Detección de armas en tiempo real mediante técnicas de CNN

Martínez Quintanar, José Alfredo

2023

https://hdl.handle.net/20.500.11777/5927

Detección de armas en tiempo real mediante técnicas de CNN

Raúl Badillo Lora (séptimo semestre en Ingeniería en Sistemas Computacionales)[1], Aldo Cova Martínez (décimo semestre en Ingeniería en Sistemas Computacionales)[1], José Alfredo Martínez Quintanar (onceavo semestre en Ingeniería Mecatrónica)[1,*], Dr. Antonio Benitez Ruiz (profesor responsable)[1], Rafael Pérez Aguirre (profesor asesor)[1]

[1]*Universidad Iberoamericana Puebla, San Andrés Cholula, Puebla, México*

Resumen

En este trabajo se desarrolla una propuesta de solución para situaciones de violencia que involucran armas de fuego y armas blancas, la cual va en incremento en el país. La propuesta se basa en el uso de algoritmos de aprendizaje automático junto con técnicas de visión por computadora y procesamiento digital de imágenes. Se desarrolla, además, un prototipo que integra un sistema de percepción a través de una cámara, que permite identificar en una imagen si existe un arma de fuego o blancas, lo cual se clasifica como una situación de violencia. Para el desarrollo del prototipo de detección se compararon tres algoritmos de aprendizaje profundo (basados en redes neuronales): una red neuronal convolucional (CNN), una red neuronal CNN-VGG19, ambas utilizando TensorFlow, y una R-CNN tomada desde la librería de YOLO. Para poder entrenar estos algoritmos se integró un conjunto de datos, los cuales se etiquetaron de acuerdo a las dos clases de interés ("*Handgun*" o "*Knife*"). Los resultados muestran que la red R-CNN obtuvo el mejor índice de aprendizaje, dado que su proceso de detección alcanzó una precisión de un 89%, un *F1-Score* de un 90% y un *recall* de un 85%.

Palabras clave: Detección de violencia, detección de armas, aprendizaje automático, visión por computadora, red R-CNN.

[*]**Autor Corresponsal:** alfredo.mquintanar13@gmail.com

Introducción

La seguridad pública se ha vuelto un tema de gran importancia en el mundo, lo que ha llevado a varios países a revolucionar la manera en la que trabajan para aplicar la ley efectivamente. En la última década, en México la incidencia delictiva ha mantenido una tendencia a la alza: el INEGI ha reportado la cifra de 30,786 incidentes por 100,000 habitantes [1], mientras que la tasa de homicidios ha llegado a su punto más alto en el último siglo, con una frecuencia de 35,625 decesos, tan solo en 2021 [2]. Además, en el 2022 en el estado de Puebla se registraron más de 912 homicidios, convirtiéndolo en el doceavo estado con más homicidios del país [3].

Debido a ello, la detección temprana de situaciones potencialmente violentas es esencial para minimizar daños y promover la seguridad de las personas. Sin embargo, el monitoreo constante y la identificación manual de amenazas resultan poco prácticos y están sujetos a errores humanos. Es por esta razón que el presente trabajo busca desarrollar un sistema que pueda detectar armas de fuego cortas y armas blancas en tiempo real, usando técnicas de visión por computadora e inteligencia artificial.

En la actualidad, las tecnologías de la información han implementado el uso de técnicas de Inteligencia Artificial (IA) y Machine Learning (ML) en una amplia variedad de sectores, teniendo aplicaciones desde el comercio electrónico hasta la agricultura, pasando por la educación, las finanzas, la robótica, entre muchos otros [4]. El objetivo fundamental de la IA es diseñar tecnología que emule la inteligencia humana [5].

Una de las áreas de especialización de esta ciencia es el Deep Learning (DL), que consiste en conjunto de clasificadores llamados redes neuronales, por la configuración que adoptan [6]. La Fig.1, presenta cómo está integrada el área de la inteligencia artificial (IA). Como se puede apreciar, las redes neuronales de aprendizaje profundo (DL) son sólo una parte de los algoritmos de redes neuronales (RN), las cuales forman parte de los algoritmos de aprendizaje automático (ML), que a su vez son un subconjunto de todas las técnicas que integran la IA. Estas redes están compuestas por múltiples capas de "neuronas" interconectadas, lo que les da una mayor capacidad de manipular características complejas y abstractas en los datos [7]. De esta idea surgen las redes neuronales convolucionales (CNN), las cuales poseen una capacidad de extracción y procesamiento de datos superior a otros tipos de redes neuronales, debido a su arquitectura interna [8].

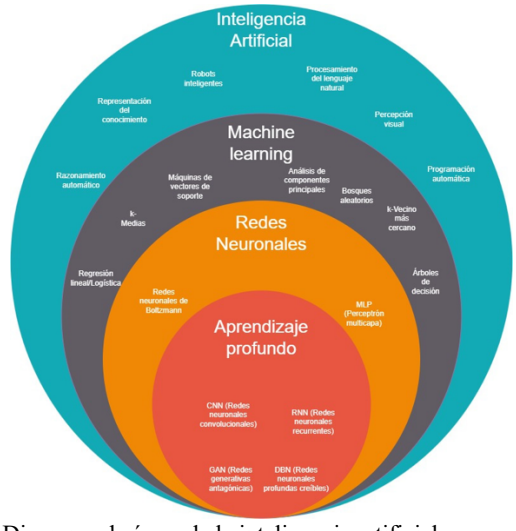


Fig.1. Diagrama de áreas de la inteligencia artificial.

Los algoritmos de ML se basan en el análisis de datos, por lo que la recopilación y preparación adecuada de éstos es

esencial [9]. Esto implica obtener conjuntos de datos relevantes y representativos del problema a resolver, asegurándose de que estén limpios y listos para su análisis, ya que su calidad tiene un impacto significativo en el rendimiento de los algoritmos de ML [10]. Cabe destacar que existen diversos algoritmos de ML disponibles y el criterio de selección del más adecuado depende, en gran medida, del problema a resolver. Una vez recopilados los datos y seleccionado un algoritmo, se puede iniciar el entrenamiento del modelo [11].
El proceso para la utilización de algoritmos de aprendizaje automático se muestra en la Fig. 2.

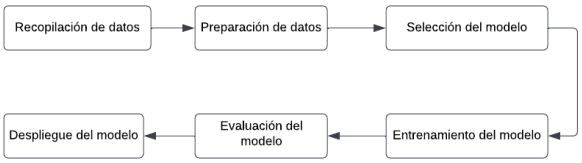


Fig.2. Diagrama con el proceso para el desarrollo de algoritmos de aprendizaje automático.

En el ámbito del desarrollo de ML, existen diversas tecnologías y librerías que desempeñan un papel fundamental en la construcción y el despliegue de modelos de IA efectivos. En este proyecto se utilizaron Ultralytics, CUDA, OpenCV y Roboflow. Cada una aporta capacidades únicas y se adapta a diferentes aspectos del ciclo de desarrollo de algoritmos de ML, desde el entrenamiento de modelos hasta la gestión de datos y la optimización del procesamiento de imágenes y videos. A continuación, se describen sus aspectos más relevantes.
CUDA, desarrollada por NVIDIA, es una plataforma de computación paralela que revoluciona la velocidad de procesamiento de datos en proyectos de IA. Su capacidad para aprovechar la potencia de las unidades de procesamiento gráfico (GPU) es esencial para acelerar tanto el entrenamiento como la inferencia de modelos de IA [12].
Ultralytics es una librería de código abierto que contribuye al aprendizaje automático y el desarrollo de IA [13]. Esta herramienta ofrece modelos de IA preentrenados, como el conocido YOLOv5, altamente valiosos en tareas de gran relevancia, como la detección de objetos, segmentación y clasificación de imágenes [14].
OpenCV, una librería de visión por computadora de código abierto, se destaca por su versatilidad en proyectos de IA relacionados con el procesamiento de imágenes y videos. Ofrece un conjunto completo de herramientas que abarcan desde la detección y seguimiento de objetos, hasta la segmentación de imágenes y calibración de cámaras, contribuyendo así a una amplia variedad de aplicaciones de IA [15].
Roboflow emerge como una plataforma integral de gestión de datos de IA. Su función principal es etiquetar y organizar conjuntos de datos de imágenes y videos, simplificando el proceso de preparación de datos, además de proporcionar herramientas para el entrenamiento y la inferencia de modelos de IA, y facilitar su implementación en dispositivos locales [16].

Metodología

Integración del Dataset

El conjunto de datos conocido como *dataset*, es vital para cada algoritmo de entrenamiento. En esta etapa, se procedió a recopilar el conjunto de datos necesario para llevar a cabo la investigación, identificando fuentes abiertas de imágenes relacionadas con el tema de estudio, armas de fuego (pistolas y revólveres) y armas blancas (cuchillos y machetes), además de ejemplos de no violencia (personas en situaciones normales sin armas) para el caso negativo de detección, que abonaran a la calidad y diversidad del conjunto de datos. En la Fig. 3 se aprecian ejemplos de imágenes correspondientes a las dos clases de los *datasets* recopilados, extraídos de [ref].
Una vez obtenidas las imágenes, se realizó su preprocesamiento, que incluyó la corrección de posibles imperfecciones, la normalización de tamaños y formatos, así como la eliminación de datos irrelevantes o ruidosos, con el propósito de uniformar los datos.
Posteriormente se etiquetó cada imagen en el conjunto de datos indicando si contenía armas o personas, para procurar mantener cierta calidad en los datos de entrenamiento.

Fig. 3. Conjunto de imágenes etiquetadas (“*Handgun*” y “*Knife*”).

El *dataset* se dividió en tres conjuntos principales: entrenamiento, validación y pruebas. El 70% de los datos se asignó al conjunto de entrenamiento, el 20% al conjunto de validación y el 10% se destinó al de pruebas; estos porcentajes se basaron en lo reportado en la referencia [9], para evitar el sobreajuste y permitir una evaluación rigurosa.
El *dataset* utilizado en este estudio incluyó más de 1000 imágenes para la clase "Riesgo" y 1000 imágenes adicionales para la clase "No riesgo", por lo que el total de imágenes procesadas fueron más de 2000. Para facilitar la preparación del *dataset*, se utilizó la herramienta Roboflow [15], que permite el etiquetado de objetos, la conversión de formatos de datos y la organización eficiente de los mismos.

Entrenamiento de algoritmos

Como se mencionó anteriormente, el *dataset* que se exportó fue el mismo para todos los algoritmos que se entrenaron y se gestionó a través de la plataforma Roboflow, separándolo en carpetas de entrenamiento, validación y pruebas, con sus respectivas clases; también cabe mencionar que cada algoritmo fue entrenado con un total de 50 épocas [ref], para ver cual se desempeñaba mejor.

El algoritmo de CNN se construyó con TensorFlow, ya que tiene capacidad de reconocimiento de imágenes, incluyendo la estructura espacial [17].
Una vez finalizada la generación de datos, se diseñó la arquitectura de la CNN, que constó de 10 capas de entrada con un *input_shape* de 250 x 250 pixeles, 8 capas internas encargadas de procesar las imágenes y una capa de salida para las clases. Luego, se procedió a compilar el modelo utilizando *optimizer*, *loss* y *metrics*.
Se usó el algoritmo de CNN-VGG19 embebido en TensorFlow, el cual es una red neuronal con la estructura básica de una CNN, con la diferencia que es un modelo ya preconstruido donde se pueden cargar los datos para ser entrenado, y el cual cuenta con 19 capas.
Para el algoritmo de entrenamiento de YOLOv8, se utilizó un archivo yaml, necesario para el entrenamiento, ya que contenía las direcciones de las imágenes, validación y pruebas, así como información sobre las dos clases a identificar: "Riesgo" y "No riesgo". El archivo yaml también contenía las rutas de las carpetas mencionadas previamente.
Una vez exportado el *dataset*, se utilizó el lenguaje de programación Python para entrenar el modelo YOLOv8 usando el modelo base de YOLO (yolov8n.pt). Para configurar estos archivos en el script de Python correspondiente al entrenamiento, se importaron las librerías de Ultralytics, se cargó el modelo base y el archivo yaml. Luego se entrenó el modelo configurando aspectos como el nombre y ubicación de dichas carpetas, y el GPU a utilizar.

Evaluación de los algoritmos

Para medir el desempeño del modelo, se evaluaron diversas métricas, como la precisión, el *recall*, la precisión por clase y la matriz de confusión, para determinar la capacidad de clasificación del algoritmo.
La matriz de confusión es una tabla construida a partir de los resultados de la clasificación del modelo, donde TP son los verdaderos positivos, FP son los falsos positivos, FN son falsos negativos y TN los verdaderos negativos, y se representa matemáticamente en la Ec. (1).

$$\begin{bmatrix} TP & FP \\ FN & TN \end{bmatrix} \quad \dots (1).$$

La precisión se define como la proporción de verdaderos positivos (TP) en relación con los valores positivos predichos (TP + FP), donde FP son los falsos positivos, como se muestra en la Ec. (2).

$$Precisión = \frac{TP}{(PT + PF)} \quad \dots (2).$$

El *Recall*, o sensibilidad de detección, se define como la proporción de los verdaderos positivos (TP), en relación con los valores de positivos reales (TP + FN) y se calcula de la siguiente forma:

$$Recall = \frac{TP}{(TP + FN)} \quad \dots (3).$$

Por último, para la *F1 Score*, que mide la exactitud del modelo, se combina la precisión y la sensibilidad en una sola medida. La fórmula es la media armónica de la precisión y sensibilidad:

$$F1 = 2\left(\frac{(Precisión \cdot Recall)}{(Precisión + Recall)}\right) \quad \dots (4).$$

Desarrollo de la interfaz

Para finalizar, se creó el diseño e implementación de una interfaz gráfica (Fig. 4), la cual tiene una ventana principal donde a través de una cámara web se puede ver lo que está sucediendo en tiempo real. Se le incluyó una ventana para el historial donde se registraría una captura de pantalla del escenario de riesgo con la fecha y hora del mismo, incluyendo dos botones, uno de descarga y otro para eliminar dicha alerta emitida. La última ventana sólo es una descripción del proyecto y especificaciones del mismo. Para evitar que el personal esté siempre en el monitor, se implementó un sistema de automatización de mensajes SMS, lanzando una alerta automática en cuanto el sistema detecte un escenario de riesgo.
Todo este proceso se realizó con Flask [18], el cual es un microframework, que se puede integrar con Python, para ejecutar un servidor de forma local dentro de un dispositivo de cómputo. Para la cama web se usó OpenCV, esto permite activar la cámara y detectar los escenarios de riesgo. Para la automatización de los SMS se utilizó Twilio [19], que es un servicio de SMS donde se configura una API_KEY [20] y el número al que se enviarán los SMS de forma automática.

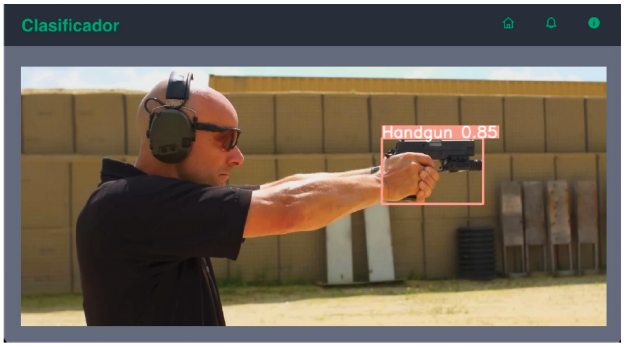
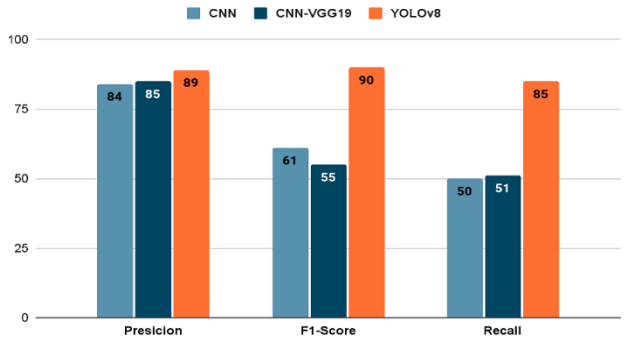


Fig. 4. Interfaz gráfica del detector de armas en tiempo real, desplegado con Flask en servidor local.

Resultados y Discusión

Se construyó un dataset con 2792 imágenes, manteniendo un balance entre las clases de "*Handgun*" y "*Knife*", para evitar que hubiera una preferencia en las predicciones hechas, además de contener imágenes nulas, para reducir la confusión con objetos que no fueran de interés.
Se entrenaron tres modelos diferentes para la detección de armas, que se evaluaron con un conjunto de imágenes distinto al utilizado durante el entrenamiento, y se calcularon métricas para comparar su desempeño.
Se determinó que el más adecuado era el que se construyó utilizando el algoritmo YOLOv8, con base en las métricas de su desempeño, que se muestran en la tabla 1.

Gráfica 1: Comparativa de promedios en las diferentes pruebas realizadas a cada algoritmo.



En ella se observa que la precisión de los tres tiene un valor similar; sin embargo, la diferencia es notoria en los demás parámetros, debido a que YOLOv8 es un modelo que ha sido previamente entrenado con un conjunto de datos enorme, mientras que los otros modelos, aunque competentes, requieren de ajustes y entrenamiento adicional para alcanzar un rendimiento similar, especialmente en tareas específicas como la detección de objetos en entornos complejos o bajo condiciones variables.

Se desarrolló un prototipo web para el modelo de detección de imágenes, demostrando la viabilidad y funcionalidad del modelo en un entorno de prueba (Fig. 5).

Fig. 5. Interfaz del historial de las detecciones de armas registradas.

A pesar de que el dataset que se utilizó en todos los entrenamientos fue el mismo, el desempeño de cada modelo se vío afectado dependiendo del preprocesamiento usado. El modelo de YOLOv8 demostró una mejora al recibir el dataset sin modificaciones, mientras que para los otros dos modelos, la CNN y CNN-VGG19, hubo que normalizar las imágenes a 250 x 250 pixeles, ya que este tamaño resultó tener el mejor equilibrio entre la calidad de resultados y la velocidad con la que los generaba.

Un punto a resaltar, fue que al iniciar los entrenamientos se aplicaron 100 épocas con el fin de obtener buenos resultados, pero se encontró que este número no subía el porcentaje de la precisión, sino provocaba el sobreentrenamiento e impedía que funcionara de forma adecuada. Analizando la gráfica de pérdidas, se observó que en 50 épocas se reducía la perdida de datos por iteración y eran suficientes para el entrenamiento.

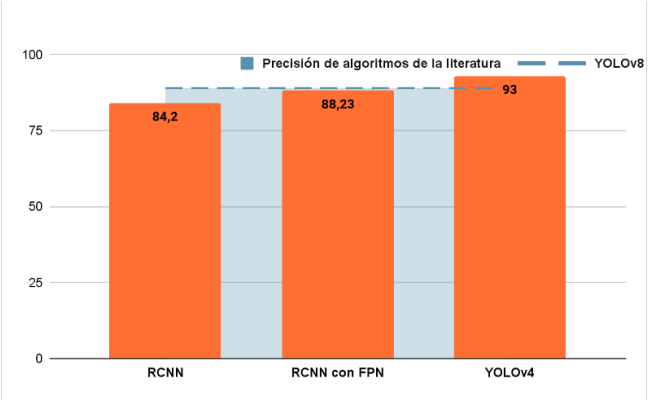
El modelo de clasificación qué se entrenó con CNN tuvo una precisión de un 84%, pero, los resultados de la matriz de confusión por clase salieron bajos, con una precisión del 59%.

Con el modelo de CNN-VGG19 se obtuvo una precisión del 88%, pero los resultados de la matriz de confusión eran bajos, ya que por cada clase mostraba un desempeño de un 51%, que fue el peor de todos.

El modelo de detección que se entrenó con YOLOv8 tiene una precisión del 89%; al observar la matriz de confusión se nota que cada clasificación se desempeñó correctamente, arriba del 90%.

Se hizo un comparación de los resultados obtenidos respecto a los encontrados en la literatura ([21]-[23]), demostrando que nuestro modelo se comportó mejor que el promedio, obteniendo el segundo mejor resultado.

Gráfica 2: Comparativa de precisión y modelos en la literatura.



Se encontraron dificultades en el despliegue del prototipo en un servidor web, debido a problemas de compatibilidad. Además, se observaron limitaciones en su rendimiento, particularmente en situaciones donde se procesaba una gran cantidad de información, lo que ocasionalmente resultaba en el bloqueo del sistema. Estos desafíos subrayan áreas clave para futuras mejoras y optimizaciones, y destacan la importancia de una evaluación exhaustiva del rendimiento del sistema en escenarios de uso intensivo de datos.

En cuanto a la interfaz gráfica, las tareas ejecutadas en Flask (mostrar el video, automatización de mensajes y registro de historial de detección) resultaron demasiado demandantes para el microframework, que se congelaba.

El proyecto ha superado desafíos en la recopilación y procesamiento de datos, así como en el entrenamiento de modelos. Aunque se han logrado resultados significativos, existen áreas de mejora identificadas en la calidad del dataset, el preprocesamiento de imágenes y la interfaz gráfica. Estos hallazgos proporcionan una base para futuras iteraciones y mejoras en la implementación de algoritmos de detección de objetos.

Conclusiones, perspectivas y recomendaciones

En este proyecto se desarrollaron tres modelos entrenados para la detección de armas blancas y de fuego, en tiempo real. De ellos, el que mejor se desempeñó fue el basado en YOLOv8, el cual superó a los otros con un porcentaje de

89% de precisión. Es importante recalcar que en las pruebas realizadas, a pesar de mostrar métricas favorables, al aplicarse en escenarios reales, los porcentajes se vieron afectados de forma sustancial, presentando predicciones tanto de alta como de baja fiabilidad, algo que es constantemente omitido en estudios similares.

Otro punto a destacar es que los modelos preentrenados, que suelen ser más referidos en la literatura, no muestran una diferencia en precisión tan significativa como se esperaría al compararlos con CNN programadas desde cero. Sin embargo, en el resto de parámetros obtuvieron mejores resultados en dichos modelos, dado que tienen ventaja debido a su aprendizaje de transferencia.

De cara a futuros estudios, se sugiere recopilar un *dataset* más extenso y variado, incluyendo más diversidad de armas detectables, ya que por el momento está limitado a pistolas y cuchillos, y realizar un preprocesamiento más exhaustivo, con el objetivo de mejorar la precisión inicial. También se propone agregar la detección de acciones que representen otros riesgos como violencia física.

Además, se propone usar un *framework* más robusto que Flask que, además de resolver los retos encontrados en la realización del proyecto, pueda soportar las mejoras sugeridas sin problemas. Finalmente, se espera desplegar el modelo en la nube, para su uso más óptimo.

Referencias

[1] Instituto Nacional de Estadística y Geografía (INEGI), "**Gobierno, Seguridad y Justicia**". [En línea]. Disponible: https://www.inegi.org.mx/temas/incidencia/#informacion_general

[2] Instituto Nacional de Estadística y Geografía (INEGI), "**DATOS PRELIMINARES REVELAN QUE EN 2021 SE REGISTRARON 35 625 HOMICIDIOS,**" *Comunicado de prensa*, 26 de julio de 2022. [En línea]. Disponible: https://www.inegi.org.mx/contenidos/saladeprensa/boletines/2022/DH/DH2021.pdf

[3] Instituto Nacional de Estadística y Geografía (INEGI), "**Mortalidad, Conjunto de datos: Defunciones por homicidio**". [En línea]. Disponible: https://www.inegi.org.mx/sistemas/olap/proyectos/bd/continuas/mortalidad/defuncioneshom.asp?s=est

[4] I. H. Sarker, "**Machine Learning: Algorithms, Real-World Applications and Research Directions**," SN Computer Science, vol. 2, no. 3, 2021. https://doi.org/10.1007/s42979-021-00592-x

[5] A. A. Khan, A. A. Laghari, y S. A. Awan, "**Machine learning in computer vision: a review**," *EAI Endorsed Transactions on Scalable Information Systems*, vol. 8, no. 32, pp. e4-e4, 2021.

[6] S. Dong, P. Wang and K. Abbas, "**A survey on deep learning and its applications**" in Computer Science Review, vol. 40, pp. 100379, 2021, doi: 10.1016/j.cosrev.2021.100379.

[7] Z. Li, F. Liu, W. Yang, S. Peng and J. Zhou, "**A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects**," in *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 12, pp. 6999-7019, Dec. 2022, doi: 10.1109/TNNLS.2021.3084827.

[8] P. Wang, E. Fan and P. Wang, "**Comparative Analysis of Image Classification Algorithms Based on Traditional Machine Learning and Deep Learning,**" in Pattern Recognition Letters, doi: 10.1016/j.patrec.2020.07.042, 2020.

[9] C. Janiesch, P. Zschech, y K. Heinrich, "**Machine learning and deep learning,**" *Electron Markets*, vol. 31, pp. 685-695, 2021. doi.org/10.1007/s12525-021-00475-2.

[10] X. Feng, Y. Jiang, X. Yang, M. Du, y X. Li, "**Computer vision algorithms and hardware implementations: A survey,**" *Integration*, vol. 69, pp. 309-320, 2019.

[11] F. Thabtah, S. Hammoud, F. Kamalov and A. Gonsalves, "**Data imbalance in classification: Experimental evaluation,**" in Information Sciences, vol. 513, pp. 429-441, 2020, doi: 10.1016/j.ins.2019.11.004.

[12] "**CUDA X**". NVIDIA. Accedido el 2 de octubre de 2023. [En línea]. Disponible: https://www.nvidia.com/es-la/technologies/cuda-x/

[13] "**Ultralytics | Revolutionizing the World of Vision AI**". Ultralytics. Accedido el 28 de agosto de 2023. [En línea]. Disponible: https://ultralytics.com/

[14] B. Jabir, N. Falih y K. Rahmani, "**Accuracy and efficiency comparison of object detection open-source models,**" *International Journal of Online & Biomedical Engineering*, vol. 17, no. 5.

[15] "**opencv-python**". PyPI. Accedido el 2 de octubre de 2023. [En línea]. Disponible: https://pypi.org/project/opencv-python/

[16] "**Roboflow: Give your software the power to see objects in images and video**". Roboflow: Give your software the power to see objects in images and video. Accedido el 2 de octubre de 2023. [En línea]. Disponible: https://roboflow.com/

[17] "**TensorFlow**". Google Brain Team. Accedido el 01 de septiembre de 2023. [En línea]. Disponible: https://www.tensorflow.org

[18] "**Welcome to Flask — Flask Documentation (3.0.x)**". Flask. Accedido el 22 de octubre de 2023. [En línea]. Disponible: https://flask.palletsprojects.com/en/3.0.x/

[19] "**Communication APIs for SMS, Voice, Video & Authentication | Twilio**". Twilio. Accedido el 22 de noviembre de 2023. [En línea]. Disponible: https://www.twilio.com/en

[20] J. Wulf and I. Blohm, "**Fostering value creation with digital platforms: A unified theory of the application programming interface design,**" Journal of Management Information Systems, vol. 37, no. 1, pp. 251-281, 2020

[21] M. T. Bhatti, M. G. Khan, M. Aslam and M. J. Fiaz, "**Weapon Detection in Real-Time CCTV Videos Using Deep Learning,**" in *IEEE Access*, vol. 9, pp. 34366-34382, 2021, doi: 10.1109/ACCESS.2021.3059170.

[22] R. Olmos, S. Tabik and F. Herrera, "**Automatic handgun detection alarm in videos using deep learning**", Neurocomputing, vol. 275, pp. 66-72, Jan. 2018.
[23] J. L. S. González, C. Zaccaro, J. A. Álvarez-García, L. M. S. Morillo and F. S. Caparrini, "**Real-time gun detection in CCTV: An open problem**", Neural Netw., vol. 132, pp. 297-308, Dec. 2020.