Localización de armas de fuego en imágenes y video usando Computer Vision

Grupo 11:
Xavier Eduardo Carlier Auz
Kevin Antonio Cevallos Pilay
Jeffrey Gabriel Prado Reyes

Descripción del problema

El crimen con armas de fuego es un problema serio que ocasiona muchas muertes alrededor del mundo, según la Oficina para Drogas y Crimen de las Naciones Unidas (UNODC) cada año alrededor de 238804 personas mueren a causa de un disparo [1]. En América Latina la Organización Mundial de la Salud (OMS) estima que hay un promedio de 28.5 homicidios por cada 100000 personas, de las cuales el 75% involucran alguna arma de fuego como pistolas, carabinas, etc [2]. Específicamente en Ecuador este problema se ha agravado mucho en los últimos años, la frecuencia de crímenes que involucren armas de fuego dentro del país está creciendo y no muestra señales de disminuir. En el 2021 se observó un incremento del 119% sobre el año anterior para asesinatos con armas de fuego en el país [3]. Para el 25 de abril del 2022 ya se han reportado 1241 muertes violentas en Ecuador, esto equivale al 49% de los homicidios en el 2021 [4]. Estas cifras dejan claro el peligro que representan los crímenes con armas de fuego en Ecuador.

Debido al narcotráfico el tráfico de armas de fuego al Ecuador ha incrementado, a pesar de los esfuerzos de decomisar armas ilícitas por parte de las autoridades. Por esta razón muchos establecimientos alrededor del país han incrementado las medidas de seguridad como contratar una mayor cantidad de guardias o el uso de detectores de metal para desalentar a los criminales [4]. Sin embargo, estas medidas no han sido efectivas en frenar el crecimiento de los crímenes violentos con armas de fuego dentro del país y no son aplicables en lugares abiertos. Entonces, es necesario recurrir a otros medios para detectar elementos peligrosos y alertar automáticamente a las autoridades. El problema que se quiere resolver en este proyecto es la falta de medidas para localizar automáticamente armas de fuego en el ambiente urbano.

Objetivo

Localizar la presencia de un arma de fuego en un ambiente urbano para alertar en el menor tiempo posible a las autoridades.

Modelos completos de análisis y definición del problema

En esta sección se presentan diagramas en lenguaje formal que describen los aspectos más importantes del problema que se quiere resolver. Por medio de estos diagramas se puede entender de mejor manera los requerimientos que se debe tomar en cuenta al diseñar una solución.

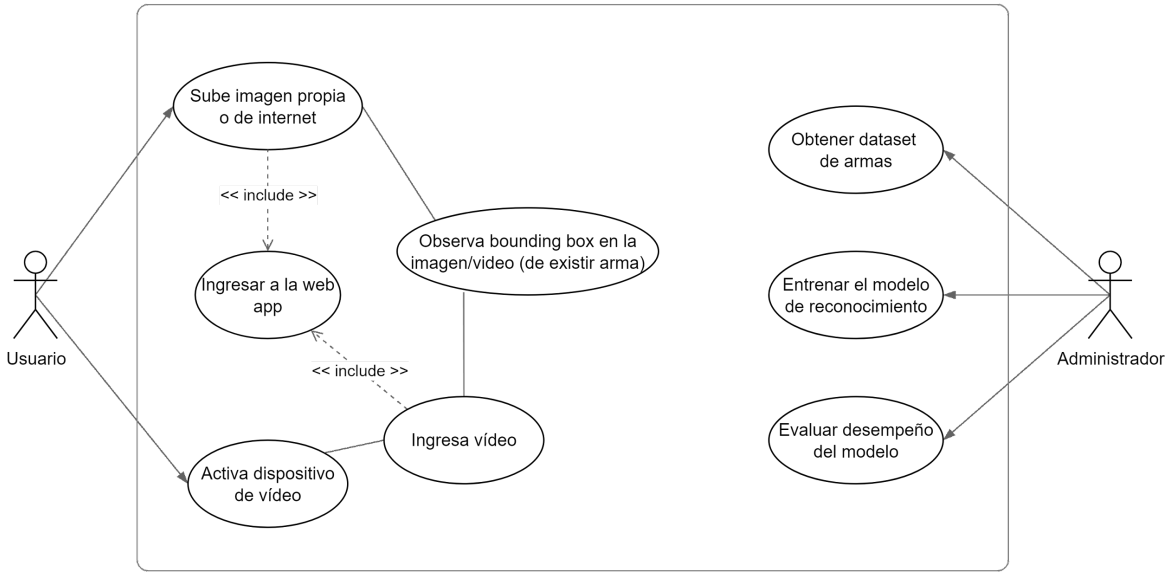


Diagrama de Casos de Uso

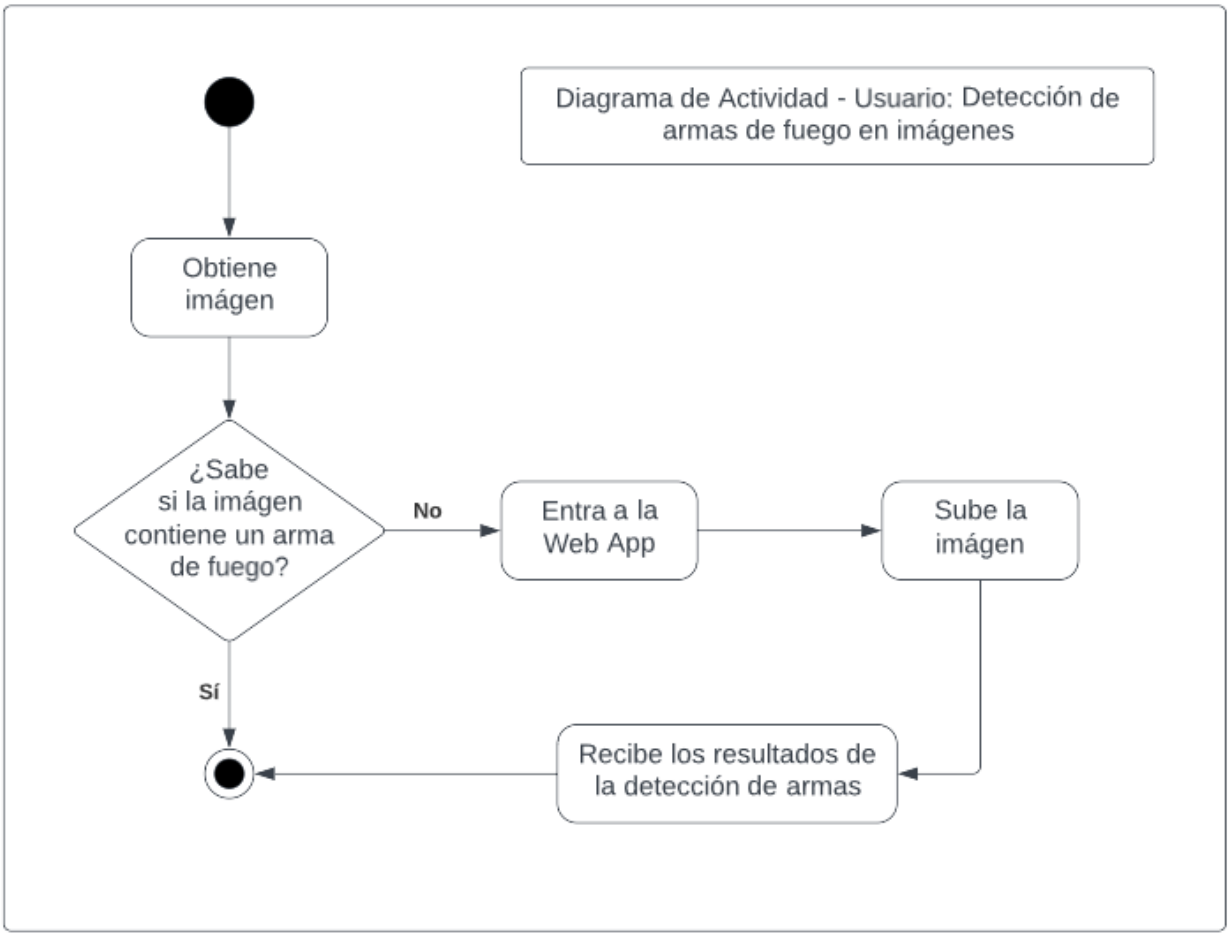


Diagrama de Actividad para la localización de armas de fuego en imágenes

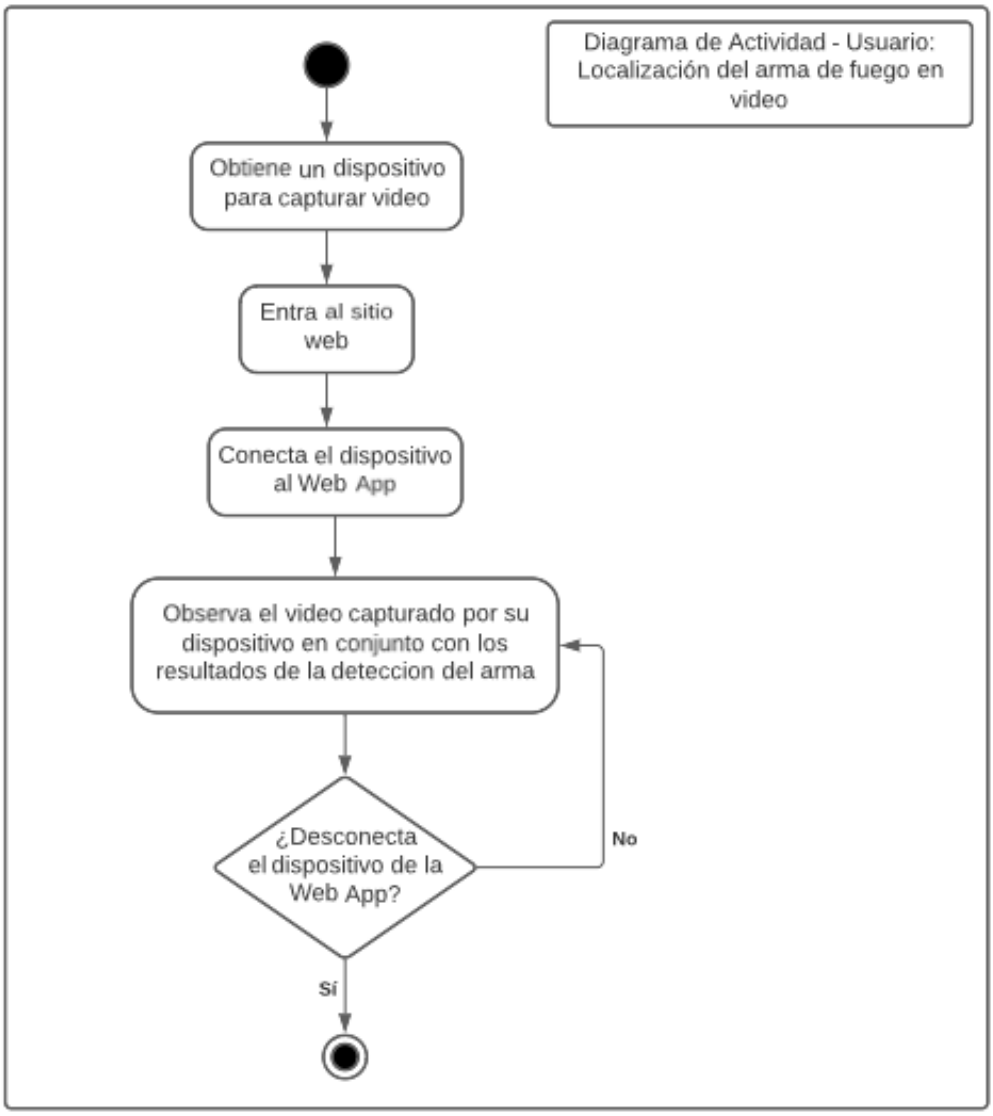


Diagrama de Actividad para la localización de un arma de fuego en video

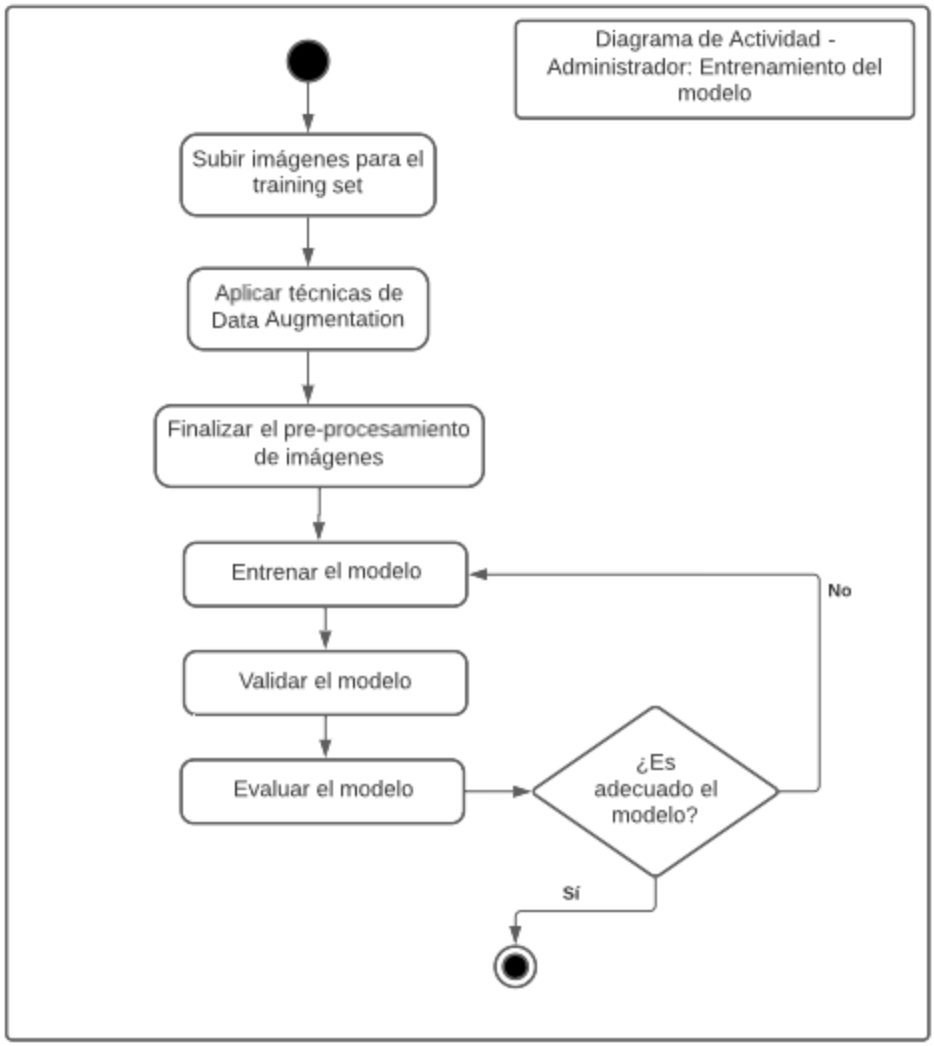


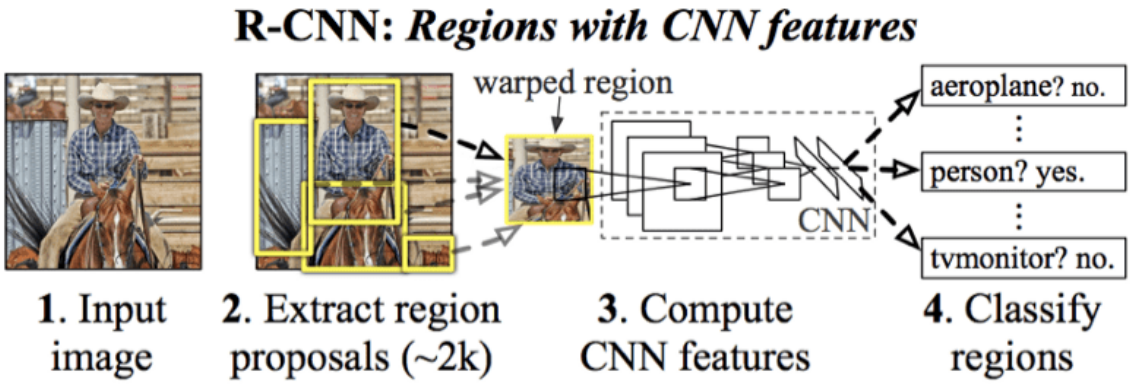
Diagrama de Actividad para el entrenamiento del modelo

Descripción preliminar de la solución

Nuestra propuesta de solución es implementar una red neuronal artificial que permita reconocer por medio de fotos o videos las personas que portan un arma, para alertar de inmediato a las autoridades para controlar y gestionar la situación reportada.

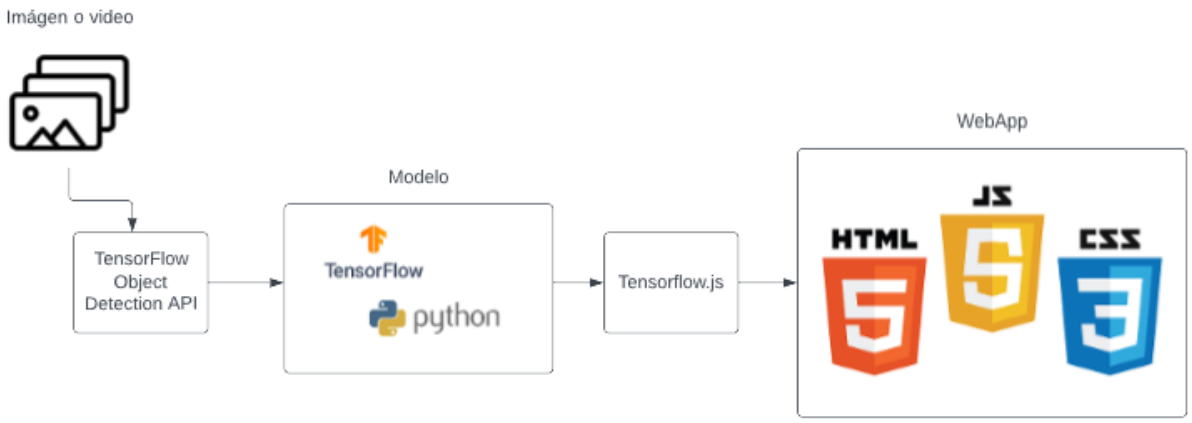
La red neuronal a ser aplicada es convolucional, dado que su aplicación en dos dimensiones resulta efectiva para tareas de visión artificial como clasificación de las imágenes. Junto con un aprendizaje supervisado y aplicado en RCNN (Region Based Convolutional Neural Networks) se ha demostrado mayor asertividad al momento de reconocer armas incluso en vídeos de baja calidad para activar alarmas en tiempos registrados de hasta 0.2 segundos. [5]

Las RCNN buscan cumplir principalmente dos tareas, clasificar y localizar objetos. La versión inicial de RCNN trabaja a partir del ingreso de una imagen, donde busca extraer de manera selectiva múltiples ROI (Region Of Interest) como en la imagen se muestran rectángulos que representan los límites de objetos y dependiendo del escenario pueden existir aproximadamente 2000 ROI. Continúa con la aparición de la red neuronal dónde se determina el tipo de objeto visualizado gracias a colecciones de SVM (Support Vector Machines) [6].



Imágen obtenida de DataScience [6]

La detección temprana de un arma portada por un civil se basa imperativamente en un buen conjunto de datos para entrenamiento del modelo donde es imperiosa la clasificación entre los diferentes tipos de armas de fuego entrando en escenario características como la longitud y el cañón del arma. Para entrenar el modelo se usará un dataset compilado por la Universidad de Granada que contiene alrededor de 3000 imágenes, tomadas de manera selectiva desde internet, de distintos tipos de pistolas [7]. Se aplicarán técnicas de data augmentation como horizontal flip para aumentar el tamaño del training set. Se tomará en cuenta que la transformación de imágenes no resulte en imágenes poco realistas para la aplicación que se quiere construir [8]. Por ejemplo realizar un flip vertical a una imágen de una persona con un arma no sería válido porque no sería un caso realista, para este caso un flip horizontal tendría más sentido.



Diseño general preliminar de la solución.

La posible solución contará con el entrenamiento del modelo por medio de servicios como el TensorFlow lite model maker, el cual recibirá imágenes o video preprocesado usando herramientas como el Object Detection Api de Tensorflow y otras librerías de Python. Los resultados de la predicción realizada por el modelo se podrían mostrar en una WebApp que mostraría la interfaz final para el usuario donde a través de la cámara se reconocerá armas en tiempo real registrando la velocidad de reacción y reconocimiento por parte del modelo. Se podría usar Tensorflow.js para desplegar el modelo entrenado en una aplicación Web que use Javascript.

Modelos preliminares de la solución

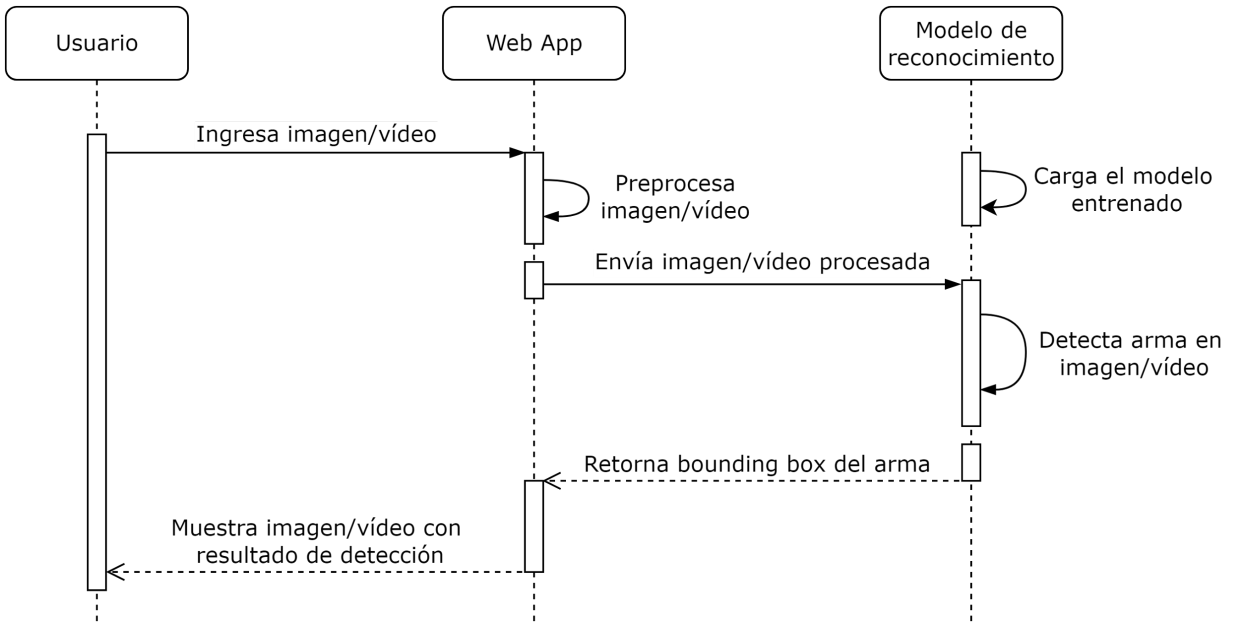
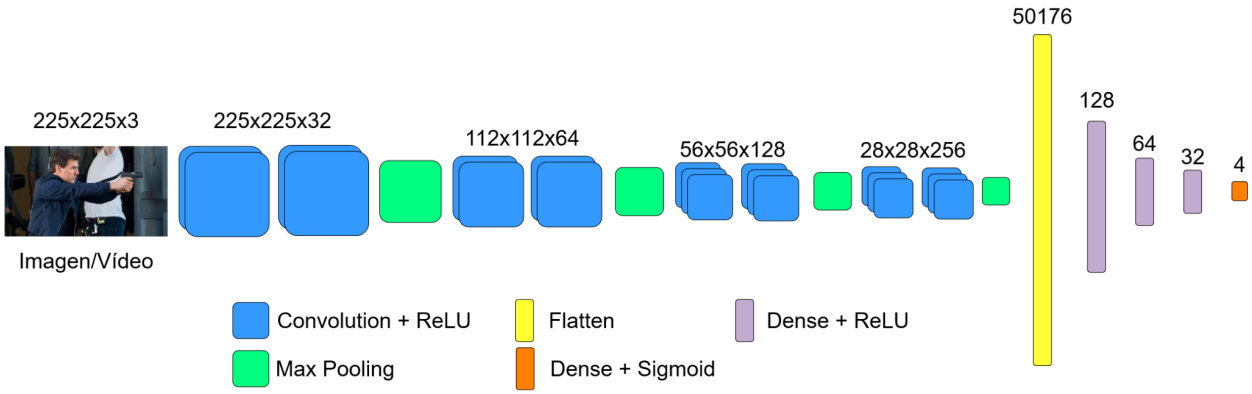


Diagrama de secuencias para la solución

Para la solución se implementará una aplicación web minimalista donde el usuario pueda cargar una imágen o cargar video por medio un dispositivo como una webcam. Esto será procesado por un modelo de reconocimiento entrenado el cual retornará la clasificación del arma con una caja (bounding box) para denotar la ubicación del arma si la encuentra. En la aplicación web se tomará el resultado del modelo para dibujar la caja y mostrar los resultados al usuario.



Diseño preliminar de la red neuronal a emplear

La solución implementada inicia cuando el usuario ingresa la imagen o el video a analizar para localizar el arma en la aplicación web mediante la carga de una imagen o un vídeo, se procede a procesar a través del modelo entrenado que verificará la existencia de armas. Si existen armas, retorna el bounding box del arma a la vista web y finalmente esta muestra el resultado de detección al usuario.

Como se indica en el diseño preliminar de la solución, se comienza ya sea con el ingreso de una imagen o un vídeo en vivo en la capa de entrada con dimensión de 225x225 y 3 canales correspondientes al formato RGB. Es importante notar que los valores de la imágen serán divididos para 255 para que se mantengan entre 0 y 1, ésto ayudará en el entrenamiento del modelo. De manera similar los valores de las 4 coordenadas de las bounding boxes se dividirán para 225 (dimensiones de la imágen) para que se mantengan dentro de 0 y 1.

La arquitectura preliminar para la solución está basada en una arquitectura VGG16, donde se utiliza una red convolucional ordinaria como base para extraer las características de una imágen de alta calidad. Esta arquitectura se diferencia de VGG16 principalmente en el número de capas de convolución utilizadas. Otra diferencia importante que se debe resaltar es el uso de Batch Normalization en cada capa convolucional par de la red antes de la activación ReLU. Además, se incorporó Dropout de 0.3 después de las activaciones ReLU de cada capa convolucional par. Este cambio fue importante porque Batch Normalization sirve para agilizar el proceso de entrenamiento, sobre todo al considerar que los valores de entrada serán números pequeños. Para que Batch Normalization sea beneficioso es recomendable colocarlo antes de la capa de

activación [15]. Por otro lado, se utilizó Dropout para tratar de reducir la posibilidad de que el modelo haga overfitting a los datos de entrenamiento. Otra característica importante que se debe resaltar es el uso de un inicializador de kernel llamado Glorot Normal, que sirvió para evitar quedar atrapado en un mínimo local al inicio del entrenamiento. También se utilizó en cada capa convolucional un regularizador de Kernel L2, el cual reduce los pesos de la red lo cual es beneficioso considerando que se utiliza Batch Normalization.

En la red convolucional base se puede notar que la primera y segunda capa oculta usa 32 filtros de 5x5, cabe recalcar que para todas las capas de convolución de esta arquitectura se ha usado un stride de 1 y la opción de padding 'same' de Keras que sirve para mantener las mismas dimensiones luego de hacer la convolución. La tercera y cuarta capa oculta cuentan con 64 filtros de 3x3, la quinta y sexta tienen 128 filtros de 3x3. Finalmente, la séptima y octava capa de esta red tienen 256 filtros de 3x3; además a cada capa de convolución descrita en esta red le sigue una capa de MaxPool con un tamaño de 2x2 para poder extraer las características importantes de la imágen. Se optó por usar MaxPool en lugar de Average Pooling, ya que es más eficiente y gasta menos recursos computacionales.

Después de las capas de convolución se utiliza Flatten antes de utilizar 3 capas Dense de 128, 64 y 32 neuronas respectivamente. Finalmente, en la capa de salida se utilizaron 4 neuronas con la activación Sigmoid. Estas cuatro salidas representan las coordenadas de la bounding box, se usa Sigmoid en lugar de Softmax ya que no se necesita que la suma de las salidas sea igual a 1. El optimizador que se utilizó para este modelo fue Adam con un aprendizaje de 0.00001.

Implementación

Para el desarrollo de la solución se utilizará el lenguaje de programación Python. Esta elección se debe al potencial del lenguaje con respecto a los demás lenguajes de programación, ya que a pesar de ser un lenguaje nuevo, en comparación al tiempo de creación con respecto a la competencia, es versátil y con una curva baja de aprendizaje, por lo que no sería difícil de aprender. Adicional, tiene una gran comunidad de desarrolladores y diversas librerías que facilitan desarrollar algoritmos para la inteligencia artificial [9].

Para la creación del modelo de la red neuronal, se trabajará con las librerías:

Seaborn: Es una librería basada en Matplotlib, pero orientada a la visualización de gráficos estadísticos, con una interfaz gráfica de alto nivel. La cual nos permite realizar diversos tipos de gráficas elegantes al analizar los datos, resultando en mejores formas de visualizar los datos [10].

Matplotlib: librería dedicada a la representación de datos en gráficos de dos dimensiones de manera sencilla, por lo que es ideal graficar rápidamente los datos [11].

NumPy: librería destinada para el cálculo y análisis de datos con arrays. Los cuales son vectores multidimensionales o matrices del mismo tipo, las cuales manejan grandes cantidades de datos y procesan rápidamente [12].

TensorFlow: librería desarrollada por Google para Deep Learning, mediante diagramas de flujo realiza operaciones matemáticas para codificar gráficos. Este gráfico está constituido por nodos, que son operaciones matemáticas, y aristas que son matrices de datos multidimensionales. Adicional incorpora una API para crear aplicaciones [13].

Keras: es un API de Tensorflow que permite implementar diseños de redes neuronales de una forma entendible para seres humanos, escondiendo la complejidad de este proceso y minimizando la cantidad de código necesario para la implementación [17].

Resultados

Se utilizó un dataset de la Universidad de Granada que contiene 3000 imágenes distintas que contenían un arma de fuego. Cada imagen tenía un archivo XML correspondiente con las coordenadas del arma de fuego. Se utilizó una distribución 70/20/10 para la creación del set de entrenamiento, validación y test.

La mejor iteración del modelo, obtuvo un 0.78 en la métrica de IoU para entrenamiento.
Con validación y test se obtuvo un IoU de 0.7 y 0.69 respectivamente. La pérdida se redujo de manera consistente en el entrenamiento, sin embargo, cerca del final en la muestra de validación el ritmo de reducción empeoró. El cambio en IoU fue más consistente en la muestra de entrenamiento y la muestra de validación.

Conclusiones

El resultado esperado de la red neuronal, es que tenga una precisión alta y que sea confiable a la hora de analizar a las personas y determinar si portan un arma o no, con lo que resultaría satisfactorio el entrenamiento y funcionamiento de la red neuronal. Adicional, reporte al instante a la persona que porte el arma de fuego a las autoridades, permitiendo actuar de manera rápida para reducir el peligro que pueda conllevar a las personas que estén cerca.

Para evaluar el modelo se utilizó la métrica IoU (Intersection over Union) que es particularmente útil para evaluar si la bounding box que predice el modelo es correcta en comparación con la bounding box real. Dicho modelo fue evaluado por medio de la librería estándar de Keras.

Contribuciones

Implementación de una versión modificada del VGG16 con hyperparámetros adaptados al problema de localización de armas.

Referencias

[1] C. Álvarez y G. Jiménez, «Lesiones con armas de fuego: sobrevivir a las balas en Ecuador,» *Estado & comunes,* pp. 119-140, 2019.

[2] C. Valdivieso, «Armas de fuego en Ecuador,» *Perfil Criminológico ,* vol. I, nº 17, pp. 6-7, 2015.

[3] Primicias, «Crímenes con armas de fuego han aumentado un 119% en el país en 2021,» Primicias, 22 Septiembre 2021. [En línea]. Disponible en: https://www.primicias.ec/noticias/sociedad/crimenes-armas-muertes-narcotrafico/. [Último acceso: 16 Junio 2022].

[4] Primicias, «Sube el tráfico de armas en las provincias en conflicto por narcotráfico,» Primicias, 27 Mayo 2022. [En línea]. Disponible en: https://www.primicias.ec/noticias/en-exclusiva/crece-trafico-armas-provincias-narco/. [Último acceso: 16 Junio 2022].

[5] Olmos, R., Tabik, S., & Herrera, F. «Automatic handgun detection alarm in videos using deep learning» Neurocomputing 275, pp. 66-72. 31 Junio 2018. doi.org/10.1016/j.neucom.2017.05.012 [Último acceso: 17 June 2022].

[6] DATA SCIENCE. 2020. R-CNN, R-CNN rápido, R-CNN más rápido, YOLO - Algoritmos de detección de objetos — Visión Artificial — DATA SCIENCE. [online] Available at: https://datascience.eu/es/vision-artificial/r-cnn-r-cnn-rapido-r-cnn-mas-rapido-yolo-algoritmos-de-deteccion-de-objetos/ [Último acceso: 17 June 2022].

[7] "OD-WeaponDetection/Pistol classification at master · ari-dasci/OD-WeaponDetection," *GitHub*. https://github.com/ari-dasci/OD-WeaponDetection [Último acceso: 13 Julio 2022].

[8] "Data Augmentation | How to use Deep Learning when you have Limited Data," *Nanonets AI & Machine Learning Blog*, May 19, 2021. https://nanonets.com/blog/data-augmentation-how-to-use-deep-learning-when-you-have-limited-data-part-2/ [Último acceso: 13 Julio 2022].

[9] "¿En qué se relaciona Python con la inteligencia Artificial?," *InGenio Learning*, [online]. https://ingenio.edu.pe/blog/en-que-se-relaciona-python-con-la-inteligencia-artificial/ [Último acceso: 14 Julio 2022].

[10] "Visualización de datos en Python con Seaborn," *Analytics Lane*, [20 Julio 2018]. https://www.analyticslane.com/2018/07/20/visualizacion-de-datos-con-seaborn/ [Último acceso: 14 Julio 2022].

[11] "Introducción a la Librería Matplotlib de Python – Parte 1 - Aprende IA," *Aprende IA*, [online]. https://aprendeia.com/libreria-pandas-de-matplotlib-tutorial/ [Último acceso: 14 Julio 2022].

[12] "La librería Numpy," *Aprende con Alf*, [12 Mayo 2022]. https://aprendeconalf.es/docencia/python/manual/numpy/ [Último acceso: 14 Julio 2022].

[13] "¿Qué es TensorFlow? ¿Cómo funciona? - Aprende IA," *Aprende IA*, [online]. https://aprendeia.com/que-es-tensorflow-como-funciona/ [Último acceso: 14 Julio 2022].

[14] R. Holbrook, A. Cook “Convolution and ReLU” Disponible en: https://www.kaggle.com/code/ryanholbrook/convolution-and-relu [Último acceso: 14 Julio 2022].

[15] S. Ioffe and C. Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.” arXiv, Mar. 02, 2015. Último acceso: Aug. 23, 2022. [Online]. Disponible: http://arxiv.org/abs/1502.03167

[16] “Data Augmentation For Object Detection.” Paperspace, Aug. 17, 2022. Último acceso: Aug. 23, 2022. [Online]. Disponible: https://github.com/Paperspace/DataAugmentationForObjectDetection

[17] “Keras: the Python deep learning API.” https://keras.io/ (accessed Aug. 04, 2022).