ISSN 1870-4069

Uso de redes neuronales convolucionales en YOLO para la detección de robos armados en establecimientos de ventas

Mario Alberto Muro-Barraza, Aldonso Becerra-Sánchez, Gustavo Zepeda-Valles, Santiago Esparza-Guerrero, Nancy Delgado-Salazar

Universidad Autónoma de Zacatecas, México

sir_mario97@hotmail.com, {a7donso, gzepeda, chago, nancydesal}@uaz.edu.mx

Resumen. El sector comercial es uno de los más afectados por la inseguridad en muchos países, ocasionado principalmente por el robo armado, el cual va creciendo a un ritmo acelerado. El objetivo de este trabajo es diseñar una arquitectura lógica para una herramienta de software capaz de apoyar en la detección de robos cometidos usando armas de fuego al interior de establecimientos de ventas, esto a través del análisis de imágenes. El diseño generado se basa en el empleo de redes neuronales convolucionales, usando en ello el esquema preentrenado "YOLOv4", el cual funciona sobre el "Darknet framework", mismo que utiliza las bibliotecas "OpenCV" para operar. Los resultados obtenidos permitieron la creación de una arquitectura de software detectora de objetos capaz de identificar armas de tres categorías: pistolas, rifles y armas punzo cortantes (tales como cuchillos, cúteres, entre otros), obteniendo un índice de precisión del 75 %; además de lograr una velocidad de 52.63 FPS, la cual es una velocidad en solicitud-respuesta con mejoras a lo existente.

Palabras clave: CNN, NN, visión computacional, detector de objetos, detección de robos armados.

Use of Convolutional Neural Networks in YOLO for the Detection of Armed Robberies in Retail Establishments

Abstract. The commercial sector is one of the most affected by insecurity in many countries, mainly due to armed robbery, which continues to grow at an accelerated pace. The objective of this work is to design a logical architecture for a software tool capable of supporting the detection of robberies committed with firearms inside retail

establishments through image analysis. The proposed design is based on the use of convolutional neural networks, employing the pre-trained "YOLOv4" scheme, which operates on the "Darknet framework" and utilizes the "OpenCV" libraries. The results obtained allowed the development of an object detection software architecture capable of identifying weapons in three categories: handguns, rifles, and sharp weapons (such as knives, box cutters, among others), achieving a precision rate of 75 %. Additionally, a speed of 52.63 FPS was obtained, which represents an improved request-response speed compared to existing solutions.

Keywords: CNN, NN, computer vision, object detector, armed robbery detection.

1. Introducción

La delincuencia evoluciona al mismo ritmo al que lo hace la sociedad [1]. Cada día nuevos métodos son desarrollados por la delincuencia para cometer asaltos, asesinatos o fraudes financieros; y ante este hecho, los gobiernos hacen, aparentemente, todo lo posible para detener y prevenir este tipo de actos delictivos, especialmente aquellos realizados con armas de fuego. No hay región en el mundo que esté exenta de las dramáticas consecuencias generadas por la violencia usando las armas de fuego [2].

Si bien el número de muertos en el contexto de los conflictos armados es bien conocido, menos evidente pero aún más dramático es el hecho de que se pierden más vidas en todo el mundo por eventos con armas de fuego fuera de los conflictos bélicos. The Global Economy [3] recopiló datos de las publicaciones de la Oficina de las Naciones Unidas contra la Droga y el Delito (UNODC), los cuales muestran las tasas y clasificaciones de los países en diferentes aspectos de la seguridad (ver Fig. 1). Se observa que los 20 países con mayor índice de homicidios en el mundo (por cada 100 mil habitantes) están conformados por áreas sudamericanas que no están en guerra.

La baja factibilidad e ineficiencia de los métodos actualmente utilizados para evitar y detener los robos a mano armada provocan pérdidas tanto materiales, económicas y humanas en la mayoría de los establecimientos de ventas, desde los pequeños y medianos, hasta incluso los grandes. Los "mejores" mecanismos de seguridad son demasiado costosos, además de que no son perfectos, principalmente debido al error humano, y los más económicos son aún menos viables por la misma razón [4].

Por otro lado, los últimos avances de tecnología tienen un gran potencial en cuanto a aplicativos nos referimos, y terminarían siendo bastante útiles; lamentablemente, solo se implementan en unos cuantos aspectos de nuestra vida [5]. En pocas palabras, se puede concluir que los ataques con armas de fuego se han vuelto (y seguirán haciéndolo) muy comunes, lo que representa un gran problema social [6]. Todas estas situaciones mencionadas dejan en claro la urgencia de un mecanismo tecnológico novedoso capaz de ayudar en la detección

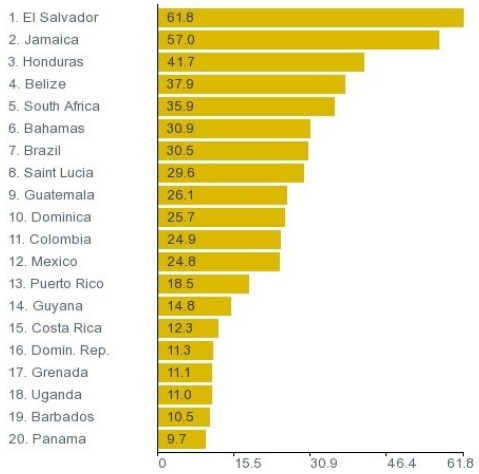

Fig. 1. Países con mayor índice de homicidios (2017).

de estos delitos sin necesidad de invertir una gran cantidad de recurso económico, principalmente orientado para pequeños o medianos establecimientos de ventas.

Las últimas afirmaciones conducen a desarrollar una aplicación de software que utilice algoritmos basados en redes neuronales convolucionales (en lo sucesivo CNN, convolutional neural networks), mediante "YOLOv4" y "Darknet framework" con propósitos de apoyar en la detección de robos a mano armada en establecimientos de ventas, lo anterior a través del análisis de imágenes (pudiendo conseguir igualmente de video). La meta a lograr es apoyar en el proceso de seguridad en establecimientos comerciales, esto mediante una sistema de alerta que notifique cuando se detecta una arma de fuego, avisando al personal de seguridad del lugar, el cual puede tomar acciones al respecto, como llamar al cuerpo policial o actuar como miembro de él. Donde los principales resultados obtenidos (detectando pistolas, rifles y armas punzo cortantes) arrojan un índice de precisión del 75 %; además de lograr una velocidad de 52.63 FPS (frames per second). Es de notar que uno de los puntos importantes a considerar en el diseño es agilizar los tiempos de respuesta, sacrificando un poco las tasas de eficiencia en ello, esto debido a la prontitud de respuesta que un sistema de este tipo debe tener en situaciones de seguridad. Este tipo de sistema podría usarse como una opción de seguridad debido al análisis rápido, la alta eficiencia y el bajo costo que brinda la CNN [7,8]. Asimismo, si se cuenta con personal de vigilancia, el sistema podría ayudar a incrementar la velocidad de detección, ayudando a señalar a través de imágenes que contengan alertas, actos sospechosos que probablemente sean ataques con armas de fuego. Esto ayudaría a reducir el impacto de las pérdidas monetarias y aumentaría la seguridad del personal del establecimiento al denunciar el delito más rápido para poder detenerlo.

El resto del documento está organizado como sigue. En la sección 2 se describen algunos trabajos relacionados relevantes, en la sección 3 se detalla el

esquema propuesto, en la sección 4 se discuten los resultados obtenidos, mientras que en la sección 5 se incluyen las conclusiones y trabajo futuro.

2. Trabajos relacionados

Existen varios trabajos e implementaciones que han sido desarrollados en la línea de la detección de intentos de robo, mientras que otros en la detección del uso de armas como cuchillos y pistolas. Estos desarrollos adoptaron distintos enfoques de la IA como la visión computacional (VC), el deep learning (DL) o lógica difusa. Por ejemplo, Morales et al. [9] crearon un sistema para apoyar en detectar robos violentos efectuados con armas de fuego o cuchillos. Utilizaron un extractor de características (CNN) llamado VGG-16, el cual se encuentra ya pre-entrenado; obteniendo una precisión del 96.69 %. Al mismo tiempo, ellos también desarrollaron un conjunto especializado de videos llamado 'UNI-Crime Dataset' [10], el cual contiene datos de robo y no robo. Este conjunto de datos es un subconjunto del original creado por Sultani, Chen y Shan [11], el cual contiene videos de CCTV (circuito cerrado de televisión, usados para la videovigilancia) de comportamientos normales, robos, peleas y explosiones, entre muchos otros. En este sentido, Vajhala et al. [12] desarrollaron un sistema para detectar personas con cuchillos o pistolas a partir de fotogramas obtenidos de un video de CCTV. Utilizaron el histograma de gradientes orientados (HOG) como vector de características y el algoritmo de propagación inversa de CNN para la clasificación. Sus rendimientos obtenidos ascienden a un índice de precisión del 83.0 %, mientras que en los tiempos de respuesta del sistema no se obtuvieron tiempos de respuesta tan rápido debido a sus dos etapas de procesamiento (verificación e identificación). Para lograr el cometido, se utilizó la base de datos INRAI [13] en el proceso de entrenar la fase de la detección humana.

En contraparte a los trabajos anteriores, Navalgund y Priyadharshini [14] crearon un sistema con la capacidad de detectar cuchillos y pistolas en la mano de una persona apuntando a otra. Ellos compararon el modelo pre-entrenado VGGNET19 con el modelo GoogleNet Inception V3, concluyendo que VGG19 obtuvo mejores resultados en la precisión del entrenamiento en menos tiempo de procesamiento (tasa de precisión del 92 % frente a un 69 % respectivamente). Bajo la misma idea, Romero y Salamea [15] propusieron un sistema basado en VGG Net utilizando imágenes en escala de grises como entradas para múltiples arquitecturas de CNN. En sus resultados obtuvieron un 86 % de precisión y un 86 % de exhaustividad. Mientras que Yahya y Ullah [16] presentaron una arquitectura de red neuronal profunda de tres flujos para la clasificación y localización temporal de eventos de robo en videos de CCTV. Con esto, obtuvieron una tasa de precisión del 75 % y una tasa de precisión de localización temporal del 73.89 %. A diferencia de otros trabajos, este sistema puede localizar el videoclip que contiene el hecho del robo.

Adicionalmente, Grega et al. [17] desarrollaron un algoritmo capaz de detectar a una persona que lleva un arma de fuego al descubierto. El algoritmo se basa en una red neuronal convencional (NN) y un descriptor MPEG-7 para

clasificar transmisiones de video de un CCTV. Los resultados obtenidos se dividen en dos categorías, con una precisión del 95.58 % para la transmisión que contenía un arma, y un 99.32 % para la que no la contenía. En pruebas posteriores obtuvieron una especificidad del 100 % para una transmisión sin arma y del 96.69 % para una que sí la contiene. Algunos años más tarde, Grega et al. [18] propusieron un nuevo enfoque para el mismo problema de este trabajo, que incluía muchas implementaciones arquitectónicas nuevas, aunque sin resultados tan diferentes. Por otra parte, Tiwari y Verma [19] presentaron y desarrollaron un marco que utiliza la segmentación basada en el color para eliminar objetos no relacionados de una imagen utilizando el algoritmo de agrupamiento k-means. Implementaron dos herramientas, la primera se denomina Harris Interest Point, mientras que la segunda es llamada Fast Retina Keypoint (FREAK), las cuales se usaron para la detección de armas, obteniendo una precisión global del sistema del 84.26 %.

3. Esquema propuesto

Para el desarrollo se siguió la metodología ágil conocida como *Prototyping*, la cual se define como un método de desarrollo de sistemas de software en el que se construye, prueba y luego se re-inventa un prototipo según sea necesario [20]. El proceso continúa hasta que finalmente se logra una versión aceptable a partir del cual se puede desarrollar el sistema o producto completo [21]. El enfoque del *Prototyping* sigue un proceso que se repite en cada iteración [22], [23] (ver Fig. 2). Durante la definición de los objetivos del prototipo (primera etapa), se busca seleccionar una de las varias funcionalidades a desarrollar para que, a continuación, se establezca hasta dónde abarcará dicha funcionalidad (segunda etapa), de tal manera que, una vez desarrollada (tercera etapa), exista concordancia entre los resultados y las métricas usadas para su evaluación (última etapa).

3.1. Definición de objetivos

El sistema plantea una meta de apoyo en detección de sucesos ilícitos como asaltos a mano armada, para ello usa imágenes obtenidas de una transmisión de video de cámaras instaladas en establecimientos comerciales o de ventas, de tal manera que puede detectar armas en escenario equiparables a tiempo real. Este proceso implica avisar a un miembro del cuerpo de seguridad del establecimiento sobre sucesos sospechosos, tomando acciones al respecto. Al seguir la metodología de *Prototyping* se plantearon los siguientes puntos para crear un producto que pueda usarse en escenarios reales; además, las métricas aumentarán a medida que mejore el prototipo: i) el sistema debe aceptar imágenes como entrada provenientes de cámaras de establecimientos de ventas o comerciales; ii) el sistema debe detectar a una velocidad considerablemente rápida la presencia de armas en la imagen analizada; iii) el sistema debe mostrar el resultado del análisis realizado, priorizando más la prontitud de respuesta que la eficiencia total; iv)

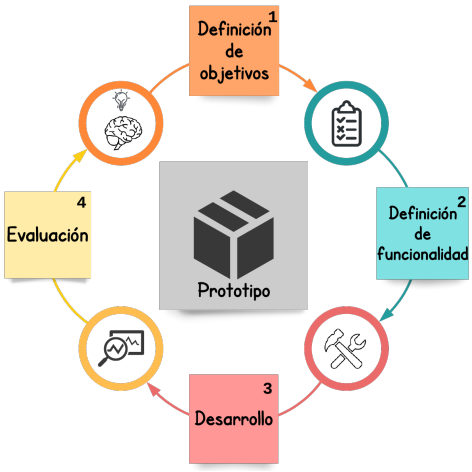

Fig. 2. Ciclo de vida del *Prototyping*.

si se detecta un arma, se debe mostrar una alerta visual de dónde se localiza; y v) el sistema consta con las bases para implementarse en un escenario real.

3.2. Definición de funcionalidad

El sistema está conformado de 3 módulos: i) receptor de imágenes, el cual conecta el prototipo a las imágenes, revisándolas y verificando formatos para pasarlas al módulo de IA; ii) agente de inteligencia artificial, el cual hace la detección, identificando en cada en imagen un objeto de tipo arma como elemento potencial; iii) transmisor de imágenes, el cual muestra la alerta, transmitiendo la cadena de imágenes, incluyendo una alerta visual que localiza el arma.

La distribución física se puede ver en la Fig. 3, donde se muestra la estructura del sistema propuesto. Dicha figura muestra cómo se deben conectar las cámaras al receptor de imágenes, el cual obtendrá la imagen o cadena de imágenes y las pasará en un formato correcto al agente de IA, quien analizará cada imagen para detectar la presencia de armas. Finalmente, el resultado será enviado al monitor de seguridad (para mostrar alertas visuales) y al disco duro (para ser guardado). Los últimos pasos son realizados por el transmisor de imágenes.

3.3. Desarrollo

Cada uno de los módulos de que consta el prototipo (Fig. 3) fue desarrollado para trabajar en un entorno de escritorio. La aplicación se ejecuta en una máquina local, donde las interfaces (tanto del receptor como del transmisor de imagen) se crearon utilizando Python y OpenCV.

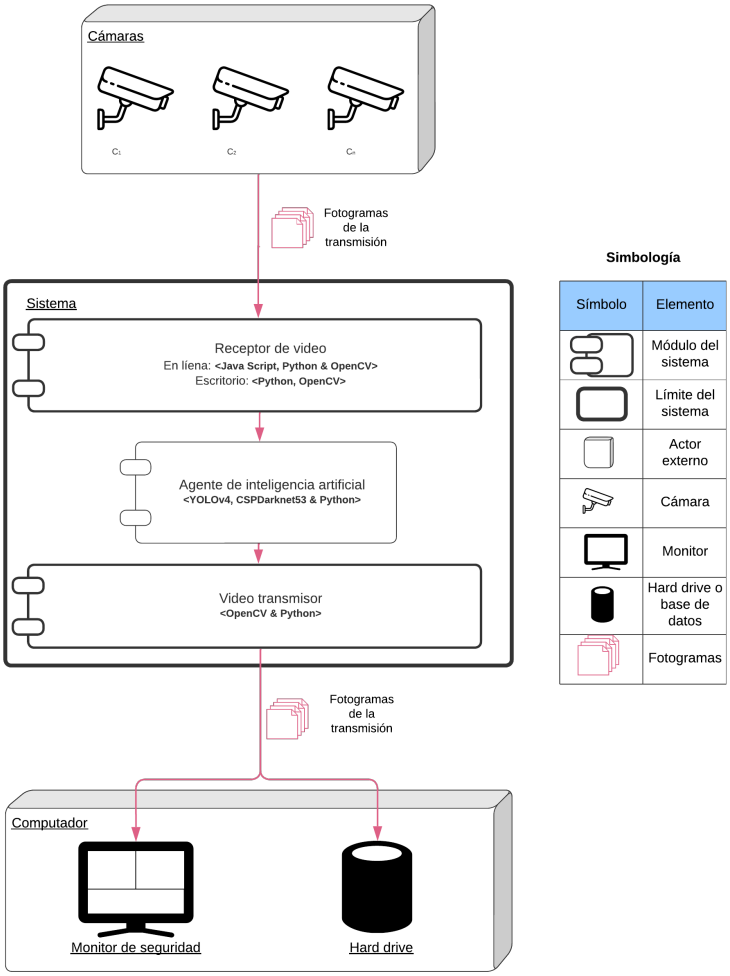

Fig. 3. Distribución de los módulos del prototipo.

Descripción del dataset Todas las imágenes utilizadas se obtuvieron a partir de dos fuentes diferentes, la primera fue de Open Images V6 (OIV6) [24]; este dataset consta de 9 millones de imágenes etiquetadas, además cuentan con sus cuadros delimitadores y máscaras de segmentación de objetos, relaciones visuales y narraciones localizadas. Contiene un total de 16 millones de cuadros delimitadores para 600 clases de objetos. El segundo repositorio fue Kaggle [25], que alberga miles de datasets diferentes, creados por su propia comunidad, en donde cada uno de ellos contiene su propio tipo de etiquetado. Para OIV6 se utilizó un script que recopila automáticamente todas las imágenes que tienen el objeto especificado en el código, siendo los siguientes los seleccionados: pistola, rifle y arma; obteniendo alrededor de 4,000 imágenes.

Fig. 4. Algunos métodos de aumento de datos aplicados en imágenes.

Después de eso se utilizaron funciones de *aumento de datos (DA - data augmentation)*, no solo para incrementar el tamaño del conjunto de datos, sino también la variedad y complejidad (tales como flip, rotation, crop, shear, grayscale, hue, saturation, brightness, exposure, blur, noise, cutout y mosaic) [26]; algunos de ellos se pueden ver en la Fig. 4. DA consiste en crear una copia del dataset original, y después aplicarle múltiples procedimientos de modificación a todos los datos (en este caso imágenes), de tal manera que incrementa el tamaño y variedad del conjunto (ya que ahora el dataset consta de las imágenes originales y las imágenes editadas). Las funciones aplicadas fueron: *'flip'*, que es una función que voltea 180° una imagen. Otra fue *'rotation'*, la cual permite rotar las imágenes en un ángulo personalizado, en este caso de 45°. La función *'crop'* genera una imagen recortada de la original, para este caso se utilizó un 30 %. La función *'shear'* añade una variabilidad en la perspectiva tridimensional de la imagen, que sería como girar la imagen en el eje Y así como en el eje X, usando parámetros como 15 % y 15 % respectivamente. La función *'grayscale'* genera imágenes extras que solo se encuentran en escala de grises, usando en ello un porcentaje del 90 % de escala. La variante de *'hue'* aplica colores distintos a los de la imagen original, al aplicarse se le especificó un parámetro del 51, lo cual genera dos variantes, una positiva del 51° y otra del -51°. La función de *'blur'* genera imágenes con una difusión, al usarse se le especificó 2px. El *'noise'* genera ruido en la imagen, para ello usamos un porcentaje del 5 % [26].

Este procedimiento se realiza también para evitar el sobre-entrenamiento e incrementar la precisión durante la fase de entrenamiento. Una vez que se aplicó el DA, el conjunto de datos incrementó su tamaño a 13,192 imágenes (ver Tablas 1 y 2); se pueden ver en las Fig. 5, 6, 7 y 8 dos ejemplos del dataset al haberse aplicado DA, donde se nota cómo una imagen cambia al aplicarse los procedimientos como la rotación, el voltear, recortar, aumento de ruido entre otras.

Fig. 5. Imagen 1 de ejemplo antes DA.

Fig. 6. Imagen 1 de ejemplo después de DA.

Fig. 7. Imagen 2 de ejemplo antes de DA.

Fig. 8. Imagen 2 de ejemplo después de DA.

Por otro lado, el segundo dataset (Kaggle) fue creado por la comunidad, el cual precisamente usa la estructura y la forma de etiquetado que necesita Darknet y YOLOv4 para administrar las imágenes y sus meta-datos (la clase del objeto en la imagen y sus coordenadas). Una vez que fue procesado usando DA, se obtuvieron 2,158 imágenes. El conjunto de datos descargado de OIV6 se usó

para el entrenamiento y el descargado de Kaggle se usó para la evaluación. Ambos suman un total de 15,350 imágenes, donde el dataset de entrenamiento representa el 85.94 % del total, y el dataset de prueba representa el 14.06 %.
Según se puede apreciar en las Tablas 1 y 2, se le dio más importancia a conseguir un dataset bastante amplio antes de que este estuviera balanceado (en cuanto a las clases contenidas dentro de él). Así pues, la variabilidad de las formas de los objetos, así como la desproporción de cantidades de estos, generó resultados bastante variados por cada una de las versiones del entrenamiento. En la Tabla 1 se aprecia como se busca incrementar el número total de imágenes del dataset (Num). Esto fue con el objetivo de realizar entrenamientos mucho más largos que terminan mejorando el modelo en general. Una vez que se experimentó con la *v_ 3*, se entendió que entre más clases estuvieran contenidas dentro del dataset, menos exacto y preciso sería el modelo de IA. En seguida se procedió a aumentar la cantidad de imágenes en las clases consideradas más importantes, quitando a la vez las de menor relevancia. En este caso, aumentando las clases de las armas, rifles y pistolas; así como de forma contraria, suprimiendo las clases de cuchillos y escopetas.

Una vez descubierto esto esto, se intentó experimentar también con las dimensiones de normalización de las imágenes. Dichas dimensiones, a las cuales YOLOv4 normaliza (Dim_N) para su procesamiento, también impactan en el tiempo de entrenamiento, así como en la precisión que se puede obtener. Ya que al aumentar el tamaño de la imagen, el proceso de convolución termina requiriendo más tiempo y procesamiento, por ende, si aumentamos mucho el tamaño y no damos el tiempo suficiente, termina siendo contraproducente. Por el contrario, al bajar la resolución, el proceso aumenta en velocidad, sin embargo minimiza su precisión debido a que los objetos empiezan a perder forma. En ambas tablas se puede observar que la suma del total de objetos de cada clase (Objs) supera a la cantidad de imágenes, y esto se debe a que varias de las imágenes contienen más de un objeto dentro de ellas.

YOLOv4 YOLO es un modelo detector de objetos para el paradigma CV. De hecho, Bochkovskiy et al. [27] explicaron que el objetivo de YOLOv4 era concebir un sistema detector de objetos de alta velocidad usando una optimización de cálculos paralelos. Es por esto que la estructura base de YOLOv4 es de tipo *detector de una etapa* (*one-stage*). Un modelo de *one-stage* es capaz de detectar objetos sin necesidad de un paso preliminar. Por el contrario, un detector de *dos etapas* utiliza una fase preliminar en donde se detectan regiones de importancia y luego, en la fase final, se analizan dichas regiones en busca del objeto.

Los detectores de una etapa pueden hacer predicciones rápidamente, lo que permite una percepción del proceso en **tiempo real**. Como se puede ver en la Fig. 9, cada parte del modelo es un subsistema que tiene un propósito muy específico. El **backbone** es la parte del detector encargada de obtener las características esenciales de la entrada; normalmente compuesto por una CNN, en este caso CSPDarknet53, la cual está compuesta por 29 capas convolucionales 3×3, un campo receptivo de 725×725 y 27.6 M de parámetros. El papel esencial

Tabla 1. Características de las diferentes versiones del dataset de entrenamiento usado para YOLOv4. Donde: Num es la cantidad total de imágenes del dataset, Objs refiere a la cantidad de objetos que contienen las imágenes (una imagen puede contener una o más armas), Dim_N representa las dimensiones de las imágenes una vez normalizadas, N_A1 representa la cantidad objetos identificados como pistolas, N_A2 es la cantidad de objetos identificados como rifles, N_A3 refiere a la cantidad de objetos etiquetados armas, N_A4 indica la cantidad de objetos marcados como escopetas y N_A5 es la cantidad de objetos marcados como cuchillos.

Versión	**Num**	**Objs**	**Dim_N**	**N_A1**	**N_A2**	**N_A3**	**N_A4**	**N_A5**
v_1	3123	3423	416×416	1737	0	1686	0	0
v_2	1486	1686	512×512	0	0	1686	0	0
v_3	4662	7657	512×512	727	2540	2960	580	850
v_4	4662	7657	512×512	727	2540	2960	0	0
v_5	13192	20964	416×416	2465	8472	10027	0	0
v_6	13192	20964	512×512	2465	8472	10027	0	0

Tabla 2. Características de las diferentes versiones del dataset de pruebas usado para YOLOv4. Donde: Num es la cantidad total de imágenes del dataset, Objs refiere a la cantidad de objetos que contienen las imágenes (una imagen puede contener una o más armas), Dim_N representa las dimensiones de las imágenes una vez normalizadas, N_A1 representa la cantidad objetos identificados como pistolas, N_A2 es la cantidad de objetos identificados como rifles, N_A3 refiere a la cantidad de objetos etiquetados armas, N_A4 indica la cantidad de objetos marcados como escopetas y N_A5 es la cantidad de objetos marcados como cuchillos.

Versión	**Num**	**Objs**	**Dim_N**	**N_A1**	**N_A2**	**N_A3**	**N_A4**	**N_A5**
v_1	241	179	416×416	24	0	155	0	0
v_2	241	179	512×512	24	0	155	0	0
v_3	241	392	512×512	24	110	155	26	77
v_4	188	289	512×512	24	155	0	0	0
v_5	188	289	512×512	24	155	0	0	0
v_6	188	289	512×512	24	155	0	0	0

del **neck** es recopilar mapas de características de diferentes etapas. Por lo general, un neck se compone de varios caminos de abajo hacia arriba y varios caminos de arriba hacia abajo; en este caso se utilizaron SPP (Spatial Pyramid Pooling) y PAN [28] (PaNet o Path Aggregation Network). SPP [29] ayuda a administrar mapas de características de la red troncal y PAN permite una mejor propagación de la información de la capa del backbone de abajo hacia arriba o de arriba hacia abajo. Ambos aumentan la precisión del detector. El rol de **head** (para el caso de un detector de una etapa) es realizar una predicción densa (dense prediction). La **dense prediction** es la predicción final, que se compone de un vector con las coordenadas del cuadro delimitador predicho (centro, alto, ancho),

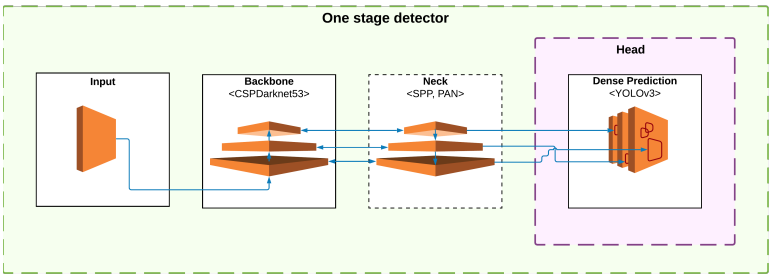

Fig. 9. Estructura de YOLOv4.

la puntuación de confianza de la predicción y la etiqueta; normalmente está conformado por una NN simple, en este caso, la misma usada para YOLOv3 [30].

Receptor de imágenes Este módulo (ver Fig. 3) es el encargado de acceder directamente al directorio donde se encuentran todas las imágenes a analizar; esto usando funciones de Python y OpenCV que automáticamente verifican y decodifican la cadena de estas para finalmente pasarlas al agente de IA.

Agente de inteligencia artificial Los aspectos importantes a mencionar son los parámetros utilizados para el enfoque del proyecto; estos parámetros son las *subdivisiones*, *anchura* - *altura*, *max_ batches* y los *pasos*. Las *subdivisiones* son subgrupos de imágenes tomadas de un conjunto mayor (llamados lotes); los lotes, por otro lado, son un subconjunto de todo el dataset de entrenamiento. Un valor estándar para lotes es de 64 imágenes y, para subdivisiones, se pueden dar una variedad de parámetros completamente válidos; dicho valor dividirá el lote creando los subgrupos de imágenes. Para este prototipo, el valor (para subdivisiones) que dio mejores resultados fue de 32; significa que el lote de 64 imágenes se dividió entre 32, dando como resultado 2 subgrupos de 32 imágenes cada uno. El parámetro *batch* define cuántas imágenes se procesarán por cada iteración de la fase de entrenamiento. Una vez que se procesaron todas las subdivisiones, la iteración finaliza. La *anchura* y la *altura* son parámetros que definen el tamaño que debe tener cada imagen para ser procesada en cada iteración; si la imagen no cumple con eso, debe ser redimensionada. A mayor tamaño, mayor precisión del modelo; por eso la resolución utilizada para la anchura y la altura fue de 512×512 píxeles. El parámetro *max_ batches* indica cuántos lotes se procesarán en toda la fase de entrenamiento; y solo un lote es procesado por una iteración. Esto significa que max_batches indica cuántas iteraciones realizará la fase de entrenamiento. Para este prototipo, el valor de max_batches fue de 6,000. El valor de los *pasos* establece en qué iteración (o número de lote) se cambiará la tasa de aprendizaje; esto se hace multiplicando la tasa de aprendizaje por una escala (esto ayuda a variar la tasa de aprendizaje y obtener mejores resultados). Para el valor de los pasos se recomienda utilizar

el 80 % y el 90 % del total de iteraciones (max_batches), por lo que el valor establecido para la fase de entrenamiento fue de 4,800 y 5,400; eso significa que la tasa de aprendizaje cambiará (se multiplicará por la escala) una vez que las iteraciones alcancen esos números.

Transmisor de imágenes Este módulo (ver Fig. 3) muestra la imagen o imágenes que fueron analizadas por el agente de IA, añadiendo además la alerta sobre el objeto encontrado.

3.4. Evaluación

Los resultados obtenidos (ver Tabla 3) son una señal de un buen progreso. El dataset de prueba incluyó la utilización de 2,158 imágenes, obteniendo resultados interesantes en métricas de precisión, exhaustividad, F1, mAP_{50} (medición de la precisión promedio de identificación) y Average IoU (promedio de la intersección sobre la unión). Estos valores hacen evidente que el modelo no tiene el mejor nivel en cuanto a precisión (cada métrica redondea del 60 % al 70 %); sin embargo, esto no quiere decir que está deficiente, principalmente porque esta investigación y la tecnología utilizada no tienen como objetivo principal alcanzar la mejor precisión máxima, si no el obtener una gran eficiencia en términos de tiempo. Esto se debe a que, si bien los resultados parciales de este trabajo aún no son tan elevados como el estado del arte, muestran la muy alta tasa de análisis de imágenes (19 ms) por cada instancia, lo cual podría proporcionar un sistema que, al ser implementado, proporcione un comportamiento en tiempo real, que de hecho es más útil que un modelo muy preciso pero lento.

Otro gran rasgo acerca de estos hallazgos radica en que están muy cerca de los publicados en la investigación oficial del proyecto YOLOv4 [27], que obtuvo un mAP_{50} de 41.66 % y tiempo por imagen de 19 ms. O incluso más en comparación con el modelo anterior (YOLOv3 [30]), con un mAP_{50} de 55.3 % y un tiempo por imagen de 35 ms (YOLOv4 y YOLOv3 fueron entrenados para el conjunto de datos MS COCO creado por [31]). Aclaremos que la columna de 'Imágenes' se refiere a la cantidad utilizada para este caso, mientras que la columna de 'Tiempo' indica el tiempo utilizado en promedio para analizar cada imagen. Teniendo en cuenta que se utilizó 'Google Colaboratory pro' para la experimentación, el cual según el uso y parámetros aleatorios, puede proveer un equipo conformado por una GPU Tesla V100-SXM2-16GB, 13 GB de RAM, CUDA 11.0, una CPU Xeon Processors @2.3Ghz, todo esto sosteniendo un sistema operativo Linux-5.4.104+-x86_64-with-Ubuntu-18.04-bionic. También puede cambiar la gráfica por una GPU Tesla T4. Además de la variabilidad del equipo, se encontró la limitación en el tiempo de uso, el cual no podía ser mayor de las 24 horas de trabajo continuo.

4. Discusión de resultados

Para obtener los resultados que se muestran en la Tabla 3, se realizaron seis fases de entrenamiento, todas ellas cambiando los parámetros de fine-tuning

Tabla 3. Resultados obtenidos en YOLOv4.

Imágenes	Precisión	Exhaustividad	F1	Average IoU	mAP_{50}	Tiempo (ms)
2158	0.75	0.63	0.68	61.13 %	41.66 %	19

Tabla 4. Parámetros de fine-tuning usados para todas las iteraciones de entrenamiento (versiones desde v1 hasta v6) en YOLOv4.

Batch	Momentum	Learning rate
64	0.949	0.001

Tabla 5. Resultados en las diferentes versiones de entrenamiento usando YOLOv4. Donde A1: pistola, A2: rifle, A3: Arma, A4: escopeta, A5: cuchillo; el símbolo '✓' indica si se utilizó y '×' si no fue utilizado.

Ver	sbdv	A&A	Maxb	Pasos	F1	AvIoU	mAP_{50}	A1	A2	A3	A4	A5	DA
v_1	16	416×416	6,000	4,800-5,400	0.33	26.62 %	49.42 %	✓	×	✓	×	×	×
v_2	16	512×512	10,000	8,000-9,000	0.02	2.06 %	1.52 %	×	×	✓	×	×	×
v_3	16	512×512	10,000	8,000-9,000	0.51	43.50 %	57.57 %	✓	✓	✓	✓	✓	×
v_4	32	512×512	10,000	8,000-9,000	0.48	38.56 %	52.78 %	✓	✓	✓	×	×	×
v_5	32	416×416	6,000	4,800-5,400	0.46	37.08 %	49.40 %	✓	✓	✓	×	×	✓
v_6	32	512×512	6,000	4,800-5,400	0.68	61.13 %	41.66 %	✓	✓	✓	×	×	✓

(quedando como los definitivos los mostrados en la Tabla 4) y variando las clases de armas; además, se buscó la orientación para utilizar y adaptar YOLOv4 [27] como detector de armas de fuego. Las últimas seis versiones previas del agente IA se pueden ver en la Tabla 5, junto con los parámetros del fine-tuning que fueron usados, así como los resultados obtenidos en cada una de ellas. Algunas versiones fueron entrenadas usando más, o incluso menos clases (pistola, rifle, arma, escopeta, cuchillo). El término 'DA' indica si se utilizó data augmentation, el término 'Sbdv' indica las subdivisiones, 'A&A' representa a la anchura y la altura, 'Maxb' refiere al valor usado para max_batches y 'AvIoU' para average IoU.

En la *v_3* se distingue que los resultados podrían verse como los más sobresalientes; pero de hecho, este alto valor se debe a que la clase *cuchillo* obtuvo un mAP_{50} individual de 89.41 %, que incrementa notoriamente el mAP_{50} total del modelo, aún cuando las otras clases no obtuvieron un gran resultado. Tomando esto en cuenta para las siguientes versiones, se establecieron las principales clases de objetos a tener en cuenta; *arma*, *rifle* y *pistola*. Una vez que se desarrolló la *v_5*, se descubrió que se pueden crear mejores modelos a partir del fine-tuning y el uso del DA. La versión *v_6* es la combinación del nuevo dataset usado en la *v_5* y el fine-tuning usado para *v_4*. Es así que la *v_6* se considera como el mejor modelo, dado que las dimensiones de entrada (512×512) permiten detectar objetos en imágenes de gran tamaño, conteniendo más obstrucciones, diferentes formas y posiciones en los objetos, obteniendo mejores resultados en la evaluación.

Los resultados mostrados en la Tabla 3 fueron los obtenidos en la *v_ 6* que se muestra en la Tabla 5. Este modelo obtuvo una precisión con el 75 % de las predicciones de presencia de armas de fuego correctas. Por otro lado, el modelo obtuvo una exhaustividad del 0.63, lo que significa que el agente de IA define correctamente el 63 % de presencia real de armas de fuego. El resultado F1 (0.68) significa que el modelo tiene una eficacia del 68 % en la detección de armas de fuego. El average IoU mostró que solo el 61.13 % de los objetos se superpusieron correctamente dentro del cuadro de predicción. El resultado de mAP_{50} indica que el modelo, para las tres clases, tiene una precisión del 41.66 %. Hay aspectos importantes que se detectaron que pudieron haber decrementado los valores obtenidos para cada métrica, uno de ellos es la integración de la clase 'pistola', pues esta es la que obtiene los peores resultados de las tres, disminuyendo de manera notoria todos los valores de todas las métricas.

Una vez que el entrenamiento del modelo del IA termina, arroja también los datos para armar una matriz de confusión, la cual es utilizada para mostrar cómo es que se está comportando. Si revisamos la Tabla 6 (*v_ 6*) encontramos ciertos datos peculiares. Los *verdaderos positivos* (VP), que aluden a todos los objetos que se clasificaron correctamente, resultan en un total de 1,119; vemos que la clase pistola identificó menos objetos (263 clasificaciones), debido probablemente a la distribución de estos en las diferentes clases, ya que precisamente la pistola es la clase que menos objetos tiene dentro del dataset (ver la Tabla 1). Cosa contraria a lo que pasa en la clase de arma (que identificó 347), siendo la que más objetos tiene dentro del dataset, pero obteniendo un resultado no tan favorable de VP (muy probablemente derivado de la complejidad de las formas de los objetos de esta clase). Lo anterior afecta el desempeño resultante de la *v_ 6*, ya que por un lado la clase pistola obtuvo resultados bajos; mientras que por otro lado, la case de arma consumió gran parte del recurso usado para el entrenamiento, arrojando también resultados bajos. Aún incluso cuando la clase rifle obtiene muy buenos resultados (509 identificaciones), en general el modelo de IA resulta afectado.

Continuando con la matriz, tenemos los *falsos positivos* (FP), que nos indican todas aquellas identificaciones erróneas por clase, que quiere decir que se identificó una clase que no se buscaba. Para el caso del arma, encontramos la mayor cantidad de identificaciones incorrectas, ya que se clasificaron erróneamente 122 objetos como rifle en vez de arma. Por otro lado tenemos a los *falsos negativos* (FN), que refiere a aquellos objetos que estaban presentes pero no fueron identificados, siendo un total de 656. Una curiosidad es la falta de los *verdaderos negativos* (VN), los cuales representan imágenes que no contenían ningún objeto de las tres clases y, de forma correcta, no se identificaron objetos en dichas imágenes. Pero dicha variable no se encuentra en nuestros resultados, ya que el conjunto de imágenes usadas pertenecen al dataset de entrenamiento, en el cual no existen imágenes que no contengan algún objeto. Al no existir imágenes vacías, no puede haber VN.

Cabe mencionar que en el diseño del prototipo se enfatiza la prontitud de respuesta del sistema antepuesto a la eficiencia total, esto debido a que en un

Tabla 6. Matriz de confusión generada de los resultados obtenidos en YOLOv4.

Clase	Pistola	Rifle	Arma
Pistola	263	67	31
Rifle	78	509	49
Arma	36	122	347

sistema de seguridad es de trascendental importancia una notificación inmediata de un suceso sospechoso. De manera adicional, se puede mencionar que existen otras arquitecturas que pueden tomarse como base para experimentación aparte de YOLOv4, como DeepSORT [32] y el prometedor YOLOv5 [33] (usando Pytorch [34]). Los resultados en dichas plataformas fueron muy inferiores a YOLOv4, por lo que aún falta experimentar con ellas para poder aplicarlo en entorno como el del presente trabajo.

5. Conclusiones y trabajo futuro

Este trabajo tuvo como objetivo presentar un esquema de software que permita apoyar en el proceso de detección de armas de fuego en asaltos en establecimientos comerciales, enfatizando en ello siempre una pronta respuesta por sobre una eficiencia total. Los resultados obtenidos demuestran (75 % de precisión con 52.63 FPS de velocidad) que el sistema podría ser utilizado en un entorno real, generando además un comportamiento en tiempo real. La estructura propuesta para todo el sistema permite la mejora para futuras iteraciones, ya que es un sistema modular que se puede adaptar a múltiples entornos.

Se empleó YOLOv4 como agente de IA, esto dado que proporciona una gran velocidad y precisión, atributos que constituyen la definición de alto 'rendimiento', el cual se busca atacar como requerimiento no funcional. Tomando en cuenta la velocidad y la taza de precisión, se hacen evidentes las ventajas que se podrían obtener automatizando las actividades del día a día mediante el uso de detectores de objetos en tiempo real. Otro aspecto importante radica en que los resultados obtenidos (precisión del 75 %, exhaustividad del 63 %, F1 con un 68 %, average IoU del 61.13 %, mAP_{50} del 41.66 %; incluyendo un 52.63 de FPS) son un base sólida para mejoras en el desarrollo de detectores de armas de fuego en tiempo real, coadyuvando en el desarrollo de metodologías precisas de *videovigilancia automatizada*. Es así que los nuevos mecanismos de IA pueden contribuir en garantizar el bienestar y la seguridad de las personas.

Para trabajo futuro, hay dos puntos a mencionar que se plantean como futuras iteraciones de acuerdo a Prototyping. Uno implica la arquitectura de agente de IA para producir una mayor precisión (Darknet framework para YOLOv4), la cual presenta algunas restricciones; donde la principal se basa en que su desarrollo está destinado a ser utilizado en la detección de objetos distintos a los propuestos en este trabajo. Es importante considerar el desarrollo de una nueva CNN en lugar de utilizar algunas de las ya desarrolladas por otros trabajos,

verificando otros parámetros para producir un mejor ajuste (fine-tuning) y crear un mejor dataset que podría mejorar mucho las métricas. El segundo punto involucra el hardware empleado, el cual produce limitaciones, principalmente porque la fase de entrenamiento es muy exigente con la GPU (Graphics Processing Unit). Teniendo en cuenta que se utilizó 'Google Colaboratory pro', se encontró la limitación en el tiempo de uso, el cual no podía cambiarse. En el futuro se podrían utilizar componentes que permitan utilizar toda la capacidad de los mismos sin ninguna restricción de tiempo.

Referencias

1. Herrera-Rodríguez, J., Vega-Zayas, J.M., Rodríguez-González, J.A.: Estrategias de afrontamiento a la criminalidad por microcomerciantes de León [Guanajuato (México)] ¿Indicador de cohesión o falla de la política criminal?. Derecho y Cambio Social (053), 1–18 (2018)
2. United Nations Office on Drugs and Crime: Firearms protocol, https://www.unodc.org/unodc/en/firearms-protocol/index.html. Last accessed 16 Feb 2020
3. The Global Economy: Homicides rate rankings, https://www.theglobaleconomy.com/rankings/homicide_rate. Last accessed 6 Mar 2020
4. Kleck, G., DeLone, M.A.: Victim resistance and offender weapon effects in robbery. Journal of Quantitative Criminology **9**(1), 55–81 (1993)
5. Xu, H.: Application of cloud computing information processing system in network education. In: Abawajy, J.H., Choo, K.K.R., Islam, R., Xu, Z., Atiquzzaman, M. (eds.) International Conference on Applications and Techniques in Cyber Intelligence ATCI 2019, vol 1017, pp. 1809-–1815. Springer International Publishing, Huainan (2020). https://doi.org/10.1007/978-3-030-25128-4_238
6. The Global Economy: Robbery rate rankings, https://www.theglobaleconomy.com/rankings/robery. Last accessed 16 Feb 2020
7. González-Arriaga, D.M., Vargas-Treviño, M.A.D., Guerrero-García, J., López-Gómez, J.: Development of a Platform for Generation of CNN and Multilayer Neural Networks. Computación y Sistemas **26**(1), 172–182 (2022)
8. Alquisiris-Quecha, O., Martinez-Carranza, J.: Depth Estimation Using Optical Flow and CNN for the NAO Robot. Research in Computing Science **148**(11), 49-58 (2019)
9. Morales, G., Salazar-Reque, I., Telles, J., Díaz, D.: Detecting Violent Robberies in CCTV Videos Using Deep Learning. In: MacIntyre, J., Maglogiannis, I., Iliadis, L., Pimenidis, E. (eds.) Artificial Intelligence Applications and Innovations AIAI 2019, IFIP Advances in Information and Communication Technology, vol. 559, pp. 282–291. Springer, Greece (2019). https://doi.org/10.1007/978-3-030-19823-7_23
10. UNI-Crime Dataset, URL: https://didt.inictel-uni.edu.pe/dataset/. Last accessed 30 Jun 2020
11. Sultani, W., Chen, C., Shah, M.: Real-world anomaly detection in surveillance videos. arXiv, 1–10 (2018)
12. Vajhala, R., Maddineni, R., Yeruva, P.R.: Weapon Detection In Surveillance Camera Images. Ph.D. thesis, Blekinge Institute of Technology (2016).

http://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1054902&dswid=-2377
13. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), pp. 886-–893. IEEE, California (2005)
14. Navalgund, U.V., Priyadharshini, P.K.: Crime Intention Detection System Using Deep Learning. In: 2018 International Conference on Circuits and Systems in Digital Enterprise Technology, ICCSDET, pp. 1–6. IEEE, India (2018)
15. Romero, D., Salamea, C.: Convolutional models for the detection of firearms in surveillance videos. Applied Sciences **9**(15), 1–11 (2019)
16. Yahya, Z., Ullah, M.M.: Classification and Temporal Localization of Robbery Events in CCTV Videos through Multi-Stream Deep Networks. In: HONET-ICT 2019 - IEEE 16th International Conference on Smart Cities: Improving Quality of Life using ICT, IoT and AI, pp. 28-–32. IEEE, Charlotte (2019)
17. Grega, M., Lach, S., Sieradzki, R.: Automated recognition of firearms in surveillance video. In: 2013 International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support, CogSIMA, pp. 45-–50. IEEE, California (2013)
18. Grega, M., Matiolański, A., Guzik, P., Leszczuk, M.: Automated detection of firearms and knives in a CCTV image. Sensors **16**(1) (2016)
19. Tiwari, R.K., Verma, G.K.: A Computer Vision based Framework for Visual Gun Detection Using Harris Interest Point Detector. Procedia Computer Science **54**, 703-–712 (2015)
20. Weitzenfeld, A., Guardati, S.: Capítulo 12: Ingeniería de software: el proceso para el desarrollo de software. Libro: Introducción a la Computación. CENGAGE Learning. México (2007)
21. Prototyping model, https://searchcio.techtarget.com/definition/Prototyping-Model. Last accessed 04 Sep 2021
22. Becerra, A., Zepeda, G., Pérez, A., Ramirez-García, U., Esparza, S.: Learning content management software personalized for a university environment. Pistas Educativas **40**(130), 347–362 (2018)
23. Sommerville, I.: Software engineering. 6th edn. Addison-Wesley, UK (2001)
24. Openimages: A public dataset for large-scale multi-label and multi-class image classification, https://storage.googleapis.com/openimages/web/index.html. Last accessed 19 Apr 2021
25. Kaggle datasets, https://www.kaggle.com/datasets. Last accessed 20 Jun 2021
26. Image leve augmentation, https://app.roboflow.com. Last accessed 10 Nov 2020
27. Bochkovskiy, A., Wang, C.Y., Liao, H.Y.M.: Yolov4: optimal speed and accuracy of object detection. arXiv, 1–17 (2020)
28. Liu, S., et al.: Path aggregation network for instance segmentation. arXiv, 1–11 (2018)
29. He, K., Zhang, X., Ren, S., Sun, J.: Spatial pyramid pooling in deep convolutional networks for visual recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence **37**(9), 1904–1916 (2015)
30. Redmon, J., Farhadi, A.: Yolov3: an incremental improvement. arXiv, 1–6 (2018)
31. Lin, T.Y. et al.: Microsoft coco: common objects in context. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) European conference on Computer Vision – ECCV 2014, vol. 8693, pp. 740-–755. Springer International Publishing, Zurich (2014). https://doi.org/10.1007/978-3-319-10602-1_48

32. Wojke, N., Bewley, A., Paulus, D.: Simple online and realtime tracking with a deep association metric. In: 2017 IEEE International Conference on Image Processing (ICIP), pp. 3645-–3649. IEEE, Beijing (2017)
33. AWS: ultralytics/yolov5: v5.0 - YOLOv5-P6 1280 models. (2021) https://doi.org/10.5281/zenodo.4679653.
34. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in pytorch. In: 31st Conference on Neural Information Processing Systems (NIPS 2017), pp. 1–4. Curran Associates, Long Beach (2017)