entrenamiento y validación para la red de los ojos.

```
Image_shape = 224
batch_size = 64

train_idg = ImageDataGenerator(rescale=1./255)
train_data = train_idg.flow_from_directory(
                                           batch_size=batch_size,
                                           directory=train_direct,
                                           shuffle=True, #deja los datos en el orden en que estan
                                           target_size=(Image_shape,Image_shape),
                                           class_mode='binary' #tipo de clasificacion que se quiere
                                          )

valid_idg = ImageDataGenerator(rescale=1./255)
valid_data = valid_idg.flow_from_directory(batch_size=batch_size,
                                           directory=val_direct,
                                           target_size=(Image_shape, Image_shape),
                                           class_mode='binary')
```

Figura 16.
Redimensionamiento y escalamiento a las imágenes de entrenamiento y validación para la red de los bostezos.

```
Image_shape = 480
batch_size = 8

train_idg = ImageDataGenerator(rescale=1./255)
train_data = train_idg.flow_from_directory(
                                            batch_size=batch_size,
                                            directory=train_direct,
                                            shuffle=True, #deja los datos en el orden en que estan
                                            target_size=(Image_shape,Image_shape),
                                            class_mode='binary' #tipo de clasificacion que se quiere
                                           )

valid_idg = ImageDataGenerator(rescale=1./255)
valid_data = valid_idg.flow_from_directory(batch_size=batch_size,
                                           directory=val_direct,
                                           target_size=(Image_shape, Image_shape),
                                           class_mode='binary')
```

Con el fin de lograr una mejor precisión y prevenir el sobreajuste de la red, se aplicó la técnica de datos aumentados a los datos de entrenamiento. Esto se realizó mediante la herramienta *ImageDataGenerator*, con la cual se efectuaron sobre las imágenes las transformaciones de rotación, desplazamiento, acercamiento y espejo horizontal, como se muestra a continuación.

Figura 17.
Datos aumentados en entrenamiento para la red de los ojos.

```
train_idg = ImageDataGenerator(
                                rescale=1./255,
                                rotation_range=45,
                                width_shift_range=.15,
                                height_shift_range=.15,
                                horizontal_flip=True,
                                zoom_range=0.1
                               )
```

Figura 18.
Datos aumentados en entrenamiento para la red de los bostezos.

```
train_idg = ImageDataGenerator(
                    rescale=1./255,
                    rotation_range=15,
                    horizontal_flip=True,
                    shear_range=0.2,   ## se le agrego
                    zoom_range=0.2
                    )
```

5.2.1. *Transfer learning con MobileNetV2 para la red de los ojos y con Xception para la red de los bostezos*.

La construcción de la red de los ojos se hizo utilizando la técnica de *transfer learning* adaptando los modelos y entrenando con la base de datos ya mencionada. Se planeó probar las diferentes topologías de CNN que ofrece *keras*, pero debido a los resultados que arrojó la primera prueba

con el modelo MobileNetV2, este fue el seleccionado para construir la red. A continuación (Figura 19) se muestra la forma como se importó el modelo de manera parcial y la definición de las capas propias de la red para la aplicación definida.

Figura 19.
Modelo y construcción de la red de los ojos.

```
import keras
from keras.applications import MobileNetV2

Mobile = MobileNetV2(weights = 'imagenet', include_top = False, input_shape = (224,224,3))

for layer in Mobile.layers:
  layer.trainable = False

x=Mobile.output
x=GlobalAveragePooling2D()(x)
predicciones=Dense(2,activation='softmax')(x) #final layer with softmax activation

model = Model(inputs = Mobile.input, outputs = predicciones)
model.compile(optimizer='Adam',loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),metrics=['accuracy'])
```

La red de los bostezos se construyó utilizando igualmente la técnica de *transfer learning*. Se probó con las diferentes topologías de CNN que ofrece *keras*, como Xception, ResNet101V2 e InceptionV3. Después de realizar pruebas con los modelos ya mencionados, el modelo Xception fue el seleccionado para construir la red, pues este proporcionó una mejor precisión en prueba. En la Figura 20 se muestra la forma como se importó el modelo de manera parcial y la definición de las capas propias de la red para la aplicación definida.

Figura 20.
Modelo y construcción de la red de los bostezos.

```
import keras
from keras.applications import Xception

Mobile = Xception(weights = 'imagenet', include_top = False, input_shape = (480,480,3))

for layer in Mobile.layers[1:80]:
  layer.trainable = False

x=Mobile.output
x=GlobalAveragePooling2D()(x)
x=Dropout(0.2)(x)  ###dropout agregado
x = Dense(1024,activation='relu')(x)
x=Dropout(0.2)(x)
predicciones=Dense(2,activation='softmax')(x) #final layer with softmax activation
model = Model(inputs = Mobile.input, outputs = predicciones)
print(model.summary())

model.compile(optimizer='Adam',loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),metrics=['accuracy'])
```

Antes de comenzar el proceso de entrenamiento, se le indicó a la red los directorios tanto de entrenamiento y validación, así como el número de épocas y *step size training* (Figura 21, Figura 22).

Figura 21.
Inicio del entrenamiento de la red de los ojos.

```
step_size_train=train_data.n//train_data.batch_size
history = model.fit(train_data,
                    steps_per_epoch=step_size_train,
                    validation_data = valid_data,
                    epochs=15)

target_direct = './modelo/'
if not os.path.exists(target_direct):
  os.mkdir(target_direct)
model.save('./modelo/modeloojos.h5')
```

Figura 22.
Inicio del entrenamiento de la red de los bostezos.

```
from keras.callbacks import EarlyStopping, ReduceLROnPlateau, ModelCheckpoint

step_size_train=train_data.n//train_data.batch_size

checkpoint_filepath = '../content/modelobostezos.h5'

callbacks_list = [
    keras.callbacks.EarlyStopping(
        monitor='val_accuracy',
        patience=10),

    keras.callbacks.ReduceLROnPlateau(
        monitor='val_loss',
        factor=0.1,
        patience=4,
        verbose=1,
        min_lr=1e-12),

    keras.callbacks.ModelCheckpoint(
        filepath=checkpoint_filepath,
        #filepath='model20-12-2020_dp03031024.h5',
        save_weights_only=True,
        monitor='val_accuracy',
        mode = 'max',
        verbose=1,
        save_best_only=True)
]


history = model.fit(train_data,
                    steps_per_epoch=step_size_train,
                    validation_data = valid_data,
                    callbacks=callbacks_list,
                    epochs=80)

model.save('../content/modelobostezos.h5')
```

Nota: Se agregaron unos *callbacks* para detener el entrenamiento en el mejor modelo posible.

Los *callbacks* se encargan de ir almacenando el mejor modelo, sin importar cuantas épocas se hayan definido para el entrenamiento; además otorgan un número de épocas límite para analizar si es posible encontrar un modelo mejor y detener el entrenamiento si se llega a ese número, en caso de que al pasar varias épocas el modelo no presente mejores resultados.

5.3. Código de detección de rostro y ojos

El código para la detección de rostro y ojos (Thomas, A., y Olarte, D., 2021c) se llevó a cabo a partir de comandos especiales del módulo cv2 perteneciente a la librería OpenCV de *Python*. Esta librería, de uso libre y orientada a la visión artificial 2D y 3D, permite el procesamiento de imágenes y ofrece aplicaciones como reconocimiento facial y de gestos, lo cual es ideal para el propósito de este proyecto. Este código hizo una detección especifica inicialmente de rostros en las imágenes y posteriormente a estos rostros hizo una detección de los ojos, para luego separar esta sección de la imágen como una imágen independiente la cual sirvió como dato de entrada para la CNN de clasificación de ojos. Cabe resaltar que las imágenes de entrada para la CNN de detección de bostezos no se les realizó detecciones de objetos sobre ellas.

El código de detección inicialmente leyó las imágenes de un directorio y luego las almacenó en una matriz de imágenes para hacer una detección de varias imágenes sucesivas. Esto se logró con las siguientes líneas de código (Figura 23).

Figura 23.
Lectura de la carpeta de imágenes.

```
mypath = '/content/drive/MyDrive/014_cv2_Faces-and-Eyes-Detection-master/fotos_pruebas'

onlyfiles = [ f for f in listdir(mypath) if isfile(join(mypath,f)) ]
image_cc = np.empty(len(onlyfiles), dtype=object)
for n in range(0, len(onlyfiles)):
  image_cc[n] = cv2.imread( join(mypath,onlyfiles[n]) )
```

La biblioteca OpenCV ofrece diferentes tipos de clasificadores en cascada preentrenados para la detección de rostros y ojos. Para la clasificación se cargaron y usaron dos clasificadores, uno para detección de rostros y otro para detección de ojos, que se cargaron al entorno de Colab como archivos *XML* (figura 24).

Figura 24.
Carga de los clasificadores.

```
face_classifier = cv2.CascadeClassifier('/content/drive/MyDrive/014_cv2_Faces-and-Eyes-Detection-master/Haarcascades/haarcascade_frontalface_default.xml')
eye_classifier = cv2.CascadeClassifier('/content/drive/MyDrive/014_cv2_Faces-and-Eyes-Detection-master/Haarcascades/haarcascade_eye.xml')
```

Para poder hacer la detección especifica de los ojos en la imágen, se detectó primero el rostro y se demarcó el área en la que se encuentra (Figura 25).

Figura 25.
Separación de la cara.

```
faces = face_classifier.detectMultiScale(image_c, 1.2, 5)

 for (x,y,w,h) in faces:
     cv2.rectangle(image_c,(x,y),(x+w,y+h),(0,255,255), 3)


 face_region = image_c[y:y+h, x:x+w]
```

En la Figura 26 se muestra como se definió el área del rostro detectada como una imágen independiente y se muestra como se procedió a hacer la detección de los ojos sobre esta imágen.

Figura 26.
Detección de los ojos.

```
eyes = eye_classifier.detectMultiScale(face_region)

  for (eyes_x, eyes_y, eyes_w,eyes_h) in eyes:
          cv2.rectangle(face_region,(eyes_x, eyes_y),(eyes_x + eyes_w, eyes_y + eyes_h), (0,255,0),3)


eyes_region=face_region[eyes_y:eyes_y+eyes_h,eyes_x:eyes_x+eyes_w]
```

Una vez se detectaron los ojos, la porción de la imágen correspondiente se almacenó como una imágen a parte a la cual se le realizó el cambio de tamaño y preprocesamiento debido para ser compatible como entrada a la red de los ojos (Figura 27). Estas mismas acciones de cambio de tamaño y preprocesamiento se realizaron con la imágen completa, que es la entrada para la red de los bostezos.

Figura 27.
Separación y preprocesamiento del ojo.

```
#PREPROCESAMIENTO IMAGEN RED OJOS

  eyes_region_res= cv2.resize(eyes_region,dsize=(224,224), interpolation = cv2.INTER_CUBIC)

  np_image_data = np.asarray(eyes_region_res)
  np_final = np.expand_dims(np_image_data,axis=0)
  input_mean = 255
  input_std = 255
  input_data = (np.float32(np_final) - input_mean) / input_std
```

5.4. Predicciones en paralelo de los modelos y clasificador final del sistema

5.4.1. *Predicciones en paralelo de los modelos*.

Las CNN's de ojos y bostezos una vez entrenadas, validadas y probadas individualmente, se cargaron al entorno de Colab como archivos con extensión .h5, llamados modeloojos.h5 y modelobostezos.h5 (Thomas, A., y Olarte, D., 2021c), los cuales pesan 9 MB y 201 MB, respectivamente. Esto se llevó a cabo con las siguientes líneas de código.

Figura 28.
Carga de los modelos.

```
  #OJOS
model_direct_eyes = '/content/drive/MyDrive/014_cv2_Faces-and-Eyes-Detection-master/modeloojos.h5'
modelo_eyes = load_model(model_direct_eyes)

  #BOSTEZOS
model_direct_bost = '/content/drive/MyDrive/014_cv2_Faces-and-Eyes-Detection-master/modelobostezos.h5'
modelo_bost = load_model(model_direct_bost)
```

Una vez cargados los modelos, se procedió a hacer las predicciones individuales de cada red trabajando en paralelo (figura 29). Las redes realizaron una clasificación binaria debido a que cada una trabaja con dos clases diferentes, por ello la respuesta de la predicción de cada una fue un 1 o un 0, representando a cada clase.

Figura 29.
Código de predicciones.

```
#PREDICCIONES

  preds_e=modelo_eyes.predict(input_data)
  preds_e = tf.squeeze(preds_e).numpy()
  preds_eyes = np.argmax(preds_e, axis=-1)

  preds_b=modelo_bost.predict(input_data_b)
  preds_b = tf.squeeze(preds_b).numpy()
  preds_bost = np.argmax(preds_b, axis=-1)
```

5.5. Clasificador final

El código escrito para la clasificación final del sistema de detección de somnolencia basa su funcionamiento a partir de dos factores, las predicciones suministradas por cada una de las redes neuronales (ojos y bostezos) y el tiempo de ejecución del modelo completo que tiene tanto la detección de rostro y ojos como la clasificación en paralelo de las redes neuronales (Thomas, A., y Olarte, D., 2021c). El tiempo de ejecución se obtuvo con las siguientes líneas de código (Figura 30), el cual arrojó un valor de 1.5654 segundos.

Figura 30.
Código para el tiempo de ejecución.

```
from time import time

tiempo_inicial = time()

Cascada(image_cc[0])

tiempo_final = time()

tiempo_ejecucion = tiempo_final - tiempo_inicial

print ('El tiempo de ejecucion fue:',tiempo_ejecucion) #En segundos
```

Teniendo en cuenta los tiempos de duración de un micro sueño y un bostezo, enunciados en capítulos anteriores, y el tiempo de respuesta del modelo, se establecieron los rangos en este clasificador. El clasificador consta de dos categorías de decisión, las cuales son somnolencia, determinada por detección de bostezos y de ojos cerrados por mucho tiempo, y ausencia de somnolencia, determinada por deteccion de ojos abiertos. Además emite alertas cuando detecta signos de somnolencia, como se puede observar en la figura 31.

Figura 31.
Código del clasificador final.

```
a=0;
b=0;
c=0;
s=0;
for i in image_cc:
  preds_eyes, preds_bost=Cascada(i)

  if preds_eyes==1:
    a=a+1
    if a>=2:
      print('Puede estar experimentando somnolencia')
      s=1

  elif preds_bost==1:
    b=b+1
    s=1
    if 4<=b<=7:
      print('Puede estar experimentando somnolencia')

  else:
    a=0
    b=0
    print('Se encuentra en estado activo, sin signos de somnolencia')
    s=0
```

5.6. Construcción de base de datos para pruebas finales

Para la construcción de la base de datos se recolectaron fotografías de personas naturales con su respectiva autorización. Dentro de esta carpeta creada se encuentran fotografías de ojos abiertos, ojos cerrados y bostezos, todas ellas en formato JPG para sean compatibles con el módulo cv2. Esta carpeta se convierte en una matriz de imágenes para poder realizar las pruebas. La base de datos utilizada para las pruebas se encuentra en (Thomas, A., y Olarte, D., 2021a).

6. Resultados

En este capítulo se presentan los resultados de los procesos de entrenamiento, validación y prueba tanto de la red neuronal de clasificación de ojos como de la red de clasificación de bostezos. Estos resultados se muestran acompañados de curvas de exactitud y pérdida de entrenamiento y validación, además de matrices de confusión. La distribución de estas matrices consiste en que su diagonal principal contiene la cantidad de aciertos y en el resto de la matriz se muestra la categoría con la que se confundieron los resultados.

Se presentan también los resultados de las pruebas del sistema completo obtenidos del clasificador final y los inconvenientes presentados por los clasificadores utilizados en la detección de rostro y ojos.

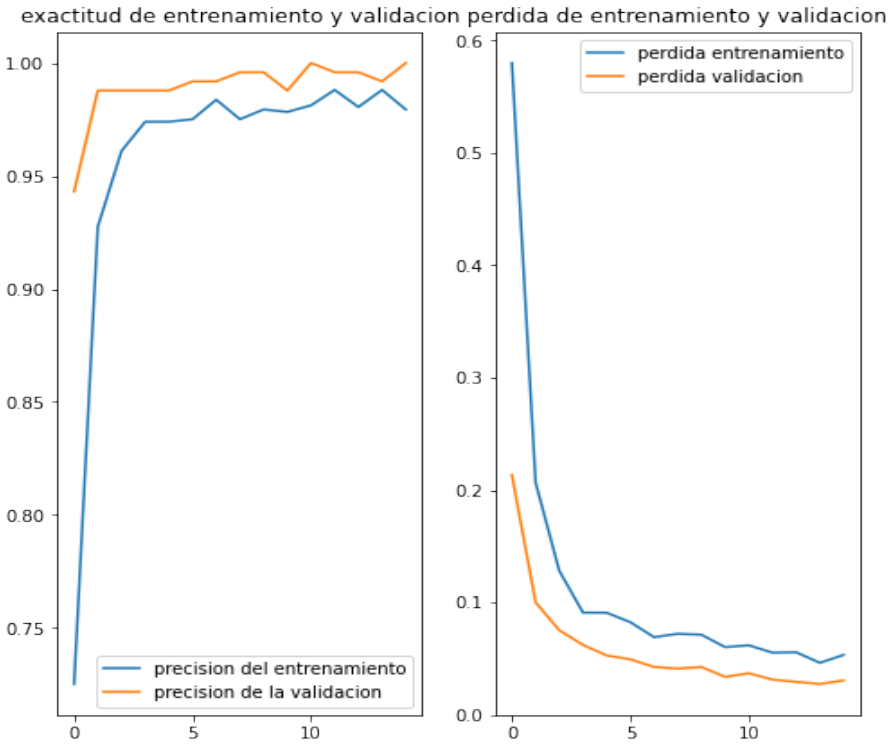
6.1. Red neuronal para la clasificación de los ojos

Para la construcción de la red neuronal encargada de la clasificación de los ojos se utilizó la técnica de *transfer learning* con el modelo MobileNetV2. En la tabla 4 , se muestra numéricamente el comportamiento del entrenamiento y validación que se obtuvo. En la figura 32, se pueden observar las curvas de exactitud y perdida tanto del entrenamiento como de la validación.

Tabla 4
Resultados del entrenamiento, validación y prueba para la red neuronal encargada de la clasificación de los ojos

Entrenamiento		**Validación**		**Prueba**
Pérdida	Exactitud	Pérdida	Exactitud	Exactitud
0.0482	0.9842	0.0306	0.9981	0.9908

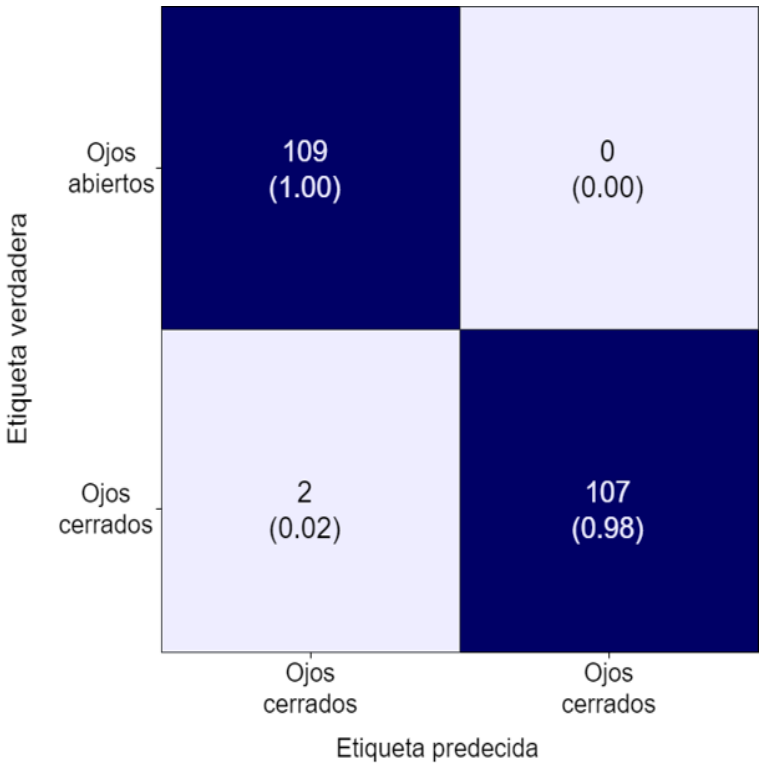
Figura 32.
Curva de exactitud y pérdida de entrenamiento y validación.



En la figura 33 se presenta la matriz de confusión, la cual muestra que clasificó 109 imágenes de ojos abiertos con una sensibilidad de 1.00, lo que quiere decir que no presentó ningún error; clasificó 107 imágenes de ojos cerrados con una sensibilidad de 0.98, en esta categoría de ojos cerrados 2 fotos salieron erróneas, lo que quiere decir que en dos imágenes de ojos cerrados

se predijo que eran ojos abiertos.

Figura 33.
Matriz de confusión.



Los resultados observados en las figuras 32 y 33 demuestran que el modelo con la arquitectura MobileNetV2 presentó resultados óptimos en su desempeño, por ello es seleccionada para el modelo definitivo de la red neuronal para la clasificación de ojos.

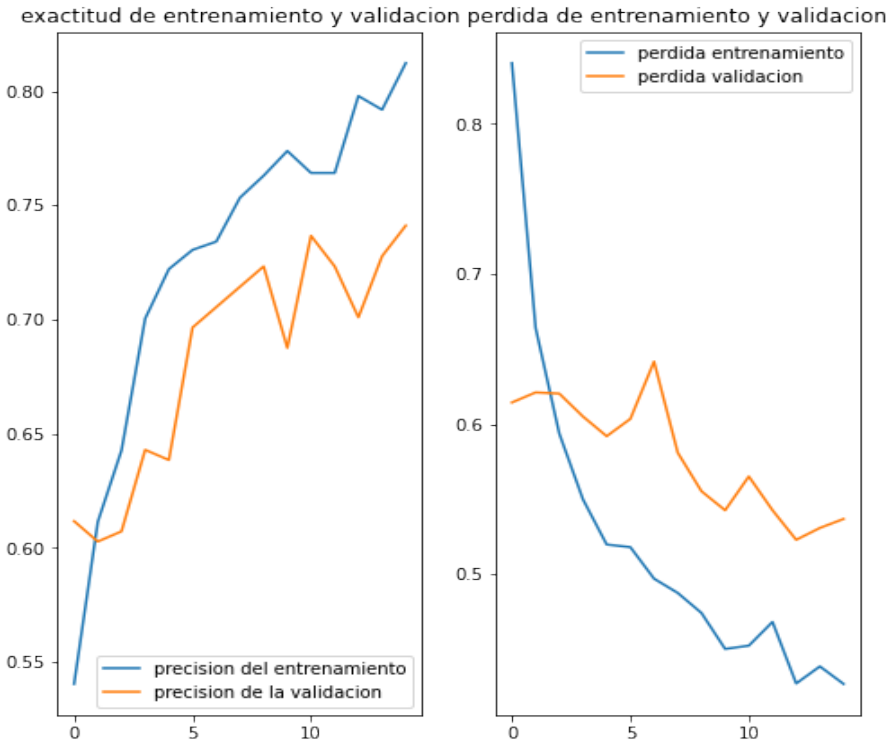
6.2. Red neuronal para la clasificación de los bostezos

Para la construcción de la red neuronal encargada de la clasificación de los bostezos se utilizó la técnica de *transfer learning* y se probó con varias arquitecturas como InceptionV3, ResNet101V2 y Xception.

6.2.1. *Red neuronal con la arquitectura ResNet101V2.*

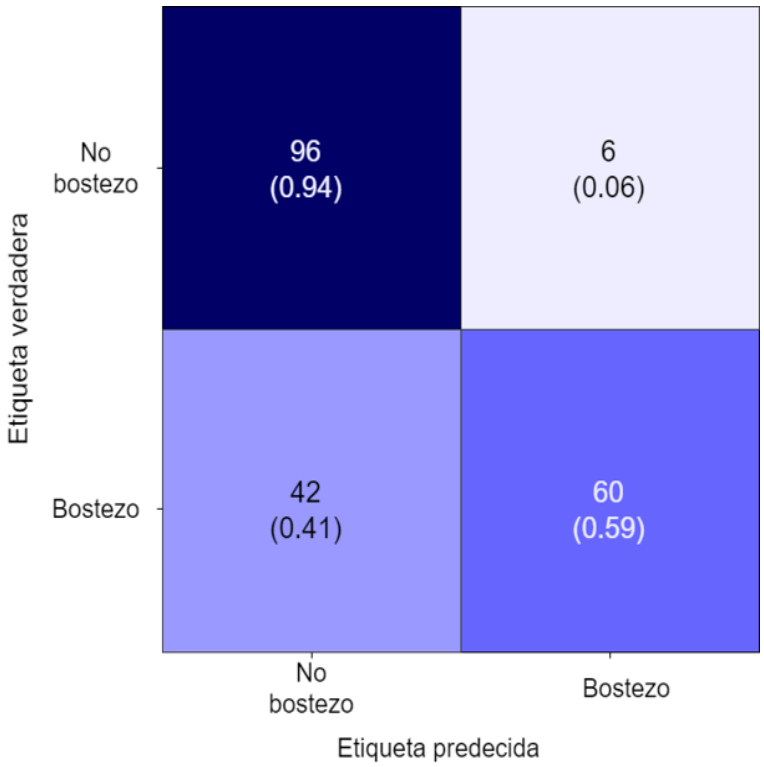
Para esta red se probó la arquitectura ResNet101V2, el cual no dió resultados óptimos, pues el valor de precisión en cuanto a entrenamiento fue de 79.06 % y en cuanto a validación fue de 74.11 %, como prueba de ello se muestra la figura 34.

Figura 34.
Curva de precisión y pérdida de entrenamiento y validación.



En la matriz confusión (figura 35), se muestra que en las categorías de bostezo y no bostezo, se tomaron 102 imágenes para cada una; en la categoría no bostezo, se clasificó 96 imágenes acertadas con una sensibilidad de 0.94 y en la categoría bostezo, se clasificó 60 imágenes correctamente equivalente a una sensibilidad de 0.59.

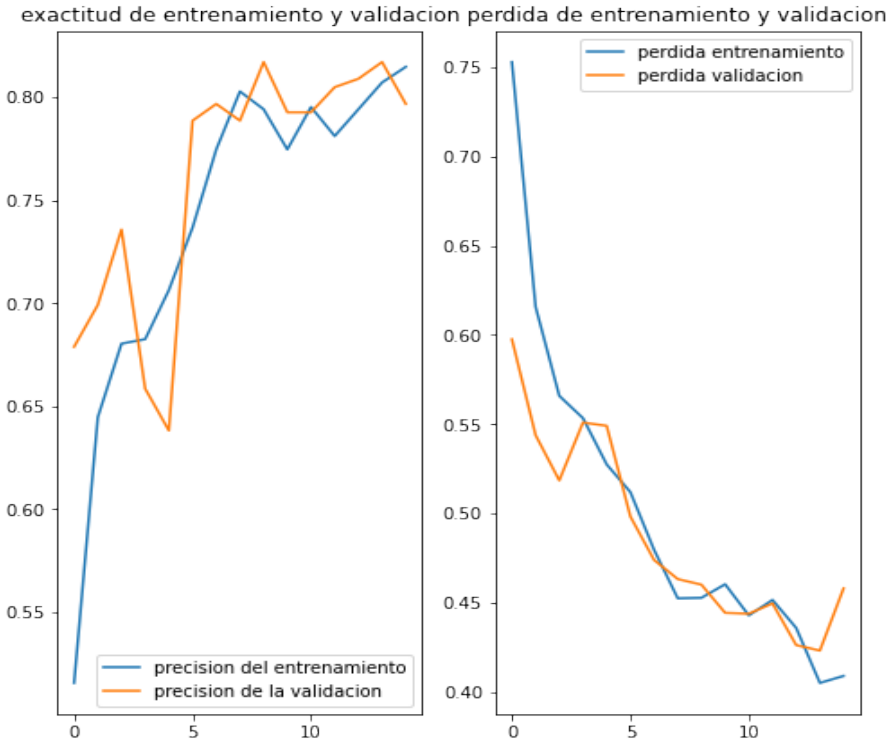
Figura 35.
Matriz de confusión.



6.2.2. *Red neuronal con la arquitectura InceptionV3*.

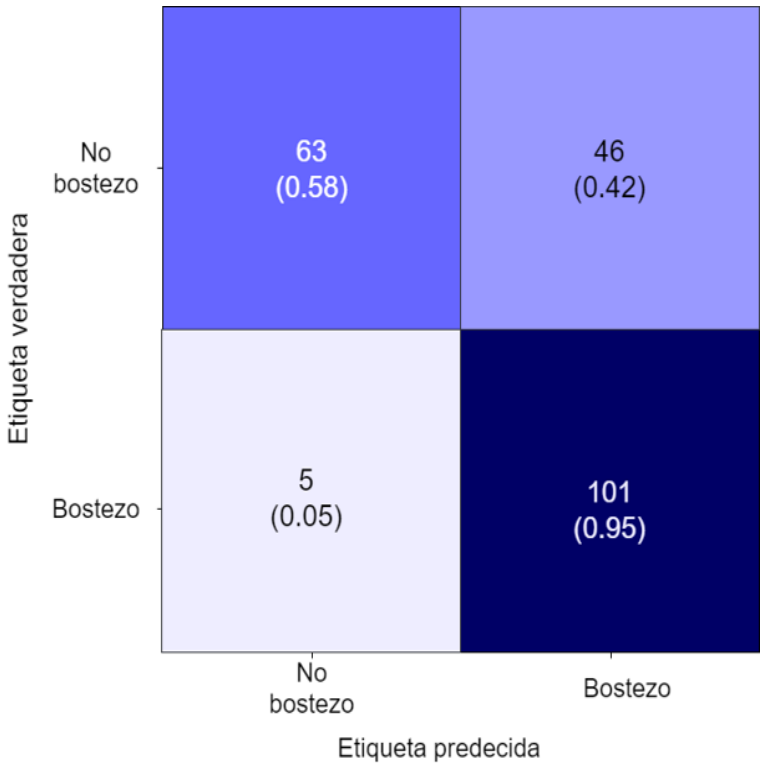
La arquitectura InceptionV3 no dió resultados óptimos pero mejoró a comparación de los resultados obtenidos por ResNet101V2, pues el valor de precisión en cuanto a entrenamiento fue de 81.45 % y en cuanto a validación fue de 79.67 %, como se muestra en la figura 36.

Figura 36.
Curva de exactitud y pérdida de entrenamiento y validación.



Los resultados de la matriz confusión (figura 37), muestra que se tomaron 109 imágenes en la categoría de no bostezo, en esta categoría se clasificó 63 imágenes acertadas, con una sensibilidad de 0.58 y 106 imágenes en la categoría de bostezo, en la que se clasificó 101 imágenes acertadas equivalente a un 0.95 en sensibilidad.

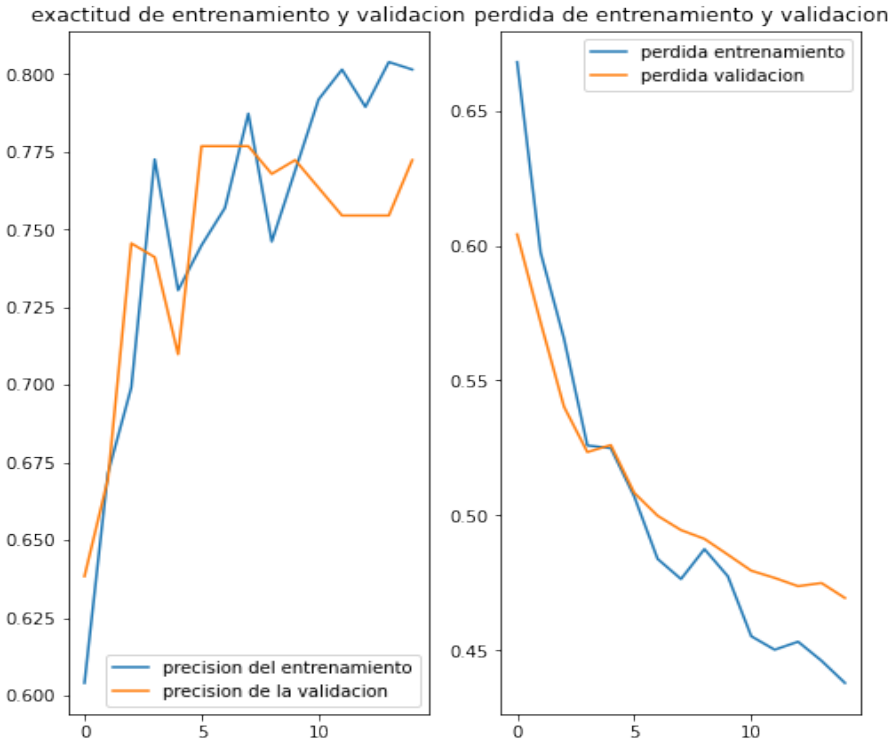
Figura 37.
Matriz de confusión.



6.2.3. *Red neuronal con la arquitectura Xception*.

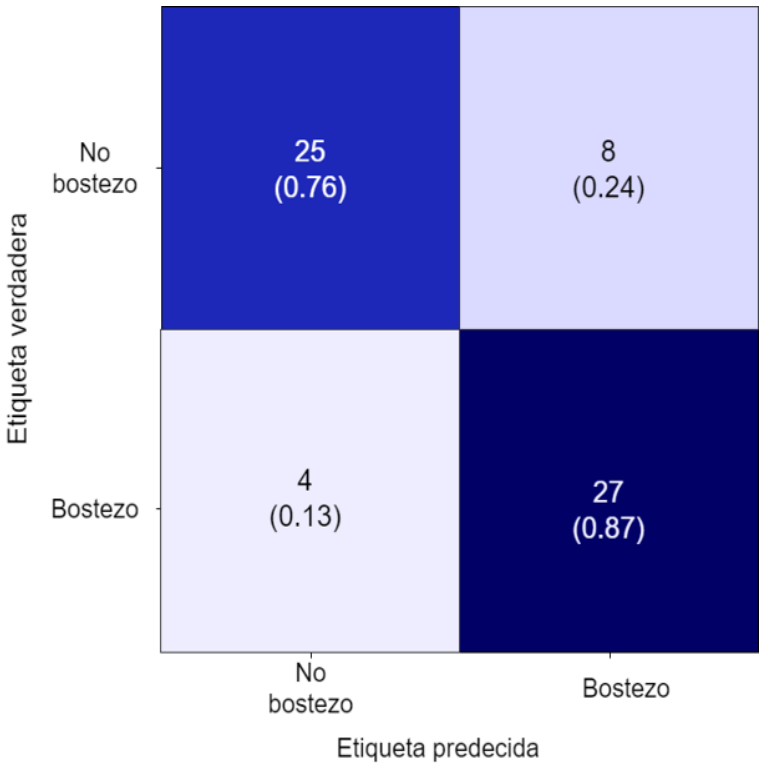
Por último se probó con la arquitecura Xception, este modelo mejoró en comparación con ResNet101V2, pero no mejoró a comparación de los resultados obtenidos con InceptionV3, pues el valor de exactitud en cuanto a entrenamiento fue de 78.81 % y en cuanto a validación fue de 77.23 %, como se muestra en la figura 38.

Figura 38.
Curva de exactitud y pérdida de entrenamiento y validación.



Se observa en la matriz de confusión (figura 39) que la categoría de no bostezos toma 33 imágenes y la categoría de bostezos toma 31 imágenes. En no bostezos se clasificó con una sensibilidad de 0.76 con 25 imágenes acertadas. La categoría de bostezos clasificó con una sensibilidad de 0.87 con un total de 27 imágenes acertadas.

Figura 39.
Matriz de confusión.



En la tabla 5 presentada a continuación se muestran los resultados del entrenamiento, validación y prueba para cada uno de los modelos presentados anteriormente. Determinando que la mejor red para dar desarrollo a la clasificación de los bostezos es Xception, ya que fue la arquitectura que mejor exactitud arrojó en la prueba.

Tabla 5
Registro de los resultados aportados por los modelos utilizados

Red	**Entrenamiento**		**Validación**		**Prueba**
	Pérdida	Exactitud	Pérdida	Exactitud	Exactitud
ResNet101V2	0.4362	0.7906	0.5367	0.7411	0.7647
InceptionV3	0.4181	0.8145	0.4581	0.7967	0.7627
Xception	0.4409	0.7881	0.4694	0.7723	0.8125

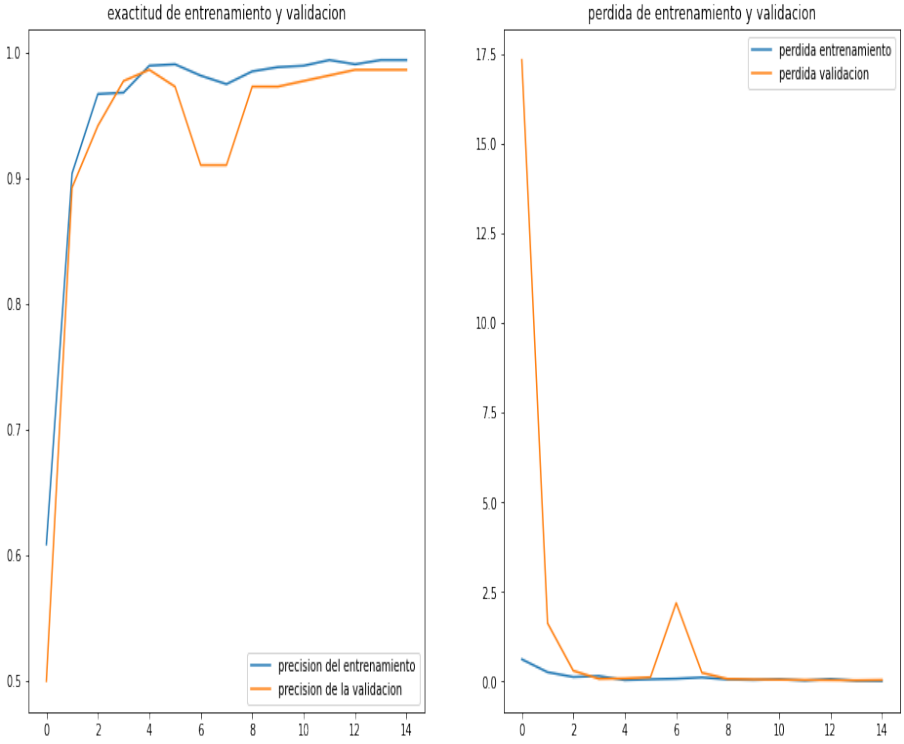
6.2.4. *Red neuronal para los bostezos con la arquitectura Xception mejorada.*

La red neuronal definitiva para la clasificación de los bostezos fue la arquitectura Xception, sobre la cual se implementaron técnias para evitar sobreajuste y así mejorar sus valores de entrenamiento, validación y prueba (tabla 6). En la figura 40, se puede observar las curvas de exactitud y perdida tanto del entrenamiento como de la validación.

Tabla 6
Resultados del entrenamiento, validación y prueba para la red neuronal encargada de la clasificación de los bostezos

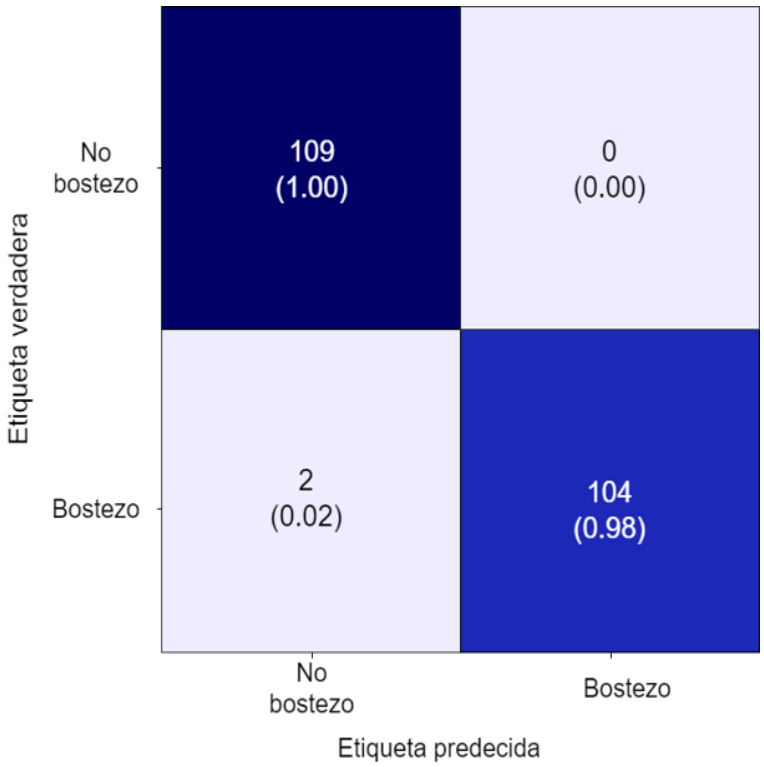
Entrenamiento		**Validación**		**Prueba**
Pérdida	Exactitud	Pérdida	Exactitud	Exactitud
0.0142	0.9944	0.0407	0.9866	0.9901

Figura 40.
Curva de exactitud y pérdida de entrenamiento y validación.



La Figura 41, muestra la matriz confusión, en ella se observa que la categoría de no bostezo toma 109 imágenes en total y arroja una sensibilidad de 1.00 en las predicciones lo que quiere decir que todas las imágenes de esta categoría fueron acertadas y en la categoría de bostezo se toman 109 imágenes, obteniendo una sensibilidad de 0.98 en las predicciones.

Figura 41.
Matriz de confusión.



Los resultados presentados en las figuras 40 y 41 demuestran que el modelo con la arquitectura Xception presentó resultados óptimos en su desempeño, por ello se seleccionó como el modelo definitivo de la red neuronal para la clasificación de bostezos.

6.3. Pruebas del sistema completo

En esta sección se presentan las pruebas del sistema completo, las cuales fueron realizadas con paquetes de datos controlados de entrada, con el fin de evaluar el desempeño del sistema en sus diferentes modos de clasificación para detectar somnolencia, mencionados en el capítulo anterior. La evaluación de los resultados se llevó a cabo mediante matrices de confusión.

6.3.1. *Prueba de detección de somnolencia para evaluar respuesta a bostezos*.

Esta prueba se realizó con tres paquetes y cada uno de ellos con una entrada de cinco imágenes diferentes de bostezos y una de ojos abiertos (Figura 42), las cuales fueron clasificadas correctamente en otro directorio. Posteriormente con las predicciones del sistema y los datos clasificados en dos categorias, se realizaron las matrices de confusión para cada uno de los paquetes de datos usados, donde se observó la sensibilidad para los datos que representan signos de somnolencia, como se muestra en las Figuras 43, 44 y 45.

Figura 42.

Paquete de imágenes utilizado para la prueba 1

Figura 43.
Matriz de confusión prueba 1.

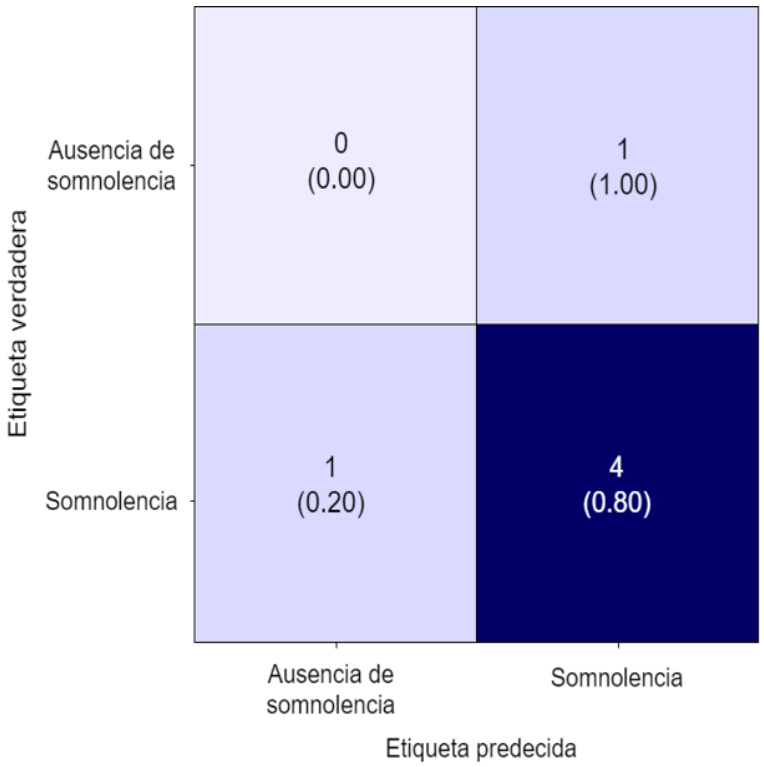


Figura 44.
Matriz de confusión prueba 2.

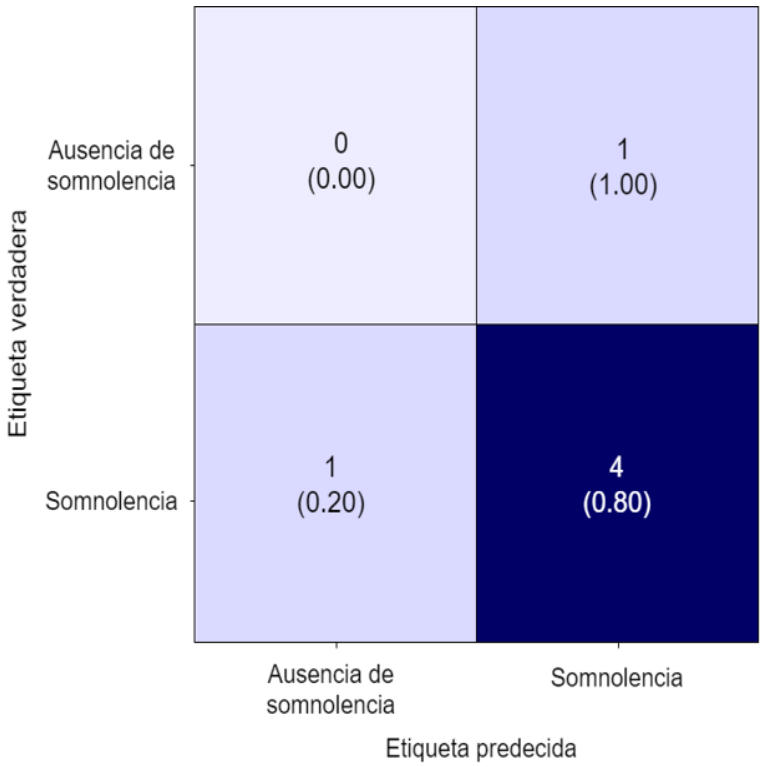
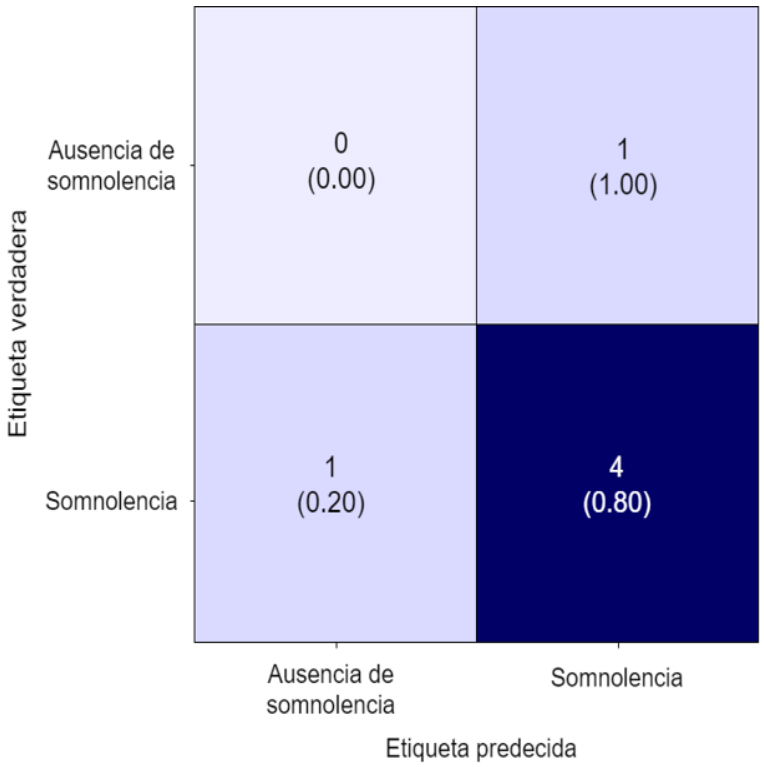


Figura 45.
Matriz de confusión prueba 3.



Como se observa en las figuras anteriores, la categoría identificada como somnolencia, arrojó un valor de sensibilidad de un 0.80, lo que equivale a que una imágen fue clasificada erroneamente. Lo anterior indica que el sistema esta respondiendo de manera adecuada ante este tipo de entrada, con un porcentaje de error reducido.

6.3.2. *Prueba de detección de ausencia de somnolencia para evaluar respuesta a ojos abiertos*.

La prueba para evaluar la respuesta ante los ojos abiertos, se realizó con tres paquetes y cada uno de ellos con cinco imágenes diferentes de ojos abiertos y una de bostezo (Figura 46). Con las predicciones del sistema y los datos clasificados en dos categorías, se crearon las matrices de confusión para cada uno de los paquetes de datos, donde se observó el valor de sensibilidad para los datos que representan signos de ausencia de somnolencia, como se muestra en las Figuras 47, 48 y 49

Figura 46.

Paquete de imágenes utilizado para prueba 2

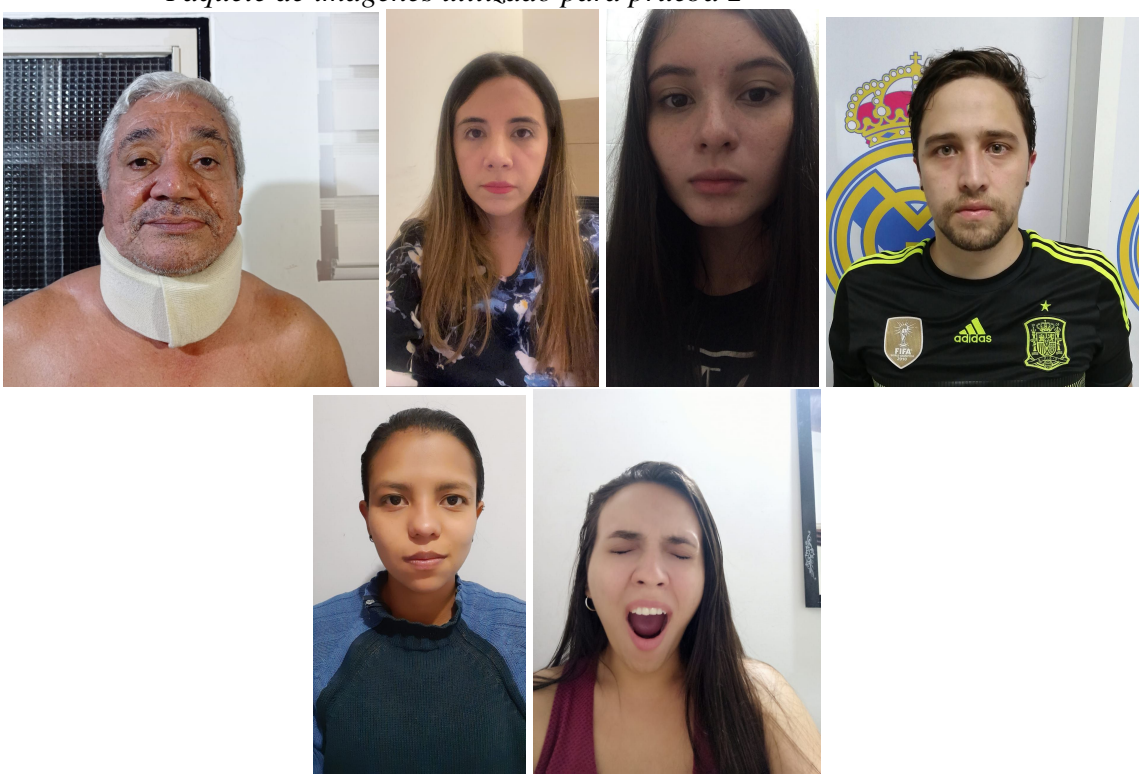


Figura 47.
Matriz de confusión prueba 1

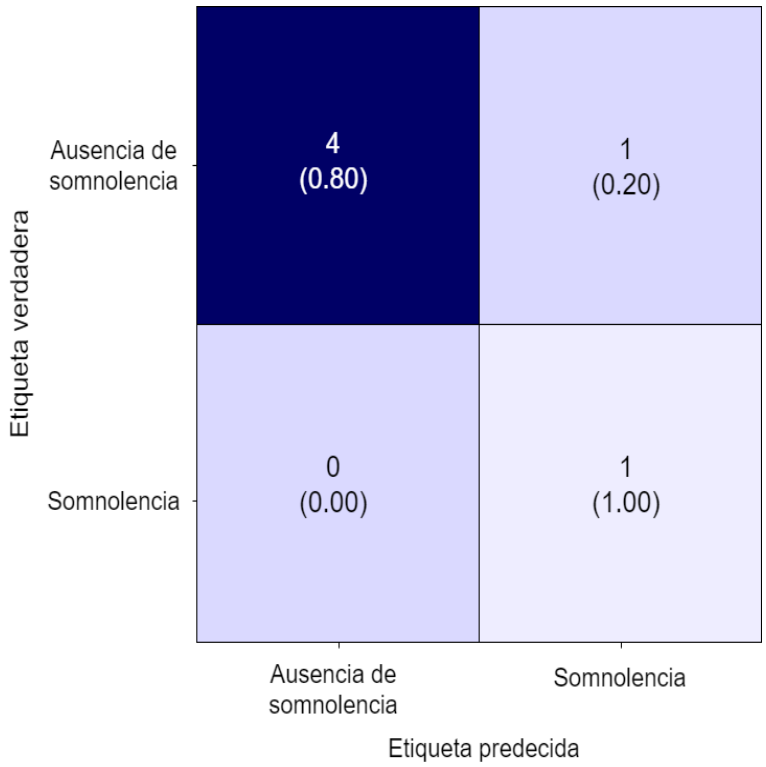


Figura 48.
Matriz de confusión prueba 2

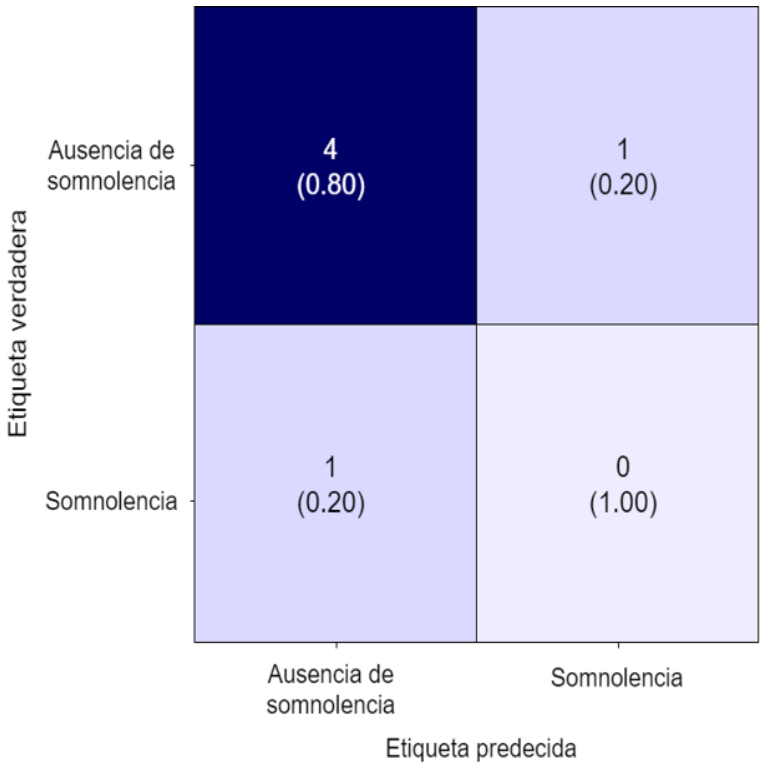
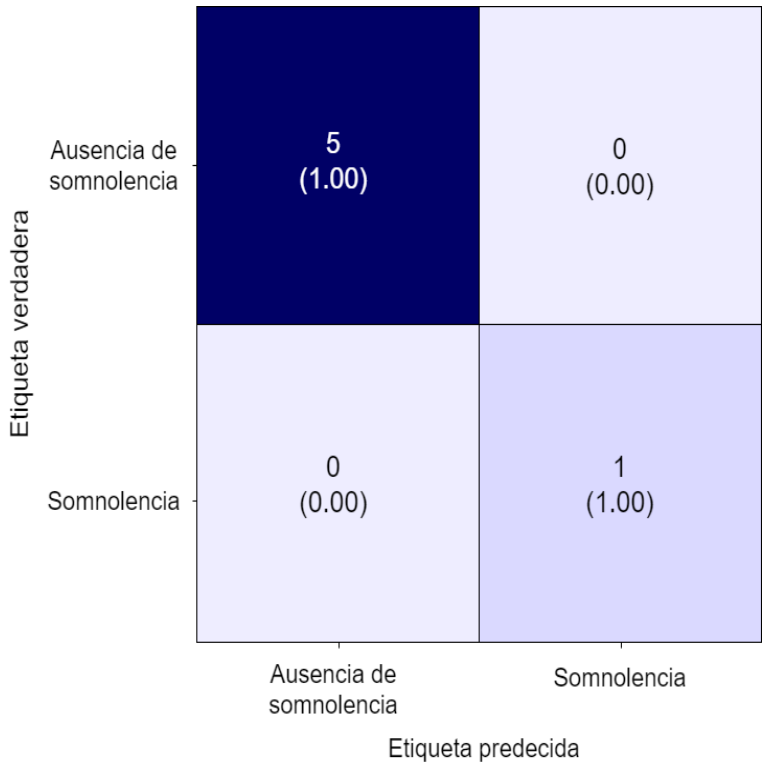


Figura 49.
Matriz de confusión prueba 3



Como se observa en las matrices de confusión, la prueba 1 y 2 en la categoría de ausencia de somnolencia, arrojaron un valor de sensibilidad del 0.80 y en la prueba 3 una sensibilidad de 1.00. Demostrando que el sistema esta respondiendo de manera adecuada ante este tipo de entrada, y posee muy poca presencia de error.

6.3.3. *Prueba de detección de somnolencia para evaluar respuesta a ojos cerrados*.

Se decidió hacer una tercera prueba con tres paquetes diferentes, cada uno de ellos con cinco imágenes diferentes de ojos cerrados y uno de ojos abiertos (Figura 45). Con las predicciones del sistema y los datos clasificados en dos categorías. Se crearon las matrices de confusión para

cada uno de los paquetes de datos, donde se observó el valor de sensibilidad para los datos que representan signos de somnolencia.

Figura 50.

Paquete de imágenes utilizado para prueba 3

Figura 51.
Matriz de confusión prueba 1

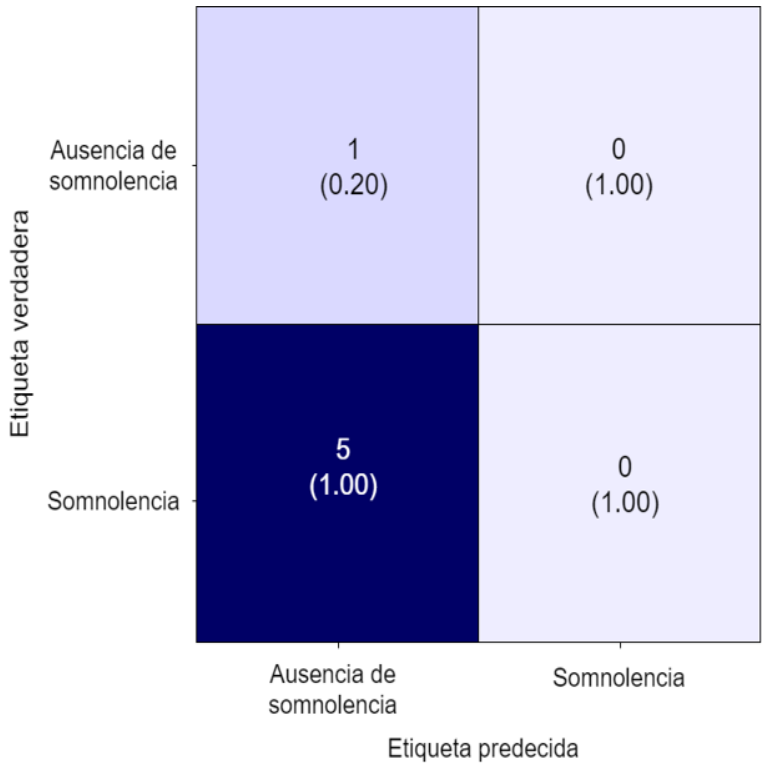


Figura 52.
Matriz de confusión prueba 2

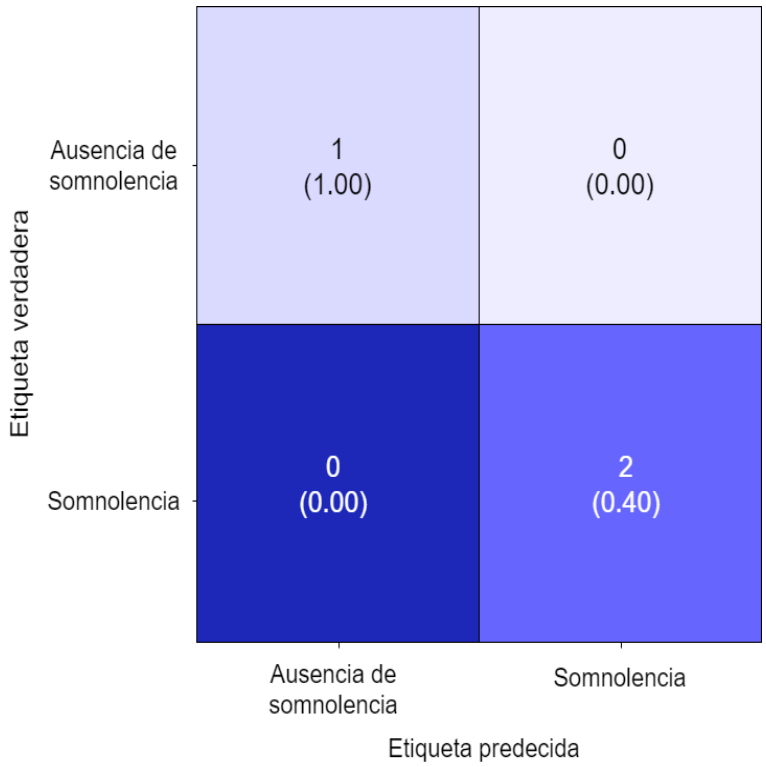
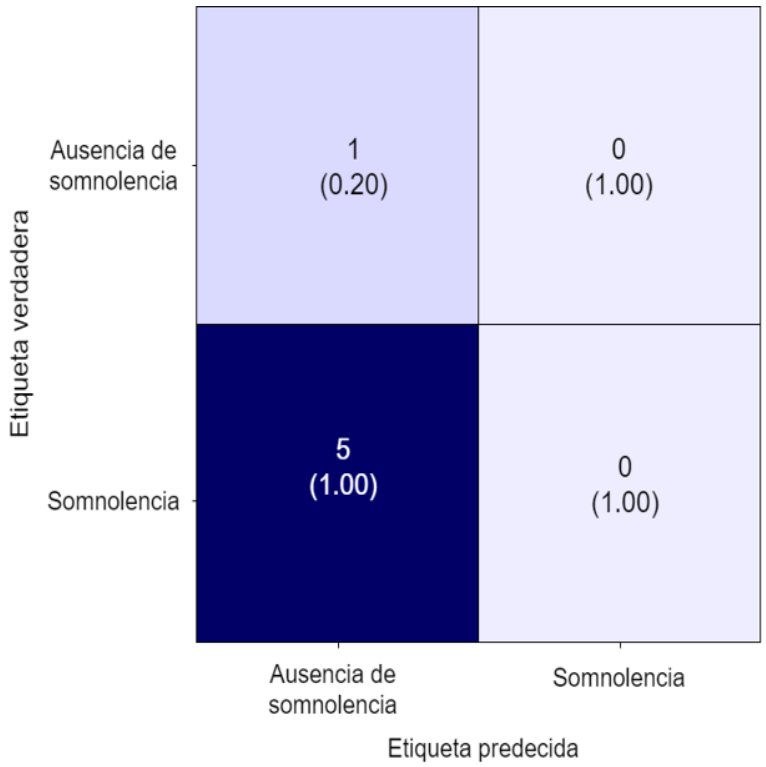


Figura 53.
Matriz de confusión prueba 3

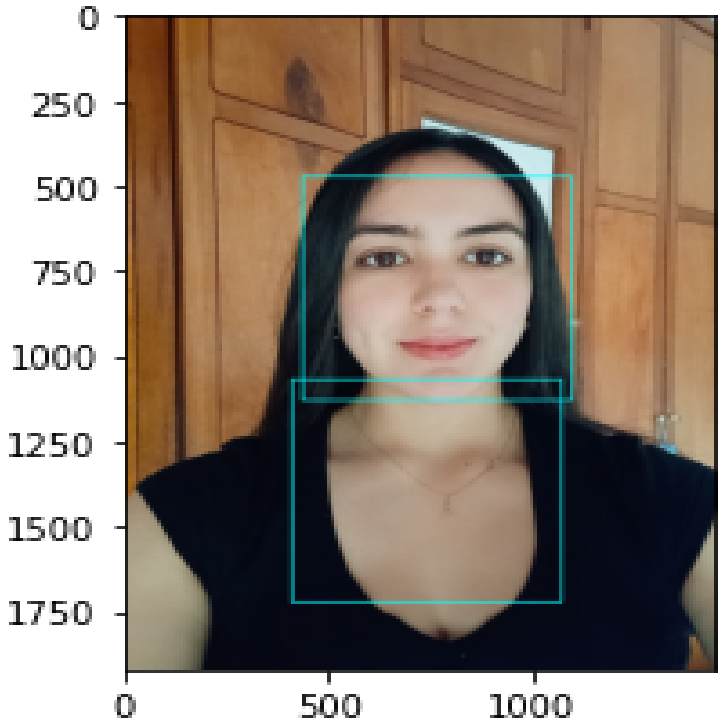


Las Figuras 51, 52 y 53 son las matrices de confusión que representan las pruebas realizadas ante esta entrada. En ellas se observa que con estas entradas el sistema no responde de una manera adecuada. Esto se debe principalmente a errores generados en la detección de rostro y ojos, propios de los calsificadores en cascada ofrecidos por la librería OpenCV, los cuales serán presentados a continuación.

6.3.3.1. Errores de detección en los Clasificadores en Cascada. Los clasificadores en cascada *haarcascade_frontalface_default.xml* y *haarcascade_eye.xml*, que se utilizaron para la detección de rostro y ojos respectivamente, durante la ejecución del sistema, presentaron errores en su funcionamiento que a su vez causaron clasificaciones erroneas en algunas pruebas del sistema.

El error generado por *haarcascade_frontalface_default.xml* consiste en una detección del rostro en un área de la imágen que no es la correcta (Figura 54).

Figura 54.
Imágen con detección de rostro en un área que no es la correcta.



Los errores generados por *haarcascade_eye.xml*, son de dos tipos, detección de otros rasgos u objetos en la imágen que no coinciden con el ojo (Figura 55), y detección de uno o varios objetos más en la imágen los cuales detectó como ojos (Figura 56).

Figura 55.
Imágen con detección de objetos que no corresponden a los ojos

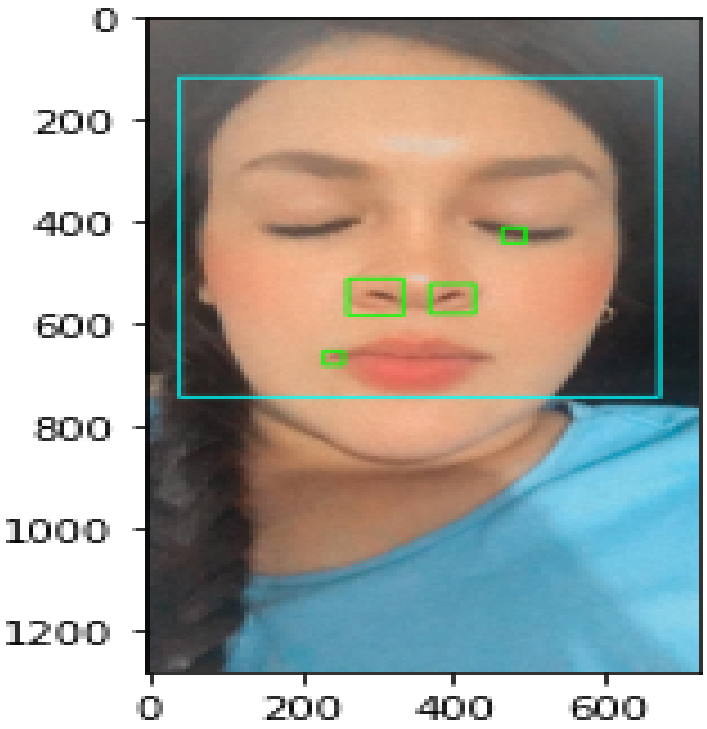
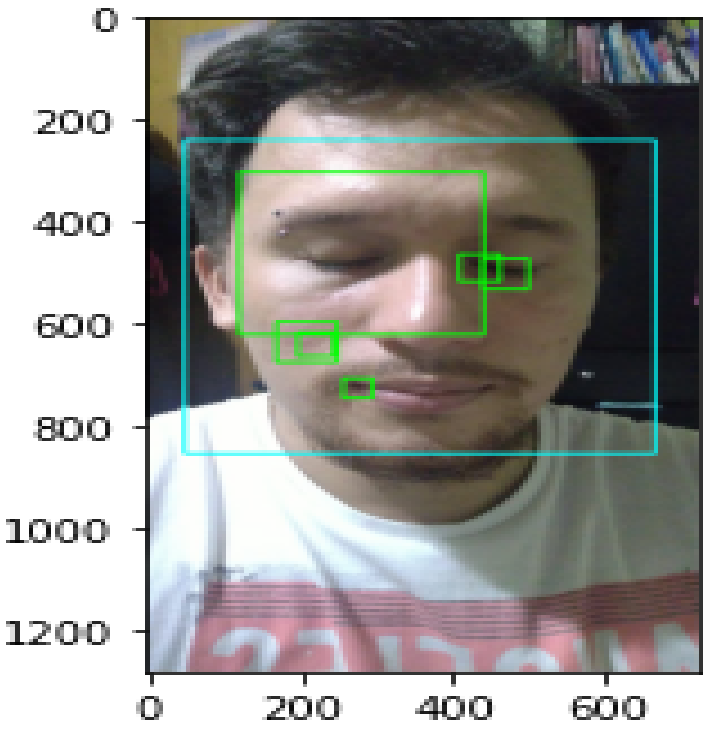


Figura 56.
Imágen de detección de ojos con más objetos detectados como ojos.



7. Conclusiones y Recomendaciones

1. Hemos desarrollado un sistema de detección de somnolencia para dos tipos de clasificaciones, estado de los ojos y bostezos. Para llevar a cabo este desarrollo utilizamos dos bases de datos públicos del repositorio kaggle y creamos una base de datos de personas naturales con sus respectivas autorizaciones, estas bases de datos están compuestas por imágenes de ojos abiertos, ojos cerrados, bostezos y no bostezos. Para llevar a cabo el proceso de entrenamiento utilizamos la técnica de transfer learning a dos modelos de CNN preentrenados, probamos diferentes arquitecturas y nuestros resultados sugirieron que MobileNetV2 mostraba el mejor rendimiento para la red de los ojos con una exactitud en prueba de 0.9908 y Xception arrojaba el mejor rendimiento para la red de los bostezos con una exactitud en prueba de 0.9901.

2. Para el clasificador final se realizaron 2 tipos de pruebas para detección de somnolencia y un tipo de prueba para detección de ausencia de somnolencia. Se obtuvo para la prueba de ausencia de somnolencia y una de las pruebas de somnolencia una sensibilidad del 0.80 en la detección del evento. Para la otra prueba de detección de somnolencia se obtuvo una sensibilidad del 0.0 en la detección del evento.

3. La base de datos es un pilar fundamental para obtener resultados óptimos en una red neuronal. Esta debe ser amplia en su contenido para cada categoría, con el fin de poder hacer una separación adecuada de datos de entrenamiento y validación; debe contener imágenes que

entre sí presenten diferencias para que la red pueda detectar la mayor cantidad de patrones posibles en el proceso de aprendizaje. Además un factor importante es que los conjuntos de datos de prueba contengan imágenes que no se encuentren en los conjuntos de entrenamiento y validación, esto para poder comprobar que el proceso de aprendizaje de la red se llevó a cabo correctamente.

4. Las redes neuronales convolucionales son ideales para aplicaciones en las que las varibles de entrada son tridimensionales, ya que cada capa de convolución por la que pasan los datos extrae patrones y características que mejoran el proceso de aprendizaje y la clasificación que pueden efectuar los modelos. Esto se pudo evidenciar tanto en el caso de clasificación de ojos abiertos/cerrados como en el caso de detección de bostezos.

5. Aplicar la técnica de *transfer learning* es una de las opciones óptimas para mejorar el desempeño de un modelo, ya que reduce el tiempo de entrenamiento debido a un menor número de parámetros y mejora la exactitud ya que las arquitecturas utilizadas son modelos preentrenados en distintas aplicaciones.

6. Las técnicas para evitar el sobreajuste como lo son el *Dropout*, las transformaciones de imágenes y la adición de capas densas en las CNN, además de el uso de *callbacks* demostraron ser efectivas para el mejoramiento de un modelo, caso que se pudo observar en la red neuro-

nal de bostezos, la cual mejoró su precisión en prueba de un 81.25 % a un 99.01 % después de aplicar lo anteriormente mencionado.

7. El desempeño de las redes neuronales individuales fue bueno, arrojando porcentajes de precisión en prueba de 99.08 % y 99.01 % para la red de ojos y bostezos, respectivamente. Esto proporciona una base confiable de predicciones para el sistema de deteccion de somnolencia.

8. El tiempo de respuesta del sistema, como se esperaba, fue aproximadamente un 50 % menor al tiempo del evento a monitorear. Aunque fue un buen registro de tiempo para la aplicación, una manera de disminuir el tiempo sería realizar una compresión del modelo, ya que el peso del sistema influye en su tiempo de ejecución. Esta compresión de modelo se recomienda como trabajo futuro para esta aplicación.

9. El sistema tuvo una respuesta bastante acertada en las pruebas individuales de somnolencia detectando bostezos y ausencia de somnolencia detectando ojos abiertos. Esto quiere decir que efectivamente la red neuronal de los bostezos está actuando según lo esperado, y los clasificadores en cascada arrojan pocos errores en la detección de ojos abiertos para ser ingresados a la red de los ojos.

10. La mayor cantidad de errores que presenta el sistema en la detección de somnolencia, se deben a las fallas propias de los clasificadores de rostro y ojos, presentando estos un mayor número de fallas en la detección de somnolencia por el criterio de ojos cerrados. Para mejorar estos errores se recomienda como trabajo futuro modificar los códigos fuente de los clasificadores en cascada.

11. A futuro, para que la operación de este sistema salga del entorno de *Google Colab* y tenga utilidad en el mundo real, se recomienda incorporarlo a un sistema embebido para que de este modo funcione sin necesidad de tener un ordenador.

Referencias Bibliográficas

Agencia Nacional de Seguridad Vial (2020). *Accidentes de transito.* https://ansv.gov.co/es/buscador-general?label=tasa+de+accidentes+por+fatiga. Accedido en marzo de 2021.

ARL Sura (2018). *La accidentalidad vial: un problema mundial*. https://www.arlsura.com/index.php/component/content/article?id=1474:la-accidentalidad-vial-un-problema. Accedido en marzo de 2021.

Basogain, X. (2008). *Redes Neuronales Artificiales y sus Aplicaciones.* Escuela centro superior de Ingeniería de Bilbao, EHU. Accedido en marzo de 2021.

Bibing.us (2000). *Procesos de aprendizaje, capítulo 3*. https://n9.cl/df0sv. Accedido en marzo de 2021.

Bohorquez, P., y Villanueva, J. (2006). *Predicción de signos a tres semanas de la acción caterpillar con redes neuronales.* Universidad de Chile. Accedido en marzo de 2021.

Brownlee, J. (2019). *A Gentle Introduction to Transfer Learning for Deep Learning*. https://machinelearningmastery.com/transfer-learning-for-deep-learning/. Accedido en marzo de 2021.

Calderón, A., y Balcero, P. (2020). *DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO DE DISPOSITIVO ADAS PARA DETECCIÓN Y ALERTA DE MICROSUEÑO, EN CONDUCTO-*

RES DE VEHÍCULOS TERRESTRES DE TRASPORTE DE CARGA Y DE PASAJEROS EN EL CONTEXTO COLOMBIANO. Universidad Piloto de Colombia. Accedido en marzo de 2021.

Carneiro, T., y Medeiros, R. (2018). *Performance Analysis of Google Colaboratory as a Tool for Accelerating Deep Learning Applications*. https://bibliotecavirtual.uis.edu.co:2272/document/8485684. Accedido en marzo de 2021.

Chokroverty, S. (2011). *Privación de sueño y somnolencia.* https://www.sciencedirect.com/sdfe/pdf/download/eid/3-s2.0-B9788480867337000036/first-page-pdf. Accedido en marzo de 2021.

Chollet, F. (2017). *Xception: Deep Learning with Depthwise Separable Convolutions*. https://arxiv.org/pdf/1610.02357.pdf. Accedido en marzo de 2021.

Fernandez, A. (2013). *Python 3 al descubierto*. RC Libros. Accedido en marzo de 2021.

Flórez, R., y Fernández, J. (2008). *Las Redes Neuronales Artificiales Fundamentos teóricos y aplicaciones prácticas.* Netbiblio,S.L. Accedido en marzo de 2021.

Fundación CEA (2015). *El sueño y la fatiga en la conducción.* https://n9.cl/xv0ui. Accedido en marzo de 2021.

Garcés, A., Orosco, L., y Laciar, E. (2014). *Automatic detection of drowsiness in EEG records based on multimodal analysis.* https://www.sciencedirect.com/science/article/abs/pii/S1350453313001690. Accedido en marzo de 2021.

García, N. (2019). *ADAPTACIÓN DE UN SISTEMA DE DETECCIÓN DE PERSONAS EN CÁMARAS OMNIDIRECCIONALES A DESCRIPTORES DEEP LEARNING*. https://repositorio.uam.es/bitstream/handle/10486/688925/garcia_crespo_nicolas_tfg.pdf?sequence=1&isAllowed=y. Accedido en marzo de 2021.

Gómez, F., Fernández, M., y López, M. (1994). *Aprendizaje con redes neuronales artificiales*. https://dialnet.unirioja.es/servlet/articulo?codigo=2281678. Accedido en marzo de 2021.

Hyttsten, M., Delgado, J., y Bailey P. (2021). *Intro to TensorFlow for Deep Learning*. https://www.udacity.com/course/intro-to-tensorflow-for-deep-learning--ud187. Accedido en marzo de 2021.

Medrano, J. (2013). *Living, yawning, dying*. http://scielo.isciii.es/scielo.php?pid=S0211-57352013000100011&script=sci_arttext&tlng=pt. Accedido en marzo de 2021.

Mohhammmed, A., Kamarudin, A., Mosa, A., y Syamsunur, D. (2019). *A review of traffic Accidents and related practices worldwide*. https://benthamopen.com/FULLTEXT/TOTJ-13-65/. Accedido en marzo de 2021.

Montesdeoca, E. (2020). *IMPLEMENTACIÓN DE UN SISTEMA DE RECONOCIMIENTO DEL USO DE MASCARILLAS COMO MEDIDA DE PRECAUCIÓN CONTRA EL COVID19 USANDO DEEP LEARNING*. http://repositorio.utmachala.edu.ec/bitstream/48000/15890/1/TTFIC-2020-IS-DE00009.pdf. Accedido en marzo de 2021.

Pede, M., Scurfield, R., y Sleet, D. (2004). *World report on road traffic injury prevention*. https://apps.who.int/iris/bitstream/handle/10665/42871/9241562609.pdf;jsessionid=17519435B752F2AF5561217909FA3FA8?sequence=1. Accedido en marzo de 2021.

Pelaez, N. (2012). *APRENDIZAJE NO SUPERVISADO Y EL ALGORITMO WAKE-SLEEP EN REDES NEURONALES*. http://jupiter.utm.mx/~tesis_dig/11612.pdf. Accedido en marzo de 2021.

Ripoll, M., y Hernández, L. (2016). *Cámaras para combatir el sueño al volante*. https://n9.cl/rqi2r. Accedido en marzo de 2021.

Rosales, E., y De Castro, J. (2010). *Somnolencia: Qué es, que la causa y cómo se mide.* http://www.scielo.org.pe/pdf/amp/v27n2/a10v27n2. Accedido en marzo de 2021.

Russel, S. (2004). *Inteligencia Artificial Un enfoque moderno*. PEARSON EDUCACIÓN.S.A. Accedido en marzo de 2021.

Saha, S. (2018). *A comprehensive Guide to Convolutional Neural Networks- the ELI5 way*. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53. Accedido en marzo de 2021.

SGMA (2017). *Inteligencia Artificial. Capítulo 4.* http://www.ptolomeo.unam.mx:8080/xmlui/bitstream/handle/132.248.52.100/219/A7.pdf?sequence=7. Accedido en marzo de 2021.

Shin, H., y Roth, H. (2016). *Deep Convolutional Neural Networks for computer-Aided detection: CNN Architectures, Dataset characteristcs and transfer Learning*. https://ieeexplore.ieee.org/abstract/document/7404017. Accedido en marzo de 2021.

Terán-Santos, J., y Egea, C. (2006). *Apnea del sueño y conducción de vehículos. Recomendaciones para la interpretación del nuevo reglamento general de conductores en España.* https://www.archbronconeumol.org/es-apnea-del-sueno-conduccion-vehiculos--articulo-S0300289616303325. Accedido en Marzo de 2021.

Thomas, A., y Olarte, D. (2021a). *Base de datos creada para las pruebas.* https://github.com/angiethomas4/Base-de-datos-prueba.

Thomas, A., y Olarte, D. (2021b). *Código fuentes de redes neuronales de ojos y bostezos* . https://github.com/olarte-droid/Codigo-principal-sistema-de-detecci-n-de-somnolencia.git. Accedido en marzo de 2021.

Thomas, A., y Olarte, D. (2021c). *Código principal del sistema de detección de somnolencia* . https://github.com/olarte-droid/

Codigo-principal-sistema-de-detecci-n-de-somnolencia.git. Accedido en marzo de 2021.

Thomas, H. (2001). *IA: Inteligencia Artificial.* https://www.redalyc.org/pdf/305/30500219.pdf. Accedido en marzo de 2021.

Toroyan, T., Peden, M., y Laych, K. (2009). *La OMS presenta el segundo informe sobre la situación mundial sobre la seguridad vial*. https://injuryprevention.bmj.com/content/19/2/150. Accedido en marzo de 2021.

Touahmia M. (2018). *Identification of Risk Factors Influencing Road Traffic Accidents*. http://etasr.com/index.php/ETASR/article/view/1615/pdf. Accedido en marzo de 2021.

Velasco, J. (2019). *DISEÑO Y DESARROLLO DE UN SISTEMA PROTOTIPO DE DIAGNÓSTICO DE AFECCIONES EN PLANTAS DE CÍTRICOS UTILIZANDO PROCESAMIENTO DE IMÁGENES Y APRENDIZAJE PROFUNDO*. http://vitela.javerianacali.edu.co/handle/11522/11943. Accedido en marzo de 2021.

Zhang, X., y Reveriano, F. (2019). *The Effect of High Performance Computer on Deep Learning: A Face Expression Recognition Case*. https://bibliotecavirtual.uis.edu.co:2272/document/8919520. Accedido en marzo de 2021.

Zuluaga, J. (2012). *Accidentes de tránsito en Colombia: calle peligrosas*. https://blog.segurossura.com.co/articulo/movilidad/accidentes-transito-colombia. Accedido en marzo de 2021.