Revista ELECTRO, Vol. 46, 2024, pp. 394 - 399
https://itchihuahua.mx/revista_electro
ISSN Electrónico: 3061-774X

DETECTOR DE SOMNOLENCIA EN CONDUCTORES UTILIZANDO VISIÓN ARTIFICIAL

Rubio Sicairos Sebastián, Prado Isiordia Miguel Abrahan, Gaxiola Sánchez Leopoldo Noel, Medina Melendrez Modesto
Tecnológico Nacional de México - Instituto Tecnológico de Culiacán
Departamento de Ingeniería Eléctrica-Electrónica
Juan de Dios Bátiz No. 310 Pte. , Col. Guadalupe, C.P. 80220 Culiacán Rosales, Sin.
Teléfono: 667 454 0100
Leopoldo.gs@culiacan.tecnm.mx

RESUMEN.

En el siguiente trabajo se presenta un detector de somnolencia para conductores realizado con la red neuronal YOLOv5. En este trabajo se mostrará la construcción del dataset seleccionado, al igual que la creación del modelo de la red neuronal con el fin de obtener su entrenamiento. Posteriormente el archivo de entrenamiento es implementado en el código de Python, el cual contiene alarmas que se activaran de acuerdo con el tiempo que el conductor permanezca con los ojos cerrados. Finalmente, se muestran los resultados obtenidos en una secuencia de video real.
Palabras Clave: Detector de somnolencia, red neuronal, YOLOv5.

ABSTRACT.

The following paper presents a drowsiness detector for drivers made with the YOLOv5 neural network. In this paper will be presented the construction of the chosen dataset, as well as the creation of the neural network model in order to obtain its training. Subsequently, the training file is implemented into the Python code, which contains alarms that will be activated according to the time that the driver remains with his eyes closed. Finally, the results obtained through a real video sequence are shown.
Keywords: Drowsiness detector, neural network, YOLOv5.

1. INTRODUCCIÓN

La real academia española define a la somnolencia como la pesadez y torpeza por el sueño [1], este estado en conductores ha mostrado ser una de las causas más frecuentes de accidentes por carretera. De acuerdo con el gobierno federal los accidentes de tránsito en México son una de las principales causas de muertes en el país, por lo que múltiples estados han impuesto diversas medidas de prevención contra consecuencias fatales.

El INEGI es la institución encargada de registrar la información de diversos eventos en el país y uno de ellos son los accidentes de tránsito. De acuerdo con sus datos, durante el 2022 se registraron 330 mil 288 accidentes [2]:

- 229 mil 940 fueron por colisión con vehículo.
- 11 mil 66 fueron colisiones con peatones.
- mil 22 fueron por colisión con objeto físico.
- 10 mil 911 fueron volcaduras
- mil 874 fueron caídas de pasajeros.
- 10 mil 857 los vehículos se salieron del camino.
- 348 derivaron en un incendio.
- 268 fueron choque con ferrocarriles.
- 53 mil 629 fueron accidentes de motocicleta.
- 4 mil 48 fueron colisiones con ciclistas.
- 6 mil 325 fueron por otras causas.

En base a estos datos, se buscó desarrollar un algoritmo capaz de prevenir accidentes de tránsito causados por la somnolencia, utilizando la visión artificial. Este proyecto va más orientado a personas que deben conducir largas jornadas, como transportistas, siendo ellos más propensos a la somnolencia.

Este proyecto consiste en la detección de somnolencia en los conductores, en el cual, se estarán monitoreando constantemente los ojos del conductor. Si se mantienen los ojos cerrados por un lapso extenso de tiempo se activarán una serie de alarmas para avisar al conductor que se está quedando dormido. Las dificultades que presenta la detección de ojos son: la iluminación no uniforme y falsas detecciones debido a un fondo variado.

Varios proyectos han sido presentados para lidiar con este problema utilizando librerías de machine learning en Python, un ejemplo representativo es mediapipe que permite obtener la relación de aspecto de ojo conocido como EAR (eye aspect ratio) por sus siglas en ingles. El cual se utiliza para calcular la distancia euclidiana entre los ojos del conductor. Para cada secuencia de vídeo, se localizan los puntos de referencia de los ojos, calibrando la relación de aspecto entre el ancho y alto de cada ojo [3]. Sin embargo, gracias a las redes neuronales como YOLO, que permiten detectar cualquier objeto dependiendo de las imágenes con las que se ha entrenado el algoritmo, se ha logrado conseguir un modelo capaz de detectar ojos abiertos o cerrados.

En este trabajo se propone un algoritmo de visión artificial que permita detectar la somnolencia en los conductores, tomando como referencia el tiempo que permanezca con los ojos cerrados. El modelo de YOLO está basado en un dataset hecho con imágenes de internet al igual que fotos tomadas recientemente. Al conseguir el archivo de pesos del modelo entrenado, este será utilizado en un código de Python donde se encuentran las alarmas y la lógica a seguir para encenderlas.

Este trabajo está organizado de la siguiente manera: En la sección 2 se explican temas claves para tener un mejor entendimiento de la visión artificial, en la sección 3 se describe el estado del arte de la detección de somnolencia en conductores, en la sección 4 se explica la lógica del código a través de diagramas de flujos, en la sección 5 se muestran imágenes del sistema funcionando y en la sección 6 se presentan las conclusiones.

2. MARCO TEORICO

A continuación, se presentan las herramientas utilizadas para un mejor entendimiento de cómo se realizó el algoritmo.

YOLO. You Only Look Once (YOLO) es un algoritmo viral y ampliamente usado, YOLO es famoso por sus características de detección de objetos. El núcleo del algoritmo de detección de objetivos de YOLO radica en el pequeño tamaño del modelo y la rápida velocidad de cálculo. La estructura de YOLO es sencilla, puede generar directamente la posición y categoría del cuadro delimitador a través de su red neuronal [4].

La arquitectura YOLO introdujo el enfoque diferenciable de extremo a extremo para la detección de objetos unificando las tareas de regresión y clasificación de objetos en una única red neuronal. Fundamentalmente, la red YOLO consta de tres componentes principales. La columna vertebral, una red neuronal convolucional, se encarga de codificar información de la imagen en mapas de características a diferentes escalas. A continuación, estos mapas de características son procesados por el cuello, una serie de capas diseñadas para integrar y refinar las representaciones de las características. Por último, el módulo de la cabeza genera predicciones para los objetos y etiquetas de clase basadas en las características procesadas [5]. Este proceso se ilustra en la Figura 1 y aplica en todas las versiones de YOLO.

Algunos de los parámetros que se deben ingresar al realizar el modelo en YOLO son [6]:

- **Número de imágenes:** Define la cantidad de las imágenes de entrada.
- **Batch:** Es un subconjunto del dataset, usado para actualizar los parámetros del modelo en cada iteración.
- **Épocas:** Define el número de épocas de entrenamiento.
- **Pesos (weights):** Este archivo contiene los resultados después que el modelo del YOLO ha sido entrenado.

YOLOv5. YOLOv5 es una mejora natural al modelo de YOLOv4, incluyendo sus características (arquitectura mejorada, entrenamiento autoadversial, optimización de hyperparametros) pero fue desarrollado en Pytorch en lugar de Darknet. YOLOv5 incorpora un algoritmo de Ultralytics llamado AutoAnclaje. Esta herramienta de preentrenamiento verifica y ajusta las cajas de anclaje si no se ajustan adecuadamente al conjunto de datos y a la configuración de entrenamiento, como el tamaño de la imagen [7].

OpenCV. Es una librería de visión por computadora de código abierto. Su biblioteca está escrita en C y C++ y se ejecuta en Linux, Windows y Mac OS X. Hay desarrollo activo en interfaces para Python, Ruby, Matlab y otros lenguajes [8].

Ultralytics. Ultralytics es el hogar de los modelos de visión por computadora más avanzados para tareas como la clasificación de imágenes, la detección de objetos, la segmentación de imágenes y la estimación de poses [9].

PyTorch. PyTorch es una librería para programas en Python que impulsan programas de deep learning. Con esta receptividad y conveniencia, PyTorch es muy útil en el desarrollo de deep neural networks (redes neuronales profundas). Tiene un alcance amplio y se aplica para diversas aplicaciones de machine learning [10].

Numpy. Es una librería de Python que proporciona un objeto de array multidimensional, varios objetos derivados (como arrays enmascarados y matrices), y un surtido de rutinas para operaciones rápidas sobre arrays, incluyendo operaciones matemáticas, lógicas, manipulación de formas, ordenación, selección, E/S, transformadas discretas de Fourier, álgebra lineal básica, operaciones estadísticas básicas, simulación aleatoria y mucho más [11].

Se especializa en el procesamiento numérico a través de ndarrays multidimensionales, donde los matrices permiten operaciones elemento a elemento, también conocidas como broadcasting.

3. ANTECEDENTES

El desarrollo de detectores de somnolencia en conductores ha sido previamente estudiado en otros trabajos. Para esto se ha utilizado la metodología de la relación de aspecto de ojo (EAR) y Haar Cascade (Chellappa et al., 2018).
A pesar de que estos modelos son funcionales bajo ciertos parámetros, los resultados de la relación de aspecto de ojo (EAR) se ha demostrado muy variante respecto un sujeto a otro, además de que no se consideran en sus resultados pruebas bajo diferentes niveles de iluminación, por lo que no presentan suficiente información como para ser considerados funcionales en entornos de operación reales.

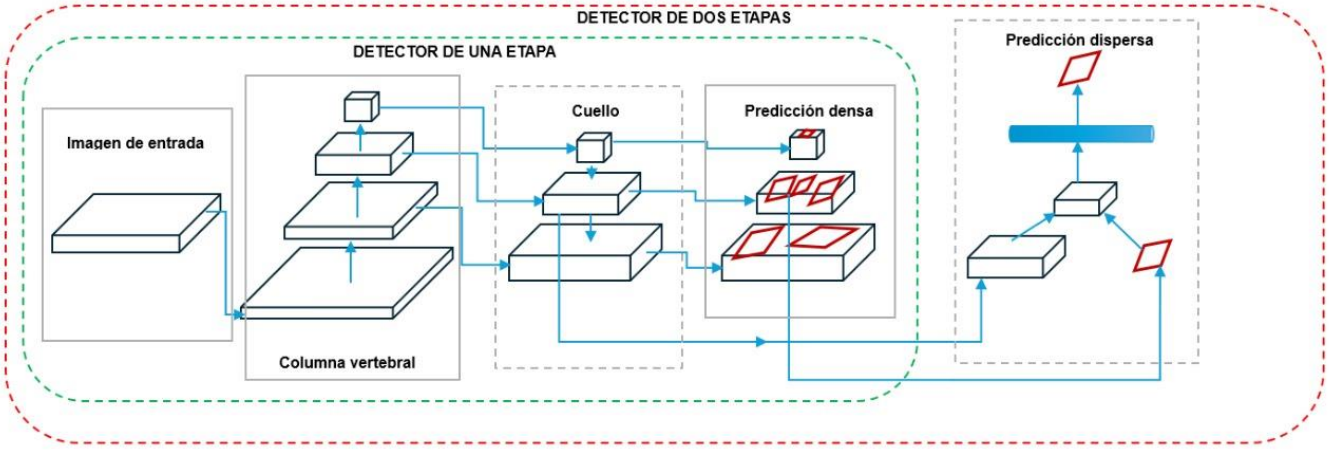

Figura 1: Proceso de la detección de objetos con YOLO.

4. METODOLOGÍA

4.1. Modelo de entrenamiento

El algoritmo propuesto consta de dos etapas principales, la primera enfocada en el entrenamiento de la red neuronal YOLOv5, para obtener un modelo capaz de reconocer ojos cerrados u ojos abiertos en una imagen, con el cual se trabaja para realizar el algoritmo en la segunda etapa.

En la Figura 2 se observa el diagrama de flujo de la primera etapa del proceso. El resultado de este proceso son los pesos de la red neuronal entrenada, los cuales podemos utilizar en Python mediante la librería Torch. Los pasos para esto se resumen en el Algoritmo 1.

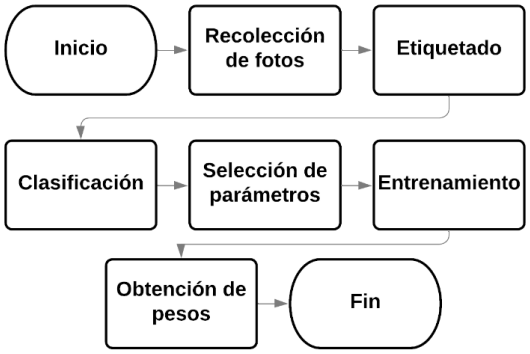

Figura 2: Diagrama de flujo del método de entrenamiento de la red YOLOv5.

Algoritmo 1 Pseudocódigo del entrenamiento de la red YOLOv5.	
PASO 1	Recolectar imágenes de ojos cerrados y abiertos a partir de repositorios públicos e imágenes propias.
PASO 2	Realizar un proceso de etiquetado de las imágenes, que consiste en indicar los segmentos de imagen que corresponden a los ojos; diferenciando entre etiquetas de ojos abiertos y cerrados.
PASO 3	Clasificar las imágenes en dos grupos: un grupo de entrenamiento, utilizado para entrenar a la red neuronal; y un grupo de validación, utilizado para verificar el rendimiento del entrenamiento.
PASO 4	Seleccionar el número de épocas, batch size y modelo de YOLOv5.
PASO 5	Entrenar el modelo y obtener sus pesos.

El modelo fue entrenado utilizando el código “train.py” dado como herramienta de entrenamiento en el repositorio de ultralytics dedicado a YOLOv5 [12] que utiliza principalmente las librerías de PyTorch y NumPy para el entrenamiento.

Para el entrenamiento del modelo de YOLOv5, se seleccionaron 150 épocas, un batch size de 8 y los pesos de YOLOv5n (nano), utilizando 570 imágenes para el entrenamiento. El GPU utilizado para el entrenamiento fue una GPU NVIDIA T4 utilizando computación en la nube.

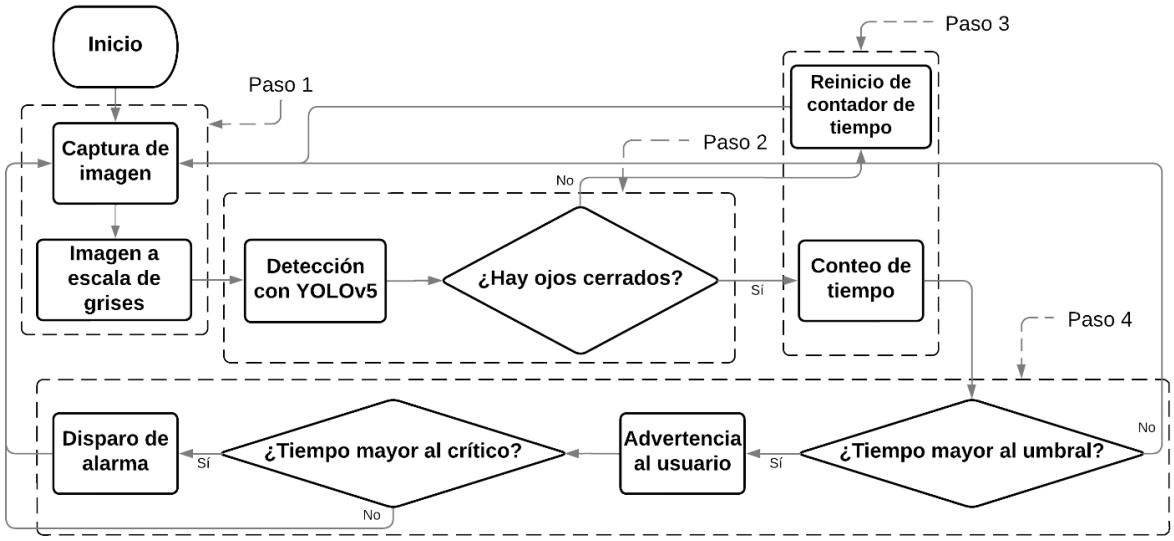

Figura 3: Algoritmo del detector de somnolencia.

4.2. Desarrollo del algoritmo

Una vez obtenidos los pesos que diferencien entre ojos abiertos y cerrados en una imagen, se puede crear el detector de somnolencia con programación en Python. El algoritmo de funcionamiento se presenta en la Figura 3 y sus pasos se resumen en el Algoritmo 2.

Algoritmo 2 Pseudocódigo del sistema de detección de somnolencia.	
PASO 1	Leer las imágenes de entrada y aplicar un filtro de ajuste de colores a escala de grises.
PASO 2	Realizar la detección con los pesos de YOLOv5, discriminando las detecciones para trabajar con las detecciones de ojos cerrados.
PASO 3	Inicializar un conteo de tiempo que se mantenga mientras se sigan obteniendo detecciones de ojos cerrados y se reinicie si se detectan ojos abiertos.
PASO 4	Si el tiempo contabilizado es mayor al tiempo de umbral (2 segundos) se activará una advertencia al usuario; en caso de llegar al tiempo crítico (4 segundos) se dispara la alarma hasta que el usuario abra los ojos.

Las advertencias mencionadas en el Algoritmo 2 tienen una duración breve y son sutiles, pues tienen un carácter informativo; mientras que la alarma es más sonora y de mayor duración; pues tiene el objetivo de despertar a un conductor dormido.

5. RESULTADOS

Se realizaron diferentes pruebas utilizando 3 secuencias de video de la vida real que incluyen cambios de sujetos (con y sin utilizar anteojos), escala e intensidad lumínica. Las secuencias contienen 1600 cuadros de escena. 1200 de ellos con 640 x 480 pixeles, y los 400 restantes con 848 x 638 pixeles. Las secuencias utilizadas para las pruebas difieren con el material de entrenamiento para verificar el funcionamiento del modelo. El algoritmo se implementó con YOLOv5n en Python 3.10.2 en una computadora portátil con un procesador Ryzen 5 2.1 GHz con 8 GB de memoria RAM y una GPU AMD Radeon Vega 8 Graphics, que se ejecuta en Windows 10 de 64 bits.

En cuanto al desempeño de velocidad de este, se logró un tiempo de procesamiento de aproximadamente 200 ms por cuadro (con tiempos picos de 300 ms en los cuadros de cambio en la detección), lo que nos arroja que el detector es funcional en entornos de tiempo real. Adicionalmente, el preprocesado de la imagen a escala de grises permite que la detección sea precisa en diferentes escalas de iluminación.

Ejemplos de los resultados del funcionamiento del algoritmo se pueden apreciar en las Figuras 3 y 4. La Figura 3 representa el estado normal de conducción, mientras el conductor mantenga los ojos abiertos, el detector estará en un ciclo de procesamiento de las imágenes hasta detectar el cambio en la detección.

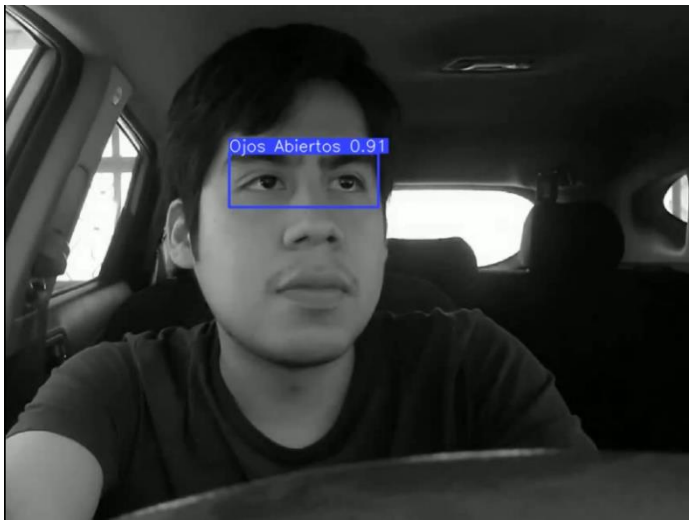

Figura 3: Detección de ojos abiertos.

En la Figura 4 se observa el estado en operación del detector, una vez que se detectan ojos cerrados en el conductor, se comienza un conteo de tiempo, se considera una alerta a los 2 segundos que permite al conductor conocer el estado del detector y que presenta signos de somnolencia; si la detección continua en el segundo 4 se considera que el conductor se encuentra dormido y se dispara la alarma final.

Figura 4: Detección de ojos cerrados.

Tabla 1: Desempeño del algoritmo en cuanto al porcentaje de aciertos.

Tipo de detección	Número de cuadros	Porcentaje del total
Exitosa	1586	99.125%
Fallida	6	0.375%
Sin detección	8	0.5%

La Tabla 1 muestra el desempeño de éxito del algoritmo en las secuencias de video analizadas, se puede observar que el detector tiene una tasa de éxito mayor al 99% en cuanto a detecciones exitosas del algoritmo, lo que es un indicador de su buen funcionamiento.

6.CONCLUSIONES

En este trabajo se presenta el diseño de un algoritmo de visión artificial para detectar somnolencia usando YOLOv5 para prevenir accidentes viales. Con este algoritmo se es capaz de diferenciar si el conductor tiene sus ojos abiertos o cerrados; con lo que se puede determinar si está dormitando (si se superan ciertos intervalos de tiempo). El algoritmo propuesto para detección de somnolencia en conductores fue capaz de detectar correctamente el 99% de los cuadros de las secuencias de video.

Se considera también a futuro la implementación del algoritmo en dispositivos microprocesadores, tales como la tarjeta NVIDIA Jetson Nano o Raspberry Pi; así como la adición de un módulo de comunicaciones para el lanzamiento de mensajes de emergencia, que permitan el uso práctico del programa y contribuir en una mejor seguridad vial.

7.REFERENCIAS

[1] Real Academia Española (RAE). “Somnolencia”, en diccionario de la lengua española. [en línea], disponible: https://dle.rae.es/somnolencia. (7 de junio 2024).

[2] Cecilia Ana, S.R. Manuel y R.A. Ricardo. “Estadístico de colisiones en carreteras federales, 2022” en Documento Técnico No. 89. [en línea], disponible: https://www.sct.gob.mx/fileadmin/DireccionesGrales/DGAF/EST_Accidentes_CF/Anuario_Estadistico_de_Accidentes_en_Carreteras_Federales_2022.pdf. (7 de junio 2024).

[3] Chellappa, A., Reddy, M. S., Ezhilarasie, R., Suguna, S. K., & Umamakeswari, A. Fatigue detection using Raspberry Pi 3. International Journal of Engineering & Technology, 7(2.24). 2018. pp. 29-32.

[4] Peiyuan Jiang, Fangyao Liu, Ying Cai, Bo Ma. “A Review of Yolo Algorithm Developments”, in The 8th International Conference on Information Technology and Quantitative Management. China. 2020 & 2021.

[5] R. Khanam M. Hussain, WHAT IS YOLOV5: A DEEP LOOK INTO THE INTERNAL FEATURES OF THE POPULAR OBJECT DETECTOR, arXiv, Julio 2024, pp. 2-3.

[6] Google colaboratory. “Step 3: Train our custom YOLOv5 model”. Custom training with YOLOv5. [en línea] disponible: https://colab.research.google.com/github/roboflow-

ai/yolov5-custom-training-tutorial/blob/main/yolov5-custom-training.ipynb. (7 de junio 2024).

[7] Terven, J., Córdova-Esparza, D. M., & Romero-González, J. A. (2023). A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas. Machine Learning and Knowledge Extraction, 5(4), 1680-1716.

[8] B. Gary, K. Adrian. Learning OpenCV. O'Reilly Media, Inc, Gravenstein Highway North, Sebastopol, CA 95472. (2008). pg.1.

[9] Weights & Biases (W&B), "Ultralytics", en Weights & Biases Documentation, [en línea] disponible: https://docs.wandb.ai/guides/integrations/ultralytics

[10] Imambi, S. Prakash, K.B. Kanagachidambaresan G.R. Programming with TensorFlow. EAI/Springer Innovations in Communication and Computing. Springer, Cham. (2021).

[11] Bressert Eli. SciPy and NumPy. Sebastopol, CA. O'Reilly Media, Inc. 2013, pp. 1-2.

[12] Jocher Glenn, "YOLOv5 / train.py", en GitHub, [en línea] disponible: https://github.com/ultralytics/yolov5/blob/master/train.py