UNIVERSIDAD CÉSAR VALLEJO

FACULTAD DE INGENIERÍA Y ARQUITECTURA

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

Sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-SPP y la iluminación y la difuminación de OpenCV

TESIS PARA OBTENER EL TÍTULO PROFESIONAL DE:

Ingeniero de Sistemas

AUTOR:

Quito Gonzales, Ernesto Edgar (orcid.org/0000-0002-8579-5943)

ASESOR:

Dr. Alfaro Paredes, Emigdio Antonio (orcid.org/0000-0002-0309-9195)

LÍNEA DE INVESTIGACIÓN:

Sistema de Información y Comunicaciones

LÍNEA DE RESPONSABILIDAD SOCIAL UNIVERSITARIA:

Desarrollo económico, empleo y emprendimiento

LIMA – PERÚ

2023

Dedicatoria

Dedico esta tesis a mi madre Martha Gonzales Crispin y a mi hermano Elmer Villena Gonzales. También, a todas aquellas personas que con su apoyo contribuyeron a finalizar el trabajo de investigación a pesar de las dificultades.

Agradecimiento

Agradezco a la universidad por las oportunidades que me ha brindado y a los docentes y compañeros en general por todo el aporte de conocimiento que me han otorgado.

Índice de contenidos

Índice de tablas

Índice de figuras

Resumen

El problema de la investigación fue ¿Cuál fue el efecto del sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV? El objetivo de la investigación fue determinar el efecto del sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV. El diseño de investigación fue pre-experimental y la metodología ágil utilizada fue Scrum. La muestra por conveniencia estuvo conformada por 2083 imágenes de cuchillos y 1327 imágenes de pistolas.

La sensibilidad del 94.2% fue menor al 100% logrado por Olmos et al. (2017), porque usaron un conjunto de datos guiados por el clasificador VGG-16. La especificad del 89.4% fue menor al 95% logrado por Elsner et al. (2019) porque utilizaron un detector de 2-Pass (2 pasadas) totalmente convolucionada en regiones (R-FCN) con un extractor de características ResNet-101. La precisión del 94.2% de esta investigación fue superior al 44.28% obtenido por Fernandez Carrobles et al. (2019) porque se usó imágenes tratadas con iluminación, difuminación y una capa Spatial Pyramid Pooling (He et al., 2015). La exactitud del 88% fue menor al 97% de Arceda et al. (2016) porque usaron un detector de escenas violentas, un algoritmo de normalización y un detector de rostros.

El tiempo promedio de entrenamiento de 2.07 s se mantuvo dentro de los mejores porque se usó una instancia con Intel(R) Xeon(R) CPU @ 2.30GHz, 12.7 GB RAM y Tesla T4 15 GB GPU similar a Nguyen et al. (2020) con Intel (R) Xeon (R) Gold 6152 CPU @ 2.10 GHz, GPU Tesla P100 con el algoritmo YOLOv3. El tiempo promedio de entrenamiento de 26.19 ms fue rápido porque se utilizó YOLOv3-spp, que aparte de usar Darknet53, adiciona una capa llamada Spatial Pyramid Pooling, similar a Nguyen et al. (2020), quienes usaron YOLOv3 con Darknet53. Se recomienda utilizar más algoritmos de aumento de datos como rotación, acercar y alejar, así como aumentar el conjunto de datos de entrenamiento e interactuar con los hiperparámetros.

Palabras clave: deep learning, YOLOv3, detección de objetos, detección cuchillos, detección pistolas

Abstract

The research problem was “What was the effect of the knife and gun detection system with YOLOv3-spp algorithms and OpenCV lighting and blurring?”. The purpose of the research was to determine the effect of the knife and gun detection system with YOLOv3-spp algorithms and OpenCV lighting and blurring. The research design was pre-experimental and the agile methodology used was Scrum. The convenience sample consisted of 3,342 images of knives and 2,778 images of pistols.

The sensitivity of 94.2% was less than the 100% achieved by Olmos et al. (2017), because they used a data set guided by the VGG-16 classifier. The specificity of 89.4% was less than the 95% achieved by Elsner et al. (2019) because they used a fully region-convolved 2-Pass (R-FCN) detector with a ResNet-101 feature extractor. The precision of 94.2% of this research was higher than the 44.28% obtained by Fernandez Carrobles et al. (2019) because images treated with lighting, blur and a Spatial Pyramid Pooling layer were used (He et al., 2015). The accuracy of 88% was lower than the 97% reported by Arceda et al. (2016) because they used a violent scene detector, a normalization algorithm, and a face detector.

The average training time of 2.07 s remained among the best because an instance with Intel(R) Xeon(R) CPU @ 2.30GHz, 12.7 GB RAM and Tesla T4 15 GB GPU was used similar to Nguyen et al. (2020) with Intel (R) Xeon (R) Gold 6152 CPU @ 2.10 GHz, Tesla P100 GPU with YOLOv3 algorithm. The average training time of 26.19 ms was fast because YOLOv3-spp was used, which apart from using Darknet53, adds a layer called Spatial Pyramid Pooling, similar to Nguyen et al. (2020), who used YOLOv3 with Darknet53. It is recommended to use more data augmentation algorithms such as rotation, zoom in and out, as well as augment the training data set and interact with hyperparameters.

Keywords: deep learning, YOLOv3, object detection, knife detection, gun detection

I. INTRODUCCIÓN

Los homicidios son el nivel más elevado de violencia en una sociedad, en particular, aquella que se ejerce de manera intencional para quitar la vida a otra persona; por ello, la tasa de homicidios se considera, a nivel internacional, como uno de los indicadores más completos y precisos para medir el grado de inseguridad ciudadana (Instituto Nacional de Estadística e Informática [INEI], 2022, p. 15). Para combatir la violencia urbana, las autoridades han instalado cámaras en puntos estratégicos; sin embargo, el personal de seguridad no tiene suficiente tiempo disponible para vigilar la situación. En muchos casos las cámaras de seguridad no están supervisadas y solo se utilizan para obtener pruebas de un acontecimiento violento. La detección de la violencia de los peatones tiene un importante valor de aplicación e importancia para la investigación en muchos campos, como la seguridad de la sociedad, la vigilancia pública y la seguridad personal (Zhonghua Guo et al., 2017).

Ahora, todos los medios o canales digitales muestran acciones de vandalismo en lugares públicos o privados. Por ejemplo, La Oficina de las Naciones Unidas contra la Droga y el Delito (UNODC) en su Estudio Mundial sobre el Homicidio dieron a conocer que en 2017 más de la mitad de los homicidios se llevó a cabo con armas de fuego, mientras que solo una quinta parte involucró objetos afilados. La representación porcentual de armas y cuchillos es de 54% y 22%. En Perú, las muertes violentas fueron realizadas con arma de fuego con un 64.8% mientras que las armas blancas representaron el 13,7% para el año 2020 (INEI, 2022).

En la actualidad, la complejidad para detectar un acto de violencia urbana mediante algoritmos es complejo. Hay diferentes métodos o técnicas que se utilizan para identificar la violencia a partir del video, como golpear algún objeto, patear, pelear y golpear a alguien, pero aun así hay un gran desafío para nosotros para identificar la violencia (Piyush et al., 2020). La seguridad de las escenas de video en una atmósfera energética, particularmente para personas y vehículos motorizados, es solo uno de los temas de estudio de investigación difíciles en la actualidad en la visión de sistemas de computadoras (Mahalingam y Subramoniam, 2019). Existen varios factores en el comportamiento de las personas que dificulta identificar si el objeto a utilizar será usado como un arma

letal. En este trabajo de investigación usaremos algoritmos de detección de objetos para identificar objetos que puedan ser usados para dañar. Algunos de los objetos son: armas de fuego, objetos punzocortantes.

La justificación teórica para realizar el trabajo de investigación fue brindar una alternativa de detección de objetos usando los algoritmos de iluminación y difuminación para enriquecer el conjunto de datos de imágenes de cuchillos y pistolas. Para ello, se utiliza la técnica de aumento de datos en las imágenes. La idea central del aumento de datos es mejorar la suficiencia y diversidad de los datos de entrenamiento mediante la generación de conjuntos de datos sintéticos (Yang et al., 2022, p. 1). El aumento de datos se ha convertido en una parte fundamental de la aplicación exitosa de modelos de aprendizaje profundo de datos de imágenes (Yang et al., 2022).

La justificación tecnológica del sistema que se desarrolló junto con el algoritmo de detección de objetos es aportar una herramienta tecnológica que pueda ser utilizado en la comunidad o ser probado por estudiantes para nuevos proyectos. La detección de objetos es una tecnología para la seguridad pública bastante útil porque el algoritmo de detección de objetos puede rastrear un objetivo particular en el video de vigilancia (Yuan Yue, et al., 2020).

La justificación práctica se basó en resolver el tiempo de respuesta del problema de la violencia urbana usando armas y cuchillos. Poder actuar de manera inmediata brinda una mayor posibilidad de salvaguardar la integridad de las personas. Además de la posibilidad de prevenir y detectar la violencia en la población con respuestas oportunas por parte de las autoridades. Fernandez-Carrobles et al. (2019) señalaron que la video vigilancia puede complementarse con el análisis automático de imágenes empleando visión artificial.

Según la realidad problemática descrita se plantea el problema general y los problemas específicos para la investigación. El problema general de la investigación fue ¿Cuál fue el efecto del sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV? Los problemas específicos fueron los siguientes:

- **PE1:** ¿Cuál fue el efecto del sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV en la sensibilidad?
- **PE2:** ¿Cuál fue el efecto del sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV en la especificidad?
- **PE3:** ¿Cuál fue el efecto del sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV en la precisión?
- **PE4:** ¿Cuál fue el efecto del sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV en la exactitud?
- **PE5:** ¿Cuál fue el efecto del sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV en el tiempo promedio de entrenamiento?
- **PE6:** ¿Cuál fue el efecto del sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV en el tiempo promedio de identificación?

El objetivo general es determinar el efecto de un sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV. Los objetivos específicos fueron los siguientes:

- **OE1:** Determinar el efecto de un sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV en la sensibilidad.
- **OE2:** Determinar el efecto de un sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV en la especificidad.
- **OE3**: Determinar el efecto de un sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV en la precisión.
- **OE4**: Determinar el efecto de un sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV en la exactitud.

- **OE5**: Determinar el efecto de un sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV en el tiempo promedio de entrenamiento.
- **OE6**: Determinar el efecto de un sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV en el tiempo promedio de identificación.

La hipótesis es el sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV tendrá una sensibilidad de por lo menos 90% (Olmos et al., 2017, p. 14; Warsi et al., 2019, p.4), una especificidad de por lo menos 95% (Elsner et al., 2019, p. 64), una precisión de por lo menos 84.21% (Fernandez Carrobles et al., 2019; Olmos et al., 2017, p. 9), una exactitud de por lo menos 90% (Arceda et al., 2016, p. 21; Zhou et al., 2018, p. 10), el tiempo promedio de entrenamiento no mayor a 70.7 s (Nguyen et al., 2020, p. 10) y un tiempo promedio de identificación de por lo menos 22 ms (Zhang et al., 2020, p. 10; Redmon y Farhadi, 2018, p. 1). Las hipótesis específicas fueron las siguientes:

- **HE1:** El sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV tendrá una sensibilidad de por lo menos 90%.

  Olmos et al. (2017) elaboraron un sistema automático de detección de armas cortas en videos. Para evaluar y comparar el rendimiento, uno de los indicadores utilizados fue la sensibilidad. Olmos et al. (2017) lograron conseguir un 100% de sensibilidad usando una base de datos de imágenes de elaboración propia y usando el extractor de características VGG-16.

  Warsi et al. (2019) elaboraron un estudio cuyo objetivo fue detectar visualmente la pistola en videos en tiempo real. Para evaluar los resultados crearon su propio conjunto de datos con pistolas y lo fusionaron con las imágenes del conjunto de datos de ImageNet. La más alta sensibilidad que obtuvieron Warsi et al. (2019) fue 61.94%.

- **HE2:** El sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV tendrá una especificidad de por lo menos 95%.

  Elsner et al. (2019) presentaron un estudio para la detección de armas en datos de imágenes. Search-Net fue entrenado con los conjuntos de datos de "Internet Movie Firearms Database" y con imágenes aleatorias del conjunto de datos "Flickr". El resultado para la especificidad que obtuvieron para Search-Net fue 85% y con la combinación de Confirmation Layer fue 95%. Por ello, esta investigación se consideró un objetivo no menor del 95% de especificidad.

- **HE3:** El sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV tendrá una precisión de por lo menos 84.21%.

  Fernandez Carrobles et al. (2019) presentaron un sistema de detección de armas y cuchillos basados en la metodología Faster R-CNN. El mejor resultado para la precisión, que obtuvieron con la arquitectura SqueezeNet, fue 84.21%.

  Olmos et al. (2017) usaron Faster RCNN con el extractor de características VGG-16 entrenado con imágenes de elaboración propia. Los resultados obtenidos fue 84.21% en la precisión. En base a los estudios realizados, se estableció el objetivo a una precisión no menor al 84.21%.

- **HE4:** El sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV tendrá una exactitud de por lo menos 90%.

  El objetivo de esta investigación se establece en base a los trabajos posteriores. Arceda Machaca et al. (2016) realizaron un trabajo para la

detección de rostros en escenas de violencia. Consiguieron obtener una exactitud del 97% utilizando un algoritmo de super resolución para mejorar la visualización en vídeos con poca claridad.

Zhou et al. (2018) sugirieron un enfoque para detectar secuencias de violencia. Para ello, la detección consta de cinco fases: preprocesamiento de vídeo, segmentación por regiones de movimiento, extracción de características de bajo nivel, procesamiento de características y clasificación/predicción. En conclusión, la exactitud que obtuvieron Zhou et al. (2018) fue 97.03%

- **HE5:** El sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV tuvo un tiempo promedio de entrenamiento menor 70.7 s.

  Se realizó un cuadro comparativo de los algoritmos YOLOv2, YOLOv3, SSD300, SSD512, RetinaNet, Fast RCNN y Faster RCNN en el tiempo promedio de entrenamiento y se obtuvo 5 días, 4 días, 9 días, 12 días, 4-12 horas, 4-12 horas y 4-12 horas (Nguyen et al., 2020, p. 10). Por ello, se ha tomado como referencia que el tiempo promedio de entrenamiento debería ser menor a 70.7 s.

- **HE6:** El sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV tuvo un tiempo promedio de identificación de por lo menos 22 ms.

  Zhang (2020) realizó una comparación de tres algoritmos: Faster RCNN, Faster RCNN-mobile y Faster RCNN-res101 para la mejora de la detección de objetos basada en el aprendizaje profundo para la enfermedad del tomate. En el cuadro comparativo de los tres algoritmos se muestra el tiempo promedio de identificación o tiempo de inferencia de los objetos. El menor tiempo promedio de identificación fue 142 ms usando el algoritmo Faster RCNN-mobile.

Redmon y Farhadi (2018) realizaron una actualización del algoritmo YOLO llamado YOLOv3. YOLOv3 fue comparado con otros algoritmos de detección de objetos mostrando una mejora en el tiempo de detección frente a los demás algoritmos. El resultado obtenido en el tiempo para la detección conseguido fue 22 ms para YOLOv3-320 y de 51 ms para YOLOv3-608.

II. MARCO TEÓRICO

Para fortalecer el tema de investigación, se recopiló articulos indexados de distintas fuentes seguras. Algunos del los articulos se centraron en desarrollar algoritmos y otras buscaron la forma de optimizar los algoritmos exitentes para aumentar la presición y tiempo de detección de objetos. Los algoritmos exitentes, descritos en este capítulo, fueron Faster RCNN, R FCN, YOLOv3, YOLOv3-spp, entre otros. Los articulos mostraron como realizar un mejor tratamiento de las imágenes, para entrenar el algoritmo eficientemente, y los resultados fueron satisfactorios. Por lo tanto, se puede concluir que para una presición alta, influye mucho como se entrena al algoritmo. En las siguientes lineas se da a conocer los antecendetes previos, que se tomaron como referencia para el desarrollo de la investigación.

Redmon y Farhadi (2018) presentaron actualizaciones a su primera versión de YOLO. En la resolución de 320 x 320, YOLOv3 se ejecuta sobre 22 ms a 28.8 mAP, fue tan preciso como el algoritmo SSD pero fue tres veces más rápido. YOLOv3 es un algoritmo muy bueno porque alcanza 57.9 AP50 en 51 ms sobre un Titan X en comparación con 57.5 AP50 en 198 ms por RetinaNet. El rendimiento de RetinaNet es similar a YOLOv3 pero 3.8 veces más rápido. Según los resultados de Redmond y Farhadi (2018) han demostrado que es útil para detecciones en tiempo real.

Yuan et al. (2020) propusieron un novedoso método de rastreo de objetos de escala adaptable y se utiliza ResNet debido a sus ventajas en cuanto a la tasa de éxito, la precisión y la eficiencia. Yuan et al. (2020) señalaron un método que tiene en cuenta tanto los rasgos profundos como la varianza de la escala y logra la mejor puntuación AUC (0.597) que es superior en un 9.5% a la de la HCF (0.502). El porcentaje de superposición de rastreo de nuestro enfoque se incrementa en las secuencias de video con la variación en escala del objetivo. La variación de la escala del objetivo perjudica notablemente a la estimación de la posición, porque el tamaño del área de búsqueda está muy correlacionado con la escala del objetivo. El método de Yuan et al. (2020) tuvo en cuenta tanto los rasgos profundos como la varianza de la escala, logrando la mejor puntuación AUC (0.597), que es superior en un 9.5% a la de la HCF (0.502) (Yuan Yue et al., 2020).

Sun et al. (2019) estudiaron el reconocimiento de la violencia en las imágenes fijas, que puede ser aplicable a la imagen y el monitoreo y filtrado de video. Se construyó una base de datos real más grande y propusieron un método de aprendizaje multivista para el reconocimiento de la violencia visual, que utiliza adecuadamente la información complementaria entre diferentes tipos de características [vistas] (Sun et al., 2019). Se validó la efectividad de MVMED+ con experimentos en múltiples conjuntos de datos. Además, el rendimiento se mejoró aún más cuando se adoptaron características de aprendizaje profundo en lugar de características de imagen artesanal (Sun et al., 2019).

Es interesante extender el modelo actual a la enseñanza semisupervisada para la aplicación del reconocimiento de la violencia visual. Además, sería factible aplicar el modelo actual a la detección de violencia en vídeos, ya que las señales de audio pueden ofrecer otra vista informativa que puede ser muy complementaria a las vistas visuales (Sun et al., 2019, p. 52).

Iqbal et al. (2019) prepararon un método eficaz de detección de armas de fuego integrado en los sistemas de vigilancia y enfocados en algoritmos científicos para notificar la violencia con armas de fuego o generar un acciones oportunas. Esas medidas no solo aumentarán la sensación de seguridad del público, sino que una respuesta oportuna también podría dar lugar a una reducción de los costos médicos y disminuir la carga de la economía. Se recoge de la Internet un amplio conjunto de datos que consiste en una gran variedad de armas de fuego (Iqbal et al., 2019). El conjunto de datos tuvo 10,973 imágenes con diversas características, incluyendo imágenes que capturan múltiples armas de fuego, diversos ambientes, variaciones de la postura de los humanos que llevan armas e imágenes con armas de fuego sin humanos (Iqbal et al., 2019).

El algoritmo OAOD sugerido fue contrastado con los métodos existentes, comprendido por FRCNN, YOLO, YOLOv2, YOLOv3, SSD y DSSD. Fueron entrenados con el conjunto de datos de imágenes de armas de fuego (Iqbal et al., 2019). OAOD ha cumplido mejor en comparación con el FRCNN y las otras redes (Iqbal et al., 2019). El OAOD se ha destacado mejor con elevados niveles de confianza donde otros han mostrado más degradación de desempeño (Iqbal et al., 2019). El OAOD sugerido fue verificado con cinco detectores existentes,

tomando en cuenta YOLOv2, YOLOv3, SSD, DSSD y FRCNN, y cuatro redes variantes sobre varios umbrales de IOU y de confianza (Iqbal et al., 2019). En una variedad de gama de experimentos, el detector sugerido ha revelado tener un mayor rendimiento de detección y localización en la detección de armas de fuego (Iqbal et al., 2019).

Fernandez-Carrobles et al. (2019) desarrollaron dos nuevos detectores de pistolas y cuchillos, usando técnicas de aprendizaje profundo y evaluando su rendimiento. El conjunto de datos estuvo conformada por 3,000 imágenes de armas de diferentes vistas y escenarios, y para incrementar la precisión del detector, se suministró una técnica de aumento de datos a todo el conjunto de datos (Fernandez-Carrobles et al., 2019). Se tomó un enfoque basado en técnicas de aprendizaje profundo y más específicamente a través de la metodología de Faster R-CNN (Fernandez-Carrobles et al., 2019).

Para el problema de la detección de armas, Faster R-CNN entrenado con GoogleNet consiguió un 55.45% de AP50 (AP a IoU=0.50) y Faster R-CNN empleando un SqueezeNet logró un 85.44% de AP50, una diferencia notable sobre GoogleNet (Fernandez-Carrobles et al., 2019). En cuanto a la detección de cuchillos, el Faster R-CNN basado en GoogleNet alcanzó un AP50 del 46.68% y un 19% utilizando SqueezeNet, siendo estos resultados mucho más bajos de lo esperado (Fernandez-Carrobles et al., 2019).

Olmos et al. (2017) indicaron que el objetivo es desarrollar un buen detector de armas en los videos que utilizan las CNN. Se utilizó un modelo de clasificación basado en VGG-16 pre-entrenado con el conjunto de datos de ImageNet (alrededor de 1,28 millones de imágenes sobre 1,000 clases de objetos genéricos) (Simonyan y Zisserman, 2014) y ajustado en el conjunto de datos de 3,000 imágenes de armas tomadas en una variedad de contextos (Olmos et al., 2017).

El modelo detectó las pistolas en 27 videos con un intervalo de tiempo medio AATpI=0,2 segundos, siendo suficiente para un sistema de alarma pero no indentificó las pistolas solo en tres escenas (Olmos et al., 2017). Esto se debe a los mismos motivos especificados anteriormente, como el contraste bajo y la

luminosidad de los cuadros, la pistola se mueve bastante rápido o cuando la pistola no se muestra en primer plano (Olmos et al., 2017).

Como trabajo futuro se debe estimar la reducción del número de falsos positivos, del detector enfocado en Faster R-CNN, mediante el preprocesamiento de los vídeos, es decir, incrementando el contraste y la luminosidad, y también incrementando el conjunto de entrenamiento con pistolas en movimiento (Olmos et al., 2017). También se determinó diferentes clasificadores basados en CNN como, GoogLenet y se consideró un mayor número de clases (Olmos et al., 2017). A pesar de hacer uso de vídeos de baja calidad para la evaluación, el modelo sugerido demostró un rendimiento bueno y es conveniente para los sistemas de alarma de detección automática de pistolas (Olmos et al., 2017).

Los resultados más prometedores se han obtenido con el modelo basado en faster R-CNN, entrenado en nuestra nueva base de datos, que proporciona cero falsos positivos, 100% de recuerdo, un alto número de verdaderos negativos y una buena precisión del 84.21% (Olmos et al., 2017)

Zhou et al. (2018) propusieron desarrollar un método que pueda detectar efectivamente comportamientos violentos tanto en escenas generales como en escenas con mucha gente con cinco fases: preprocesamiento de vídeo, segmentación por regiones de movimiento, extracción de características de bajo nivel, procesamiento de características y clasificación/predicción (Zhou et al., 2018). Durante la fase de preprocesamiento de vídeo, se extrajo cuadros de una larga secuencia de vídeo utilizando una estrategia de muestreo temporal disperso, que se denomina marco de segmento temporal (Wang et al., 2016) (Zhou et al., 2018). En conclusión, el método propuesto mejoró el rendimiento en los tres conjuntos de datos y los resultados experimentales podrían demostrar en la práctica la eficacia del enfoque propuesto tanto para las secuencias de violencia general como para las de violencia de masas (Zhou et al., 2018).

Verma y Dhillon (2017) presentaron un sistema de detección automática de armas de mano que utiliza el aprendizaje profundo, en particular el modelo de la CNN. Se implementó y evaluó el sistema a través de Internet Movie Firearms

Database IMFDB, una base de datos de referencia de armas de fuego (Verma y Dhillon, 2017, p. 1). Las imágenes negativas se recuperaron de Internet aleatoriamente de diferentes categorías como flores, paisajes, animales, etc. Se implementó CNN usando MatConvNet (Vedaldi et al., 2014, p. 2), una caja de herramientas de MATLAB que implementa redes neuronales convolucionales (CNN) de última generación sin unidad de procesamiento gráfico (GPU) para aplicaciones de visión artificial (Verma et al., 2017, p. 1).

El sistema incorporó una arquitectura VGG-16 enfocado en la CNN como extractor de características, continuado con unos clasificadores de última generación implementados en una base de datos de armas estándar. Con un 93% de precisión, se han demostrado los mejores resultados (Verma et al., 2017).

Lejmi et al. (2017) propusieron reconocer las acciones de violencia tomando como primitivas de entrada los descriptores que extraen los puntos de interés detectados. Se utilizaron dos estrategias de fusión, dependiendo de si se aplica antes o después de la clasificación: a) La fusión temprana y b) La fusión tardía (Lejmi et al., 2017). La fusión tardía es más simple, ya que ambas modalidades se valoran de forma individual y solo se tienen en cuenta sus resultados para la fusión (Lejmi et al., 2017). Sobre la base de los resultados experimentales, el descriptor HOG descubrió más características en comparación con HARRIS y SURF, pero SURF fue el más lento pero tiene un buen rendimiento (Lejmi et al., 2017). Se determinó la robustez de cada detector y se apreció que el descriptor SURF es el más relevante y detecta más características (Lejmi et al., 2017).

Fan et al. (2017) buscaron reducir la redundancia de parámetros en diferentes capas en un modelo de detección utilizando diferentes técnicas. El enfoque utilizó núcleos escasos deterministas que permiten entrenar un modelo desde cero y se distingue de trabajos anteriores que dependen de modelos pre-entrenados (Fan et al., 2017). Se utilizaron dos conjuntos de datos de cámaras corporales para evaluar nuestro enfoque propuesto (Fan et al., 2017).

El primer set GoPro fue grabado usando una cámara GoPro Hero4 con una resolución ultra HD de 1920×1080 a 29 FPS, este conjunto de datos incluye tres escenas llenas: un mercado de granjeros bajo techo, una estación de metro y una acera urbana (Fan et al., 2017). El segundo conjunto de Indigo tiene seis clips policiales realistas obtenidos con Indigo Vision HD 1280×720 a 30 FPS, incorporando dos paradas de tráfico, una escena de entrevista, una escena de violencia doméstica y una escena de arresto en un aparcamiento (Fan et al., 2017). El segundo conjunto, compromete a una o dos personas en la vista, detecta muchos objetos complicados como caras/personas con distorción o parcialmente visibles (Fan et al., 2017).

Su enfoque redujo un detector de Faster RCNN basado en VGG16 en un factor de más de 10× mientras que sigue funcionando de manera comparable con el detector de referencia. Además, nuestro detector alcanza una velocidad de cálculo de casi 2× (Fan et al., 2017).

Senst et al. (2017) propusieron un método de detección de vídeo violento, basado en la metodología de Lagrange, centrándose en un rendimiento robusto para datos de videovigilancia desafiantes. Se pueden hallar dos conceptos fundamentales para separar los vídeos violentos: descriptores globales o representaciones de características locales (Senst et al., 2017). Los métodos de Lagrange se aplican regularmente para especificar sistemas dinámicos no lineales que pueden ser explicados por una serie de campos dependientes del tiempo (Senst et al., 2017).

Los métodos de Lagrange cuantifican las propiedades de cada partícula mientras se mueve dentro del campo de flujo y desvelan patrones de movimiento intrínsecos que se rigen por la evolución temporal del sistema (Senst et al., 2017). Para esta situación, adecuaron los conceptos a una serie de campos de flujo óptico, que determinan el móvil de información dentro de la secuencia de imágenes durante el tiempo de duración (Senst et al., 2017). El marco propuesto ha sido ampliamente probado en varios conjuntos de datos difíciles y en datos no públicos del mundo real obtenidos por la Policía Metropolitana de Londres (Senst et al., 2017). Su método evidenció una ventajosa precisión e incrementó los múltiples algoritmos de última generación (Senst et al., 2017).

Arceda et al. (2016) propusieron el desarrollo de un sistema que apoye a las cámaras de vigilancia que controlan algunos eventos criminales (Arceda et al., 2016). Se usó algoritmos de súper resolución para mejorar los resultados de un algoritmo de detección facial para reconocer a las personas en una escena violenta (Arceda et al., 2016). La propuesta se fracciona en tres partes: Un detector de escenas violentas, un algoritmo de normalización y un detector de rostros (Arceda et al., 2016).

El empleo de ViF con Horn-Schunck fue superiormente acogible debido a su mínimo costo computacional y buenos resultados, seguido con el tiempo de procesamiento del descriptor durante 2 segundos. Se reveló que la aplicación de la super-resolución aumenta la resolución de videos, y también produce más píxeles por cuadro, así como un mayor flujo de datos (Arceda et al., 2016). Se calculó el rendimiento de la ViF con Horn-Schunck en un clasificador SVM con un núcleo polinómico y validación cruzada (k=10) (Arceda et al., 2016). Para este experimento se han reunido los conjuntos de datos de SV y Boss (Arceda et al., 2016).

Se sabe que estos conjuntos de datos contienen videos en situaciones reales, con muy mala calidad y por eso obtuvimos una baja precisión, pero con un AUC aceptable (Arceda et al., 2016). El detector de rostros de Lucas Kanade y Tomasi (KLT) no logra detectar ninguna cara al principio, pero después de un algoritmo de súper resolución, la resolución del vídeo se elevó a 700 x 497 píxeles, el detector localizó dos rostros (Arceda et al., 2016).

Zhang et al. (2016) propusieron un marco rápido y robusto para la detección de violencia en las escenas de vigilancia. El método expuesto tiene dos módulos: el módulo de capacitación y el módulo de detección (Zhang et al., 2016). El módulo de capacitación se basó en escoger datos de capacitación representativos y separar el descriptor del OHOF, y luego conseguir el modelo de características utilizando un SVM lineal mientras que el módulo de detección detecta las regiones candidatas a la violencia en la secuencia de vídeo, y determina las regiones precisas de las actividades violentas (Zhang et al., 2016). Finalmente, los resultados, en tres desafiantes conjuntos de datos, han revelado ser superior que el método propuesto, que no solo detecta sino que también

ubica la violencia en videos de vigilancia concurridas y no concurridas (Zhang, Yang et al., 2016).

Ren et al. (2015) indicaron que el trabajo se centra en conseguir un método con el menor costo computacional y una precisión aceptable. Evaluamos nuestro método en los puntos de referencia de detección de COVs de PASCAL (Everingham et al., 2007), donde las RPNs con Fast R-CNNs producen una precisión de detección mejor que la fuerte línea de base de la Búsqueda Selectiva con Fast R-CNNs (Ren et al., 2015). Mientras tanto, el método renuncia a casi todas las cargas computacionales de las SS en tiempo de prueba - el tiempo efectivo de ejecución de las propuestas es de solo 10 milisegundos (Ren et al., 2015).

Simonyan y Zisserman (2014) desarrollaron un método de detección que tiene una velocidad de cuadro de 5fps en una GPU y por lo tanto es un sistema de detección de objetos práctico en términos de velocidad y precisión con 73.2% mAP en PASCAL VOC 2007 y 70.4% mAP en 2012 (Simonyan y Zisserman, 2014). El método permitió que un sistema unificado de detección de objetos basado en el aprendizaje profundo funcione a 5-17 fps (Ren et al., 2015). La RPN aprendida también mejora la calidad de la propuesta de la región y la precisión (Ren et al., 2015).

Serrano et al. (2015) propusieron un método eficiente basado en la detección de las manchas de movimiento. Se calculó la diferencia de imagen absoluta entre cuadros consecutivos, la imagen resultante se binariza, lo que da lugar a una serie de manchas de movimiento (Serrano et al., 2015). Los experimentos muestraron que el método no supera a los mejores métodos considerados. Sin embargo, fue mucho más rápido y, al mismo tiempo, mantiene precisiones útiles que oscilan entre el 70% y cerca del 98%, dependiendo del conjunto de datos (Serrano et al., 2015). En el futuro se procurará mejorar la precisión utilizando características adicionales para detectar las zonas violentas y también se pueden obtener más mejoras en la velocidad explotando el paralelismo en el procesamiento de las manchas (Serrano et al., 2015).

Senst et al. (2015) propusieron una característica local enfocada en el algoritmo SIFT que integra modelos de apariencia y de movimiento basados en

Lagrange. Utilizaron el concepto de medidas de Lagrange para la tarea de detección de video violento siguiendo el marco Lagrangiano (Kuhn et al., 2012) se utilizó el concepto de línea de trayectoria como las líneas de campo integrales en el campo de flujo inestable (Senst et al., 2015) . Los experimentos se llevaron a cabo en los conjuntos de datos de detección de violencia Crowd Violence y Hockey Fight (Senst et al., 2015). En conclusión, se evaluó el algoritmo con un enfoque de bolsa de palabras y se mostró mejoras significativas con respecto al estado actual de los conjuntos de datos de detección de violencia, es decir, Violencia de multitudes, Lucha de Hockey (Senst et al., 2015). Se recomendó que en el futuro se amplíe el campo de aplicación y aplicar el algoritmo LaSIFT al reconocimiento del campo de acción más general (Senst et al., 2015).

En las siguientes páginas se dará a conocer las teorias relacionadas al trabajo de investigación. De los algortimos existentes en la comunidad de desarrolladoreas, se utilizará un algoritmo de detección de objetos basados en redes neuronales convolucionales nombrada como YOLOv3 con la capa Spatial Pyramid Pooling (He et al., 2015, p. 1).

Dai et al. (2016) dieron a conocer una red que consiste en arquitecturas compartidas, totalmente convolutivas como es el caso de FCN (Long et al., 2015). Para integrar la variación de la traducción en FCN, se construye un conjunto de mapas de puntuación sensibles a la posición utilizando un banco de capas convolucionales especializadas como salida de FCN (Dai et al., 2016). Cada uno de estos mapas de puntuación compila la información de posición con respecto a una posición espacial (Dai et al., 2016). En la parte superior de este FCN, añadimos una capa de agrupación de RoI sensible a la posición que recoge la información de estos mapas de puntuación, sin que le sigan las capas de peso (convolucionales/fc) (Dai et al., 2016).

Para la elaboración del trabajo de investigación, se han utilzado librerías y herramientas de programación. Asimismo, se seleccionó la metodologia de desarrollo de software.

Se hará uso de la librería PyTorch para la construcción del algortimo de deep learning. Esta herramienta es un apoyo con relación a acelerar la construcción del algoritmo. Esta basado en el lenguaje de programación python.

La iluminación y difuminación son dos algoritmos que forman parte de la técnica de data augmentation o aumento de datos (Zoph et al., 2019, p. 2). Se agregó esta técnica para la generalización del conjunto de datos (Zoph et al., 2019, p. 2). Los algoritmos de iluminación y difuminación se encuentran dentro de las funciones de OpenCV. OpenCV, Open Source Computer Vision Library, es una biblioteca para la visión por computadora y el aprendizaje automático de código abierto (OpenCV, s.f.). Dentro de todo el catálogo de módulos, el presente trabajo de investigación hace uso del módulo convertScaleAbs y para interacturar con la difuminación se usa GaussianBlur.

Dentro del módulo convertScaleAbs, Scales evalua valores absolutos y transforma el resultado a 8 bits. En cada elemento de la matriz de entrada, la función convertScaleAbs ejecuta tres operaciones secuencialmente: escalado, toma de un valor absoluto, conversión a un tipo de 8 bits sin signo (OpenCV, s.f.). El resultado, según los parámetros, es una imagen con alta iluminación o con baja iluminación.

El módulo para interactuar con la difuminación en OpenCV es GaussianBlur (OpenCV, s.f.), el cual desenfoca una imagen utilizando un filtro gaussiano. La función convoluciona la imagen de origen con el núcleo gaussiano especifico (OpenCV, s.f.). El resultado es una imagen nítida como una imagen desenfocada, según los parámetros.

Existen varias librerías de deep learning en el mercado. Para fines del estudio se nombrará solamente las librerías con mayor uso. Matlab es una herramienta de análisis de datos propietario, por lo que se necesita de una licencia pagada para el uso académico. Matlab se aplica para el aprendizaje automático, tratamiento de señales, tratamiento de imágenes, visión artificial, comunicaciones, finanzas computacionales, diseño de control, robótica, etc. (The MathWorks, 2020).

Pytorch es una librería open-source, el uso es libre sin ningún tipo de licencia. Es un marco de trabajo de computación científica apoyado en Python dirigido a dos audiencias: Un reemplazo para NumPy para usar el poder de las GPU y una plataforma de investigación de aprendizaje profundo que proporciona la máxima flexibilidad y velocidad (PyTorch, 2020).

Tensorflow es otra librería de open-source desarrollador por Google. Es un marco de trabajo de aprendizaje automático que funciona a gran escala y en entornos variados. Además soporta una diversidad de aplicaciones, con un tratamiento en la formación y la inferencia en redes neuronales profundas. (Abadi et al., 2016).

La metodología de analisis de los datos elegida es Knowledge Discovery in Databases o KDD. KDD es un proceso para encontrar conocimiento importante y util de una base de datos (como se citó en Daderman et al., 2018). El KDD es un proceso iterativo que posee muchos pasos (Daderman et al., 2018). KDD posee cinco fases: a) Selección, b) Pre-procesamiento, c) Transformación, d) Data Mining, e) Interpretacion/Evaluación (Daderman et al., 2018).

La metodología de desarrollo de software ágil elegida es Scrum. Este término fue dado por Ikujiro Nonaka e Hirotaka Takeuchi alrededor de 1980 por la coyuntura tecnológica que en esos años empezaba a darse con mayor fuerza (López Gil et al., 2018). La metodología Scrum fue aplicado primero en el desarrollo de software, cuando antes se utilizaba para planificar y gestionar los proyectos los métodos en cascada y secuencial (López Gil et al., 2018).

Scrum es el marco de trabajo de procesos usados para la gestión del trabajo en los productos muy complejos a partir de los años 90, se emplean varios procesos y técnicas, se muestra la eficacia de las técnicas de gestión del producto y las técnicas de trabajo para la mejora continua del producto, equipo y el entorno de trabajo (Schwaber et al., 2017, p. 3).

Scrum tiene cuatro eventos formales o tambíen llamados etapas: a) Planificación del Sprint (Sprint Planning), b) Scrum Diario (Daily Scrum), c) Revisión del Sprint (Sprint Review) d) Retrospectiva del Sprint (Sprint Retrospective) (Schwaber et al., 2017, p. 5). El sprint es un bloque de tiempo menor o igual a un mes durante el cual se va desarrollando el producto “Terminado” con potencial para ser desplegado a producción, cada nuevo sprint se inicia al terminar el sprint anterior (Schwaber et al., 2017, p. 9). Scrum tiene tres tipos de roles: el dueño del producto (Product Owner), el equipo de desarrollo (Development Team) y el Scrum Master (Schwaber et al., 2017, p. 5).

El sistema se va a desarrollar tomando en cuenta la referencia de la metodología ágil scrum, motivo por el cual se omitirá algunos roles para adaptarlo a la situación actual del trabajo de investigación por ser de carácter individual y no contar con un equipo grande de desarrolladores. Como otras metodologías de desarrollo, Scrum define a los documentos como Artefactos. Los artefactos se han establecido para aumentar la transparencia de la información importante y necesaria para asegurar que todos posean el mismo entendimiento (Schwaber et al., 2017, p 15).

La lista de productos (Product Backlog) es una lista ordenada de las cosas que son necesarias en el producto, se enumera todas las características, funcionalidades, requisitos, mejoras y correcciones, nunca está completa y evoluciona a medida que el producto y su entorno donde se utilizará también cambien (Schwaber et al., 2017, p 15). Lista de Productos es dinámica y cambia según la situación del producto. Así mismo este artefacto solo es modificado y ordenado por el Dueño del Producto (Product Owner).

La Lista de Pendientes (Sprint Backlog) es el conjunto de ítems de la Lista de Producto con una planificación para dar el Incremento del producto y llegar al Objetivo del Sprint (Schwaber et al., 2017, p 15). La Lista de Pendientes se emplea para observar el trabajo que el Equipo de Desarrollo señala como necesaria para alcanzar el Objetivo del Sprint. Solo el Equipo de Desarrollo realiza modificaciones a la Lista de Pendientes del Sprint durante el Sprint y cuando se necesite o requiera un nuevo trabajo, el Equipo de Desarrollo lo suma la Lista de Pendientes del Sprint (Schwaber et al., 2017, p. 5).

La elección de la metodología de desarrollo de software se estableció mediante el estudio comparativo de metodologías tradicionales y ágiles para proyectos de Desarrollo de Software (López Gil et al., 2018). El estudio consultado realizó un cuadro comparativo tomando en cuenta los FCE (Factores críticos de éxito), en el que se concluye que las metodologías ágiles son mejores para dirigir proyectos de desarrollo de software obteniendo mayor puntuación en cuatro de las cinco áreas propuestas: a) Procesos, b) Recursos humanos, c) Calidad, d) Tecnología e innovación (López Gil et al., 2018, p. 128).

III. METODOLOGÍA

El presente capitulo aborda las estrategias metodológicas de la investigación. El tipo y diseño de investigación es aplicada cuantitativa. La operacionalización de la variable se resume junto al problema, los objetivos, las hipótesis, las variables y las dimensiones de la investigación. Se define la población como al conjunto de datos de imágenes y vídeos cuyas muestras serán específicamente de cuchillos y pistolas. La técnica e instrumento de recolección de datos se establece sobre una hoja de cálculo con los indicadores de la sensibilidad, la especificidad, la precisión, la exactitud, el tiempo promedio de entrenamiento y el tiempo promedio de identificación. El procedimiento, el método y el análisis de datos se describen secuencialmente. Finalmente, se lista los aspectos éticos que se cumplen en la investigación.

3.1 Tipo y diseño de investigación

Hernández et al. (2014) mencionaron que toda investigación cumple con dos propósitos importantes: a) producir conocimiento y teorías y b) resolver problemas (investigación aplicada). En ese sentido, se estableció que es una investigación de tipo aplicada cuantitativa porque el motivo de esta investigación es resolver un problema social como la violencia con cuchillos y pistolas.

El diseño de investigación es de tipo pre-experimental. La elección del tipo de diseño es porque el estudio tiene una sola variable independiente y se llama así porque el grado de control es mínimo (Hernández et al., 2014, p. 141). El diagrama a utilizar es la siguiente:

G X 0

G: Grupo de sujetos o casos.
X: Tratamiento, estímulo o condición experimental.
0: Una medición de los sujetos de un grupo.
(HERNÁNDEZ, y otros, 2014, p. 141)

3.2 Variables y operacionalización

Para una mejor comprensión de las variables y su operacionalización, por favor revise el anexo 1.

3.3 Población (criterios de selección), muestra, muestreo, unidad de análisis

Para el entrenamiento del algoritmo YOLOv3-spp se usarán imágenes y videos de los tipos cuchillos y pistolas. Los objetos en las imágenes que se van a seleccionar deben estar en diferentes contextos para obtener mejores resultados de detección de objetos.

Para la evaluación de los objetivos planteados, se ha seleccionado a la población de imágenes y vídeos obtenidos de diferentes repositorios en internet, Open Images Dataset V7, Kaggle y fuentes propias.

La muestra es por conveniencia del tipo no probabilístico. Para Hernández et al. (2018) las muestras que son no probabilísticas, la elección de las unidades son independientes de la probabilidad, y dependen de las razones relacionadas con las características y contexto de la investigación.

Para seleccionar la cantidad de muestras de imágenes de cuchillos y pistolas, se revisó estudios anteriores de detección de objetos. Olmos et al. (2017) utilizaron un conjunto de datos de 3,000 imágenes de armas tomadas en una variedad de contextos, mientras que Fan et al. (2017) generaron un total de 1,089 imágenes GoPro y 5,049 imágenes Indigo.

Las muestras seleccionadas para cuchillos y pistolas fueron 2,083 y 1,327 imágenes, respectivamente. Estas muestras estuvieron conformadas por imágenes de cuchillos y pistolas en diferentes contextos. Los contextos fueron: ataques, cortando un objeto, solo el objeto o con una entidad.

3.4 Técnicas e instrumentos de recolección de datos

La técnica para la recolección de datos estuvo basado en la recolección de imágenes de fuentes de Kaggle, Open Images Dataset V7 y fuentes propias. Kaggle es la comunidad de ciencia de datos más grande del mundo con herramientas y recurosos para la ciencia de datos. Dentro de estos recursos están los repositorios de conjunto de datos como las imágenes. En Open Images, tienen un conjunto de datos con más de 9 millones de imágenes que están etiquetadas con cuadros delimitadores, máscaras de segmentación, relaciones visuales y narraciones (Open Images Dataset V7, 2022). Las imágenes de fuentes propias son obtenidas de videos realizados para la investigación. Estos videos son realizados por una cámara web. El instrumento de recolección de datos fue la hoja de tabulación de datos en la que se preciso los campos:

a) **Clase**: Identifica al objeto a evaluar.
b) **VP**: Es la cantidad de elementos que el algoritmo identificó correctamente (Chicco et al., 2021, p. 2).
c) **VN**: Cantidad de elementos negativos que se etiquetan corectamente como negativos (Chicco et al., 2021, p. 2).
d) **FP**: Son los que se pronostican incorrectamente como positivos (Chicco et al., 2021, p. 2).
e) **FN** Son los clasificados erróneamente como negativos (Chicco et al., 2021, p. 2).
f) **Sensibilidad**: Mide el número de instancias correctas recuperadas dividido por todas las instancias correctas (Dalianis, 2018, p. 47). La fórmula es VP / (VP + FN).
g) **Especificidad**: Mide la proporción de negativos que se identifican correctamente como negativos o que no tienen la condición (Dalianis, 2018, p. 48). La fórmula VN / (VN + FP).
h) **Precisión**: Mide el número de instancias correctas recuperadas dividido por todas las instancias recuperadas (Dalianis, 2018, p. 47). La fórmula es VP / (VP + FP).
i) **Exactitud**: Es una medida definida como la proporción de instancias verdaderas recuperadas, tanto positivas como negativas, entre todas las

instancias recuperadas (Dalianis, 2018, p. 48). La fórmula es (VP + VN) / (VP + VN + FP + FN).

j) **Tiempo promedio de entrenamiento (s):** Tiempo total de entrenamiento. La formula es (FechaHoraFin – FechaHora Inicio)/Cantidad Imágenes Entrenadas.

k) **Tiempo promedio de identificación (ms):** Es el tiempo que toma al algoritmo identificar el objeto en una imagen. La fórmula es:

$$\frac{\sum_{i=1}^{n} t_i}{n}$$

El instrumento de recolección de datos se encuentra adjunto en el anexo 11.

3.5 Procedimientos

Paso 1: Cantidad de imágenes

Para obtener los datos se hará uso de las bases de datos de internet. Las imágenes y vídeos se descargan de diferentes repositorios en internet. Entre los repositorios existentes en internet, se eligió Kaggle y también imágenes propias. De la fuente se logra obtener un total de 1,327 imágenes de pistolas y 1,067 cuchillos para el aprendizaje del algoritmo.

Tabla 1: Cantidad de imágenes

	Pistola	**Cuchillo**
Kaggle	1,327	1,067
Propias	0	1,016
TOTAL	**1,327**	**2,083**

Paso 2: Etiquetados de imágenes

Tratamiento de las imágenes para el entrenamiento. En cada una de las imágenes se dibujará un cuadro delimitador del objeto que se detectará. Asimismo, se agregará una etiqueta que indique si la imagen tiene “Cuchillos” o “Pistolas”.

Figura 1: Etiquetado de imagen

Paso 3: División de conjuntos de datos

La cantidad de imágenes será dividida en train/test. La división train es para entrenar el modelo (Van Der Goot, 2021, p. 4485). Finalmente, test se utiliza para confirma la respuesta final a la pregunta de investigación (Van Der Goot, 2021, p. 4485). El indicador de tiempo promedio de entrenamiento será evaluado con el conjunto de datos train y los indicadores de sensibilidad, especificidad, precisión, exactitud y tiempo promedio de identificación se evaluará con el conjunto de datos test.

Paso 4: Recopilación de resultados

Para analizar el resultado de los algoritmos se utilizará una matriz de confusión para obtener la precisión, la especificidad, la sensibilidad y la exactitud.

Tabla 2: Matriz de confusión para cuchillos y pistolas

Predicción	**Cuchillo**	**TP_C/VN_P**	**FP_C/FN_P**	**FP_C/VN_P**
	Pistola	**FN_C/FP_P**	**VN_C/TP_P**	**VN_C/FP_P**
	Background	**FN_C/VN_P**	**VN_C/FN_P**	**VN_C/VN_P**
		Cuchillo	**Pistola**	**Background**
		Actual		

Se está agregando una tercera clase a la matriz de confusión. La clase Background también es considera en otros trabajos de investigación. La etiqueta

de background FN corresponde a casos de coccinélidos que el modelo no identificó en absoluto, mientras que background FP corresponde a predicciones de coccinélidos cuando no hay ningún coccinélido en la imagen de entrada original (Wang et al., 2023, p. 9748). Podemos concluir que las imágenes background o clase de fondo, hace referencia a las imágenes que no tiene cuchillos ni pistolas o que el algoritmo YOLOv3-SPP no ha encontrado que las imágenes tengan una etiqueta valida.

El tiempo promedio de entrenamiento se obtendrá luego de cada fase de entrenamiento. El tiempo promedio de identificación se obtendrá de cada detección y el tiempo que tardó en identificar el objeto. La suma de todos esos tiempos se dividirá por cada imagen utilizada en test obteniendo el promedio del tiempo promedio de identificación.

3.6 Método de análisis de datos

Los datos obtenidos se insertarán en una tabla para ser evaluadas mediante los siguientes indicadores: sensibilidad, especificidad, precisión, exactitud, tiempo promedio de entrenamiento y tiempo promedio de identificación. Luego serán comparados con el resultado de los dos algoritmos propuesto como base.

$$\boldsymbol{Sensibilidad} = \frac{\mathrm{VP}}{(\mathrm{VP} + \mathrm{FN})}$$

Donde:

VP = Verdadero Positivo

FN = Falso Negativo

(Olmos et al., 2017, p. 14; Warsi et al., 2019, p.4)

$$\boldsymbol{Especificidad} = \frac{\mathrm{VN}}{(\mathrm{VN} + \mathrm{FP})}$$

Donde:

VN= Verdadero Negativo

FP = Falso Positivo

(Elsner et al., 2019, p. 64)

$$\boldsymbol{Precisión} = \frac{\mathrm{VP}}{(\mathrm{VP} + \mathrm{FP})}$$

Donde:

VP = Verdadero Positivo

FP= Falso Positivo

(Fernandez Carrobles et al., 2019, p. 7; Olmos et al., 2017, p. 7)

$$\boldsymbol{Exactitud} = \frac{\mathrm{VP} + \mathrm{VN}}{(\mathrm{VP} + \mathrm{VN} + \mathrm{FP} + \mathrm{FN})}$$

Donde:

VP = Verdadero Positivo

VN = Verdadero Negativo

FP = Falso Positivo

FN = Falso Negativo

(Arceda et al., 2016, p. 21; Zhou et al., 2018, p. 10)

$$\boldsymbol{Tiempo\ de\ entrenamiento}\ (\boldsymbol{s}) = \frac{\mathrm{FechaHora}_{final} - \mathrm{FechaHora}_{inicial}}{Cantidad\ Imagenes\ Entrenadas}$$

Donde:

$\mathrm{FechaHora}_{final}$ = Fecha y hora final del entrenamiento.

$\mathrm{FechaHora}_{inicial}$ = Fecha y hora inicial del entrenamiento.

Cantidad Imágenes Entrenadas = Es la cantidad de imágenes que fueron entrenadas

(Nguyen et al., 2020, p. 10)

$$\boldsymbol{Tiempo\ de\ identificación}\ (\boldsymbol{ms}) = \frac{\sum_{i=1}^{n} t_i}{n}$$

Donde:

t = Tiempo de cada identificación del conjunto de datos test.

n = Total de imágenes del conjunto de datos test.

(Zhang et al., 2020, p. 10; Redmon y Farhadi, 2018, p. 1)

3.7 Aspectos éticos

Los hallazgos e innovaciones científicas han permitido que las personas tengan una mejor calidad de vida. Todas las tecnologías se elaboran con el propósito de

ayudar a la humanidad en diferentes aspectos. Esta investigación se inclina por el aspecto social, busca obtener un algoritmo capaz de detectar objetos que causen daño físico a las personas. Los aspectos éticos que se tuvo en cuenta en la investigación son:

- Para realizar el trabajo de investigación se han tomado ideas y conceptos de otros autores, los que fueron debidamente citados y referenciados para preservar la auditoría correspondiente. El principio de respeto de la propiedad intelectual es mantener el respeto por los derechos de la propiedad intelectual de otros investigadores y evitar el plagio (Universidad César Vallejo, 2021, p. 6).
- Las metodologías fueron seleccionadas en base a otros trabajos de investigación. Al igual, que los resultados fueron obtenidos por instrumentos validados por otros autores. Con ello, la investigación se alinea con el principio de transparencia, el cual menciona que la investigación debe ser difundida de tal modo que sea posible replicar la metodología y verificar la validez de los resultados (Universidad César Vallejo, 2021, p. 6).
- La investigación, sistema de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación, cumple con el principio de libertad. Cumple con el principio de libertad principalmente, porque fue desarrollado de manera libre e independiente sin ningún tipo de intereses económicos, políticos o de cualquier otra forma (Universidad César Vallejo, 2021, p. 9).
- En base a los principios fundamentales, en este trabajo de investigación se promueve y defiende la integridad, el honor y la dignidad al contribuir con la seguridad ciudadana (Colegio de Ingenieros del Perú, 2018).

IV. RESULTADOS

A continuación, se presenta los resultados de los indicadores de investigación para los algoritmos YOLOv3-spp y la iluminación y difuminación de OpenCV entrenado con imágenes de cuchillos y pistolas. Los indicadores del estudio son los siguientes: sensibilidad, especificidad, precisión, exactitud, tiempo promedio de entrenamiento y tiempo promedio de identificación.

4.1 Preparación de las muestras

En la Tabla 1 se tiene el conjunto de datos sin imágenes procesadas. Las imágenes de cuchillos se dividirán en conjunto de datos train/test en porcentajes de 89%/11%. Las imágenes de pistolas se dividirán en conjunto de datos de train/test en porcentajes de 88%/12%.

Tabla 3: Conjunto de datos con imágenes sin procesar

Conjunto de datos sin imágenes procesadas				
Clases	**Tipo**	**Cantidad**	**% imágenes**	**Total**
Cuchillos	**Train**	1,847	89%	**2,083**
	Test	236	11%	
Pistolas	**Train**	1,165	88%	**1,327**
	Test	162	12%	
Total		**3,410**		**3,410**

Ahora, se preparó un nuevo conjunto de datos, usando los algoritmos de iluminación y difuminación de OpenCV. Se seleccionó 1,000 imágenes aleatorias de cuchillos y 1000 imágenes aleatorias de pistolas. Primero, se pasó las 1,000 imágenes de cuchillos al algoritmo de iluminación. El algoritmo de iluminación retornó 1,000 nuevas imágenes que tuvieron iluminación alta e iluminación baja.

Luego se pasaron las 1,000 imágenes originales al algoritmo de difuminación, obteniendo 1000 imágenes nuevas que tuvieron imágenes con alta difuminación y baja difuminación. Posteriormente, las 1,000 imágenes de pistolas pasaron por ambos algoritmos, obteniendo 1,000 imágenes nuevas con alta iluminación y baja iluminación y con 1,000 imágenes nuevas con alta

difuminación y baja difuminación. Finalmente, se logró obtener 2,000 imágenes nuevas para cuchillos y 2,000 imágenes nuevas para pistolas.

Tabla 4: Imágenes procesadas con los algoritmos de iluminación y difuminación de OpenCV

T. Procesamiento	Cuchillos	Pistolas
Img. Originales (Entrada)	1,000	1,000
Iluminación (Aumento o reducción aleatorio) (Procesamiento)	1,000	1,000
Difuminación (Aumento o disminución aleatorio) (Procesamiento)	1,000	1,000
Total, resultado iluminación + resultado difuminación.	2,000	2,000
Imágenes Eliminadas	741	549
Total, iluminación + difuminación (etiquetadas)	**1,259**	**1,451**

Las nuevas imágenes procesadas fueron juntadas con el conjunto de datos. La unión de todas las imágenes creó un nuevo conjunto de datos propio y personalizado.

Tabla 5: Conjunto de datos 2 con imágenes procesadas y sin procesar

Conjunto de datos sin imágenes procesadas				
Clases	**Tipo**	**Cantidad**	**% imágenes**	**Total**
Cuchillos	**Train**	2,824	85%	**3,342**
	Test	518	15%	
Pistolas	**Train**	2,531	91%	**2,778**
	Test	247	9%	
Total		**6,120**		**6,120**

En la tabla 5 se muestra las cantidades de imágenes que se unieron con el conjunto de datos anterior y el conjunto de datos con imágenes procesadas con los algoritmos de iluminación y difuminación.

4.2 Entrenamiento del modelo de detección de objetos

Se realizan dos modelos de detección de objetos con dos diferentes conjuntos de datos. Un entrenamiento fue realizado con el conjunto de datos con imágenes

sin procesar de la tabla 3 y el segundo entrenamiento fue realizado con el conjunto de datos con imágenes procesadas y sin procesar de la tabla 5.

Al realizar el primer entrenamiento, el modelo pasa por las pruebas. Para obtener la matriz de confusión se usa la división test. En “test” se tienen 236 imágenes de cuchillos junto con 162 imágenes de pistolas. Una sola imagen puede tener de 1 a más objetos iguales. Ante ello, la cantidad de cuchillos o pistolas puede subir. En la tabla 6 se muestra lo descrito.

Figura 2: Ejemplo de objetos instanciados en una sola imagen

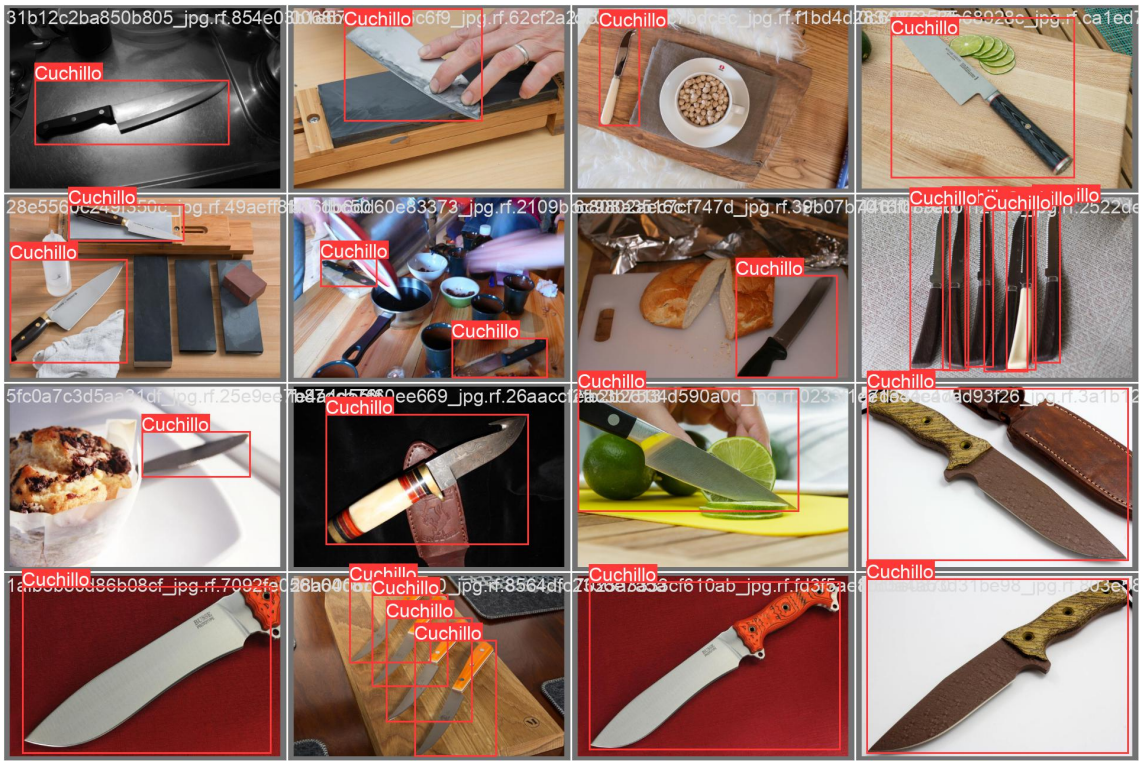

Tabla 6: Cantidad de imágenes de cuchillos y pistolas con cantidad de objetos instanciados de cuchillos y pistolas por imagen del conjunto de datos 1

Clases	Cantidad Imágenes	Cantidad Objetos Instanciados
Cuchillos	236	267
Pistolas	162	177
Total	398	444

Al realizar el segundo entrenamiento, con las imágenes procesadas con los algoritmos de iluminación y difuminación de OpenCV, el modelo pasa por las pruebas. Para obtener la matriz de confusión se usa la división test. En “test” se tienen 518 imágenes de cuchillos junto con 284 imágenes de pistolas. Una sola imagen puede tener de 1 a más objetos iguales. Ante ello, la cantidad de cuchillos o pistolas puede subir. En la tabla 7 se muestra lo descrito.

Tabla 7: Cantidad de imágenes de cuchillos y pistolas con cantidad de objetos instanciados de cuchillos y pistolas por imagen del conjunto de datos 2

Clases	Cantidad Imágenes	Cantidad Objetos Instanciados
Cuchillos	518	656
Pistolas	247	284
Total	765	940

Los resultados se analizaron con el procesamiento de las imágenes de cuchillos y pistolas.

4.3 Prueba de la hipótesis especifica HE1

$HE3_0$: El sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV no incrementó la sensibilidad de por lo menos 90%.

$HE3_1$: El sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV incrementó la sensibilidad de por lo menos 90%.

Tabla 8: Sensibilidad del entrenamiento del conjunto de datos 1

Algoritmo	Clase	VP	VN	FP	FN	Sensibilidad
YOLOv3-spp	Cuchillo	248	194	20	19	**92.9%**
	Pistola	161	286	18	16	**91.0%**

Tabla 9: Sensibilidad del entrenamiento del conjunto de datos 2

Algoritmo	Clase	VP	VN	FP	FN	Sensibilidad
YOLOv3-spp – Iluminación - Difuminación	Cuchillo	573	294	35	83	**87.3%**
	Pistola	273	689	12	11	**96.1%**

La sensibilidad para el primer conjunto de datos es del 92.9% para cuchillos y 91.0% para pistolas. Para el segundo conjunto de datos es del 87.3% para cuchillos y 96.1% para pistolas. Los resultados demuestran que se acepta la hipótesis nula y se rechaza la hipótesis alternativa; por lo tanto, el sistema no incrementó la sensibilidad de por lo menos 90%.

4.4 Prueba de la hipótesis especifica HE2

$HE2_0$: El sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV no incremento la especificidad de por lo menos 95%.

$HE2_1$: El sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV incrementó la especificidad de por lo menos 95%.

Tabla 10: Especificidad del entrenamiento del conjunto de datos 1

Algoritmo	Clase	VP	VN	FP	FN	Especificidad
YOLOv3-spp	Cuchillo	248	194	20	19	**90.7%**
	Pistola	161	286	18	16	**94.1%**

Tabla 11: Especificidad del entrenamiento del conjunto de datos 2

Algoritmo	**Clase**	**VP**	**VN**	**FP**	**FN**	**Especificidad**
YOLOv3-spp – Iluminación - Difuminación	Cuchillo	573	294	35	83	**89.4%**
	Pistola	273	689	12	11	**98,3%**

Los resultados de la especificidad de detección de objeto para cuchillos con el algoritmo YOLOv3-spp muestra un 90.7% para cuchillos y un 94.1% para pistolas. Para el modelo entrenado con YOLOv3-spp con los algoritmos de iluminación y difuminación muestran 89.4% para cuchillos y 98.3% para pistolas. Para este indicador se acepta la hipótesis nula y se rechaza la hipótesis alternativa; por lo tanto, el sistema no incrementó la especificidad de por lo menos 95%.

4.5 Prueba de la hipótesis especifica HE3

$HE1_0$: El sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV no incremento la precisión de por lo menos 84.21%.

$HE1_1$: El sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV incrementó la precisión de por lo menos 84.21%.

Tabla 12: Matriz de confusión del entrenamiento del conjunto de datos 1

Predicción	**Cuchillo**	248	0	20
	Pistola	1	161	17
	Background	18	16	0
		Cuchillo	**Pistola**	**Background**
		Actual		

Tabla 13: Precisión del entrenamiento del conjunto de datos 1

Algoritmo	**Clase**	**VP**	**VN**	**FP**	**FN**	**Precisión**
YOLOv3-spp	Cuchillo	248	194	20	19	**92.5%**
	Pistola	161	286	18	16	**89,9%**

Tabla 14: Matriz de confusión del entrenamiento del conjunto de datos 2

Predicción	**Cuchillo**	573	0	35
	Pistola	2	273	10
	Background	81	11	0
		Cuchillo	**Pistola**	**Background**
		Actual		

Tabla 15: Precisión del entrenamiento del conjunto de datos 2

Algoritmo	**Clase**	**VP**	**VN**	**FP**	**FN**	**Precisión**
YOLOv3-spp – Iluminación - Difuminación	Cuchillo	573	294	35	83	**94.2%**
	Pistola	273	689	12	11	**95,8%**

En base a los resultados anteriores, se comprueba el incremento de la precisión al 94.2% para cuchillos y al 95.8% para pistola; por lo tanto, se rechaza la hipótesis nula y se acepta la hipótesis alternativa con respecto al incremento de la precisión por lo menos al 84.21% en la detección de cuchillos y pistolas.

4.6 Prueba de la hipótesis especifica HE4

$H4_0$: El sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV no incrementó la exactitud de por lo menos 90%.

$H4_1$: El sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV incrementó la exactitud de por lo menos 90%.

Tabla 16: Exactitud del entrenamiento del conjunto de datos 1

Algoritmo	**Clase**	**VP**	**VN**	**FP**	**FN**	**Exactitud**
YOLOv3-spp	Cuchillo	248	194	20	19	**91.9%**
	Pistola	161	286	18	16	**92.9%**

Tabla 17: Exactitud del entrenamiento del conjunto de datos 2

Algoritmo	Clase	VP	VN	FP	FN	Exactitud
YOLOv3-spp – Iluminación - Difuminación	Cuchillo	573	294	35	83	**88.0%**
	Pistola	273	689	12	11	**97.7%**

En la exactitud el primer algoritmo tuvo un 91.9% para cuchillos y 92.9% para pistolas. Para el segundo algoritmo se consiguió un 88.0% para cuchillos y un 97.7% de pistolas. Frente a estos resultados, se acepta la hipótesis nula y se rechaza la hipótesis alternativa; por lo tanto, el sistema no incrementó la sensibilidad de por lo menos 90%.

4.7 Prueba de la hipótesis especifica HE5

$HE5_0$: El sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV no tuvo un tiempo promedio de entrenamiento menor o igual a 70.7 s.

$HE5_1$: El sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV tuvo un tiempo promedio de entrenamiento de por lo menos 70.7 s.

Tabla 18: Tiempo promedio de entrenamiento del conjunto de datos 1 con el conjunto de datos 2

Algoritmo	Img	Batch	Epoch	Device	Tiempo promedio de entrenamiento (s)
YOLOv3-spp	640	16	35	GPU	3.59
YOLOv3-spp – Iluminación - Difuminación	640	16	35	GPU	2.07

En la tabla 18 se puede observar la diferencia entre el tiempo promedio de entrenamiento del conjunto de datos 1 y el conjunto de datos 2. El resultado obtenido para el primer entrenamiento es de 3.59 s, mientras que para el

segundo entrenamiento fue 2.07 s. Los parámetros de entrada para ambos entrenamientos son iguales. En base a estos tiempos de entrenamiento se rechaza la hipótesis nula y se acepta la hipótesis alternativa; por lo tanto, el sistema tuvo un tiempo promedio de entrenamiento menor o igual a 70.7 s.

4.8 Prueba de la hipótesis especifica HE6

$HE6_0$: El sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV no tuvo un tiempo promedio de identificación de por lo menos 22 ms.

$HE6_1$: El sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV tuvo un tiempo promedio de identificación de por lo menos 22 ms.

Tabla 19: Tiempo promedio de identificación del conjunto de datos 1 con el conjunto de datos 2

Algoritmo	Clases	Tiempo promedio de identificación (ms)
YOLOv3-spp	Cuchillos	26.77
	Pistolas	24.14
YOLOv3-spp – Iluminación - Difuminación	Cuchillos	25.35
	Pistolas	26.19

El tiempo promedio de identificación es el promedio del tiempo de 236 imágenes de cuchillos y 162 imágenes de pistolas para el primer conjunto de datos de la división test. El tiempo promedio de identificación es el tiempo de 518 imágenes de cuchillos y 247 imágenes de pistolas para el segundo conjunto de datos de la división test.

El tiempo promedio de identificación se mantiene sobre los 22 ms de los mejores algoritmos de identificación. Ante esto, se rechaza la hipótesis nula y se acepta la hipótesis alternativa; por lo tanto, el sistema tuvo un tiempo promedio de identificación de por lo menos 22 ms.

4.9 Hipótesis general

HG_0: El sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV no tendrá una sensibilidad de por lo menos 90, una especificidad de por lo menos 95%, una precisión de por lo menos 84.21%, una exactitud de por lo menos 90%, el tiempo promedio de entrenamiento no mayor a 70.7 s y un tiempo promedio de identificación de por lo menos 22 ms.

HG_1: El sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV tendrá una sensibilidad de por lo menos 90, una especificidad de por lo menos 95%, una precisión de por lo menos 84.21%, una exactitud de por lo menos 90%, el tiempo promedio de entrenamiento no mayor a 70.7 s y un tiempo promedio de identificación de por lo menos 22 ms.

Posterior a la revisión de las hipótesis especificas H1, H2, H3, H4, H5 y H6 se puede concluir que solo fue aceptado las hipótesis H3, H5 y H6; por ello, no se aceptó la hipótesis general.

4.10 Resumen

Después de detallar el resultado de las hipótesis, ahora se resume todos los resultados en la tabla 20. La columna “Hipótesis” tiene las hipótesis específicas y la hipótesis general. En la columna “Aceptada” se indica si la hipótesis se acepta o no.

Tabla 20: Tabla de resumen de las hipótesis

Cod.	Hipótesis	Aceptada (Sí/No)
H1	El sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV tuvo una sensibilidad de por lo menos 90%.	No
H2	El sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV tuvo una especificidad de por lo menos 95%.	No
H3	El sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV tuvo una precisión de por lo menos 84,21%.	Sí
H4	El sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV tuvo una exactitud de por lo menos 90%.	No
H5	El sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV tuvo un tiempo promedio de entrenamiento menor o igual 70.7 s.	Sí
H6	El sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV tuvo un tiempo promedio de identificación de por lo menos 22 ms.	Sí
HG	El sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV tendrá una sensibilidad de por lo menos 90%, una especificidad de por lo menos 95%, una precisión de por lo menos 84.21%, una exactitud de por lo menos 90%, el tiempo promedio de entrenamiento no mayor a 70.7 s y un tiempo promedio de identificación de por lo menos 22 ms.	No

V. DISCUSIÓN

Los resultados de esta investigación para los indicadores de detección de cuchillos, tales como sensibilidad, especificidad, precisión, exactitud, tiempo promedio de entrenamiento y tiempo promedio de identificación fueron 87.3%, del 89.4%, 94.2%, 88.0%, 2.07 s y 25.35, respectivamente. Para pistolas, la sensibilidad, la especificidad, la precisión, la exactitud, el tiempo promedio de entrenamiento y el tiempo promedio de identificación fueron 96.1%, 98.3%, 95.8%, 97.7%, 2.07 s y 26.19 ms, respectivamente.

Los indicadores demostraron un aumento en la precisión para cuchillos y pistolas. El tiempo promedio de entrenamiento se mantuvo por debajo de los 70.7 s y el tiempo promedio de identificación se mantuvo entre el tiempo de los algoritmos más rápidos. Sin embargo, para los indicadores como la especificidad, sensibilidad y exactitud en la detección de los cuchillos se mantuvo por debajo de los objetivos. En las próximas líneas se realizará la discusión para cada indicador.

La sensibilidad lograda en esta investigación fue 87.3% para cuchillos y 96.1% para pistolas; sin embargo, Olmos et al. (2017) obtuvieron un 100% de sensibilidad, la que fue mayor a los resultados de esta investigación; dado que, utilizaron un modelo de clasificación basado en VGG-16 pre-entrenado con un conjunto de datos de ImageNet (Alrededor de 1,28 millones de imágenes sobre 1,000 clases de objetos genéricos y ajustado en el conjunto de datos de 3,000 imágenes de armas. Posteriormente, midieron la sensibilidad usando Faster RCNN con el backbone VGG-16.

La sensibilidad que obtuvieron Warsi et al. (2019) fue 75%. Sin embargo, la sensibilidad de esta investigación fue mayor debido al uso del algoritmo YOLOv3-spp y un conjunto de datos de entrenamiento con 2,824 imágenes de cuchillos y 2,531 imágenes de pistolas. Además, YOLOv3-spp combina Darknet-53 con una capa final llamada Spatial Pyramid Pooling (He et al., 2015). La capa Spatial Pyramid Pooling genera una representación de la longitud fija independientemente del tamaño/escala de la imagen. La agrupación piramidal es resistente a las deformaciones de los objetos (He et al., 2015). En cambio, Warsi et al. (2019) usaron YOLOv3 que usa solamente Darknet-53 como su

backbone principal y también usaron un conjunto de datos propio con imágenes de pistolas.

La especificidad para la detección de cuchillos fue 89.4%, siendo menor al 95% obtenido por Elsner et al. (2019) porque utilizaron un detector de 2-Pass (2 pasadas) que consta de dos módulos. Primero, la red de búsqueda que son R-CFN con un extractor de características ResNet-101 y una segunda red llamada Confirmation Layer que utilizaron para revisar la salida de la primera red. Además, R-FCN ResNet-101 fue entrenada con 1162 pistolas y 387 rifles de asalto.

La especificidad de esta investigación para la detección de pistolas fue 98.3%, la que fue mayor al 95% obtenido por Elsner et al. (2019) para la detección de pistolas y rifles de asalto, gracias a que los algoritmos de iluminación y difuminación en las imágenes de pistolas de esta investigación aumentaron la variedad de características en diferentes contextos que fueron recopilados al momento del entrenamiento de YOLOv3-spp. Además, Spatial Pyramid Pooling [la capa final] (He et al., 2015) permitió evitar la pérdida de estas nuevas características en las imágenes de pistolas.

Las precisiones para cuchillos y pistolas fueron 94.2% y 95.8%, respectivamente. Las precisiones para esta investigación fueron mayores a las precisiones obtenidas por Fernandez Carrobles et al. (2019), quienes obtuvieron 85.45% para pistolas y 46.68% para cuchillos. El aumento en la precisión de esta investigación se debió a que se consideró un 44.28% de imágenes nuevas que fueron procesadas por los algoritmos de iluminación y difuminación y entrenadas con YOLOv3-spp que usa una capa final para extraer características de objetos pequeños, Spatial Pyramid Pooling (He et al., 2015). La capa Spatial Pyramid Pooling evitó la perdida de características de las imágenes de cuchillos y pistolas porque utilizó una agrupación piramidal que es resistente a las deformaciones de los objetos (He et al., 2015).

La exactitud de los algoritmos YOLOv3-spp con la iluminación y difuminación fue 88.0% para cuchillos; pero, Arceda et al. (2016) obtuvieron una mayor exactitud del 97% para la detección de rostros. La exactitud del estudio de Arceda et al. (2016) fue mayor porque su propuesta fue combinar tres algoritmos en tres etapas secuenciales: (1) un detector de escenas violentas, (2) un algoritmo de normalización y (3) un detector de rostro. El algoritmo de super-resolución que estuvo en la etapa de normalización ayudó a que los videos con poca resolución sean optimizados y aumenten la cantidad de píxeles para conseguir mejores resultados.

La exactitud para la detección de pistolas de esta investigación fue 97.7%, siendo mayor al 94.31% obtenido por Zhou et al. (2017) porque se realizaron cambios en la iluminación de una imagen, agregando un poco más de brillo o restándole iluminación. Similarmente, el algoritmo de difuminación permitió aumentar la claridad de la imagen o hacer que se difumine levemente para aumentar el contexto del entorno donde pueda detectarse el objeto en la vida real. Además, la obtención de estas características nuevas de los objetos ocurrió gracias a que YOLOv3-spp añadió una última capa denominada Spatial Pyramid Pooling (He et al., 2015).

El tiempo promedio de entrenamiento para cuchillos y pistolas fue 2.07 s siendo menor a los 70.7 s obtenidos por Nguyen et al. (2020) con el algoritmo YOLOv3 porque se usaron especificaciones de hardware de equipo de cómputo más potentes. Nguyen et al. (2020) realizaron el entrenamiento con un Intel (R) Xeon (R) Gold 6152 CPU @ 2.10 GHz, GPU Tesla P100. Para el entrenamiento de YOLOv3-spp con la iluminación y difuminación de esta investigación se utilizó un Intel(R) Xeon(R) CPU @ 2.30GHz, 12.7 GB RAM y Tesla T4 16 GB GPU implementado en la nube con Google Colab. Tanto en esta investigación como en el estudio de Nguyen et al. (2020) se usó el algoritmo YOLOv3.

Los tiempos promedio de identificación para cuchillos y pistolas de esta investigación fueron 25.35 ms y 26.19 ms, respectivamente, los que fueron menores a los 142 ms obtenidos por Zhang et al. (2020) porque YOLOv3 utiliza una única red neuronal convolucional a diferencia de Faster RCNN que utiliza un

enfoque basado en regiones con una combinación de dos redes neuronales. Por ello, YOLOv3 es usado para detecciones en tiempo real. Además, el tiempo promedio de identificación para esta investigación fue menor porque se usó un AMD Ryzen 5 5600H 3.30 GHz, 16 GB RAM y NVIDIA GeForce RTX 3060 6GB, mientras que Zhang et al. (2020) usaron un ordenador con una NVIDIA GeForce GTX 1060 GPU con 12 GB.

VI. CONCLUSIONES

A continuación, se muestra las conclusiones:

1. Se puede obtener un mayor porcentaje de sensibilidad utilizando un modelo de clasificación basado en VGG-16 pre-entrenado con un conjunto de datos de ImageNet con alrededor de 1.28 millones de imágenes sobre 1,000 clases de objetos genéricos y ajustado en el conjunto de datos de 3,000 imágenes de armas con Faster RCNN con el backbone VGG-16. Puesto que, un modelo pre-entrenado tiene las características generalizadas de los objetos que se van a detectar. Además, ahorra tiempo y mejora el rendimiento con menos ejemplos de entrenamiento. A esta forma de reutilizar modelos pre-entrenados se le conoce como transferencia de conocimiento.

2. La sensibilidad en la detección de cuchillos y pistolas es buena usando el algoritmo YOLOv3-spp y un conjunto de datos de entrenamiento con 2,824 imágenes de cuchillos y 2,531 imágenes de pistolas, ya que, combina Darknet-53 con una capa final llamada Spatial Pyramid Pooling (He et al., 2015). La capa Spatial Pyramid Pooling genera una representación de la longitud fija independientemente del tamaño/escala de la imagen. La agrupación piramidal es resistente a las deformaciones de los objetos (He et al., 2015).

3. La especificidad en la detección de cuchillos aumenta si se utiliza un detector de 2-Pass (2 pasadas) debido a que la detección se divide en dos pasos o módulos. Primero, el objeto pasa por la red de búsqueda R-CFN con el extractor de características llamada ResNet-101 y luego el resultado se transfiere una segunda red llamada Confirmation Layer que sirve para revisar la salida de la primera red. El detector de 2-Pass (2 pasadas) asegura que el resultado sea el objeto que se quiere detectar.

4. Para mejorar el resultado de la especificidad en la detección de pistolas, los algoritmos de iluminación y difuminación en las imágenes de pistolas brindaron una variedad de contextos ricos en características que fueron recopilados al momento del entrenamiento de YOLOv3-spp. Además, la

capa final, Spatial Pyramid Pooling (He et al., 2015) permitió evitar la pérdida de estas nuevas características en las imágenes de pistolas.

5. YOLOv3-spp con los algoritmos de iluminación y difuminación de OpenCV lograron una precisión muy alta para la detección de cuchillos y pistolas, ya que se consideró un 44.28% de imágenes nuevas que fueron procesadas por los algoritmos de iluminación y difuminación y entrenadas con YOLOv3-spp que usa una capa final para extraer características de objetos pequeños, Spatial Pyramid Pooling (He et al., 2015). La capa Spatial Pyramid Pooling evitó la perdida de características de las imágenes de cuchillos y pistolas porque utiliza una agrupación piramidal que es resistente a las deformaciones de los objetos (He et al., 2015).

6. Para obtener un porcentaje de exactitud superior al 97% se combinó más de un algoritmo en diferentes fases o etapas. Esto sucede debido a que Arceda et al. (2016) obtuvieron una exactitud de 97% al combinar tres algoritmos en tres etapas secuenciales. La primera etapa incorpora un detector de escenas violentas, la segunda etapa tiene un algoritmo de normalización y la tercera etapa utiliza un algoritmo de detector de rostro. Si la imagen tiene poca resolución, el algoritmo de super-resolución mejora la calidad y el algoritmo de detección de rostro puede recuperar las características del objeto y lo detecta con mayor claridad.

7. La exactitud en la detección de pistolas fue superior al 94.31% por realizar cambios en la iluminación de una imagen, ya sea agregando un poco más de brillo o restándole iluminación. Similarmente, el algoritmo de difuminación permitió aumentar la claridad de la imagen o hacer que se difumine levemente para aumentar el contexto del entorno donde pueda detectarse el objeto en la vida real. Además, la obtención de estas características nuevas de los objetos ocurrió gracias a YOLOv3-spp, el que añade una última capa denominada como Spatial Pyramid Pooling (He et al., 2015).

8. El tiempo promedio de entrenamiento conseguido en esta investigación fue inferior a 70.7 s por usar especificaciones de hardware de equipo de cómputo más potentes que el usado en el estudio de Nguyen et al. (2020). Las características utilizadas fueron Intel(R) Xeon(R) CPU @ 2.30GHz, 12.7 GB RAM y Tesla T4 15 GB GPU implementado en la nube con Google Colab.

9. El tiempo promedio de identificación para cuchillos y pistolas fue inferior a 142 ms con YOLOv3 que utiliza una única red neuronal convolucional a diferencia de Faster RCNN que utiliza un enfoque basado en regiones con una combinación de dos redes neuronales. YOLOv3 es usado para detecciones en tiempo real. Además, el tiempo promedio de identificación para esta investigación fue menor por el uso de una AMD Ryzen 5 5600H 3.30 GHz, 16 GB RAM y NVIDIA GeForce RTX 3060 6GB.

VII. RECOMENDACIONES

Para futuras investigaciones, se sugiere lo siguiente:

1. Probar los algoritmos de detección de objetos con los servicios de cloud computing como Google Cloud Plataform, Amazon Web Services, Microsoft Azure, etc., con los que se tenga mayor capacidad de procesamiento para las pruebas de los algoritmos.

2. Interactuar con los hiper parámetros del Algoritmo YOLOv3-spp antes del entrenamiento; ya que, permite obtener mejores resultados en la detección. Los hiper parámetros indican al algoritmo los parámetros que afectarán al modelo al momento del entrenamiento como, por ejemplo: el número de épocas, la tasa de aprendizaje, el tamaño de lote, el tamaño de la celda que extraerá las características de la imagen, entre otros. Todos los parámetros están en el anexo 18.

3. Crear un conjunto de datos de imágenes con los algoritmos para rotar, alejar y acercar junto a los algoritmos de iluminación y difuminación para entrenar otros algoritmos de detección de objetos evaluando la sensibilidad, la especificidad, la precisión y la exactitud. La razón de ello es debido a que, con una mayor cantidad de imágenes en diferentes posiciones servirá para lograr mejores resultados en la detección, además sirve para que el conjunto de datos sea útil en el entrenamiento de algoritmos como Faster RCNN, SSD, ResNet-101, en otros.

4. El sistema de detección de cuchillos y pistolas fue probado y usado en una instancia física local, pero se recomienda llevar este sistema a un entorno Cloud para que pueda ser usado por más personas o clientes. El objetivo de tenerlo en un entorno Cloud es por la disponibilidad y porque el cliente o usuario final no necesita tener un buen equipo mínimo OnPremise.

5. Cambiar la arquitectura del sistema de detección de objetos por una arquitectura API REST que permita escalar y distribuir el sistema como un servicio.

REFERENCIAS

ABADI, Martín, et al., 2016. *TensorFlow: A system for large-scale machine learning [TensorFlow: Un sistema para el aprendizaje automático a gran escala].* s.l. : 12th symposium on operating systems design and implementation, 2016. págs. 265-283. Vol. 16.

ARCEDA Machaca, V., et al., 2016. *Fast Face Detection in Violent Video Scenes [Detección rápida de rostros en escenas de video violentas].* 2016. págs. 5-26. Vol. 329.

CHICCO, Davide, TOTSCH, Niklas y JURMAN, Giuseppe. 2021. *The Matthews correlation coefficient (MCC) is more reliable than balanced accuracy, bookmaker informedness, and markedness in two-class confusion matrix evaluation.* Toronto : BioData mining, 2021.

COLEGIO DE INGENIEROS DEL PERÚ. 2018. *Código de ética del Colegio de Ingenieros del Perú.* Lima : s.n., 2018.

CSURKA, Gabriella, et al., 2004. *Visual categorization with bags of keypoints [Categorización visual con bolsas de puntos clave].* 2004. págs. 1-2. Vols. Vol. 1, No. 1-22.

DADERMAN, Antonia y ROSANDER, Sara. 2018. *Evaluating Frameworks forImplementing Machine Learning in Signal Processing [Evaluación de marcos para implementar el aprendizaje automático en el procesamiento de señales].* 2018.

DAI, Jifeng, et al., 2016. *R-FCN: Object Detection via Region-based Fully Convolutional Networks.* s.l. : Advances in neural information processing systems, 2016. págs. 379-387.

DALIANIS, H. 2018. *Evaluation Metrics and Evaluation.* s.l. : Clinical Text Mining: secondary use of electronic patient records, 2018.

ELSNER, Jens, et al., 2019. *Automatic Weapon Detection in Social Media Image Data Using a Two-Pass Convolutional Neural Network[La detección automática de armas en Los datos de imagen de los medios sociales usando una neurona convolucional de dos pasos Red].* s.l. : European Law Enforcement Research Bulletin, 2019. págs. 61-65. Vol. 4.

EVERINGHAM, Mark, et al., 2009. *The Pascal Visual Object Classes Challenge 2006 (VOC2006) Results [Los resultados del Desafío de Clases de Objetos Visuales Pascal 2006 (VOC2006)].* 2009.

FAN, Quanfu, CHEN, Chun-Fu y LEE Giun, Gwo. 2017. *A Sparse Deep Feature Representation for Object Detection from Wearable Cameras [Una representación de rasgos de poca profundidad para la detección de objetos de cámaras portátiles].* 2017.

FERNANDEZ Carrobles, Milagros, DENIZ, Oscar y MAROTO, Fernando. 2019. *Gun and knife detection based on Faster R-CNN for video surveillance [Detección de armas y cuchillos basada en Faster R-CNN para la vigilancia por video].* [ed.] Cham Springer. 2019. págs. 441-452.

FIERES, J, Schemmel, J. y Meier, K. 2006. *Training convolutional networks of threshold neurons suited for low-power hardware implementantion [Entrenamiento de redes convolucionales de neuronas umbral adecuadas para la implementación de hardware de baja potencia].* s.l. : Neural Networks, 2006. IJCNN'06. International Joint Conference on. IEEE, 2006. págs. 21-28.

HE, Kaiming, et al., 2016. *Identity Mappings in Deep Residual Networks [Mapeo de identidad en redes residuales profundas].* [ed.] Cham Springer. 2016. págs. 630-645.

HE, Kaiming, et al., 2015. *Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition.* s.l. : IEEE transactions on pattern analysis and machine intelligence, 2015.

HERNÁNDEZ Sampieri, Roberto y MENDOZA Torres, Christian Paulina. 2018. *Metodología de la investigación: Las rutas cuantitativa, cualitativa y mixta.* Ciudad de México : Mc-Graw-Hill Interamericana Editores, 2018. 978-1-4562-6096-5.

HERNÁNDEZ, Roberto, FERNÁNDEZ, Carlos y BAPTISTA, Pilar. 2014. *Metodología de la investigación.* Sexta. México D.F : McGRAW-HILL, 2014. 978-1-4562-2396-0.

Instituto Nacional de Estadística e Informática. 2022. *Homicidios en el Perú, contandolos uno a uno 2019 y 2020.* Lima : s.n., 2022.

IQBAL, Javed, et al., 2019. *Orientation Aware Object Detection with Application to Firearms [Detección de objetos de orientación con aplicación a las armas de fuego].* 2019.

KUHN, Alexander, et al., 2012. *A Lagrangian Framework for Video Analytics [Un marco lagrangiano para el análisis de video].* s.l. : 2012 IEEE 14th International Workshop on Multimedia Signal Processing (MMSP). IEEE, 2012. págs. 387-392.

LEJMI, Wafa, MAHJOUB, Mohamed Ali y KHALIFA, Anouar Ben. 2017. *Fusion Strategies for Recognition of Violence Actions [Estrategias de fusión para el reconocimiento de acciones de violencia].* 2017. págs. 178-183.

LONG, Jonathan, SHELHAMER, Evan y DARRELL, Trevor. 2015. *Fully Convolutional Networks for Semantic Segmentation [Redes totalmente convolutivas para la segmentación semántica].* 2015. págs. 3431-3440.

LÓPEZ Gil, Alba y PAJARES Gutiérrez, Javier. 2018. *Estudio comparativo de metodologías tradicionales y ágiles para proyectos de Desarrollo de Software.* Valladolid : s.n., 2018.

MAHALINGAM, T. y SUBRAMONIAM, M. 2019. *A trusted waterfall framework based moving object detection using FACO-MKFCM techniques [Un marco de confianza basado en la detección de objetos móviles en cascada usando técnicas FACO-MKFCM].* 2019. págs. 26427-26452. Vol. 78.

NEBAUER, C. 1998. *Evaluation of convolutional neural networks for visual recognition [Evaluación de las redes neuronales convolutivas para el reconocimiento visual].* s.l. : IEEE Transactions on Neural Networks, 1998. págs. 685-696.

NGUYEN, Nhat-Duy, et al., 2020. *An Evaluation of Deep Learning Methods for Small Object Detection.* 2020. págs. 1-18.

ÑAUPAS Paitan, Humberto, MEJIA Mejia, Elias y NOVOA Ramirez, Eliana. 2014. *Metodología de la investigación cuantitativa - cualitativa y redacción de la tesis.* Cuarta. Bogotá, Colombía : Ediciones de la U, 2014.

Oficina de las Naciones Unidas contra la Droga y el Delito (UNODC). 2019. ESTUDIO MUNDIAL SOBRE EL HOMICIDIO. [En línea] julio de 2019. [Citado el: 28 de 06 de 2023.] https://www.unodc.org/ropan/es/estudio-mundial-sobre-el-homicidio-en-espaol.html.

OLMOS, Roberto, TABIK, Siham y HERRERA, Fransisco. 2017. *Automatic Handgun Detection Alarm in Videos Using Deep Learning [Alarma de detección automática de armas de mano en videos que utilizan el aprendizaje profundo].* 2017. págs. 66-72. Vol. 275.

Open Images Dataset V7. 2022. Open Images Dataset V7. *Open Images Dataset V7.* [En línea] 01 de 10 de 2022. [Citado el: 08 de 07 de 2023.] https://storage.googleapis.com/openimages/web/factsfigures_v7.html.

OpenCV. OpenCV. *OpenCV.* [En línea] [Citado el: 08 de 07 de 2023.] https://opencv.org/about/.

PIYUSH, Vashistha, JUGINDER Pal, Singh y MOHD Aamir, Khan. 2020. *A Comparative Analysis of Different Violence Detection Algorithms from Videos [Un análisis comparativo de diferentes algoritmos de detección de la violencia a partir de vídeos].* 2020.

PYTORCH. What is PyTorch? [En línea] [Citado el: 02 de 07 de 2020.] https://pytorch.org/tutorials/beginner/blitz/tensor_tutorial.html#sphx-glr-beginner-blitz-tensor-tutorial-py.

REDMON, Joseph y FARHADI, Ali. 2018. *YOLOv3: An Incremental Improvement.* Washington : s.n., 2018. págs. 1-6.

REN, Shaoqing, et al., 2015. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks [Faster R-CNN: Hacia la detección de objetos en tiempo real con las redes de propuestas de regiones].* s.l. : Advances in neural information processing systems, 2015. págs. 91-99.

SCHWABER, Ken y SUTHERLAND, Jeff. 2020. *La guía de Scrum. La guía definitiva de Scrum: Las reglas del juego.* 2020.

SENST, Tobias, EISELEIN, Volker y SIKORA, Thomas. 2015. *A Local Feature based on Lagrangian Measures for Violent Video Classification [Un reportaje local basado en las medidas de Lagrange para la clasificación de videos violentos].* 2015.

SENST, Tobias, et al., 2017. *Crowd Violence Detection Using Global Motion-Compensated Lagrangian Features and Scale-Sensitive Video-Level Representation.* 2017. págs. 2945-2956.

SERRANO Gracia, Ismael , et al., 2015. *Fast Fight Detection [Detección de lucha rápida].* s.l. : PLoS One, 2015.

SIMONYAN, Karen y ZISSERMAN, Andrew. 2014. *Very deep convolutional networks for large-scale image recognition [Redes convolucionales muy profundas para el reconocimiento de imágenes a gran escala].* 2014.

SUN, Shiliang, LIU, Yuhan y MAO, Liang. 2019. *Multi-view learning for visual violence recognition with maximum entropy discrimination and deep features [Aprendizaje multivisión para el reconocimiento de la violencia visual con la máxima discriminación de entropía y características profundas].* 2019. págs. 43-53. Vol. 50.

THE MATHWORKS. Descripción del producto MATLAB. [En línea] [Citado el: 02 de 07 de 2020.] https://la.mathworks.com/help/matlab/learn_matlab/product-description.html.

UNIVERSIDAD CÉSAR VALLEJO. 2021. *RESOLUCIÓN DE CONSEJO UNIVERSITARIO N° 0340-2021/UCV.* Trujillo : s.n., 2021.

VAN DER GOOT, Rob. 2021. *We Need to Talk About train-dev-test Splits.* Punta Cana : Association for Computational Linguistics, 2021.

VEDALDI, Andrea y LENC, Karel. 2014. *MatConvNet - Convolutional Neural Networks for MATLAB.* 2014.

VERMA, Gyanendra y DHILLON, Anamika. 2017. *A Handheld Gun Detection using Faster R-CNN Deep Learning [Una detección de armas de mano usando un Faster R-CNN de aprendizaje profundo].* 2017. págs. 84-88.

WANG, CHAOXIN, et al., 2023. *Detecting common coccinellids found in sorghum using deep learning models.* s.l. : Nature, 2023. pág. 9748.

WANG, Limin, et al., 2016. *Temporal segment networks: Towards good practices for deep action recognition [Redes de segmentos temporales: Hacia las buenas prácticas para el reconocimiento de la acción profunda].* s.l. : En European conference on computer vision, 2016. págs. 20-36.

WARSI, Arif, et al., 2019. *Gun Detection System Using YOLOv3.* 2019. págs. 1-5.

YANG, Suorong, et al., 2022. *Image Data Augmentation for Deep Learning: A Survey.* s.l. : arXiv preprint arXiv:2204.08610, 2022.

YUAN, Yue, et al., 2020. *A scale-adaptive object-tracking algorithm with occlusion detection [Un algoritmo de rastreo de objetos adaptable a la escala con detección de oclusión].* 2020. págs. 1-15. Vol. 1.

ZEILER, Matthew D. y FERGUS, Rob. 2014. *Visualizing and understanding convolutional neural networks [Visualizar y comprender las redes neuronales convolucionales].* 2014. págs. 6-12.

ZHANG, Tao, et al., 2016. *A new method for violence detection in surveillance scenes [Un nuevo método para la detección de la violencia en las escenas de vigilancia].* 2016. págs. 7327-7349. Vol. 75.

ZHANG, Yang, SONG, Chenglong y ZHANG, Dongwen. 2020. *Deep Learning-Based Object Detection Improvement for Tomato Disease.* 2020. págs. 1-8.

ZHANG, Zhifei. 2016. *Derivation of Backpropagation in Convolutional Neural Network (CNN) [Derivación de la retropropagación en la Red Neural Convolutiva (CNN)].* s.l. : University of Tennessee, Knoxville, TN, 2016.

ZHONGHUA, Guo, et al., 2017. *Pedestrian violence detection based on optical flow energy characteristics [Detección de la violencia peatonal basada en las características de la energía del flujo óptico].* 2017.

ZHOU, Peipei, et al., 2018. *Violence detection in surveillance video using low-level features [Detección de la violencia en los videos de vigilancia usando características de bajo nivel].* s.l. : PLoS one, 2018. Vol. 13.

ZOPH, Barret, et al., 2019. *Learning Data Augmentation Strategies for Object Detection.* Reino Unido : Springer International Publishing, 2019.

Anexo 1: Matriz de operacionalización de variables

Tabla 21: Matriz de operacionalización de variables

Variable	Definición conceptual	Definición operacional	Dimensiones	Indicadores	Instrumento	Escala de medición
Efecto de un sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV	El uso de armas en lugares públicos se ha convertido en un gran problema en nuestra sociedad (Fernandez-Carrobles et al., 2019, p. 1). Está principalmente involucrada en el aumento de la seguridad de los seres humanos porque es una herramienta importante para monitorear el comportamiento de las personas (Lejmi et al., 2017, p. 1).	La seguridad podrá incrementarse aplicando algoritmos de visión artificial a las imágenes obtenidas de los sistemas de vigilancia por vídeo (Fernandez-Carrobles et al., 2019, p. 1). Ayudará a prevenir, detectar y reducir el crimen (Lejmi et al., 2017, p. 1).	Sensibilidad (Olmos et al., 2017, p. 14; Warsi et al., 2019, p.4)	Sensibilidad = VP / (VP + FN)	Matriz de confusión	Razón
			Especificidad (Elsner et al., 2019, p. 64)	Especificidad = VN / (VN + FP) VN = Verdadero Negativo FP = Falso Positivo (Elsner et al., 2019, p. 64)	Matriz de confusión	Razón
			Precisión (Fernandez Carrobles et al., 2019, p. 7; Olmos et al., 2017, p. 7)	Precisión = VP / (VP + FP) VP = Verdadero Positivo FP = Falso Positivo (Fernandez Carrobles et al., 2019, p. 7; Olmos et al., 2017, p. 7)	Matriz de confusión	Razón
			Exactitud (Arceda et al., 2016, p. 21; Zhou et al., 2018, p. 10)	Exactitud = (VP + VN) / (VP + VN + FP + FN) VP = Verdadero Positivo VN = Verdadero Negativo FP = Falso Positivo FN = Falso Negativo (Arceda et al., 2016, p. 21; Zhou et al., 2018, p. 10)	Matriz de confusión	Razón
			Tiempo promedio de entrenamiento (Nguyen et al., 2020, p. 10)	$\text{Tiempo promedio de entrenamiento (s)} = \frac{\text{FechaHora}_{final} - \text{FechaHora}_{inicial}}{Cantidad\ Imagenes\ entrenadas}$ FechaHora_{final} = Fecha y hora final del entrenamiento. $\text{FechaHora}_{inicial}$ = Fecha y hora inicial del entrenamiento. Cantidad imágenes entrenamiento = Contar todas las imágenes usadas para el entrenamiento.	Hoja de tabulación	Razón
			Tiempo promedio de identificación (Zhang et al., 2020, p. 10; Redmon y Farhadi, 2018, p. 1)	$\text{Tiempo promedio de identificación (ms)} = \frac{\sum_{i=1}^{n} t_i}{n}$ t = Tiempo de cada identificación del conjunto de datos test. n = Total de imágenes del conjunto de datos test.	Hoja de tabulación	Razón

Anexo 2: Matriz de consistencia

Tabla 22: Matriz de consistencia

PROBLEMAS	OBJETIVOS	HIPOTESIS	VARIABLE	DIMENSIONES	INDICADORES
General	**General**	**General**			
¿Cuál fue el efecto del sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV?	Determinar el efecto de un sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV.	El sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV tuvo una sensibilidad de por lo menos 90% (Olmos et al., 2017, p. 14; Warsi et al., 2019, p.4), una especificidad de por lo menos 95%. (Elsner et al., 2019, p. 64), una precisión de por lo menos 84.21% (Fernandez Carrobles et al., 2019; Olmos et al., 2017, p. 9), una exactitud de por lo menos 90% (Arceda et al., 2016, p. 21; Zhou et al., 2018, p. 10), el tiempo promedio de entrenamiento no mayor a 70.7 s (Nguyen et al., 2020, p. 10) y un tiempo promedio de identificación de por lo menos 22 ms (Zhang et al., 2020, p. 10; Redmon y Farhadi, 2018, p. 1).	-	-	-
Específicos	**Específicos**	**Específicos**			**Indicadores**
¿Cuál fue el efecto del sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV en la precisión?	Determinar el efecto de un sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV en la precisión.	El sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV tuvo una sensibilidad de por lo menos 90% (Olmos et al., 2017, p. 14; Mohd et al., 2019, p.4).	Efecto del sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV	Sensibilidad (Olmos et al., 2017, p. 14; Mohd et al., 2019, p.4)	Sensibilidad = VP / (VP + FN) VP = Verdaderos Positivos FN = Falso Negativo (Olmos et al., 2017, p. 14; Mohd et al., 2019, p.4)
¿Cuál fue el efecto del sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV en la especificidad?	Determinar el efecto de un sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV en la especificidad.	El sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV tuvo una especificidad de por lo menos 95% (Elsner et al., 2019, p. 64).		Especificidad (Elsner et al., 2019, p. 64)	Especificidad=VN/(VN+FP) VN = Verdadero Negativo FP = Falso Positivo (Elsner et al., 2019, p. 64)
¿Cuál fue el efecto del sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV en la sensibilidad?	Determinar el efecto de un sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV en la sensibilidad.	El sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV tuvo una precisión de por lo menos 84,21% (Fernandez Carrobles et al., 2019; Olmos et al., 2017, p. 9).		Precisión (Fernandez Carrobles et al., 2019, p. 7; Olmos et al., 2017, p. 7)	Precisión = VP / (VP + FP) VP = Verdadero Positivo FP = Falso Positivo (Fernandez Carrobles et al., 2019, p. 7; Olmos et al., 2017, p. 7)

PROBLEMAS	OBJETIVOS	HIPOTESIS	VARIABLE	DIMENSIONES	INDICADORES
Específicos	**Específicos**	**Específicos**			**Indicadores**
¿Cuál fue el efecto del sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV en la exactitud?	Determinar el efecto de un sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV en la exactitud.	El sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV tuvo una exactitud de por lo menos 90% (Arceda et al., 2016, p. 21; Zhou et al., 2018, p. 10).		Exactitud (Arceda et al., 2016, p. 21; Zhou et al., 2018, p. 10)	Exactitud = (VP + VN) / (VP + VN + FP + FN) VP = Verdadero Positivo VN = Verdadero Negativo FP = Falso Positivo FN = Falso Negativo (Arceda et al., 2016, p. 21; Zhou et al., 2018, p. 10)
¿Cuál fue el efecto del sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV en el tiempo promedio de entrenamiento?	Determinar el efecto de un sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV en el tiempo promedio de entrenamiento.	El sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV tuvo un tiempo promedio de entrenamiento no mayor a 70.7 s (Nguyen et al., 2020, p. 10).		Tiempo promedio de entrenamiento (Nguyen et al., 2020, p. 10)	Tiempo promedio de entrenamien $= \frac{\text{FechaHora}_{final} - \text{FechaHora}_{ini}}{Cantidad\ Imagenes\ Entrenad}$ FechaHora_{final} = Fecha y hora final del entrenamiento. $\text{FechaHora}_{inicial}$ = Fecha y hora inicial del entrenamiento. Cantidad imágenes entrenamiento = Contar todas las imágenes usadas para el entrenamiento.
¿Cuál fue el efecto del sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV en el tiempo promedio de identificación?	Determinar el efecto de un sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV en el tiempo promedio de identificación.	El sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV tuvo un tiempo promedio de identificación de por lo menos 22 ms (Zhang et al., 2020, p. 10; Redmon y Farhadi, 2018, p. 1).		Tiempo promedio de identificación (Zhang et al., 2020, p. 10; Redmon y Farhadi, 2018, p. 1)	Tiempo promedio de identificació $= \frac{\sum_{i=1}^{n} t_i}{n}$ t = Tiempo de cada identificación del conjunto de datos test. n = Total de imágenes del conjunto de datos test.

Anexo 3. Prototipo de pantallas del sistema

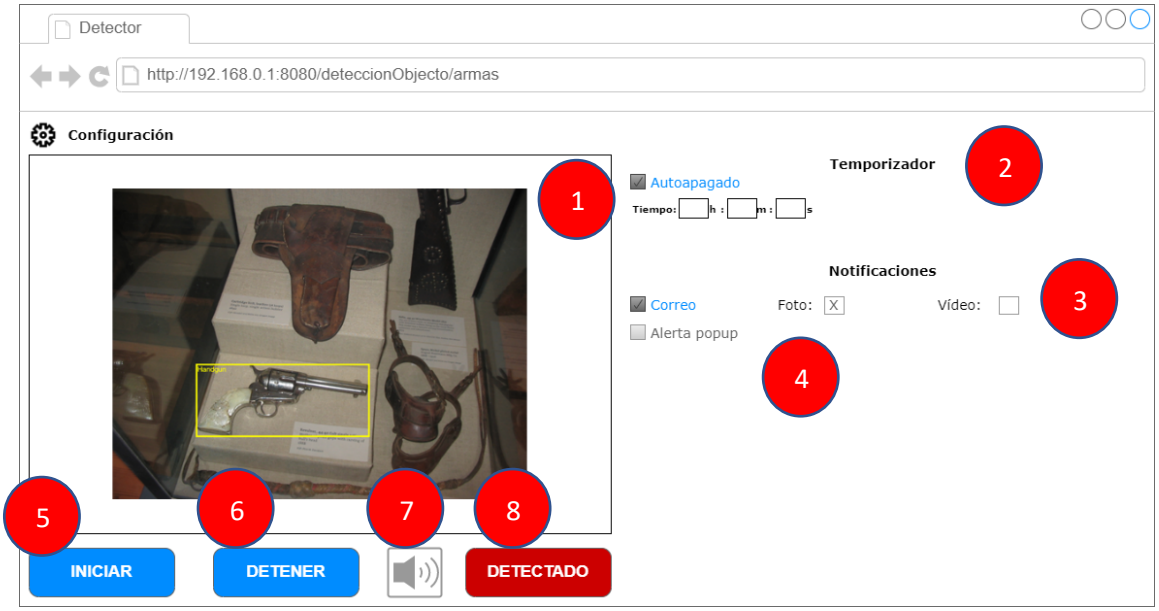

Figura 3: Pantalla principal

La pantalla principal se observará cuando se inicie el sistema. La vista tiene 8 componentes.

1) Componente pantalla: Para visualizar el video de la cámara.
2) Componente temporizador: Se utiliza para indicar al sistema que debe mostrar el video durante el tiempo configurado en los cuadros **h**, **m y s**.
3) Componente notificaciones correo: Se utiliza para indicar al sistema si la alerta debe ser enviado por correo. Además, puede indicarse si al momento de enviar la alerta por correo electrónico, se adjunte una foto del momento o un video con una corta duración.
4) Componente notificación alerta popup: Se utiliza para indicar al sistema si debe mostrar una notificación popup. Popup es una ventana que se mostrará en la parte inferior derecha.
5) Componente iniciar: Es un botón para encender la cámara para transmitir el video.
6) Componente detener: Es un botón para apagar o finalizar la transmisión del video.
7) Componente sonido: Indica al sistema si debe emitir un sonido como alerta al detectar el objeto.
8) Componente estado: Sirve para visualizar el estado de la detección. Si se detecta un arma mostrará el texto **DETECTADO** con un fondo rojo, de lo contrario mostrará el texto **NO DETECTADO** con un fondo verde.

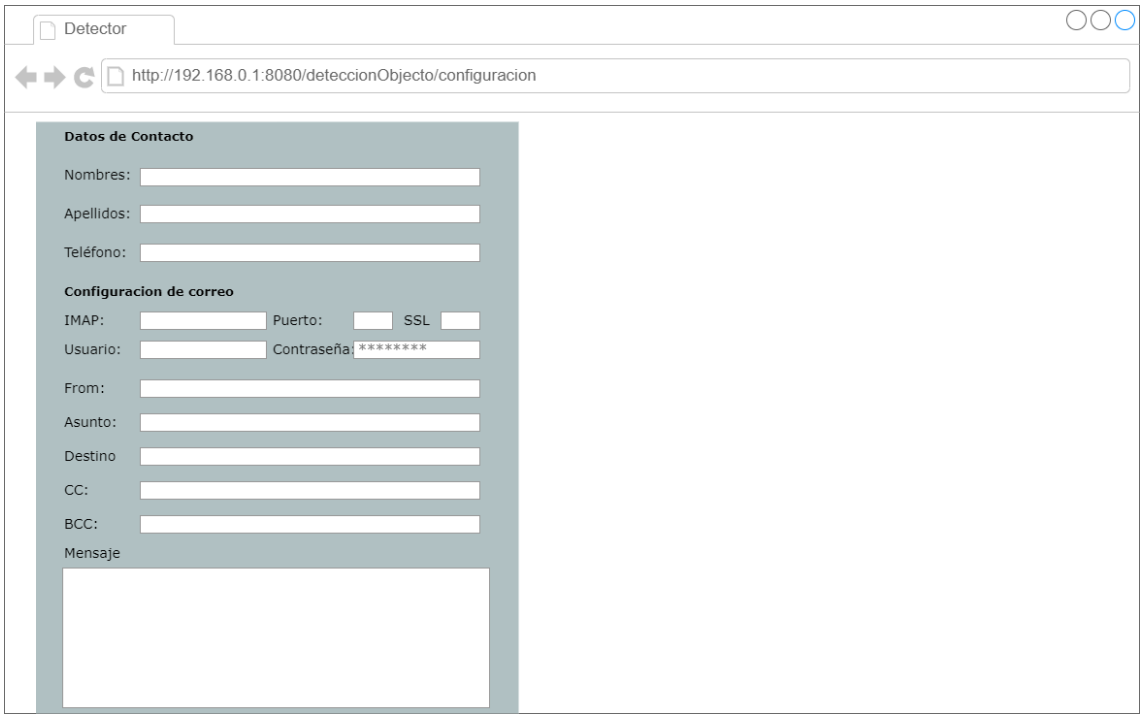

Figura 4: Vista configuración correo y datos personales

La vista presenta campos para registrar datos personales y la configuración del correo electrónico. Los datos personales servirán para ser enviados en el cuerpo del correo. Los campos **IMAP**, **Puerto**, **SSL**, **Usuario** y **Contraseña** sirven para conectarse con el servidor de correos. Se puede utilizar cualquier servidor de correo como Google, Hotmail (Outlook) entre otros. El campo **from** debe ser el mismo en **Usuario**. El campo **asunto** es el encabezado del correo. El campo **destino** sirve para agregar uno o más correos que serán enviados. El campo **CC** son correos con copia. El campo **BCC** son correos ocultos. Finalmente, el campo **mensaje** es el cuerpo del correo.

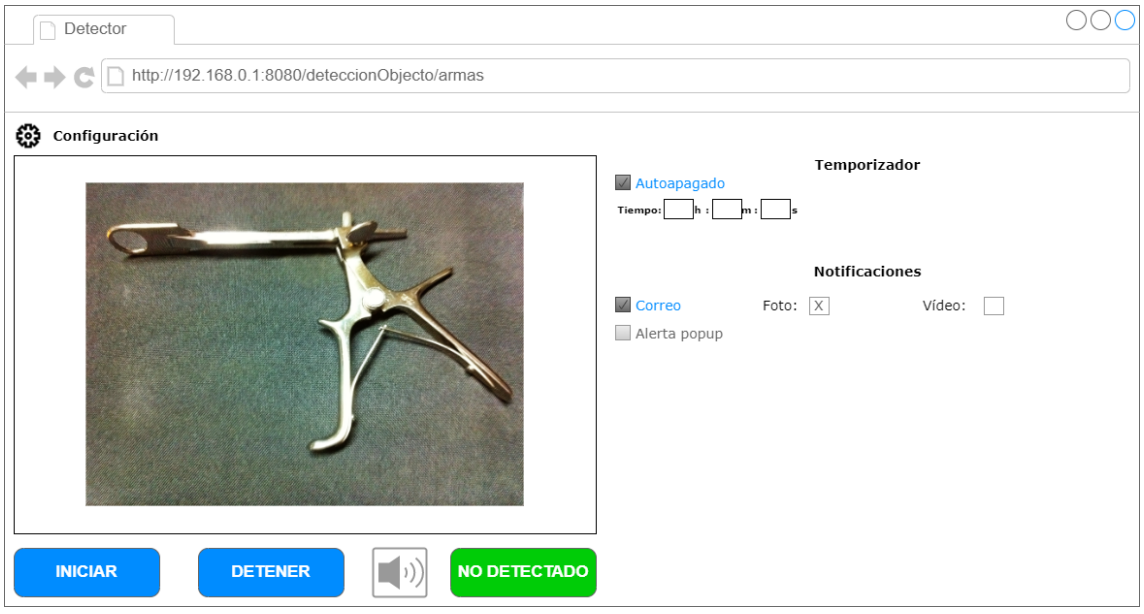

Figura 5: Objeto sin detectar arma o cuchillo.

La vista cuando no se detecte arma o cuchillo. El componente estado muestra el texto **NO DETECTADO** y con el fondo de color verde.

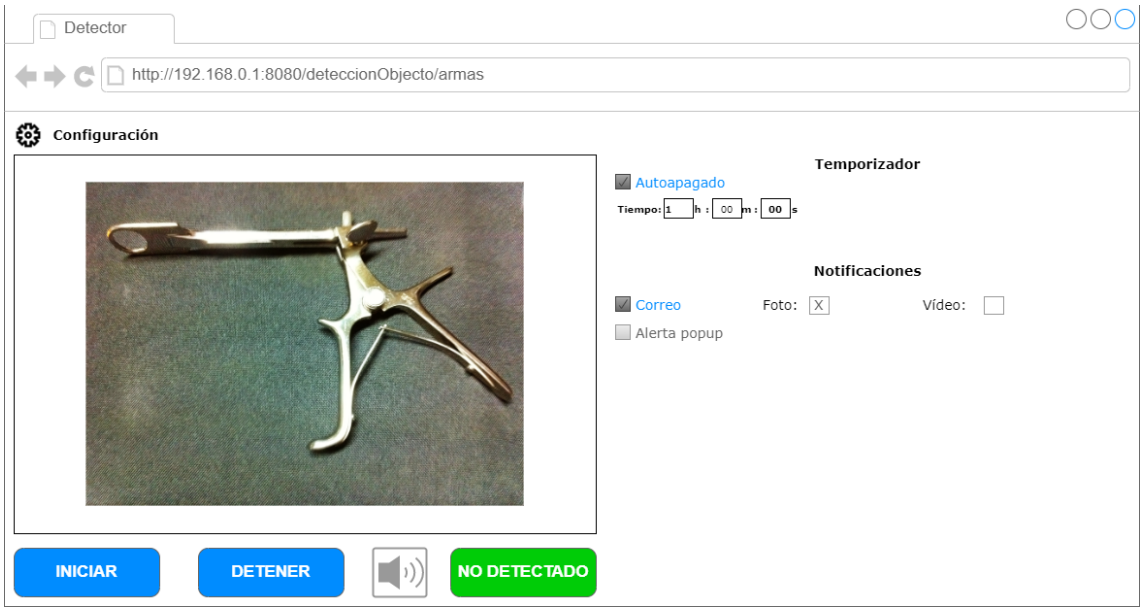

Figura 6: Temporizador

El componente temporizador, para este ejemplo, está programado por una hora.

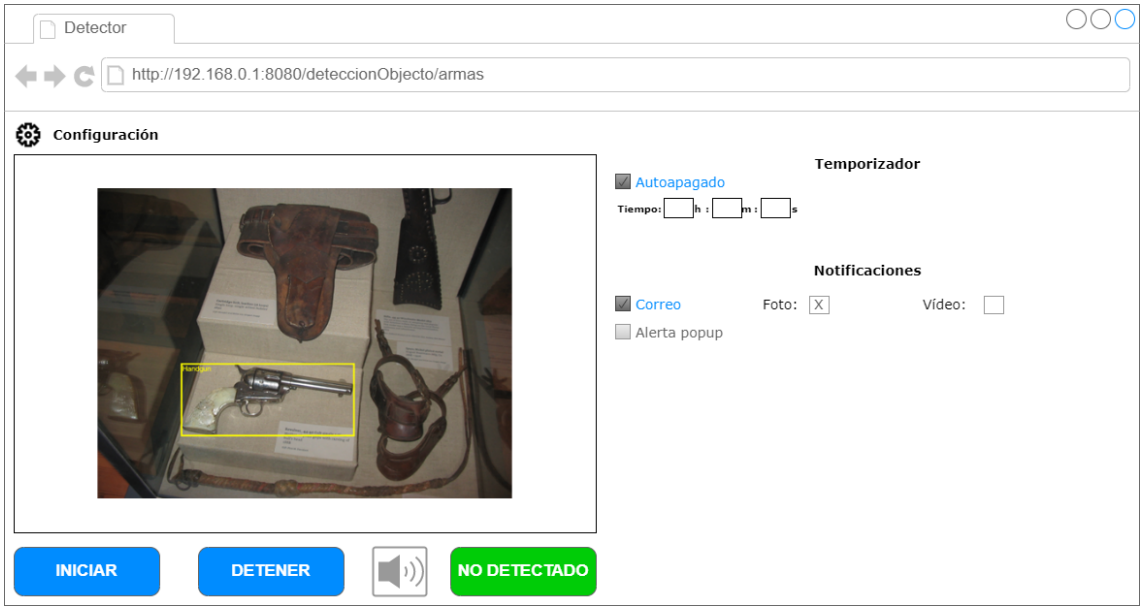

Figura 7: Notificación por correo con foto o video

En el componente notificación correo indica al sistema que, al momento de detectar un objeto (arma o cuchillo) debe enviar un mensaje por correo electrónico. Además, se puede especificar si se adjuntará una foto o video con el objeto detectado. En la **Figura 6** se muestra un ejemplo de la alerta enviada al correo.

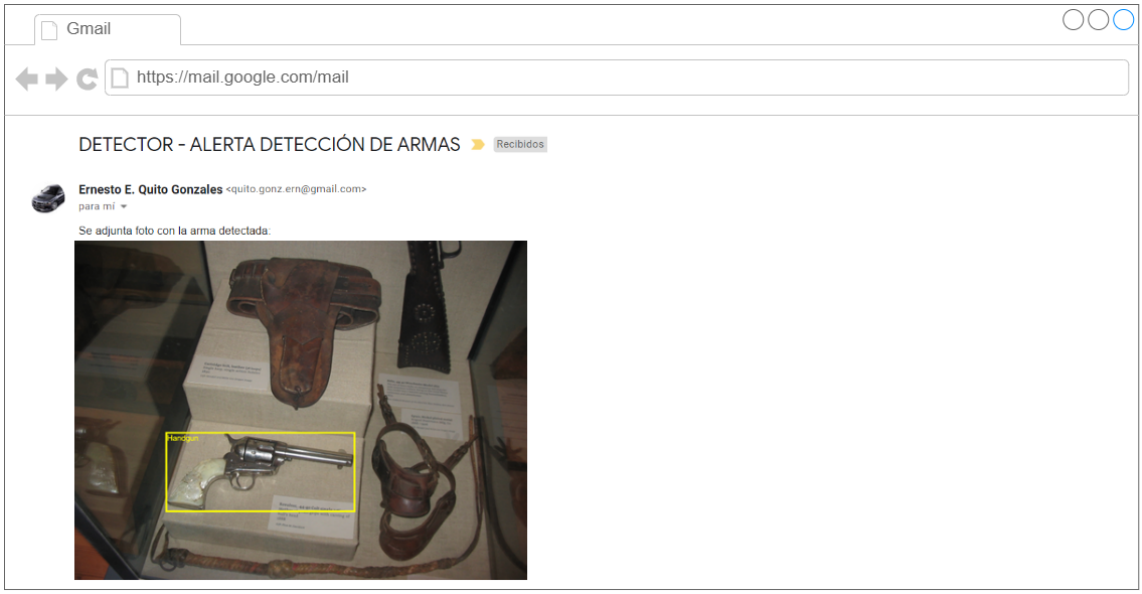

Figura 8: Alerta enviada al correo

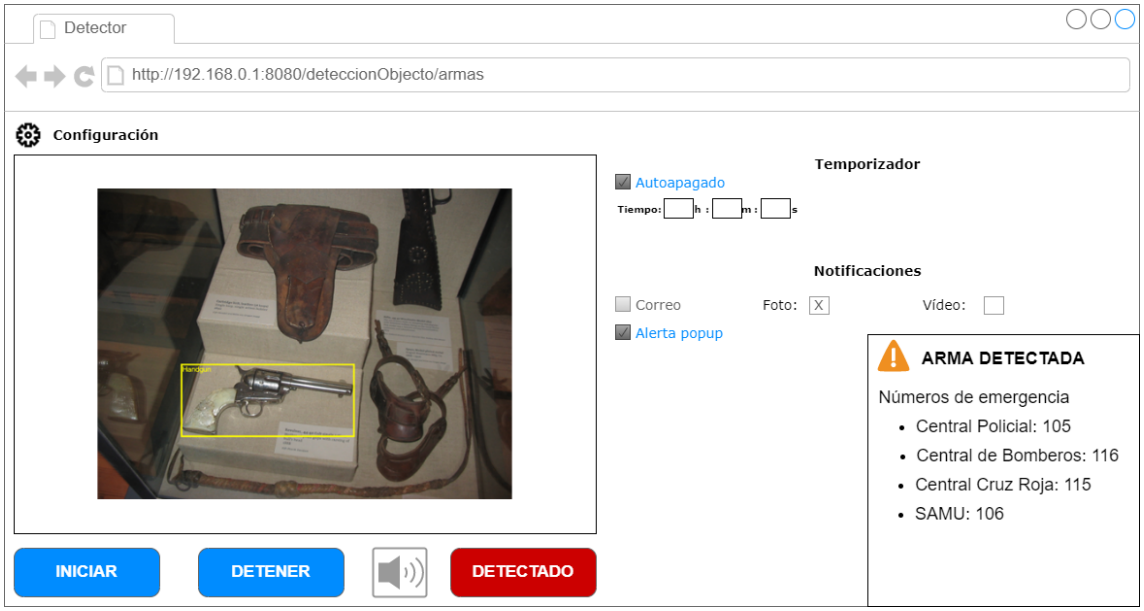

Figura 9: Alerta Popup

El componente notificaciones, tiene la opción para mostrar un popup o ventana emergente cuando se detecte un objeto.

Anexo 4: Pseudocódigo del algoritmo YOLOv3

Algoritmo para el entrenamiento del modelo de detección de objetos

```
INICIO
  Entradas
  img o imgsz ← número  # Indica tamaño de imagen.
  batch ← número # Indica el tamaño del lote a procesar durante el entrenamiento.
  epochs ← número #  Indica el número de iteración o ciclos.
  data ← cadena # Indica la ruta del archivo yaml que tiene información del conjunto
de datos.
  weights ← cadena  # Indica la ruta done se encuentra los pesos iniciales.
  device ← cadena  # Indica del 0 al 5 cuando es una GPU o cpu.
  hyp ← cadena  # Indica la ruta del archivo con los hiperparámetros.

  data_dict ← Leer data.yaml
  nc ← data_dict['nc']
  names ← data_dict['names']
  weights ← attempt_download(weights)
  ckpt ← torch.load (weights)
  model ← Class Model (cfg=ckpt, ch=3, nc=nc, anchors=hyp.get('anchors'))
  ---
  gs ← max(int(model.stride.max()), 32)
  imgsz ← check_img_size(imgsz, gs, floor=gs * 2)
  start_epoch ← 0
  Para cada epoch en el rango(start_epoch, epochs)
    model.train()
    ckpt <- {
              'epoch': epoch,
              'best_fitness':best_fitness,
              'model':model,
              'ema': ema.ema,
              'updates': ema.updates,
              'optimizer': optimizer.state_dict(),
              'opt': vars(opt)}
           }
    torch.save(ckpt, last)
    torch.save(ckpt, best)
  FinPara
  validate.run(best)
  model <- best
  confusion_matrix <- ConfusionMatrix(nc)
  dataloader <- create_dataloader(data)
  Para cada im de dataloader
    preds <- model(im)
    preds <- non_max_suppression(preds)
      Para cada pred en preds:
        confusion_matrix.process_batch(predn)
      FinPara
  FinPara
  confusion_matrix.plot(predn)
FIN
```

Anexo 5: Pseudocódigo del algoritmo de iluminación

```
INICIO
  # Se define la clase
  Clase TratamientoImagen:
    # Definir la variable que almacenará la imagen. Por defecto es None
    _establecer_imagen ← None

    # Definir el método para leer la imagen
    Func leer_imagen(self, ruta_imagen)
      # El resultado de la lectura de la imagen se establece en la variable
      # de clase.
      self._establecer_imagen = cv2.imread(ruta_imagen)
    FinFunc

    # Se instancia el método para el filtro de difuminación.
    # Los parámetros que entrada son:
    #   imagen: La ruta de la imagen.
    #   alpha: Nivel de contraste que se aplicará a la imagen.
    #   beta: Nivel de brillo que se aplicará a la imagen.
    Func establecer_iluminacion(self, contraste, brillo):

      # Utilizar el filtro Scale Absolute para establecer los niveles
      # de brillo.
      self.establecer_imagen = cv2.convertScaleAbs(self.establecer_imagen,
contraste, brillo)

      # Se asegura que los valores de los píxeles estén dentro del
      # rango válido. El resultado se establece a la variable de la clase.
      self._establecer_imagen = np.clip(imagen_cambiado, 0, 255)
    FinFunc
  FinClase
FIN
```

Anexo 6: Pseudocódigo del algoritmo difuminación

```
INICIO
  # Se define la clase
  Clase TratamientoImagen:
    # Definir la variable que almacenará la imagen. Por defecto es None
    _establecer_imagen ← None

    # Definir el método para leer la imagen
    Func leer_imagen(self, ruta_imagen)
      # El resultado de la lectura de la imagen se establece en la variable
      # de clase
      self._establecer_imagen = cv2.imread(ruta_imagen)
    FinFunc

    # Se instancia el método para el filtro de difuminación.
    # Los parámetros que entrada son:
    #   imagen: La ruta de la imagen.
    #   tamanio_kernel: Tamaño del kernel del filtro de difuminación.
    #   sigma: Desviación estándar para el cálculo del kernel.
    Func establecer_difuminacion(self, tamanio_kernel, sigma):

      # Utilizar el filtro de difuminación Gaussiano a la imagen
      # El resultado se establece en la variable de clase.
      self.establecer_imagen = cv2.GaussianBlur(self.establecer_imagen,
tamanio_kernel, sigma)
    FinFunc
  FinClase
FIN
```

Anexo 7: Pseudocódigo utilización de los dos algoritmos para el tratamiento de las imágenes

```
INICIO
  FILE ← Path(__file__).resolve
  ROOT ← FILE.parents[0]
  ROOT ← Path(os.path.relpath(ROOT, Path.cwd()))
  ruta_imagenes ← ‘c\\.’
  cant_tranformaciones ← ‘1000’
  tipo_tranformacion ← ‘difuminar,iluminar’
  img_salida ← 640
  off_auto_dir_save ← Falso
  dir_save ← ‘/salida’
  Func transformacion(op):
    ruta_imagenes ← [os.path.abspath(img.path) Para img en
os.scadir(ruta_imagenes)]
    Si len(ruta_imagenes) <cant_transformaciones:
      imprimir ’La cantidad de imágenes tiene que ser mayor
a la cantidad de tranformaciones’
      sys.exit(0)
    muestras ← random.sample(ruta_imagenes,
k=cant_tranformaciones)
    tratamiento_imagen ← TratamientoImagen()
    tipo_transformacion ← tipo_tranformacion.split(‘.’)
    ksizes ← [i Para i en range(1, 35) if i % 2 != 0]
    valores_brillo ← [i Para i en range(-50, 50)]
    valores_contraste ← [round(2-(i-1)/10, 1) Para i en
range(1, 41) Si round(2-(i-1)/10, 1) < -0.2 O round(2-(i-
1)/10, 1) > 0.2]

    Para tf en tipo_transformacion:
      Si tf == ‘difuminar’:
        Para di en tqdm(muestras,desc=’difuminar’):
          ksize ← random.choice(ksizes)
          tratamiento_imagen.leer_imagen(di)

tratamiento_imagen.establecer_difuminado(tamanio_kernel=(ksi
ze,ksize), sigma=0)

tratamiento_imagen.ajustar_tamanio((img_salida,img_salida))
          nombre_final_imagen ← str(uuid.uuid4())
          nombre_imagen_procesada ← os.path.join(dir_save,
f'pist_{nombre_final_imagen}.jpg')

tratamiento_imagen.exportar_imagen(ruta_imagen=nombre_imagen
_procesada)
        FinPara
      FinSi
      Si tf == 'iluminar':
        Para di in tqdm(muestras, desc='iluminar'):
```

```
            beta ← random.choice(valores_brillo)
            alpha ← random.choice(valores_contraste)
            tratamiento_imagen.leer_imagen(di)

tratamiento_imagen.establecer_iluminacion(brillo=beta,
contraste=alpha)
            tratamiento_imagen.ajustar_tamanio((img_salida,
img_salida))
            nombre_final_imagen ← str(uuid.uuid4())
            nombre_imagen_procesada ← os.path.join(dir_save,
f'pist_{nombre_final_imagen}.jpg')

tratamiento_imagen.exportar_imagen(ruta_imagen=nombre_imagen
_procesada)
          FinPara
        FinSi
      FinPara
    FinFunc

    Func main():
      Si not off_auto_dir_save:
        auto_incrementable ← 0
        Mientras Verdadero:
          carpeta ← f'proc_datos{auto_incrementable}'
          nueva_ruta ← os.path.join(dir_save, carpeta)
          Si no os.path.exists(nueva_ruta):
            os.mkdir(nueva_ruta)
            break
          FinSi
          auto_incrementable += 1
        FinMientras
        dir_save ← nueva_ruta
      FinSi
      transformacion()
    FinMain
FIN
```

Anexo 8: Flujograma del algoritmo YOLOv3

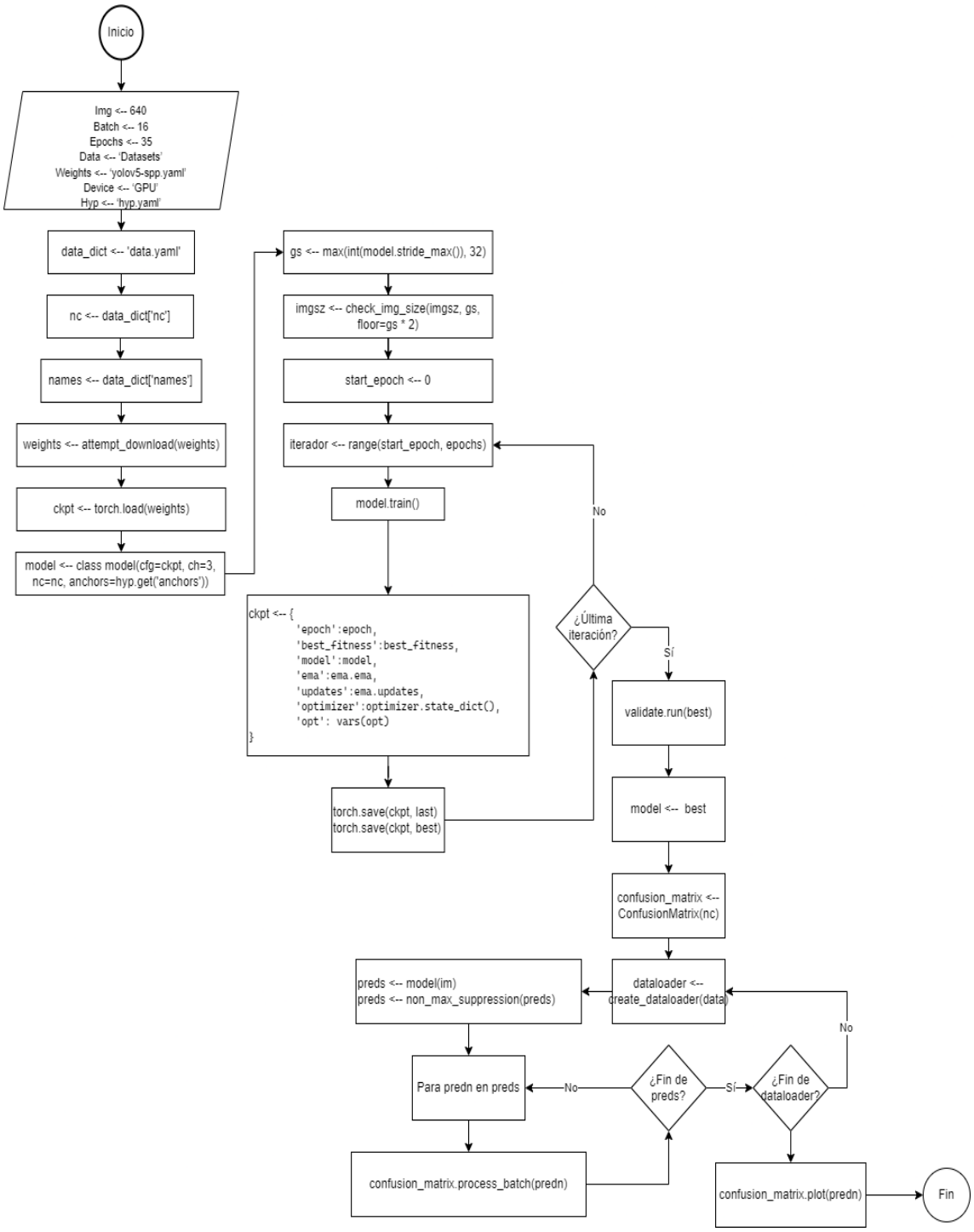

Figura 10: Flujograma de entrenamiento YOLOv3-spp

Anexo 9: Flujograma del algoritmo de iluminación

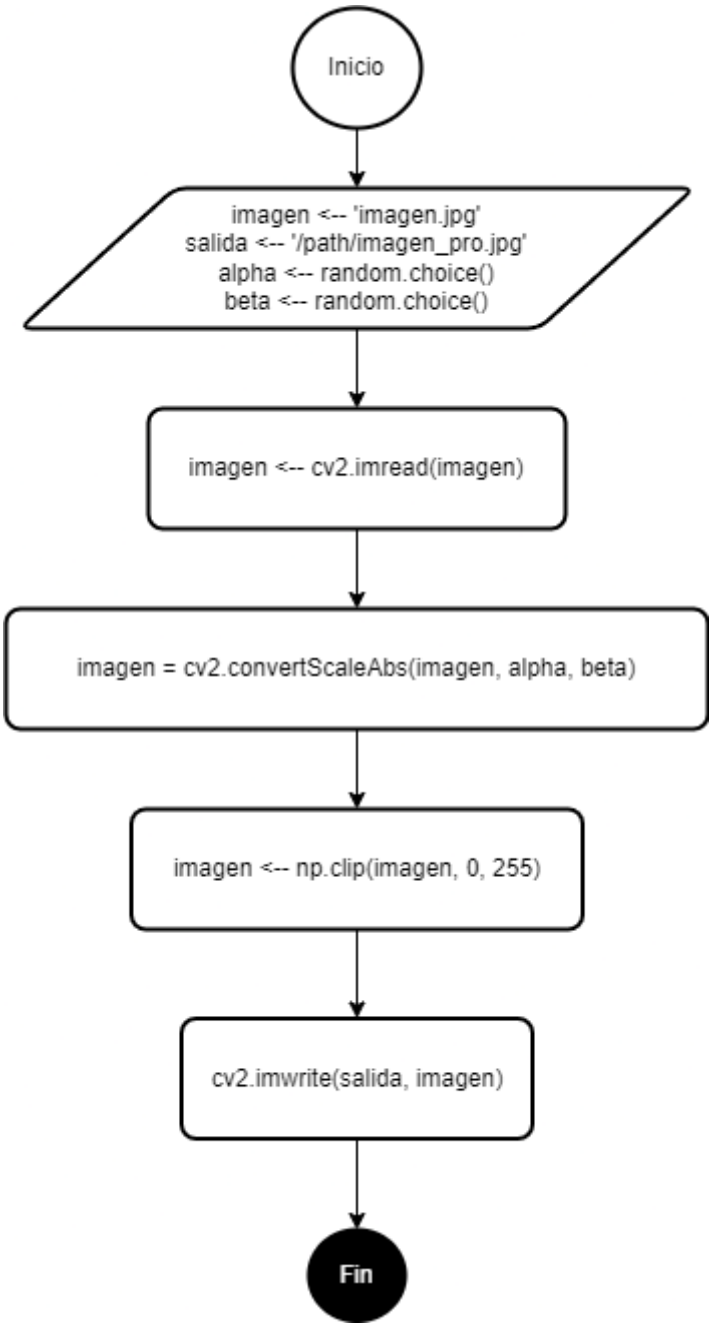

Figura 11: Flujograma del algoritmo de iluminación

Anexo 10: Flujograma del algoritmo de difuminación

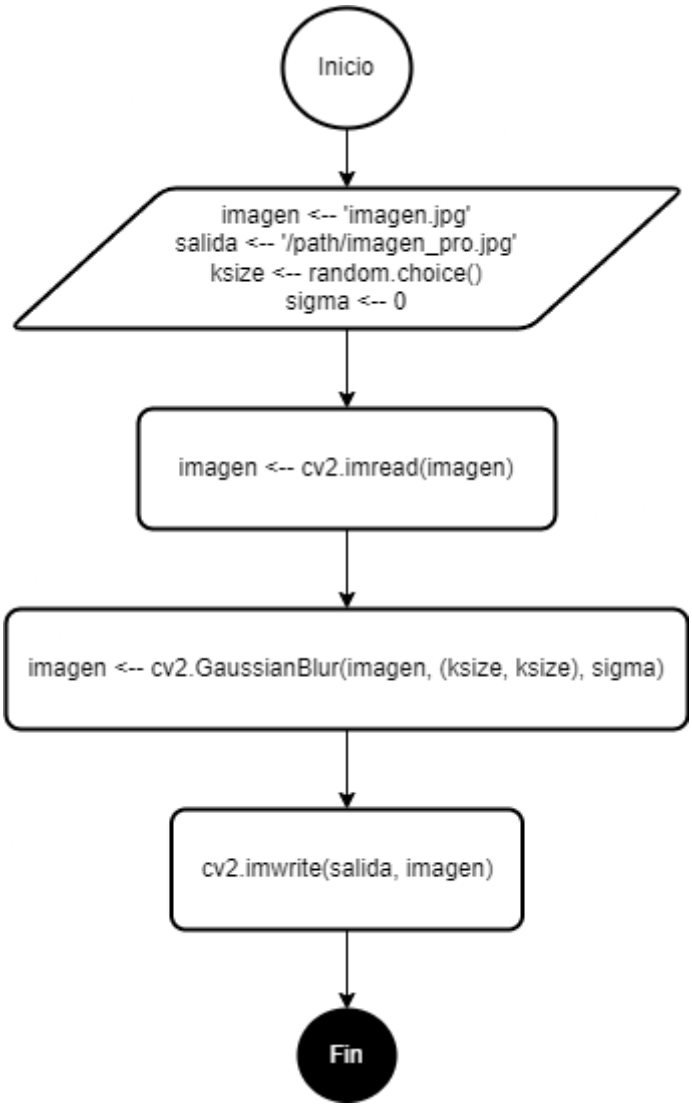

Figura 12: Flujograma del algoritmo de difuminación

Anexo 11: Flujograma con el uso de iluminación y difuminación para el tratamiento de las imágenes

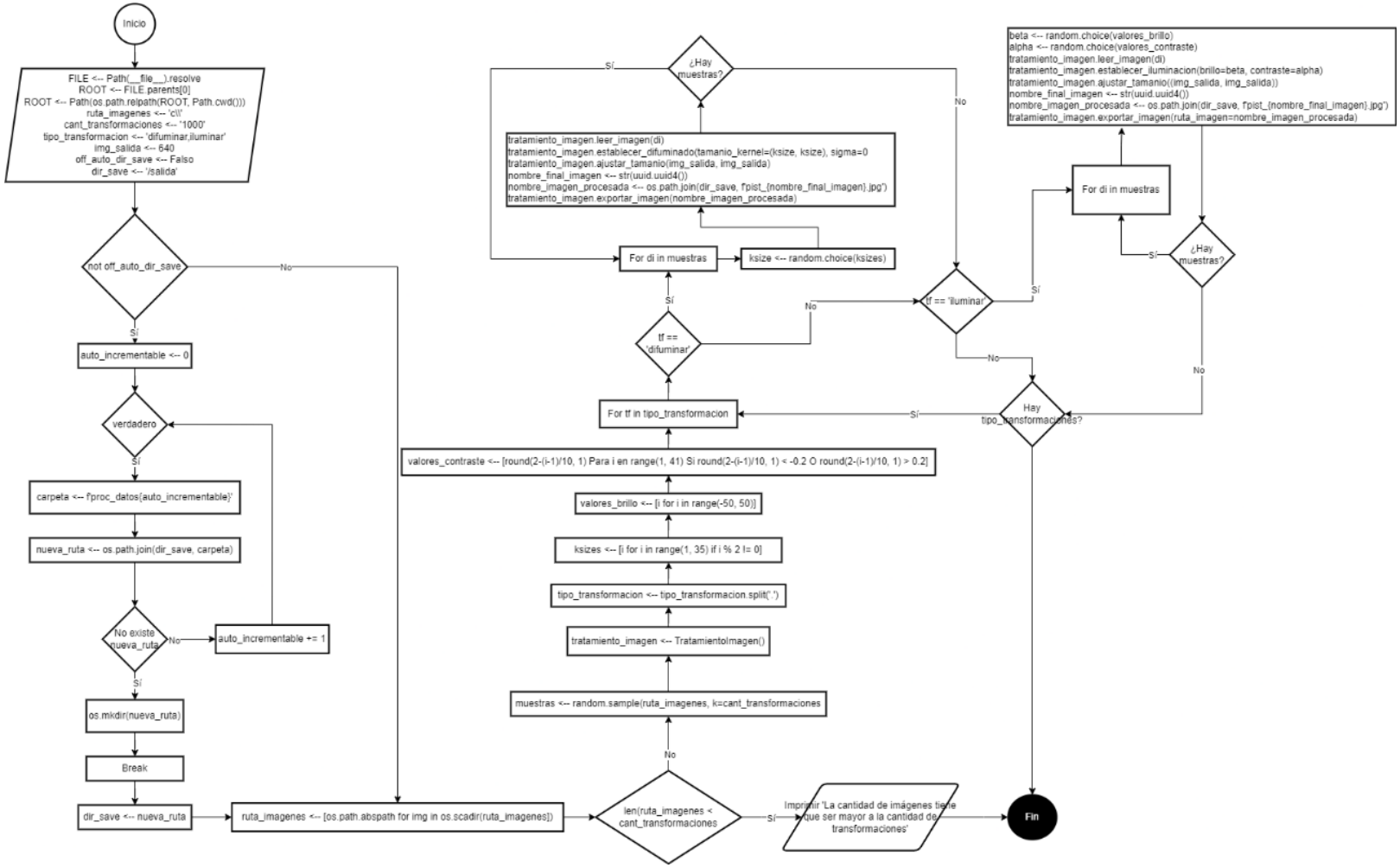

Figura 13: Flujograma con el uso iluminación y difuminación para el tratamiento de las imágenes

Anexo 12: Arquitectura tecnológica (Producción)

El entrenamiento del modelo para la detección de objetos requiere de mucha potencia computacional. Por otro lado, al llevar el modelo entrenado es menos costoso en términos de procesamiento computacional. Por ello, se presenta la arquitectura con las siguientes características de hardware en un ambiente productivo.

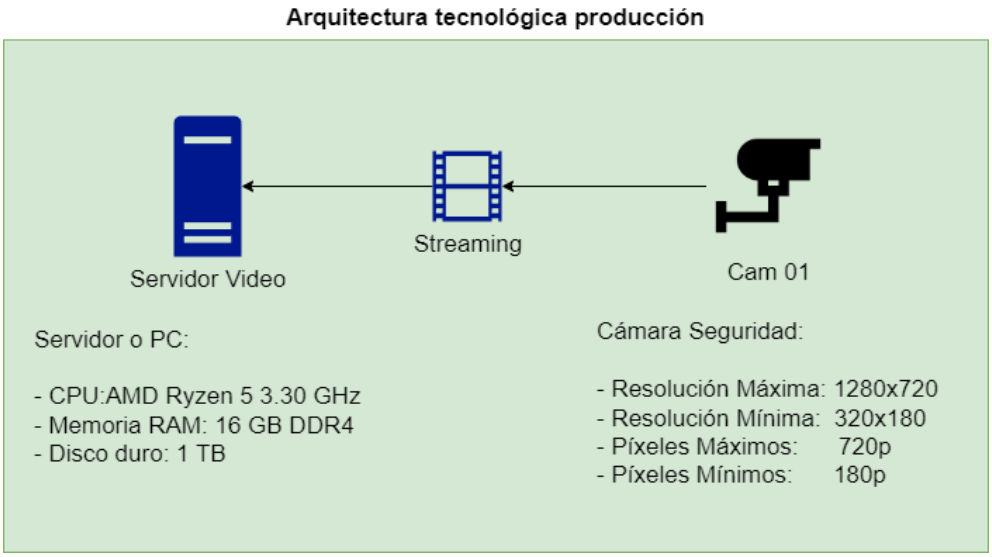

Figura 14: Arquitectura tecnológica en producción

Anexo 13: Arquitectura Tecnológica (Desarrollo)

El ambiente de desarrollo permite realizar modificaciones y pruebas en los algoritmos YOLOv3-spp y la iluminación y difuminación. En caso de no conseguir buenas prestaciones de hardware se puede optar por computo en la nube. Se recomienda Google Colab que ayudó a realizar el entrenamiento para la detección de cuchillos y pistolas.

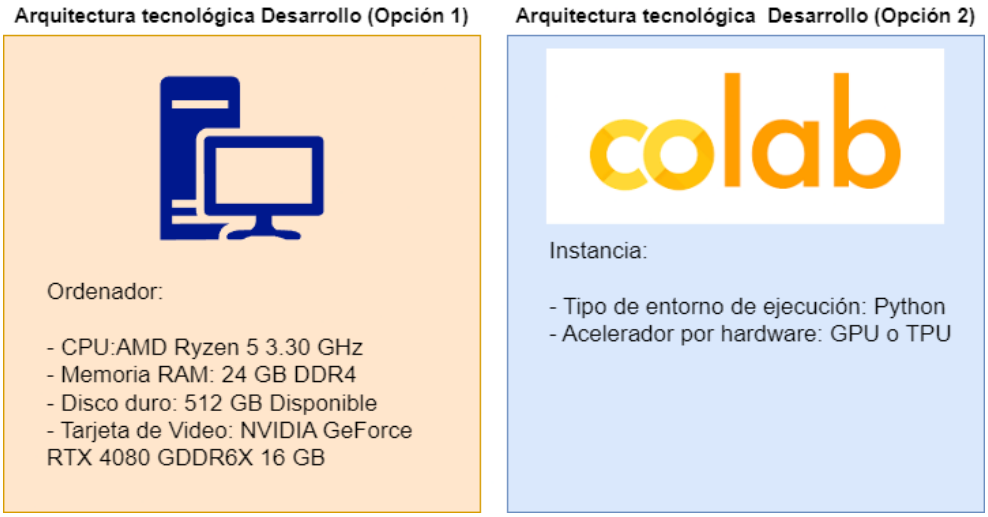

Figura 15: Arquitectura tecnológica de desarrollo

Anexo 14: Instrumento de recolección de datos

El instrumento de recolección de datos es una hoja tabulada con el detalle de los objetos identificados junto con los indicadores para medir el rendimiento del algoritmo YOLOv3-spp.

Clase	VP	VN	FP	FN	Sensibilidad	Especificidad	Precisión	Exactitud	Tiempo promedio de entrenamiento (s)	Tiempo promedio de identificación (ms)
Cuchillo										
Pistola										

Anexo 15: Metodología de desarrollo ágil SCRUM

La metodología utilizada para el desarrollo del sistema de detección de cuchillos y pistolas fue SCRUM. SCRUM fue adaptado para este proyecto de investigación tomando solo las siguientes características: a) Product Backlog y b) Sprint Backlog.

La lista de requerimientos es ingresada al Product Backlog.

Product Backlog

Identificador	Enunciado	Estado	Dimensión / Esfuerzo	Prioridad	Comentarios
1000	Como cliente, necesito mostrar en la aplicación el video de la camara, con la finalidad de visualizar la imagen en la aplicación.	Hecho	3h	MEDIO	
1001	Como un cliente, necesito un botón con la función de "Iniciar video", con la finalidad presentar el video en la app.	Hecho	3h	MEDIO	
1002	Como cliente, necesito un boton con la función de "Parar el video", con la finalidad de dejar de mostrar el video en la app	Hecho	3h	MEDIO	
1003	Como un cliente, necesito deshabilitar el botón "Iniciar video" cuando sea presionado, con la finalidad de causar algún tipo de incidente al iniciar muchas veces la camara.	Hecho	3h	MEDIO	
1004	Como un cliente, necesito deshabilitar el botón "Parar video" cuando sea presionado, con la finalidad de causar algún tipo de incidente al iniciar muchas veces la camara.	Hecho	3h	MEDIO	
1005	Como cliente, necesito habilitar el boton "Iniciar video" cuando "Parar video" este deshabilitado, con la finalidad de volver a visualizar el video.	Hecho	3h	MEDIO	
1006	Como cliente, necesito habilitar el boton "Parar video" cuando "Iniciar video" este deshabilitado, con la finalidad de volver a visualizar el video.	Hecho	3h	MEDIO	
1007	Como cliente, necesito visualizar las detecciones realizadas.	Hecho	24h	ALTO	
1008	Como cliente, necesito configurar un correo dónde me llegará las alertas.	Hecho	24h	ALTO	
1009	Cómo cliente, necesito grabar o tomar un foto del momento de la detección.	Hecho	24h	ALTO	

Figura 16: Lista de requerimientos agregados al Backlog

Del Product Backlog, los requerimientos son pesados y se agrega el tiempo (horas) de desarrollo.

Sprint Backlog

						Semana 1			Semana 2			Semana 3			Semana 4			Semana 4				Total	
Identificador	Enunciado	Tarea	Asignado	Estado	Horas estimadas totales	Cons.	Rest.		Cons.	Rest.		Cons.	Rest.		Cons.	Rest.		Cons.	Rest.			Cons.	Rest.
1000	Como cliente, necesito mostrar en la aplicación el video de la camara, con la finalidad de visualizar la imagen en la aplicación.	Crear proyecto	EQUITOG	HECHO	2	2	0			0			0			0			0			2	0
		Crear label en el Main	EQUITOG	HECHO	1	1	0			0			0			0			0			1	0
1001	Como un cliente, necesito un botón con la función de "Iniciar video", con la finalidad presentar el video en la app.	Agregar el objeto Button	EQUITOG	HECHO	2	2	0			0			0			0			0			2	0
		Agregar personalización	EQUITOG	HECHO	2	2	0			0			0			0			0			2	0
1002	Como cliente, necesito un boton con la función de "Parar el video", con la finalidad de dejar de mostrar el video en la app.	Agregar el objeto Button	EQUITOG	HECHO	1		1		1	0			0			0			0			1	0
		Agregar personalización	EQUITOG	HECHO	1		1		1	0			0			0			0			1	0
		Exportar archivo .ui	EQUITOG	HECHO	1		1		1	0			0			0			0			1	0
1003	Como un cliente, necesito deshabilitar el botón "Iniciar video" cuando sea presionado, con la finalidad de causar	Agregar configuración al boton	EQUITOG	HECHO	3		3		3	0			0			0			0			3	0
1004	Como un cliente, necesito deshabilitar el botón "Parar video" cuando sea presionado, con la finalidad de causar algún tipo de incidente al iniciar muchas veces la camara.	Agregar configuración al boton	EQUITOG	HECHO	2		2			2		2	0		0	0		0	0			2	0
1005	Como cliente, necesito habilitar el boton "Iniciar video" cuando "Parar video" este deshabilitado, con la finalidad de volver a visualizar el video.	Agregar configuración al boton	EQUITOG	HECHO	2		2			2		2	0		0	0		0	0			2	0

Figura 17: Lista de tareas en el Sprint Backlog. Parte 1

Sprint Backlog

Identificador	Enunciado	Tarea	Asignado	Estado	Horas estimadas totales	Semana 1		Semana 2		Semana 3		Semana 4		Semana 4		Total	
						Cons.	Rest.	Cons.	Rest.	Cons.	Rest.	Cons.	Rest.	Cons.	Rest.	Cons.	Rest.
1006	Como cliente, necesito un boton con la función de "Parar el video", con la Como cliente, necesito habilitar el boton "Parar video" cuando "Iniciar video" este deshabilitado, con la finalidad de volver a visualizar el video.	Agregar configuración al boton	EQUITOG	HECHO	2		2		2	2	0	0	0	0	0	2	0
1007	Como cliente, necesito visualizar las detecciones realizadas.	Añadir el modelo entrenado al sistema de detección	EQUITOG	HECHO	24		24		24		24	18	6			18	6
1008	Como cliente, necesito configurar un correo dónde me llegará las alertas.	Agregar funcionalidad para añadir el correo electrónico	EQUITOG	HECHO	24		24		24		24	18	6			18	6
1009	Cómo cliente, necesito grabar o tomar una foto del momento de la detección.	Agregar la funcionalidad para grabar o tomar una foto al sistema	EQUITOG	HECHO	24		24		24		24			21	3	21	3

Figura 18: Lista de tareas en el Sprint Backlog. Parte 2

Anexo 16: Reporte del tiempo de detección del modelo entrenado con el conjunto de datos con los algoritmos de iluminación y difuminación

El reporte es extraído usando el modelo entrenado con imágenes de iluminación y difuminación de OpenCV. Se utilizan 765 imágenes de la división del conjunto de datos Test.

	A	B	C	D	E
1	Deteccion	Imagen	tiempo_inferen	tiempo inferencia (ms)	Objeto
2	image 1/1: 682x1024 (no detections)	Speed: 12.8ms pre-process	75.0ms inference	75	Sin detección
3	image 1/1: 680x1024 2 Cuchillos	Speed: 8.5ms pre-process	25.6ms inference	25,6	Cuchillo
4	image 1/1: 683x1024 (no detections)	Speed: 6.8ms pre-process	25.6ms inference	25,6	Sin detección
5	image 1/1: 768x1024 1 Cuchillo	Speed: 6.0ms pre-process	90.6ms inference	90,6	Cuchillo
6	image 1/1: 683x1024 2 Cuchillos	Speed: 7.0ms pre-process	25.0ms inference	25	Cuchillo
7	image 1/1: 1024x1024 1 Cuchillo	Speed: 8.5ms pre-process	91.3ms inference	91,3	Cuchillo
8	image 1/1: 768x1024 1 Cuchillo	Speed: 6.5ms pre-process	26.0ms inference	26	Cuchillo
9	image 1/1: 1024x768 1 Cuchillo	Speed: 5.5ms pre-process	77.6ms inference	77,6	Cuchillo
10	image 1/1: 768x768 1 Cuchillo	Speed: 6.0ms pre-process	28.6ms inference	28,6	Cuchillo
11	image 1/1: 768x1024 1 Cuchillo	Speed: 8.4ms pre-process	23.8ms inference	23,8	Cuchillo
12	image 1/1: 1024x683 (no detections)	Speed: 5.1ms pre-process	76.0ms inference	76	Sin detección
13	image 1/1: 1024x1024 1 Cuchillo	Speed: 9.5ms pre-process	27.4ms inference	27,4	Cuchillo
14	image 1/1: 768x1024 1 Cuchillo	Speed: 6.8ms pre-process	23.8ms inference	23,8	Cuchillo
15	image 1/1: 371x1024 1 Cuchillo	Speed: 4.0ms pre-process	102.6ms inference	102,6	Cuchillo
16	image 1/1: 1024x1024 (no detections)	Speed: 8.9ms pre-process	27.9ms inference	27,9	Sin detección
17	image 1/1: 681x1024 1 Cuchillo	Speed: 7.6ms pre-process	21.4ms inference	21,4	Cuchillo
18	image 1/1: 1024x819 2 Cuchillos	Speed: 6.6ms pre-process	92.6ms inference	92,6	Cuchillo
19	image 1/1: 768x1024 2 Cuchillos	Speed: 7.6ms pre-process	22.5ms inference	22,5	Cuchillo
20	image 1/1: 683x1024 1 Cuchillo	Speed: 5.7ms pre-process	21.6ms inference	21,6	Cuchillo
21	image 1/1: 768x1024 1 Cuchillo	Speed: 5.0ms pre-process	22.0ms inference	22	Cuchillo
22	image 1/1: 1024x684 1 Cuchillo	Speed: 5.5ms pre-process	21.1ms inference	21,1	Cuchillo
23	image 1/1: 768x1024 1 Cuchillo	Speed: 5.3ms pre-process	21.2ms inference	21,2	Cuchillo
24	image 1/1: 593x1024 1 Cuchillo	Speed: 5.4ms pre-process	82.1ms inference	82,1	Cuchillo
25	image 1/1: 680x1024 1 Cuchillo	Speed: 5.6ms pre-process	20.4ms inference	20,4	Cuchillo
26	image 1/1: 682x1024 1 Cuchillo	Speed: 6.7ms pre-process	19.1ms inference	19,1	Cuchillo
27	image 1/1: 680x1024 3 Cuchillos	Speed: 5.5ms pre-process	19.4ms inference	19,4	Cuchillo
28	image 1/1: 683x1024 1 Cuchillo	Speed: 6.5ms pre-process	20.1ms inference	20,1	Cuchillo
29	image 1/1: 768x1024 1 Cuchillo	Speed: 6.1ms pre-process	20.7ms inference	20,7	Cuchillo
30	image 1/1: 1024x1024 1 Cuchillo	Speed: 7.5ms pre-process	25.4ms inference	25,4	Cuchillo
31	image 1/1: 768x1024 1 Cuchillo	Speed: 5.7ms pre-process	19.6ms inference	19,6	Cuchillo

Figura 19: Reporte de detecciones con los algoritmos de iluminación y difuminación

Anexo 17: Reporte del tiempo de detección del modelo entrenado con el conjunto de datos sin los algoritmos de iluminación y difuminación

El reporte es extraído usando el modelo entrenado con imágenes del conjunto de datos sin los algoritmos de iluminación y difuminación de OpenCV. Se utilizan 765 imágenes de la división del conjunto de datos Test.

	A	B	C	D	E
1	Deteccion	Imagen	tiempo_inferencia	tiempo inferencia (ms)	Objeto
2	image 1/1: 768x1024 2 Cuchillos	Speed: 15.2ms pre-process	204.7ms inference	204,7	Cuchillo
3	image 1/1: 683x1024 1 Cuchillo	Speed: 6.4ms pre-process	77.2ms inference	77,2	Cuchillo
4	image 1/1: 768x1024 2 Cuchillos	Speed: 16.0ms pre-process	15.6ms inference	15,6	Cuchillo
5	image 1/1: 639x640 1 Pistola	Speed: 1.5ms pre-process	85.1ms inference	85,1	Pistola
6	image 1/1: 412x1024 1 Cuchillo	Speed: 7.7ms pre-process	72.1ms inference	72,1	Cuchillo
7	image 1/1: 768x1024 1 Cuchillo	Speed: 0.0ms pre-process	31.3ms inference	31,3	Cuchillo
8	image 1/1: 639x640 1 Pistola	Speed: 15.6ms pre-process	31.3ms inference	31,3	Pistola
9	image 1/1: 768x1024 1 Cuchillo	Speed: 17.0ms pre-process	25.2ms inference	25,2	Cuchillo
10	image 1/1: 640x640 1 Pistola	Speed: 10.4ms pre-process	29.4ms inference	29,4	Pistola
11	image 1/1: 640x640 1 Pistola	Speed: 6.5ms pre-process	31.5ms inference	31,5	Pistola
12	image 1/1: 640x640 1 Pistola	Speed: 7.9ms pre-process	28.5ms inference	28,5	Pistola
13	image 1/1: 768x1024 2 Cuchillos	Speed: 9.6ms pre-process	20.2ms inference	20,2	Cuchillo
14	image 1/1: 1024x1024 2 Cuchillos	Speed: 14.1ms pre-process	23.1ms inference	23,1	Cuchillo
15	image 1/1: 1024x1024 4 Cuchillos	Speed: 12.5ms pre-process	26.1ms inference	26,1	Cuchillo
16	image 1/1: 639x640 1 Pistola	Speed: 7.9ms pre-process	23.9ms inference	23,9	Pistola
17	image 1/1: 640x640 1 Pistola	Speed: 8.0ms pre-process	25.3ms inference	25,3	Pistola
18	image 1/1: 683x1024 1 Cuchillo	Speed: 7.6ms pre-process	21.4ms inference	21,4	Cuchillo
19	image 1/1: 1024x870 1 Cuchillo	Speed: 4.1ms pre-process	81.8ms inference	81,8	Cuchillo
20	image 1/1: 487x1024 2 Cuchillos	Speed: 7.4ms pre-process	62.5ms inference	62,5	Cuchillo
21	image 1/1: 683x1024 1 Cuchillo	Speed: 2.8ms pre-process	27.7ms inference	27,7	Cuchillo
22	image 1/1: 640x640 2 Pistolas	Speed: 9.8ms pre-process	23.9ms inference	23,9	Pistola
23	image 1/1: 640x640 1 Pistola	Speed: 6.8ms pre-process	25.8ms inference	25,8	Pistola
24	image 1/1: 1024x1024 1 Cuchillo	Speed: 12.8ms pre-process	24.0ms inference	24	Cuchillo
25	image 1/1: 1024x768 2 Cuchillos	Speed: 5.8ms pre-process	65.2ms inference	65,2	Cuchillo
26	image 1/1: 768x1024 2 Cuchillos	Speed: 6.8ms pre-process	20.8ms inference	20,8	Cuchillo
27	image 1/1: 678x1024 1 Cuchillo	Speed: 7.3ms pre-process	18.9ms inference	18,9	Cuchillo
28	image 1/1: 685x1024 1 Cuchillo	Speed: 8.7ms pre-process	21.1ms inference	21,1	Cuchillo
29	image 1/1: 768x1024 1 Cuchillo	Speed: 12.1ms pre-process	18.8ms inference	18,8	Cuchillo
30	image 1/1: 683x1024 1 Cuchillo	Speed: 7.8ms pre-process	10.3ms inference	10,3	Cuchillo
31	image 1/1: 768x1024 1 Cuchillo	Speed: 11.5ms pre-process	21.2ms inference	21,2	Cuchillo
32	image 1/1: 768x1024 1 Cuchillo	Speed: 8.6ms pre-process	21.5ms inference	21,5	Cuchillo
33	image 1/1: 436x1024 5 Cuchillos	Speed: 8.9ms pre-process	13.2ms inference	13,2	Cuchillo
34	image 1/1: 634x1024 3 Cuchillos	Speed: 10.4ms pre-process	69.1ms inference	69,1	Cuchillo

Figura 20: Reporte de detecciones sin los algoritmos de iluminación y difuminación

Anexo 18: Configuración de los Hyperparameters

hyp.scratch-low.yaml

```yaml
lr0: 0.01  # tasa de aprendizaje inicial (SGD=1E-2, Adam=1E-3)
lrf: 0.01  # tasa de aprendizaje final de OneCycleLR (lr0 * lrf)
momentum: 0.937  # SGD impulso/Adam beta1
weight_decay: 0.0005  # optimizador peso decaimiento 5e-4
warmup_epochs: 3.0  # épocas de calentamiento (fracciones ok)
warmup_momentum: 0.8  # impulso inicial de calentamiento
warmup_bias_lr: 0.1  # polarización inicial de calentamiento lr
box: 0.05  # caja pérdida ganancia
cls: 0.5  # cls pérdida ganancia
cls_pw: 1.0  # cls BCELoss positivo_peso
obj: 1.0  # obj pérdida ganancia (escala con píxeles)
obj_pw: 1.0  # obj BCELoss positivo_peso
iou_t: 0.20  # Umbral de entrenamiento IoU
anchor_t: 4.0  # ancla-umbral múltiple
# anchors: 3  # anclas por capa de salida (0 para ignorar)
fl_gamma: 0.0  # gamma de pérdida focal (eficiente gamma predeterminado = 1.5)
hsv_h: 0.015  # aumento de imagen HSV-Hue (fracción)
hsv_s: 0.7  # imagen HSV-Aumento de saturación (fracción)
hsv_v: 0.4  # aumento de valor de HSV de imagen (fracción)
degrees: 0.0  # rotación de imagen (+/- grados)
translate: 0.1  # traducción de imagen (+/- fracción)
scale: 0.5  # escala de imagen (+/- ganancia)
shear: 0.0  # corte de imagen (+/- grados)
perspective: 0.0  # perspectiva de imagen (+/- fracción), rango 0-0.001
flipud: 0.0  # image flip up-down (probabilidad)
fliplr: 0.5  # image flip izquierda-derecha (probabilidad)
mosaic: 1.0  # mosaico de imagen (probabilidad)
mixup: 0.0  # confusión de imágenes (probabilidad)
copy_paste: 0.0  # segmento copiar y pegar (probabilidad)
```

Figura 21: Imagen del archivo con los hyperparameters

Anexo 19: Manual de usuario del sistema de detección de cuchillos y pistolas

Se realiza la documentación del manual de usuario del sistema de detección de cuchillos y pistolas con los algoritmos YOLOv3-spp y la iluminación y la difuminación de OpenCV.

Requisitos del sistema

Especificaciones técnicas:

- Windows 11 Home Single Language
- AMD Ryzen 5 5600H with Radeon 3.30 GHz
- RAM 16.0 GB
- HDD o SATA 512 GB
- NVIDIA GeForce RTX 3060
- Cámara Web o Cámara de Seguridad IP 720p.

Software

- Python 3.9
- VSCode
- PyQt5
- Pytorch
- CUDA 11 o superior
- CDNN 8 o superior

Interfaz de usuario

Descripción de los elementos de la interfaz

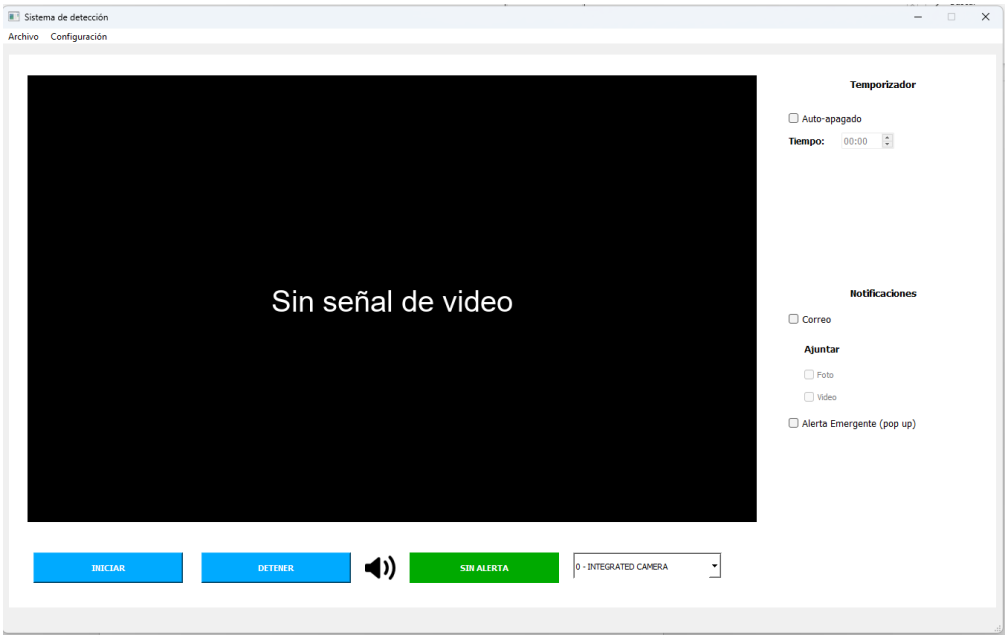

Figura 22: Pantalla principal del sistema

La pantalla principal del sistema de detección de cuchillos y pistolas muestra las opciones para iniciar con la interacción. En la pantalla principal están los botones para iniciar, pausar, silenciar, mostrar la alerta y seleccionar el dispositivo del video. Por otro lado, están las funciones de envío de los objetos detectados. En la parte superior está el menú de configuraciones. En la configuración se puede ingresar un servidor de correo para el envío de alertas mediante correo electrónico. Solo se puede configurar un servidor de correo.

Navegación por los menús y opciones

En la pantalla de video, se mostrará las imágenes de la cámara web o del dispositivo de video configurado en el ordenador.

Figura 23: Pantalla de video

El botón de “INICIAR” cargará el modelo entrenado con la detección de cuchillos y pistolas y luego inicializará la captura de video del dispositivo seleccionado.

Figura 24: Botón INICIAR del sistema de detección

El botón de “DETENER” envía una señal para detener la captura de video del dispositivo.

Figura 25: Botón DETENER del sistema de detección

El botón de “silenciar” le indica al sistema que no emita el sonido de alerta cuando se detecte el objeto.

Figura 26: Botón SILENCIAR del sistema de detección

El indicador de alertas muestra el mensaje “SIN ALERTA” cuando no detecte objeto alguno y muestra “CON ALERTA” y un fondo rojo cuando detecte un objeto.

Figura 27: Indicador de alerta del sistema de detección

La opción de “seleccionar” muestra una lista de dispositivos que se encuentren configurados en el ordenador.

Figura 28: Seleccionar dispositivos en el sistema de detección

La opción de “Correo” permite habilitar o deshabilitar la alerta por correo electrónico. Además, al habilitar la opción de “Correo” tendrá la posibilidad de indicar al sistema si quiere que se envíe una foto o vídeo de los últimos minutos del objeto detectado.

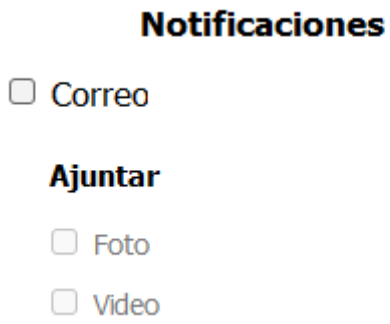

Figura 29: Opciones de envío de alertas en el sistema de detección

En la parte superior, en la barra de “menús”, se encuentra un botón “Configuración”. Como sub-opción, está la configuración del servidor de correo electrónico.

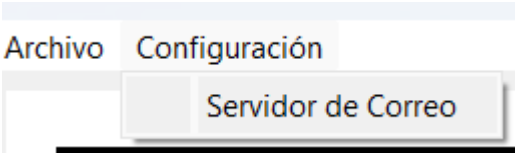

Figura 30: Bar del menú para la configuración de correo electrónico en el sistema de detección

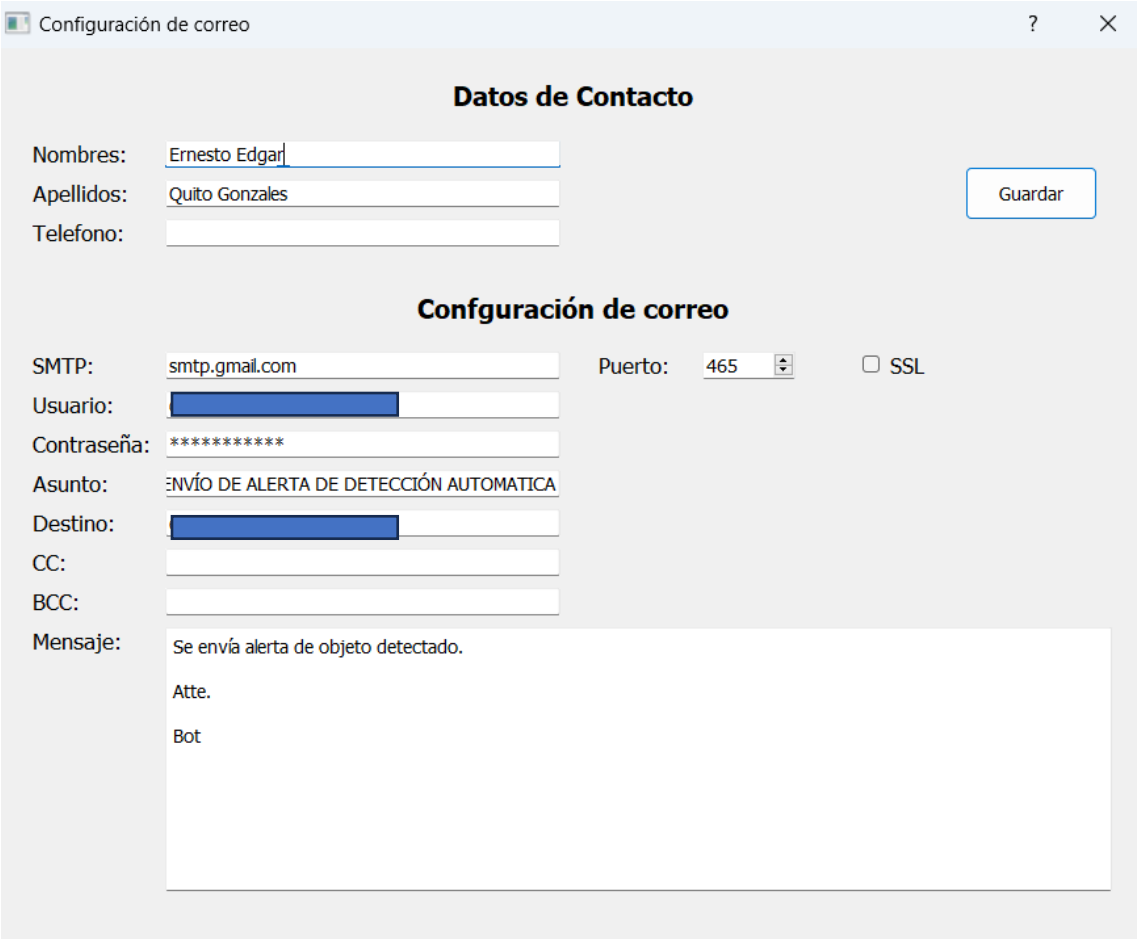

Figura 31: Configuración del servidor de correo electrónico en el sistema de detección

UNIVERSIDAD CÉSAR VALLEJO

FACULTAD DE INGENIERÍA Y ARQUITECTURA

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

Declaratoria de Autenticidad del Asesor

Yo, ALFARO PAREDES EMIGDIO ANTONIO, docente de la FACULTAD DE INGENIERÍA Y ARQUITECTURA de la escuela profesional de INGENIERÍA DE SISTEMAS de la UNIVERSIDAD CÉSAR VALLEJO SAC - LIMA ESTE, asesor de Tesis titulada: "SISTEMA DE DETECCIÓN DE CUCHILLOS Y PISTOLAS CON LOS ALGORITMOS YOLOV3-SPP Y LA ILUMINACIÓN Y LA DIFUMINACIÓN DE OPENCV", cuyo autor es QUITO GONZALES ERNESTO EDGAR, constato que la investigación tiene un índice de similitud de 12.00%, verificable en el reporte de originalidad del programa Turnitin, el cual ha sido realizado sin filtros, ni exclusiones.

He revisado dicho reporte y concluyo que cada una de las coincidencias detectadas no constituyen plagio. A mi leal saber y entender la Tesis cumple con todas las normas para el uso de citas y referencias establecidas por la Universidad César Vallejo.

En tal sentido, asumo la responsabilidad que corresponda ante cualquier falsedad, ocultamiento u omisión tanto de los documentos como de información aportada, por lo cual me someto a lo dispuesto en las normas académicas vigentes de la Universidad César Vallejo.

LIMA, 21 de Julio del 2023

Apellidos y Nombres del Asesor:	**Firma**
ALFARO PAREDES EMIGDIO ANTONIO **DNI:** 10288238 **ORCID:** 0000-0002-0309-9195	Firmado electrónicamente por: EALFAROP el 21-07-2023 22:26:11

Código documento Trilce: TRI - 0608532

INVESTIGA UCV