Universidad Nacional de Loja

Facultad de la Energía, las Industrias y los Recursos Naturales no Renovables

Carrera de Ingeniería en Electrónica y Telecomunicaciones

Sistema de detección y reconocimiento de intrusos de bajo costo basado en visión artificial, aplicado a sistemas de videovigilancia

Trabajo de Titulación previo a optar por el Título de Ingeniero en Electrónica y Telecomunicaciones

AUTOR:

Jorge Enrique Ortega Jaramillo.

DIRECTOR:

Ing. Luis Eduardo Rodríguez Montoya, Mg. Sc.

LOJA – ECUADOR

2022

Certificación

Loja, 24 de marzo del 2022

Ing. Luis Eduardo Rodríguez Montoya, Mg. Sc.
DIRECTOR DEL TRABAJO DE TITULACIÓN

CERTIFICO:

Que he revisado y orientado todo proceso de la elaboración del trabajo de titulación: **"Sistema de detección y reconocimiento de intrusos de bajo costo basado en visión artificial, aplicado a sistemas de videovigilancia"**, de autoría del estudiante: **Jorge Enrique Ortega Jaramillo,** previo a la obtención del título de **Ingeniero en Electrónica y Telecomunicaciones**, una vez que el trabajo cumple con todos los requisitos exigidos por la Universidad Nacional de Loja para el efecto, autorizo la presentación para la respectiva sustentación y defensa.

Firmado electrónicamente por:
LUIS EDUARDO RODRIGUEZ MONTOYA

Ing. Luis Eduardo Rodríguez Montoya, Mg. Sc.
DIRECTOR DEL TRABAJO DE TITULACIÓN

Autoría

Yo, **Jorge Enrique Ortega Jaramillo**, declaro ser autor del presente trabajo de titulación y eximo expresamente a la Universidad Nacional de Loja y a sus representantes jurídicos de posibles reclamos y acciones legales, por el contenido del mismo. Adicionalmente acepto y autorizo a la Universidad Nacional de Loja la publicación de mi trabajo de titulación en el Repositorio Digital Institucional – Biblioteca Virtual.

Firma: Firmado electrónicamente por: JORGE ENRIQUE ORTEGA JARAMILLO

Cédula de Identidad: 1105335002

Fecha: 14/07/2022

Correo electrónico: jorge.ortega@unl.edu.ec

Teléfono o Celular: 0979090127

Carta de autorización

Carta de autorización del trabajo de titulación por parte del autor para la consulta de reproducción parcial o total, y publicación electrónica del texto completo.

Yo, **Jorge Enrique Ortega Jaramillo**, declaro ser autor del trabajo de titulación: **Sistema de detección y reconocimiento de intrusos de bajo costo basado en visión artificial, aplicado a sistemas de videovigilancia** como requisito para optar el título de **Ingeniero en Electrónica y Telecomunicaciones**, autorizo al sistema Bibliotecario de la Universidad Nacional de Loja para que con fines académicos muestre la producción intelectual de la Universidad, a través de la visibilidad de su contenido de la siguiente manera en el Repositorio Digital Institucional.

Los usuarios pueden consultar el contenido de este trabajo en el Repositorio Institucional, en las redes de información del país y del exterior, con las cuales tenga convenio la Universidad.

La Universidad Nacional de Loja, no se responsabiliza por el plagio o copia del trabajo de titulación que realice un tercero.

Para constancia de esta autorización, en la ciudad de Loja, a los 14 días del mes de julio del dos mil veintidós.

Firma: Firmado electrónicamente por: JORGE ENRIQUE ORTEGA JARAMILLO

Autor: Jorge Enrique Ortega Jaramillo
Cédula: 1105335002
Dirección: Calles Sucre y Saraguro
Celular: 0979090127

DATOS COMPLEMENTARIOS:
Director de Trabajo de Titulación: Ing. Luis Eduardo Rodríguez Montoya, Mg. Sc.

Dedicatoria

A mis padres, María Jaramillo y Jorge Ortega; a mis hermanos, César, Juan y Doménica; y, a mi abuela, María Vallejo, por su apoyo incondicional a lo largo de toda mi vida y por ser la principal inspiración para seguir adelante cumpliendo cada una de las metas propuestas.

Jorge Enrique Ortega Jaramillo

Agradecimiento

A mis padres, hermanos y abuela, por todo el apoyo y cariño brindado.

Al Ing. Luis Eduardo Rodríguez Montoya, Mg. Sc., quien con su guía fue posible el desarrollo y finalización del proyecto.

De igual forma, quiero agradecer a mis amigos y compañeros de carrera, por toda la ayuda y motivación brindada.

Jorge Enrique Ortega Jaramillo

Índice de Contenidos

Índice de Tablas

Índice de Figuras

Índice de Anexos

1. Título

Sistema de detección y reconocimiento de intrusos de bajo costo basado en visión artificial, aplicado a sistemas de videovigilancia.

2. Resumen

El presente proyecto de titulación versa sobre la construcción de un sistema de detección de intrusos basado en visión artificial, aplicado a sistemas de video vigilancia, encaminado a solucionar las limitaciones que poseen los sensores de movimiento y/o de presencia, además de establecer una solución versátil, eficiente y de bajo costo.

Para su desarrollo, se empleó una revisión bibliográfica que permita obtener un modelo de detección adecuado, además de adquirir los conocimientos necesarios para establecer los parámetros correctos y generar un software capaz de detectar la presencia humana dado un entorno específico. En consecuencia, se construyeron los algoritmos necesarios para obtener un conjunto de datos que dio paso al entrenamiento del modelo; aplicando criterios que permitan aumentar información de aprendizaje, generando una solución eficiente capaz de diferenciar la forma humana desde un escenario provisto por un sistema de CCTV.

Además, se diseñó una aplicación móvil en conjunto con una API web, para brindar la posibilidad de desplegar la solución a un entorno real; permitiendo a quien lo requiera, un acceso al monitoreo de la situación del entorno vigilado. Se utilizaron WebSockets como el protocolo de transmisión de datos, lo que permitió actualizar en tiempo real la información expuesta en la aplicación utilizando hilos de ejecución única, evitando añadir sobrecarga de procesos en el sistema.

De esta forma, una vez generados los distintos algoritmos para el correcto funcionamiento del sistema, se estableció una prueba de rendimiento en diferentes escenarios de vigilancia para medir su eficacia, lo que demostró que el producto final es adecuado para el despliegue en producción.

Palabras clave: redes neuronales, imágenes digitales, procesamiento digital, servicios web, aplicaciones móviles

2.1. Abstract

This degree project deals with the construction of an intrusion detection system based on artificial vision, applied to video surveillance systems, aimed at solving the limitations of motion and presence sensors, in addition to establishing a versatile, efficient and low-cost solution.

For its development, a bibliographic review was used to obtain an adequate detection model, in addition to acquiring the necessary knowledge to establish the correct parameters and generate a software capable of detecting human presence given a specific environment. Consequently, the necessary algorithms were built to obtain a set of data that gave way to the training of the neural network by applying criteria that allow increasing learning information, generating an efficient solution capable of differentiating the human form from a scenario provided by a CCTV system.

In addition, a mobile application was designed in conjunction with a web API to provide the ability to deploy the solution to a realistic environment allowing anyone who requires its access to monitor the situation of the monitored environment. WebSockets were used as the data transmission protocol, which allowed real-time updating of the information displayed in the application using single execution threads, avoiding adding process overload to the system.

Thus, once the different algorithms were generated for the correct operation of the system, a performance test was established in different monitoring scenarios to measure its effectiveness, which showed that the final product is suitable for production deployment.

Keywords: neural networks, digital images, digital processing, web services, mobile applications

3. Introducción

Durante los últimos años el desarrollo de una inteligencia artificial capaz de replicar los cinco sentidos del humano ha sido de gran interés para científicos e investigadores en el campo de procesamiento digital de señales y ciencias de la computación. Una de las principales ramas que ha tenido un desarrollo considerable, es la visión por computadora, cuyo objetivo es brindar a un sistema informático la capacidad de identificar y comprender automáticamente el mundo visual, dando paso a la toma de decisiones inteligentes (Olveres et al., 2021).

Dados los recientes avances, se ha evidenciado el desarrollo distintos algoritmos y modelos capaces de obtener resultados satisfactorios utilizando pocos recursos computacionales, lo que conlleva a una implementación sin complicaciones en sistemas simples. Es por ello que hoy en día se puede apreciar tecnología con características estándar que incluyen el concepto de visión artificial; desde computadoras personales hasta microcontroladores, cuyo propósito suele ser el apoyo en el desarrollo de tareas en los campos de la ciencia e ingeniería.

En la actualidad, existen una variedad de sistemas en los que es posible implementar este enfoque y poder brindar la capacidad de inferir acciones en base al análisis de imágenes adquiridas; uno de estos son los sistemas de circuito cerrado de televisión (CCTV), cuyo propósito es el de brindar seguridad adicional en un área privada. Estas soluciones por si solas cumplen la principal función de brindar visión en puntos estratégicos en donde se requiera reforzar la seguridad, pero para establecer una respuesta en base a un evento de intrusión, suelen estar acompañados de dispositivos electrónicos, tales como sensores de movimiento o presencia, con el propósito de desplegar una alerta según sea el caso.

Según el tipo de dispositivo que se utilice para la detección de intrusiones, puede dar paso a imprecisiones que suelen materializarse en falsas alarmas que, a la larga, disminuyen la fiabilidad del sistema afectando su desempeño y disminuyendo la confianza del usuario hacia la solución. Como consecuencia de ello, los nuevos sistemas de videovigilancia se han visto en la necesidad de implementar arquitecturas inteligentes que les permitan procesar las imágenes captadas por las cámaras y responder en base al resultado del análisis.

Asimismo, se presenta la necesidad de establecer sistemas robustos de seguridad dados los datos actuales brindados por la Fiscalía General del Estado, en su informe mensual “Ecuador: Las Cifras de Robos” (Dirección de Estadística y Sistemas de Información - FGE, 2021), donde se denota un crecimiento en la tasa de robos a domicilios y unidades económicas,

específicamente en una cantidad de 14,2% y 17,3% respectivamente, en comparación al año 2020.

De la misma forma, otro inconveniente que se suscita es el que muchas de las arquitecturas que incluyen la característica de reconocimiento de intrusos suelen ser exclusivas del sistema en cuestión, obligando al usuario adquirir un nuevo producto y desechar el antiguo implementado, en caso de poseerlo. Es por ello que, en el presente proyecto, se propone un sistema de detección y reconocimiento de intrusos basado en visión artificial en conjunto con herramientas web, con el fin de generar una solución disponible para su implementación en un sistema de videovigilancia.

4. Marco Teórico

4.1. Entorno Nacional de Seguridad

Los actuales índices de seguridad en el Ecuador, denotan un aumento considerable en temas correspondientes a vulneración en domicilios y la propiedad privada, que se derivan en hurtos o pérdidas materiales. Toda esta situación se puede caracterizar a partir de los datos proporcionados en el siguiente punto, y posteriormente, en el análisis de la tecnología destinada al apoyo de la seguridad privada.

4.1.1. Índice de intrusiones y robos suscitados en propiedad privada.

La Fiscalía General del Estado, en su reporte mensual de las estadísticas de robos con corte hasta el 08 de julio de 2021 (Dirección de Estadística y Sistemas de Información - FGE, 2021), expone que, en lo que va del año, se registran 5904 robos a domicilios, lo cual presenta un aumento del 14,2% comparado a la cantidad de 5169 que se presentó el año pasado en el mismo periodo de tiempo, y, además, se registran 3423 robos a unidades económicas, que denota un aumento del 17,3% con respecto al periodo anterior, en donde se registraron 2919 robos.

A partir de la información anterior, se establecen la cantidad de robos que se suscitaron en el periodo Enero – Septiembre 2021, especificados por mes como se observa en la Tabla 1 correspondiente a robos a domicilios y en la Tabla 2 correspondiente a robos a unidades económicas. Estos datos presentan un comportamiento constante, que, si bien presenta una disminución a mitad de año, la tendencia de cometer este tipo de delitos aumenta en periodos importantes, tales como el inicio y fin de año, donde se concentran el mayor número de festividades en el calendario, y por ende un mayor movimiento en la economía del país.

Tabla 1. Número de robos a domicilio durante el periodo enero – septiembre 2021

Mes	Enero	Febrero	Marzo	Abril	Mayo	Junio	Julio	Agosto	Septiembre
Cantidad	681	678	656	592	595	607	709	690	696

Las actividades de intrusión en esta categoría, se registran en una mayor cantidad los fines de semana, específicamente los días sábados por la noche. **Fuente:** Estadísticas FGE, Ecuador: Las cifras de Robos (Dirección de Estadística y Sistemas de Información - FGE, 2021).

Tabla 2. Número de robos a unidades económicas durante el periodo enero – septiembre 2021

Mes	Enero	Febrero	Marzo	Abril	Mayo	Junio	Julio	Agosto	Septiembre
Cantidad	681	678	656	592	595	607	709	690	696

Las actividades de intrusión en esta categoría, se registran en una mayor cantidad los fines de semana, específicamente los días miércoles por la mañana. **Fuente:** Estadísticas FGE, Ecuador: Las cifras de Robos (Dirección de Estadística y Sistemas de Información - FGE, 2021).

Así pues, con la información que se expone en el anterior párrafo, se establece una segmentación de las diferentes etapas del día en los que se suscitan vulneraciones a la propiedad privada, en donde se evidencia que, a partir de la Figura 1, en la noche es donde ocurre la mayor actividad sospechosa en la categoría de robo a domicilios; y, en el caso de los robos a unidades económicas, se presentan en la mañana en su mayoría, tal y como lo indica la Figura 2.

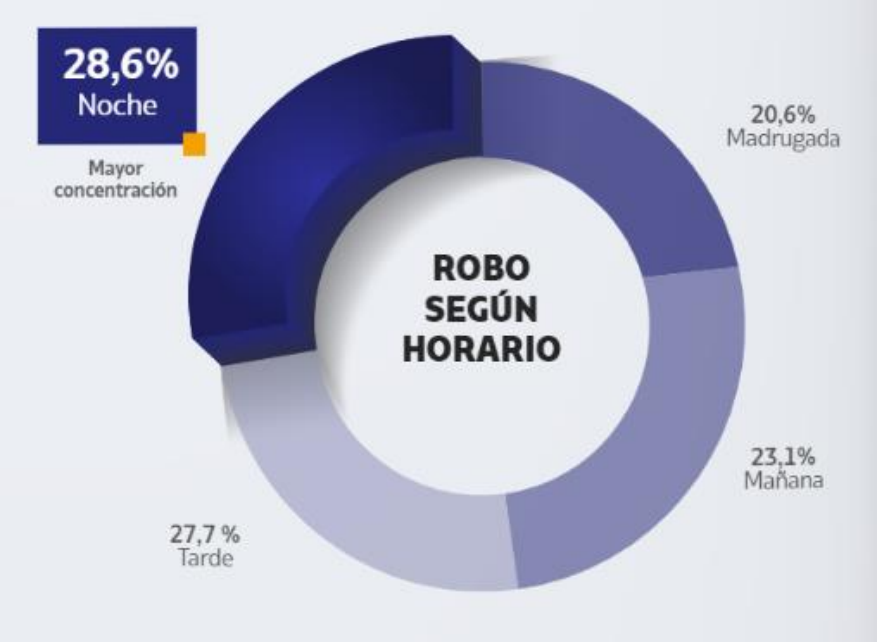


Figura 1. Porcentaje de robos a domicilios según el horario del día
Fuente: Adaptado de Estadísticas FGE, Ecuador: Las cifras de Robos, de Fiscalía General del Estado, 2021, FGE(https://www.fiscalia.gob.ec/estadisticas-de-robos/).

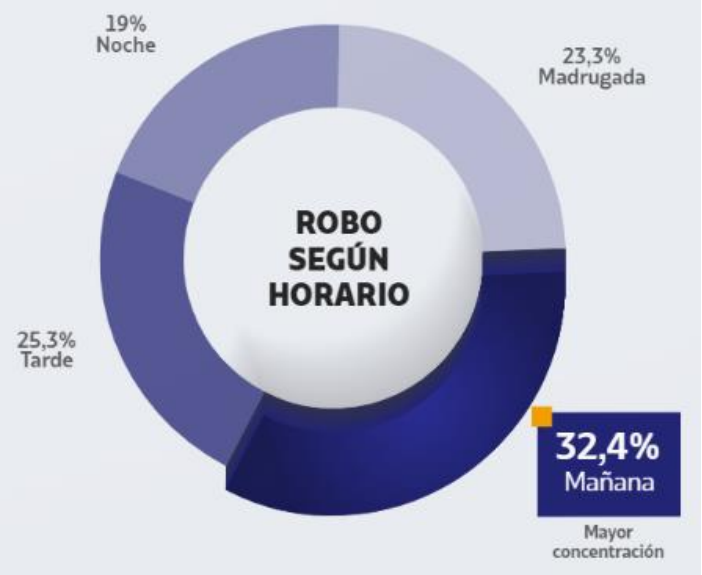


Figura 2. Porcentaje de robos a unidades económicas según el horario del día
Fuente: Adaptado de Estadísticas FGE, Ecuador: Las cifras de Robos, de Fiscalía General del Estado, 2021, FGE(https://www.fiscalia.gob.ec/estadisticas-de-robos/).

4.1.2. Tecnología actual destinada a la seguridad

4.1.2.1. Sensores de movimiento (PIR).

Los principales medios de detección de movimiento que se emplean en los sistemas de videovigilancia, son los llamados sensores PIR. Un sensor infrarrojo pasivo (PIR o Passive Infrared) es un dispositivo que permite captar las señales de calor de objetos que se encuentran en su campo de visión y cuando alcanza un umbral, activará lo que esté conectado al mismo (Vigren, 2020); lo que denota que su funcionamiento se asemeja a un comportamiento con respuesta lógica que, a través de un estímulo, envía una respuesta de uno o cero, expresado en un límite de voltaje de operación.

4.1.2.2. Sistemas de videovigilancia (CCTV).

Un circuito cerrado de televisión es un sistema que involucra un conjunto de cámaras ya sean conectadas por cable o por conexión inalámbrica, derivadas de un concentrador central que integra funciones variadas, utilizado para el monitoreo en tiempo real de un sitio (Khan et al., 2020).

Estos sistemas son esenciales para una variedad de usos públicos y privados, los cuales, combinados con tecnologías de Internet de las cosas (IoT), pueden generar soluciones inteligentes que permitan garantizar la seguridad y protección de una comunidad en específico (Khan et al., 2020).

Figura 3. Esquema de un Circuito Cerrado de Televisión (CCTV)
Fuente: Adaptado de Implementación De Los Sistemas De Detección De Incendio E Intrusión (p. 33), por C. Valbuena, 2012, Universidad Santo Tomás (https://repository.usta.edu.co/bitstream /handle/11634/30567/2012carlosvalbuena.pdf).

4.2. Procesamiento Digital de Imágenes

El procesamiento digital de imágenes se ocupa de la manipulación y transformación de imágenes digitales, obtenidas desde un dispositivo capturador, a través de un computador para un posterior análisis de su resultado (Alvarado Moya, 2012).

Principalmente el procesamiento de una imagen digital se utiliza para la restauración de la misma, siendo que, al transmitir los datos de la imagen a través de un canal, este suele ser afectado por ruido externo adquirido por el mismo canal. Otro de los enfoques principales que se obtiene de esta disciplina, es el manipular la imagen para obtener un resultado deseado, como por ejemplo la manipulación del color a conveniencia, o el ajuste de la iluminación en caso de presentar un escenario con tonos oscuros que dificultan la observación de la información que contienen (Alvarado Moya, 2012).

Dentro del presente proyecto, se propone la utilización de esta disciplina, para aplicar un procesamiento de un conjunto de datos compuesto principalmente por imágenes, con el objetivo de la extracción de características específicas y su manipulación libre, con resultados que brinden variedad en la entrada de un modelo de detección de objectos y así aumentar su

media de precisión promedio aplicado a un conjunto de clases o, en este caso, a una clase específica.

Es por ello que en las secciones a continuación, se detalla brevemente las características fundamentales que permitirán el óptimo desempeño de la solución a generar.

4.2.1. Definición de una imagen digital

Una imagen digital de puede definir matemáticamente como una función bidimensional cuyas coordenadas de valores discretos en el espacio se definen por x y y, dentro de la función $f(x, y)$, que determina los niveles de intensidad y de grises en la imagen en un punto específico. Es decir, la imagen digital está compuesta por un número finito de elementos, llamados píxeles (Mejía Vilet, 2005).

4.2.2. Imágenes en color

A diferencia de las imágenes monocromáticas, cuya composición se basa en un solo canal en donde se describe cada tonalidad de gris y formar la imagen digital; la imagen a color convencional usualmente se compone de 3 canales de información monocromática, en donde cada uno representa un color específico: rojo, verde y azul (Alvarado Moya, 2012).

4.2.3. Histograma

El histograma es un gráfico que contiene información estadística y de frecuencia de un valor o sistema concreto. En el caso del histograma de una imagen, surge a partir de la realización de una normalización de los canales de colores (en el caso de una imagen a color), y así obtener una interpretación monocromática y poder representar la ocurrencia o el nivel de color e información que contiene dicha imagen procesada (Mejía Vilet, 2005).

Los histogramas se suelen utilizar como base de muchas técnicas de procesamiento de imagen en el dominio espacial, dado que logra esbozar una interpretación de toda la información que contiene una imagen, y a partir de la misma, realizar las acciones pertinentes según la aplicación requerida (Alvarado Moya, 2012).

4.2.4. Índice de similitud estructural

El índice de similitud estructural permite medir la semejanza entre dos imágenes, permitiendo obtener, entre otras cosas, la degradación de una imagen que pasa a través de un procesamiento específico o para medir el desempeño del canal por el que se transmite la información (Datta, 2020).

Se basa en evaluar la calidad de la imagen a partir de la cuantificación de los errores que pueden existir en una imagen distorsionada con respecto a una imagen de referencia, a partir de la extracción de 3 características clave, las cuales son la Luminancia, Contraste y la Estructura (Imatest, 2021).

Generalmente los valores que se obtiene a partir del análisis aplicado, está en un rango de -1 a +1 en donde +1 establece que las dos imágenes son idénticas, y el valor de -1 indica que son totalmente diferentes. Frecuentemente, estos valores son normalizados a 0 y 1 para obtener una representación en formato de porcentajes (Datta, 2020).

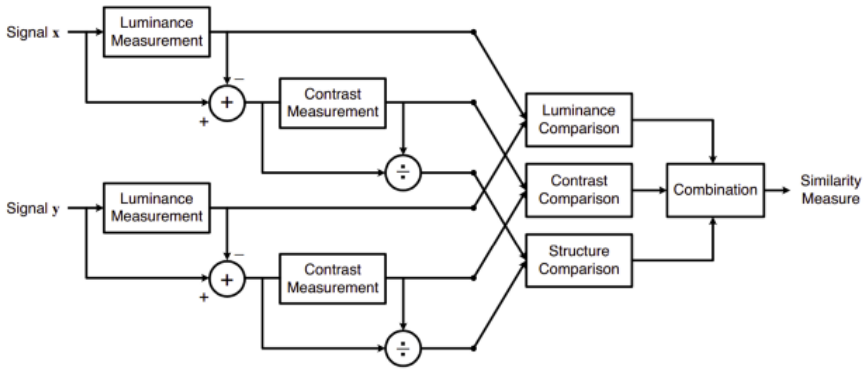


Figura 4. Diagrama de bloques del funcionamiento del SSIM
Fuente: Adaptado de All about Structural Similarity Index (SSIM), por P. Datta, 2020, (https://medium.com/srm-mic/all-about-structural-similarity-index-ssim-theory-code-in-pytorch-6551b455541e).

4.3. Visión Artificial

La visión artificial es una disciplina que se ocupa de problemas como la segmentación detección y reconstrucción de imágenes, el reconocimiento de objetos, etc.; cuyo objetivo es modelar y comprender el mundo visual, extrayendo información útil a partir de imágenes digitales, a menudo inspirado en tareas complejas que realiza la visión humana (Olveres et al., 2021). Generalmente se lo considera una subárea de estudio de la inteligencia artificial y del aprendizaje automático (Machine Learning), debido a que brinda a una máquina la capacidad de dar una respuesta, haciendo uso de un conjunto de datos recopilados a través de un dispositivo que permita adquirir la imagen de cierto escenario en concreto.

La extracción de la información útil se la realiza a partir de la aplicación de distintos filtros y preprocesamiento de los datos de la imagen digital, haciendo posible que un sistema computacional sea capaz de clasificar, segmentar o reconocer una clase dentro de dicha imagen, y proporcionar una acción específica a partir del resultado obtenido.

Para que una computadora adquiera la capacidad de identificar por sí mismo lo que está viendo, es necesario la aplicación de un "aprendizaje" continuo, y esto es posible a partir de la aplicación de las redes neuronales, las cuales están atadas al concepto del Aprendizaje Profundo (de la traducción de Deep Learning), en donde se busca crear un modelo que permita simular el funcionamiento del cerebro.

Con esto en mente, se establecen a continuación, distintos conceptos y descripciones que ayudarán a comprender de mejor manera los términos correspondientes a redes neuronales y los distintos problemas que se busca solucionar, tales como la clasificación, detección y segmentación de clases u objetos dentro de una imagen, además de obtener un mejor entendimiento del porqué de las distintas decisiones tomadas a lo largo del desarrollo del presente proyecto.

4.3.1. Redes neuronales

El concepto de las redes neuronales utilizadas en las ciencias computacionales para producir distintas respuestas y mapas de datos a partir de una entrada específica, se origina a partir de la inspiración biológica y funcional del sistema nervioso humano, en donde su eje central es el cerebro, el cual es representado, valga la redundancia, por una red neuronal que, tal y como lo describe Moreno (2019), "recibe información, la trata y a partir de esta toma determinadas decisiones". Entonces, el cerebro humado se puede modelar como un sistema de tres capas, tal y como se muestra en la Figura 5.

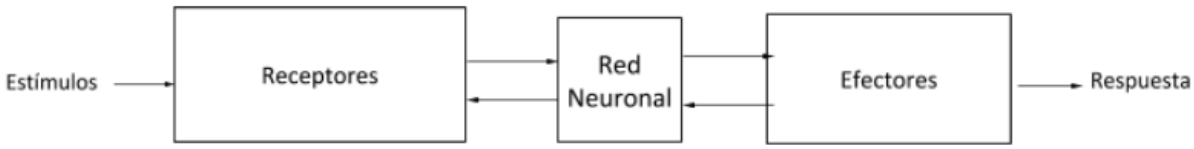


Figura 5. Esquema simple de red neuronal
Fuente: Adaptado de Clasificación de imágenes usando redes neuronales convolucionales en Python (p. 23), por A. Moreno, 2019, Universidad de Sevilla (https://idus.us.es/bitstream/handle/11441/89506/TFG-2402-ARTOLA.pdf).

4.3.1.1. Modelo de una neurona artificial. El modelo de una neurona artificial se basa en el funcionamiento de la neurona real, en donde a través de un conjunto de delicadas estructuras llamadas dendritas, recoge señales procedentes de otras neuronas, por medio de impulsos de actividad eléctrica (Moreno, 2019). Siguiendo este concepto se establece una representación del modelo genérico de la neurona artificial, en donde se puede apreciar en la Figura 6 las siguientes características (Moreno, 2019):

- Un conjunto de sinapsis o conectores, los cuales toman un conjunto de valores, positivos o negativos,
- Una función de red, encargada de ponderar las señales entrantes,
- Una función de activación, para limitar la amplitud de salida de una neurona, y, por último;
- La salida, que se calcula en base al valor resultante de la función de activación.

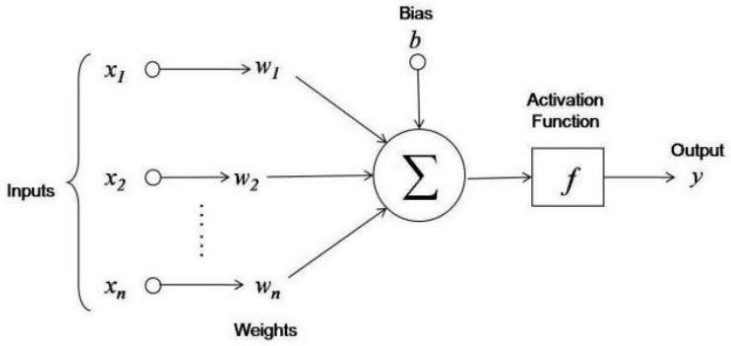


Figura 6. Modelo genérico de una neurona artificial
Fuente: Adaptado de Clasificación de imágenes usando redes neuronales convolucionales en Python (p. 24), por A. Moreno, 2019, Universidad de Sevilla (https://idus.us.es/bitstream/handle/11441/89506/TFG-2402-ARTOLA.pdf).

4.3.2. Aprendizaje de una red neuronal artificial

Para que una red neuronal artificial tenga la capacidad de resolver un determinado problema sometido a distintos entornos, debe realizarse un entrenamiento previo de la misma en base de alimentarlo con un conjunto de datos variado para obtener resultados satisfactorios.

Tal como lo menciona (Isasi Viñuela & León Galván, 2004), las redes neuronales artificiales son sistemas de aprendizaje basados en ejemplos, de los cuales, se debe establecer un conjunto de datos que debe ser significativo, es decir, con un número suficiente de ejemplos; y representativo, lo cual denota que las muestras deben ser de varios tipos. Con estas características, la información que se tome como entrada de la red neuronal en el proceso de aprendizaje, podrá brindar una capacitación adecuada a la red, obteniendo como resultado, una resolución de problemas eficiente.

Así mismo, dependiendo de la estructura del aprendizaje a la cual va a ser sometida la red y del problema a resolver, se distinguen tres tipos de esquemas (Isasi Viñuela & León Galván, 2004):

4.3.2.1. Aprendizaje supervisado. Esquema en el cual, los datos del conjunto de aprendizaje contienen atributos y cierta información concerniente a la solución del

problema. En síntesis, existe un agente externo que determina si la salida es la adecuada en base a una entrada específica, y de no ser así, modifica los pesos resultantes para que la red se adapte y pueda resolver un problema establecido de forma eficiente.

4.3.2.2. Aprendizaje no supervisado. En este esquema el conjunto de datos solamente presenta información de los ejemplos, y no hay nada que permita guiar el proceso de aprendizaje, por lo cual, la propia red será la encargada de modificar los pesos a partir de la información proporcionada, extrayendo información y rasgos característicos del conjunto de entrenamiento.

4.3.2.3. Aprendizaje por refuerzo. Se presenta como variante del aprendizaje no supervisado, con la diferencia de que no dispone de información concreta que permita determinar el error que ha cometido en procesamiento del conjunto de entrenamiento, la propia red determina si el patrón producido es el adecuado o no.

4.3.3. Reconocimiento de imágenes

Dentro del reconocimiento de imágenes existen tres tareas fundamentales que se abordan dependiendo del problema a resolver y la aplicación que brindarán los sistemas finales que se basan en cualquiera de estas tareas. Estas se presentan a continuación como:

4.3.3.1. Clasificación de imágenes. En este tipo, se intenta comprender el contenido de una imagen como un todo, etiquetándola en una clase específica y permitiendo ordenar el resultado en una sola categoría. Por lo general, la entrada es una imagen que contiene uno o varios objetos de una misma clase.

4.3.3.2. Reconocimiento de objetos. En contraste con la clasificación de imágenes, en este tipo de tarea se busca clasificar y encontrar la ubicación de una o varias clases en una misma imagen, lo que permite establecer etiquetas de los distintos objetos y obtener información relevante para la aplicación de acciones pertinentes al problema que se intenta resolver. Por lo general este tipo de tareas se las utiliza en detecciones en tiempo real, lo que lo convierte en el componente ideal para el desarrollo del presente trabajo.

4.3.3.3. Segmentación de imágenes. A diferencia de las dos tareas mencionadas anteriormente, esta se centra en clasificar y reconocer cada pixel de uno o varios objetos pertenecientes a una o varias clases dentro de una misma imagen, segmentando así el área perteneciente al resultado esperado de forma precisa, permitiendo obtener mucha más información, tanto de su ubicación como de la forma que preside. Sus aplicaciones pueden

estar enfocada a la detección y reconocimiento de una acción específica de una persona, por ejemplo.

4.4. Algoritmos/Modelos de Reconocimiento Desarrollados

Actualmente existen distintos modelos de reconocimiento de imágenes, que se enfocan de distinta manera, en clasificar, reconocer o segmentar una imagen a través del uso de redes neuronales convolucionales, lo que ciertamente permite obtener un sistema capaz de responder y brindar información, según la imagen de entrada con la que se alimente y el problema a resolver.

A lo largo de esta subsección, se presentarán las distintas características únicas de cada uno de los modelos, y su desarrollo a lo largo del tiempo, siendo así, que se podrá obtener un entendimiento de la situación de cada uno y aportar con una perspectiva adecuada para la elección del modelo a aplicar en la solución del presente trabajo.

4.4.1. Parámetro de Precisión Media.

El promedio de parámetros de precisión media (en inglés Mean Average Precision – mAP) proporciona una medida de calidad en todos los niveles de funcionamiento en la clasificación y detección de clases en una imagen o video, el cuál suele mostrarse como una curva que indica el progreso de la eficiencia del modelo (Yohanandan, 2020).

Muchos algoritmos de detección, tales como Faster R-CNN, MobileNet SSD y YOLO, usan el parámetro mAP para evaluar sus modelos para publicar sus investigaciones. Generalmente se calcula a partir de la cuantización de las predicciones realmente correctas que hizo el modelo elegido, versus la cantidad de falsos positivos (Yohanandan, 2020).

4.4.2. Extracción de fondo con K Nearest Neightbor.

La extracción de fondo en una imagen, generalmente es utilizado para detectar cambios y movimiento dentro de secuencias de vídeo, con el fin de procesar dicho cambio y obtener una respuesta concreta según sea el caso (Trnovszký et al., 2017).

El método de extracción de fondo K Nearest Neightbot (KNN), está basado en Modelos de Mezcla Gaussianos, el cual generalmente es utilizado para extraer una máscara de un objeto detectado, en base a un cambio generado en el ambiente una vez que se ha establecido un modelo de fondo (Lendave, 2021).

Es utilizado principalmente, como base, en la detección y clasificación de objetos de imágenes obtenidas por una cámara estática colocada en un ambiente. Se lo considera un

algoritmo supervisado de Machine Learning, debido a que no existe un aprendizaje en sí; al contrario, se basa en un valor límite para decidir si los pixeles de una imagen coinciden con el modelo de fondo generado, y, en consecuencia generar una máscara para aislar el objeto que causó el movimiento (Trnovszký et al., 2017).

4.4.3. Support Vector Machine SVM

La Máquina de Vector Soporte (Support Vector Machine – denominado en adelante como SVM), es un método efectivo para construir un clasificador, que permite predecir la clase del objeto que se está analizando en cuestión, a partir de unos datos de entrada. Todo esto logra realizarlo a través de cálculos de regresión lineal y representación de los datos en hiperplanos para su respectivo análisis (Nalepa & Kawulok, 2019).

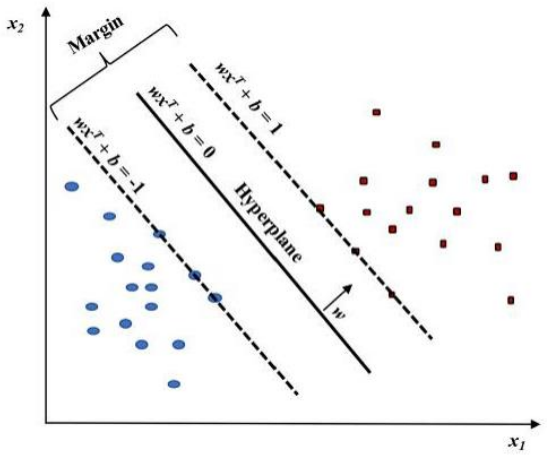


Figura 7. Modelo linear SVM. Clasificación de dos clases a través del SVM
Fuente: Adaptado de Applications of support vector machine (SVM) learning in cancer genomics, por S. Huang, 2018, Cancer Genomics Proteomics.

Las SVM es un modelo que aplica el aprendizaje supervisado, cuya importancia ha sido relevante a lo largo de su desarrollo, el cual, en sus inicios fue pensado como un clasificador binario, que ha evolucionado y ahora se concibe como un clasificador multiclase, logrando resultados eficientes en la práctica y aplicaciones de la vida real (Guenther & Schonlau, 2016).

Los inconvenientes con este modelo es las limitaciones que imponen el tamaño de los datos que se requiere y se ingresan como entrada en el proceso de entrenamiento y los largos tiempos que requiere para una eficiencia óptima, lo cual presenta una desventaja en el uso de recursos tales como la memoria y procesamiento (Nalepa & Kawulok, 2019).

En aplicaciones que requieren un análisis de tiempo real se limita por las condiciones mencionadas en el anterior párrafo. Sin embargo, las ventajas que brinda han consolidado su uso en diferentes modelos como un soporte, tal y como se expone en el trabajo de investigación

(Lili Zhu, Petros Spachos, 2021) en donde se lo utiliza como una capa de procesamiento en la clasificación y verificación de calidad de productos comestibles, tal como el banano, y en donde aporta con la extracción eficaz de las características para servir como apoyo al algoritmo YOLO y obtener una eficiencia superior a la estándar.

Dentro de otras aplicaciones, se encuentra el uso de este algoritmo como una capa del procesamiento de una imagen dentro del modelo SSD (Single Shot Multibox Detector – del cual más adelante se abordará su estudio), para funcionar, una vez más, como soporte para la extracción de características importantes y aportar en el entrenamiento de la red neuronal.

Por las razones mencionadas, es importante tener una idea de su funcionamiento, ya que, si bien no se hace un uso directo del modelo para su aplicación en la detección de intrusos, es imprescindible su uso como soporte en los distintos algoritmos de detección.

4.4.4. Single Shot Multibox Detector

El modelo Single Shot MultiBox Detector (en adelante SSD) consiste en la división de la imagen para formar una cuadrícula, en lugar de utilizar una ventana deslizante, en donde cada celda es responsable de la detección de objetos en la región donde se encuentran (Kumar et al., 2020).

Este método utiliza una única red neuronal para analizar el mapa de características y generar resultados a partir de discretizar el espacio de salida de los cuadros delimitadores en un conjunto de cuadros predeterminados en diferentes proporciones y escalas. De esta forma, se obtienen predicciones en base a puntuaciones generadas según la clase del objeto a la que pertenezca, y una ubicación precisa del mismo dentro de la imagen, encerrado en un cuadro ajustado a las dimensiones de dicho objeto (Miao et al., 2019).

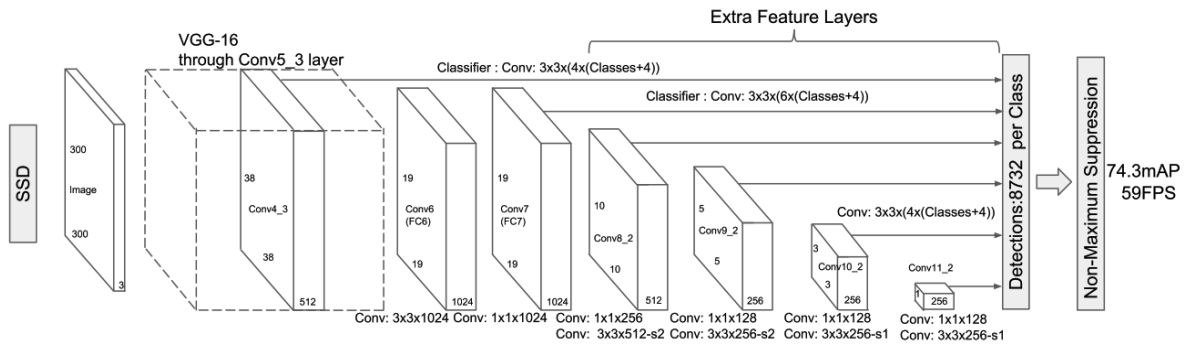


Figura 8. Arquitectura del modelo SSD
Fuente: Adaptado de Insulator detection in aerial images for transmission line inspection using single shot multibox detector, por X. Miao et al, 2016, IEEE Access.

El modelo presenta tres fases primordiales para la detección de objetos en una imagen, cuya organización se presenta en la Figura 8, y se describen de la siguiente forma (Miao et al., 2019):

- **Extracción de características.** Es una pila de redes convolucionales que generan mapas de características que codifican información útil sobre la imagen, en donde también se tiene en énfasis la funcionalidad a partir de varias escalas de codificación para una detección eficaz, independientemente de la imagen que se le brinde como entrada.

- **Cabezales de detección.** Su estructura se define también por redes convolucionales, pero su rol es distinto, dado que su tarea es generar cuadros de predicciones en conjunto con un porcentaje que determina la clase a la que pertenece, más no la de extraer información características de las regiones de la imagen analizada.

- **Supresión no máxima (non maximal supression).** Este paso es requerido dado que, al realizar la detección por las divisiones de la imagen, se pueden suscitar múltiples resultados provocando que se generen múltiples cuadros que encierren a un mismo objeto perteneciente a una misma clase, por lo que es necesario remover dichos cuadros con el fin de obtener un detector con un rendimiento óptimo.

En el caso del rendimiento de este modelo de detección, Miao et al. (2019) lo presentan comparándolo con otros modelos tales como YOLO (cuya naturaleza y funcionamiento es similar al presente modelo descrito), y Faster R-CNN, obteniendo buenos resultados en distintos entornos de prueba. Todo esto se analiza más adelante, con el fin de realizar una selección adecuada para implementar en la solución planteada del presente proyecto de investigación.

4.4.5. Region Proposal Network (RPN)

RPN, es una red totalmente convolucional que predice simultáneamente los límites de los objetos y las puntuaciones de objetividad en cada posición. Permite generar propuestas de posibles objetos dentro de una imagen y, dado su naturaleza, es posible fusionar esta red con algoritmos como Fast R-CNN compartiendo sus características convolucionales, generando resultados altamente confiables (Ren et al., 2017).

RPN tiene un clasificador que permite determinar la probabilidad de que una propuesta del conjunto generado, contenga el objeto señalado a detectar; y un regresor el cual retrocede las coordenadas de las propuestas (Karmarkar, 2018).

RPN se asemeja a los modelos que utilizan el principio de ventana deslizante, dado su funcionamiento principal es el de deslizar una pequeña red de procesamiento a través de todo el mapa de características generado por la última capa convolucional compartida. Todo este proceso lo realiza en el dominio espacial de dicho mapa de características (Ren et al., 2017).

Con la descripción anterior, se pueden resumir los pasos del funcionamiento del modelo, y así presentar un esbozo del proceso de detección que realiza:

- Generar propuestas de detecciones a través de cuadros de ancla, en cuyos límites se encuentra la clase o clases a detectar.
- Clasificar cada uno de los cuadros de anclaje, ya sea en primer plano o en segundo plano.
- Aprender las dimensiones del objeto en cuestión, para ajustar los cuadros y obtener su localización dentro de la imagen.

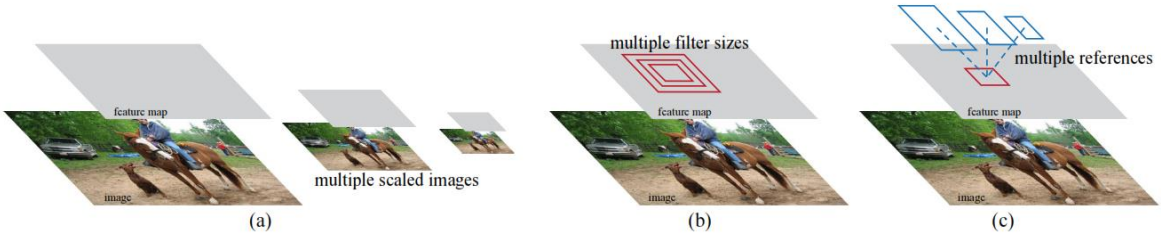


Figura 9. Diferentes esquemas para abordar múltiples escalas y tamaños
Fuente: Adaptado de Faster R-CNN: Towards Real-Time Object Detection with Region Proposal, por S. Ren et al, 2017, IEEE Transactions on Pattern Analysis and Machine Intelligence.

4.4.6. Region Based CNN

R-CNN hace referencia a una Red Neuronal Convolucional basada en Regiones, dado que se basa en seleccionar 2000 regiones a través de una búsqueda selectiva, a partir de una imagen de entrada, en donde se extraen mapas de características para detectar objetos y/o realizar reconocimiento de las clases incluidas en los datos de entrada (Gandhi, 2018).

El mapa de características dentro de este modelo, es extraído a través de una CNN cuyo resultado es útil para alimentar un SVM para clasificar la presencia de objetos dentro las regiones propuestas en la imagen (Gandhi, 2018).

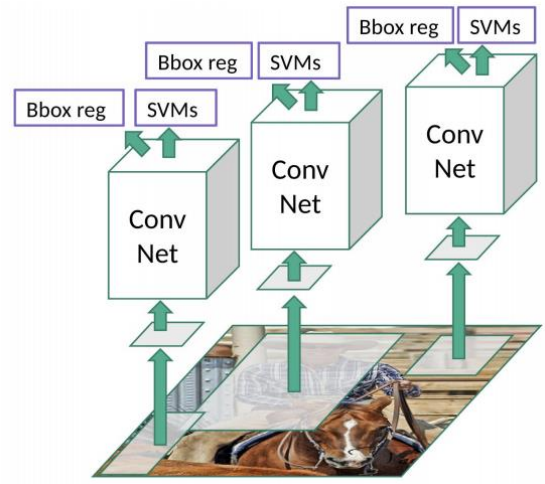


Figura 10. Esquema del funcionamiento del modelo R-CNN
Fuente: Adaptado de R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms, por R. Gandhi, 2018, Towards Data Science (https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e).

4.4.6.1. Problemas con R-CNN

Para Gandhi, (2018), existen varios problemas de implementación y uso con respecto a R-CNN, los cuales se listan a continuación:

- Se necesita una gran cantidad de tiempo para entrenar a la red, dado que requiere procesar 2000 propuestas de región por imagen.
- No es adecuado para aplicaciones en tiempo real dado que toma varias decenas de segundos en procesar cada imagen.
- El proceso de búsqueda selectiva es un algoritmo fijo, por lo tanto, no se está produciendo ningún aprendizaje en esa etapa. Esto puede resultar en tomar propuestas de regiones erróneas como falsos positivos.

4.4.6.2. Fast/Faster R-CNN

El algoritmo Fast R-CNN, al igual que R-CNN, se basa en una búsqueda selectiva de las propuestas de regiones para la detección y clasificación de objetos dentro de una imagen; sin embargo, la diferencia reside en que la red neuronal se alimenta con la imagen directamente, más no con las propuestas de regiones que antes se requerían para la clasificación, lo que resulta en una reducción en el procesamiento de la red, dado que ya no se debe procesar 2000 regiones por cada imagen (Girshick et al., 2014).

A pesar de la reducción de procesamiento que presenta Fast R-CNN y, por consiguiente, un aumento en la rapidez de procesamiento que permite obtener resultados en pocos segundos,

sigue siendo una opción poco viable para aplicaciones en tiempo real. Es por ello que (Ren et al., 2017), modificó el algoritmo para obtener el mapa de características convolucional directamente de la imagen, y de esta forma, utilizar una red neuronal convolucional adicional para predecir las propuestas de regiones, en lugar de utilizar una búsqueda selectiva, aumentando significativamente el rendimiento al reducir los parámetros a procesar. Por último, se realiza una agrupación de las regiones de interés (RoI pooling), permitiendo así obtener la clasificación y detección de la clase deseada dentro de la imagen, siendo resaltado a través de cuadros delimitadores.

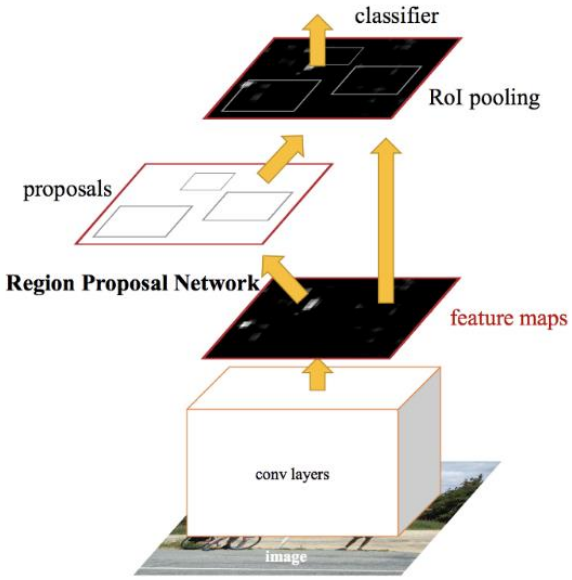


Figura 11. Arquitectura del algoritmo Faster R-CNN
Fuente: Adaptado de R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms, por R. Gandhi, 2018, Towards Data Science (https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e).

4.4.7. You Only Look Once - YOLO

El algoritmo You Only Look Once (en adelante YOLO) se basa en aplicar una red neuronal única a la imagen, lo cual la divide en regiones, pronostica cuadros delimitadores y establece probabilidades para cada región. El modelo basa sus ventajas en mirar completamente el gráfico en el momento que se realiza la prueba, por esta razón se tiene el contexto global (Moreira, 2021).

En YOLO se toma la detección de objetos como un problema único de regresión, una única red convolucional predice simultáneamente múltiples cuadros delimitadores que enmarcan los objetos en la imagen y predice probabilidades condicionales por cada clase para cada uno de estos cuadros delimitadores (Massiris et al., 2020).

4.4.7.1. YOLOv4

El objetivo de la versión 4 del algoritmo YOLO, es el aumentar la velocidad de detección a la vez de optimizar la precisión en comparación con otros modelos, debido a que existen muy pocas opciones disponibles que se centren en aplicaciones de tiempo real.

Como resultado del cambio de backbone y de la aplicación de diferentes métodos de procesamiento de imágenes para la optimización de la red neuronal, es posible utilizar este modelo en hardware de un único GPU, lo que deriva en realizar un entrenamiento exhaustivo, utilizando marcas de tarjetas gráficas convencionales, sin tener que pensar en la adquisición de un sistema especializado y costoso (Bochkovskiy et al., 2020).

Además de la ventaja de costo que se menciona anteriormente, se obtiene un aumento significativo en la velocidad de detección (FPS) (Bochkovskiy et al., 2020), siendo aplicable para situaciones que requieran un uso en tiempo real, tales como el presente proyecto, en donde se necesita una detección rápida de algún tipo de intrusión en la propiedad privada.

4.4.8. Comparativa entre modelos de detección de objetos:

La comparativa que se presenta de los distintos modelos de detección se desarrolla a partir de distintos factores que cuantizan su rendimiento. Tales parámetros son: el mAP (mean Average Precision) y la velocidad de detección medida en cuadros por segundo (FPS). Cabe destacar que las pruebas que presentan los distintos autores, se realizaron bajo ciertas circunstancias en donde existe una Unidad de Procesamiento Gráfico (GPU) que permite acelerar cientos de veces las tareas y cálculos que se realizan, a comparación de utilizar solamente la CPU.

Las pruebas de rendimiento que presentan Liu et al., (2016) se apoyaron en la base de distintos conjuntos de datos que permitieron entrenar a cada una de las opciones para obtener resultados y generar conclusiones al respecto. Con los dataset VOC2012 y VOC2007, se observa que el modelo SSD entrenado con una resolución de imagen de 300x300 tiene una mejora en la precisión por sobre los modelos Fast y Faster R-CNN; además, si se aumenta la resolución a 512x512 se tiene un resultado de 4,5% más de precisión que Faster R-CNN. Por otra parte, comparado con versiones anteriores a la 4 del modelo YOLO, se tiene un rendimiento significativamente mayor. En la Tabla 3 se observa con más detalle la diferencia que existe entre los modelos de detección, puestos a prueba en la detección de distintos objectos, y obteniendo un promedio que se refleja en el parámetro mAP.

Tabla 3. Pruebas de detecciones con distintas clases de objetos, realizados con el dataset PASCAL VOC2007

Method	Fast R-CNN	Faster R-CNN	Faster R-CNN	YOLO	SSD300	SSD300	SSD512	SSD512
Data	07++12	07++12	07++12+COCO	07++12	07++12	07++12+COCO	07++12	07++12+COCO
mAP	68.4	70.4	75.9	57.9	72.4	77.5	74.9	**80.0**
aero	82.3	84.9	87.4	77.0	85.6	90.2	87.4	**90.7**
bicicleta	78.4	79.8	83.6	67.2	80.1	83.3	82.3	**86.8**
pájaro	70.8	74.3	76.8	57.7	70.5	76.3	75.8	**80.5**
bote	52.3	53.9	62.9	38.3	57.6	63.0	59.0	**67.8**
botella	38.7	49.8	59.6	22.7	46.2	53.6	52.6	**60.8**
bus	77.8	77.5	81.9	68.3	79.4	83.8	81.7	**86.3**
auto	71.6	75.9	82.0	55.9	76.1	82.8	81.5	**85.5**
gato	89.3	88.5	91.3	81.4	89.2	92.0	90.0	**93.5**
silla	44.2	45.6	54.9	36.2	53.0	59.7	55.4	**63.2**
vaca	73.0	77.1	82.6	60.8	77.0	82.7	79.0	**85.7**
mesa	55.0	55.3	59.0	48.5	60.8	63.5	59.8	**64.4**
perro	87.5	86.9	89.0	77.2	87.0	89.3	88.4	**90.9**
caballo	80.5	81.7	85.5	72.3	83.1	87.6	84.3	**89.0**
persona	72.0	79.6	84.1	63.5	79.4	84.3	83.3	**86.8**
planta	35.1	40.1	52.2	28.9	45.9	52.6	50.2	**57.2**
oveja	68.3	72.6	78.9	52.2	75.9	82.5	78.0	**85.1**
sofá	65.7	60.9	65.5	54.8	69.5	**74.1**	66.3	72.8
tren	80.4	81.2	85.4	73..9	81.9	**88.4**	86.3	**88.4**
tv	64.2	61.5	70.2	50.8	67.5	74.2	72.0	**75.9**

Region Based CNN (R-CNN); You Only Look Once (YOLO); Single Shot Multibox Detector (SSD), Mean Average Precision (mAP). **Fuente:** *SSD: Single Shot Multibox Detector*, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Liu et al., (2016).

En otro acercamiento, Bochkovskiy et al., (2020) brindan una comparación de YOLOv4 con otros modelos de detección, incluyendo SSD, en donde existen resultados favorables con la nueva versión, utilizando el conjunto de datos MS COCO para entrenamiento y evaluación. En la investigación, se expone que las pruebas se realizaron con GPU de Maxwell, siendo que la precisión promedio (AP) de YOLOv4 ronda el rango de entre 41 – 43% priorizando su arquitectura para las aplicaciones en tiempo real obteniendo una velocidad de 22 a 38 FPS, en comparación a SSD, que, bajo las mismas circunstancias, presenta resultados de un AP de 25 a 29% y una velocidad de 22 a 44 FPS, siendo más rápido que YOLOv4, pero con una precisión más baja.

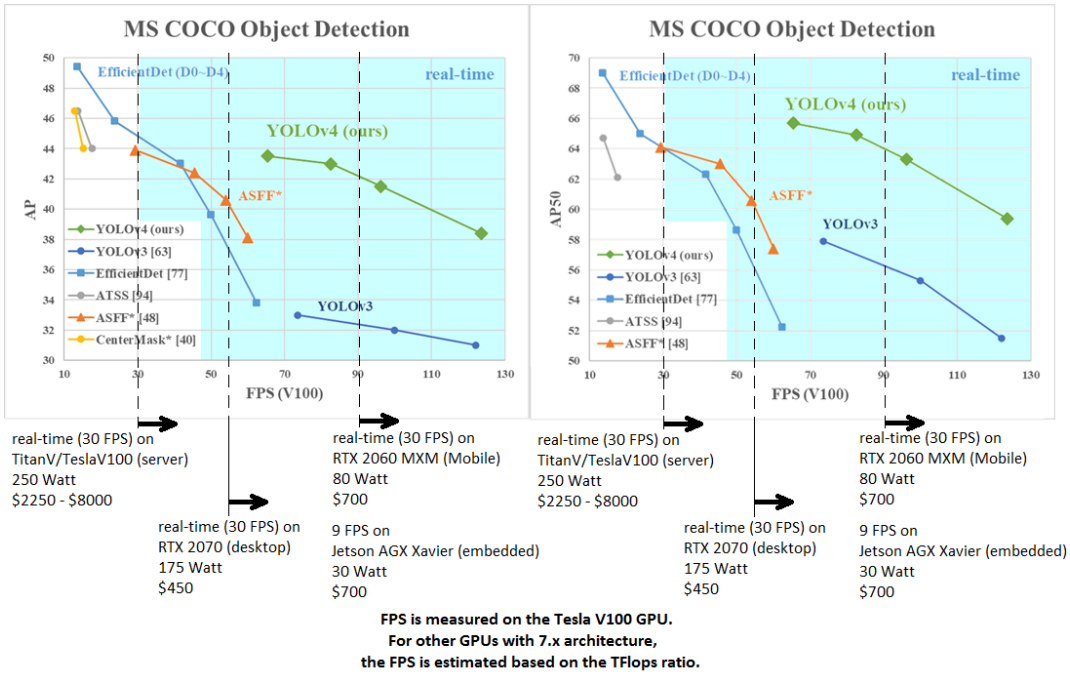


Figura 12. Pruebas realizadas de YOLOv4 en relación a otros modelos de detección
Nota. Adaptado de YOLOv4: Optimal Speed and Accuracy of Object Detection, por A, Bochkovskiy et al., 2020, (https://arxiv.org/abs/2004.10934v1).

La elección entre los distintos modelos dista entre las características que presentan cada uno, siendo que la opción más adecuada para aplicar en el presente proyecto es YOLOv4, debido a que ofrece un ambiente equilibrado entre precisión y velocidad, convirtiéndose en la elección perfecta para aplicaciones en tiempo real; uno de los requerimientos actuales.

Otro factor importante es que el modelo presenta la opción de utilizar a MobileNet o ShuffleNet como backbone, lo cual permite trabajar en arquitecturas que presenten solamente la CPU como único medio de procesamiento (Bochkovskiy et al., 2020), lo que consiente la generación de soluciones que se puedan implementar en sistemas económicos, con resultados óptimos y a la altura de la tarea a cumplir.

Ciertamente el modelo provee la posibilidad de implementar o activar backbones cuyo uso requiera de GPU convencional, lo cual brinda una ventaja adicional en aplicaciones en donde el presupuesto no sea tan estricto, dado que dicha GPU no requiere ser muy sofisticada para obtener resultados excelentes.

4.5. Aplicaciones Móviles – Descripción General

4.5.1. Flutter

Flutter es un entorno de trabajo (framework) de código abierto y multiplataforma creado por Google, que permite el desarrollo de aplicaciones móviles de alto rendimiento, lo

cual se traduce en la construcción de un mismo código capaz de ejecutarse en sistemas operativos tales como Android, iOS y Google Fuschia (Barot, 2021).

El lenguaje de programación en el que se basa este framework es Dart, el cual es un lenguaje de programación desarrollado y soportado por Google. Se desarrolló originalmente como reemplazo y sucesor de JavaScript, por lo que, implementa la mayoría de las características importantes de este, pero con la diferencia de que se adaptó su sintaxis para que sea similar a Java y así poder facilitar la curva de aprendizaje de programadores que estén familiarizados con el desarrollo de aplicaciones móviles en Android (Wu, 2018).

4.5.2. Kotlin

Kotlin es un lenguaje de programación pragmático que se ejecuta en la máquina virtual Java y Android. Combina características funcionales y orientadas a objetos, y es totalmente interoperable con Java. En consecuencia, es posible mezclar código Kotlin y Java en la misma aplicación, para llamar código Kotlin desde código Java y también lo contrario (Mateus & Martinez, 2018).

Es preferible el uso de Kotlin como lenguaje de propósito genera en comparación con otras opciones, debido a que brinda soluciones fáciles a los errores y conceptos erróneos de programación. Además, las aplicaciones creadas en Kotlin generalmente observan mínimos bloqueos o interrupciones abruptas y tiempo de inactividad escasos (Barot, 2021).

4.5.3. React Native

React Native es un marco de trabajo que utiliza el lenguaje Javascript como núcleo para el desarrollo de aplicaciones móviles. Se basa en React, la biblioteca de Javascript de Facebook para crear interfaces de usuario, pero en lugar de tener como entorno al desarrollo web, apunta a plataformas móviles.

El producto resultante del desarrollo en React Native, es una aplicación cuya experiencia de usuario no dista a la de una aplicación desarrollada con herramientas de código nativo, sea cual sea en el sistema operativo de un dispositivo móvil en el que se esté ejecutando. El escribir código basado principalmente en Javascript, permite establecer una compatibilidad universal, facilitando el desarrollo simultáneo para Android y iOS.

4.5.4. Comparativa general

4.5.4.1. Ventajas de React Native

React Native posee una amplia comunidad al tener el respaldo y soporte de Facebook, debido a que se encuentra basado en el framework de desarrollo web React, el cual goza de una popularidad significiativa. Brinda una sensación de fluidez en la experiencia de usuario, lo cual se logra al estar constituido principalmente por el lenguaje de programación Javascript; lo que permite contar con una documentación variada en la red (Wilfred, 2020).

4.5.4.2. Ventajas de Flutter

Flutter, al ser multiplataforma, permite hacer uso del mismo producto entregable, en cualquier sistema operativo móvil. Esto se traduce en un desarrollo rápido y eficaz, dado que no requiere de pruebas adicionales en los distintos entornos que existen actualmente en el marcado (InVerita, 2020).

4.5.4.3. Ventajas de Kotlin

En general, Kotlin establece un ambiente de acceso sencillo a la información dado que Google provee un soporte continuo a través de documentación en línea que se actualiza frecuentemente. Proporciona soporte de componentes nativos, que permiten el acceso a los recursos del dispositivo de forma sencilla y eficaz. Provee soporte para el desarrollo multiplataforma, haciendo uso de herramientas tales como SwiftUI y Jetpack Compose (Barot, 2021).

4.5.4.4. Panorama general de las tecnologías de desarrollo móvil

En primera instancia, React Native provee una biblioteca de librerías amplia para distintas soluciones, con la principal desventaja que la mayoría son de baja calidad o han sido completamente abandonadas (Wilfred, 2020).

Otra desventaja que no puede pasar desapercibida, es la baja seguridad que presenta React Native, al basar toda su estructura en Javascript, agregando susceptibilidad a distintos aspectos al producto final (InVerita, 2020).

Por otra parte, en Fluter, el tamaño de la aplicación resultante que ocupa en memoria es bastante elevado en comparación con las soluciones nativas. La comunidad con la que cuenta Flutter es relativamente pequeña, lo cual se debe a que es una tecnología que no ha estado mucho tiempo en circulación (Wilfred, 2020).

En última instancia, Kotlin se caracteriza por tener una curva de aprendizaje más complicada en comparación a las otras dos soluciones presentadas, a pesar de tener una variedad de cursos disponibles en la red. Se requiere tiempo adicional para obtener un Producto Mínimo Viable (MVP), pero con una estabilidad y soporte bastante buenos (Barot, 2021).

4.6. Tecnologías Web

4.6.1. Interfaz de programación de aplicaciones de transferencia de estado representacional (API REST)

Son servidores web que permiten el uso de lenguajes de consultas del lado del servidor, los cuales priorizan el proveer de los datos que se solicitan por parte de los usuarios, a través de aplicaciones que se conectan directamente a estos. Proveen un entorno para los desarrolladores el cual permite realizar solicitudes para la extracción de datos de múltiples fuentes, en donde se concentran en un único punto final (Masdiyasa et al., 2020).

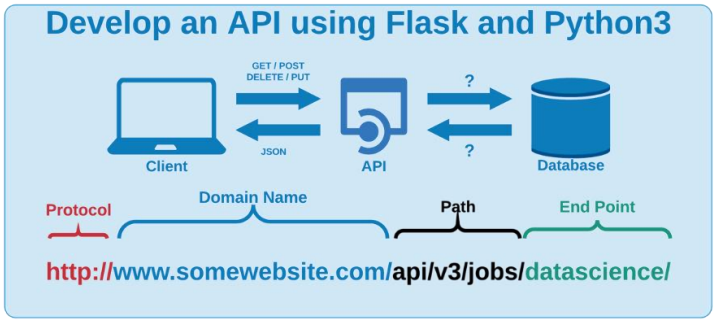


Figura 13. Arquictectura de una API REST
Fuente: Adaptado de *Develo pan API using Flask and Python3*, por C. Padberg, 2021, (https://morioh.com/p/8a7d070e6e96).

4.6.2. Websockets

WebSockets es un protocolo de comunicación que permite establecer una comunicación interactiva entre el servidor y cliente, dando origen al envío de mensajes de forma bidireccional controlados por eventos, sin el requerimiento de enviar una respuesta por cada una de las peticiones que realicen los terminales conectados (Fette & Melnikov, 2011).

El objetivo de esta tecnología es proveer un mecanismo para que las aplicaciones basadas en la web, establezcan una comunicación bidireccional con servidores que no dependen de la apertura de múltiples conexiones HTTP (Soewito et al., 2019).

4.6.3. Ventajas Websockets vs HTTP

La ventaja que proveen los websockets por encima de las peticiones HTTP clásicas es la comunicación dinámica en la comunicación de cliente-servidor, dado que no es necesario

que el cliente consulte constantemente el estado de los datos al servidor para actualizar la aplicación en caso de detectar un nuevo cambio, como se realiza comúnmente con la técnica de Sondeo Largo de HTTP (en inglés Long Polling HTTP) (Kitamura & Ubl, 2010).

El Long Polling HTTP permite al cliente establecer una conexión HTTP con el servidor, la cual se mantiene activa hasta que se envíe una respuesta con algún cambio en específico. El problema de esta solución es el excesivo uso de peticiones HTTP que afectan al rendimiento del servidor, lo cual los hace no aptas para aplicaciones con requerimientos de baja latencia (Kitamura & Ubl, 2010).

Por la arquitectura que presentan los websockets, es posible su aplicación en soluciones que requieran una actualización de los datos en tiempo real; esto debido a la creciente demanda de la implementación de Hogares Inteligentes, en donde constantemente se comparten datos de sensores a través de internet, por medio de protocolos de comunicación confiables (Soewito et al., 2019).

5. Metodología

En la presente sección se detallan los materiales y métodos utilizados para el desarrollo del Trabajo de Titulación. Se explica tanto la metodología de trabajo, así como el proceso a nivel general que se llevó a cabo durante el periodo dispuesto para la terminación del proyecto.

5.1. Contexto

El proyecto de investigación se llevó a cabo dentro de un contexto académico durante el periodo de octubre 2021 a marzo 2022, en la Carrera de Ingeniería en Electrónica y Telecomunicaciones de la Facultad de Energía, las Industrias y los Recursos Naturales No Renovables perteneciente a la Universidad Nacional de Loja, con la colaboración de docentes que imparten asignaturas afines al tema en cuestión.

5.2. Procedimiento

Se aplicó un enfoque cuantitativo dado que la investigación requirió un proceso sistemático y con un orden establecido para llevar a cabo determinados pasos; partiendo desde la selección del tema a abordar, hasta la documentación y difusión de resultados obtenidos (Monje, 2011).

El estudio se realizó en base a las estadísticas expuestas por la Dirección de Estadística y Sistemas de Información - FGE (2021), en donde se modela el comportamiento de los individuos envueltos en un evento de intrusión a la propiedad privada y pública. Para ello, la recolección de los datos necesarios para el modelamiento del sistema de detección se realizó a través de un muestreo simple al azar, con el fin de brindar la capacidad de aprendizaje, mas no de memorización, a la red neuronal.

Para alcanzar el objetivo general del presente proyecto de investigación se usó el siguiente procedimiento por cada uno de los objetivos específicos planteados:

1. Implementación de un algoritmo de detección de individuos para la diferenciación de las formas captadas por el sistema de videovigilancia.
    a. Se elaboró una investigación acerca de los distintos algoritmos de detección de objetos, analizando cada una de las opciones y determinando el adecuado a los requisitos planteados.
    b. Se desarrolló algoritmos para la recolección y procesamiento de imágenes, necesarias para la construcción de un conjunto de datos personalizado;

requerimiento primordial para el entrenamiento de los modelos de detección seleccionados durante la investigación inicial.

c. Se realizaron pruebas de rendimiento iniciales a los productos resultantes, estableciendo el modelo adecuado a los requisitos planteados.
d. Se establecieron algoritmos con las clases necesarias para la construcción del entorno de ejecución del sistema de detección en cuestión.

2. Desarrollo de un aplicativo móvil para el despliegue del servicio al usuario, con la finalidad de presentar las imágenes obtenidas por el sistema de videovigilancia y el estado de la situación del sitio vigilado.
    a. Se elaboró el diseño de la base de datos, a través de un diagrama entidad-relación UML, estableciendo la estructura de la información a almacenar; requisito necesario para la construcción de la API web.
    b. Se construyó una API Web, a través de la librería de Python, Flask, cuya estructura consta de una base de datos relacional PostgreSQL y el protocolo de comunicación WebSockets, para la transmisión de datos hacia la aplicación móvil.
    c. Se realizó la captura de requisitos a implementar dentro del aplicativo móvil, en base a la demanda obtenida del primer objetivo específico.
    d. Se elaboraron diagramas de prototipado para el análisis del aplicativo móvil.
    e. Se codificaron los módulos de la aplicación, en base a los artefactos previamente obtenidos.
    f. Se realizó la conexión con el sistema obtenido en el primer objetivo, y las respectivas pruebas de funcionamiento.
3. Evaluación estadística de la eficiencia del sistema en la detección y reconocimiento de intrusos, en base a distintos ambientes de prueba.
    a. Se generaron distintos escenarios de prueba.
    b. Se registraron todos los datos obtenidos en las pruebas, y, finalmente;
    c. Se generó la documentación respectiva al sistema completo (Manual de Usuario).

5.3. Procesamiento y análisis de datos

Para el análisis de datos se aplicaron distintos métodos y modelos para una representación de la estructura y funcionamiento de los resultados generados en los respectivos objetivos planteados. Para la evaluación de la precisión del modelo de detección generado, se

aplicó el cálculo del valor mAP, donde cada uno de los promedios de precisión que se generan en base a un conjunto de datos destinado para evaluación, se obtienen a partir del valor medio de 11 puntos de la curva PR, para cada umbral posible (Bochkovskiy et al., 2020).

Por otra parte, para la evaluación de la eficiencia del sistema en general, se aplicó un promedio de detecciones, donde se analizan falsos positivos, falsos negativos y detecciones realizadas con éxito para establecer un valor estadístico que describa el rendimiento de la solución generada. La recolección de estos datos se las realiza a partir de la toma de muestras aleatorias en base a la detección de movimiento y de intrusos, cuyos resultados se almacenan en una base de datos relacional aplicada en el sistema, específicamente PostreSQL, con el motivo de establecer un medio por el cual se pueda realizar consultas específicas de forma sencilla.

5.4. Recursos

Con el propósito de cumplir con los objetivos planteados, se utilizaron los siguientes recursos:

5.4.1. Recursos Científicos

Método analítico: Este método consiste en la descomposición de un objeto en partes o elementos consecutivos (Lopera et al., 2010). Para el presente Trabajo de Titulación, se hizo uso de este método para dividir el objeto de estudio en etapas, las mismas que fueron plasmadas como objetivos específicos junto con las actividades a realizar de manera secuencial.

Estudio del estado del arte: Este proceso se define como una revisión de propuestas de investigación acerca de un objeto o fenómeno (Guevara Patiño, 2016). Durante el desarrollo del presente TT, se llevó a cabo esta técnica para buscar información en fuentes bibliográficas que hagan referencia a las etapas, metodologías y procesos que permiten gestionar de manera eficiente el ciclo de vida de desarrollo de software.

5.4.2. Recursos Técnicos

Herramientas colaborativas: Se hizo uso de herramientas colaborativas disponibles en la web tales como: Google Drive como área de trabajo para la gestión de documentos, imágenes y otros recursos. Zoom como principal herramienta de comunicación entre los principales participantes del proyecto de investigación.

Entornos Virtuales de desarrollo: Google Colaboratory como editor de código y entorno de aprendizaje de los modelos de detección. Google Cloud como entorno de desarrollo para la superación de las limitaciones de hardware.

Editor de código Visual Studio Code: Se utilizó esta herramienta para la elaboración del código fuente durante la ejecución del primer objetivo específico del presente proyecto.

Entorno de desarrollo integrado (IDE) IntelliJ Idea: Herramienta utilizada para la elaboración del código fuente, producto del segundo objetivo específico del presente trabajo de titulación.

Repositorio de alojamiento Github: Este servicio en la nube se utilizó para la administración y seguimiento de las modificaciones del código fuente generado de los distintos artefactos generados durante el desarrollo del presente proyecto.

Sistemas de Circuito Cerrado de Televisión: Se utilizaron para obtener los recursos necesarios para la generación de pruebas de rendimiento del sistema generado a partir de los objetivos específicos uno y dos.

5.5. Participantes

El presente TT está enfocado en la línea de investigación de Desarrollo de Software, el mismo que fue llevado a cabo por los siguientes participantes:

- Jorge Enrique Ortega Jaramillo, como estudiante autor del presente Trabajo de Titulación, quien ejecutó todas las actividades desde el planteamiento del tema, hasta la finalización de los objetivos propuestos (véase sección Metodología, apartado 5.2).
- El Ing. Luis Eduardo Rodríguez Montoya, Mg. Sc. como tutor académico y director del presente Trabajo de Titulación, quien llevó a cabo la tarea de supervisión de avances académicos y técnicos desarrollados por el autor.
- El Ing. Kleber Rolando Morillo Aguilar, Mg Sc. como tutor académico y colaborador externo del Trabajo de Titulación.

6. Resultados

En la presente sección se detalla la evidencia de los resultados obtenidos a lo largo de la ejecución del Trabajo de Titulación, en base a los objetivos específicos propuestos.

6.1. Objetivo 1: Implementar un algoritmo de detección de individuos capaz de diferenciar las formas captadas por el sistema de videovigilancia, permitiendo un sistema robusto a las falsas alarmas.

En este apartado se realizaron las acciones pertinentes para seleccionar un modelo de detección de objetos pertinente para la aplicación en el proyecto. Es por ello que, en base a la investigación realizada en el marco teórico, se optó por el modelo YOLOv4 en su versión ligera, el cual se denomina YOLOv4-Tiny.

6.1.1. Construcción del conjunto de datos

Para obtener un modelo adecuado, que permita detectar y reconocer una persona dentro de un escenario específico que se obtiene a través de un sistema de CCTV, fue requerido generar un conjunto de datos apropiado, el cual constó de un gran número de imágenes adquiridas a partir de conjuntos de datos certificados tales como Open Image Datasets de Google (OID).

Adicionalmente, se generó un conjunto de datos personalizado seleccionando imágenes pertinentes a partir de vídeos obtenidos de sistemas de CCTV, utilizando como entrada los parámetros especificados en la Tabla 4 para la ejecución de los algoritmos desarrollados y expuestos en el Anexo 2. Asimismo, en la Figura 14 de puede observar un ejemplo de las imágenes seleccionadas para el entrenamiento del modelo de detección en cuestión.

Tabla 4. Parámetros de configuración para los algoritmos de construcción del conjunto de datos

Descripción	Valor
Similitud entre imágenes;	75 – 80%
Extensión de imágenes válidas:	JPG; PNG
Extensión de vídeos soportados:	MPG; MP4
Formado de nombramiento de imágenes:	out-**n**-jpg
Probabilidad de alteración aleatoria:	Iluminación: 25% Rotación: 10% Sin alteración: 65%

El valor de la similitud entre imágenes indica el porcentaje de igualdad entre la última imagen tomada y la actual, con el objetivo de obtener imágenes en diferentes contextos para el aumento de información. La probabilidad de alteración aleatoria se establece para el proceso de etiquetado de imágenes, detallado en el Anexo 2.

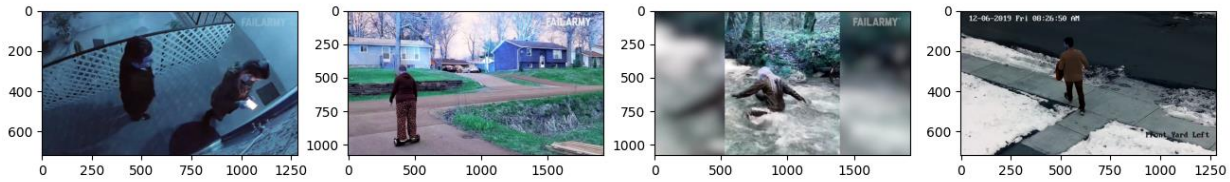


Figura 14. Muestra de imágenes tomadas del conjunto de datos personalizado generado.
Fuente: Elaborado por el autor.

Las probabilidades de alteración aleatoria de la imagen, fueron de utilidad para añadir la característica de imagen única a las distintas muestras generadas para la construcción del conjunto de datos; por lo que, tal como lo menciona la Tabla 4, se añadió una probabilidad de 25% para el cambio de iluminación aleatorio y un 10% en la rotación aleatoria de la imagen. Ambos parámetros pueden suceder una vez por imagen. Dichos porcentajes son establecidos tal que solamente una pequeña parte del conjunto de datos se vea afectado.

Con respecto a la alteración de la iluminación de la imagen, se utilizó el método Corrección Gamma, donde el rango de valores que modifican la iluminación, se selecciona aleatoriamente en un rango de 1 a 3. En la Figura 15 se presenta un ejemplo tomado del conjunto de datos, en donde se aplicó esta corrección. Todo este proceso de alteración ocurre en la fase de generación de anotaciones, necesarias para el entrenamiento del modelo personalizado YOLOv4-Tiny.

Figura 15. Corrección de Iluminación a través de la Corrección Gamma
Fuente: Elaborado por el autor.

6.1.2. Entrenamiento del modelo de detección.

Para el entrenamiento de la red neuronal, fue necesario una configuración previa de la estructura requerida para su entrenamiento y generación de los pesos (weights) cuyos parámetros se encuentran alojados en un archivo de extensión [**.**cfg]. Se destacan algunos parámetros de esta configuración, necesarios para el entrenamiento del modelo:

- **classes:** Señala el número de clases diferentes que se van a analizar, tanto en entrenamiento como en detección, en una misma imagen. El valor se especificó en **classes = 1** dado se toma en consideración solamente la clase "persona".
- **filters:** número de kernels utilizado para la inferencia de la red. Este valor se modifica en base a la fórmula $filtros = (clases + 5) \times 3$ únicamente en las capas convolucionales ubicadas antes de cada una de las capas [yolo]. Para este caso, como se trabaja con una sola clase, el valor fue de **filters = 18**.
- **max_batches:** Número de iteraciones en total para el entrenamiento del modelo, se calcula a partir del cálculo ($clases * 2000$). Como se está considerando una sola clase (persona), se establece en un valor mínimo igual a 6000.
- **subdivisions:** Divisor para el número de muestras que se van a procesar al mismo tiempo, a través del uso de la GPU disponible, en caso de ser necesario.
- **batch:** Refiriéndose al número de imágenes que serán procesadas en una sola iteración. El valor estándar y utilizado es de 64. Adicionalmente este valor se utiliza en conjunto con **subdivisions**, por lo que, en caso de presentarse que no existen los recursos suficientes para analizar todas esas divisiones al mismo tiempo a través del a GPU, se puede editar a un valor de 32 o 16.
- **steps:** Número de iteraciones en donde la tasa de aprendizaje aumenta en un factor definido en **scales**. Se establecen calculando el 80% y el 90% del número de iteraciones total (max_batches).
- **width/height:** Dimensiones específicas para escalar la imagen y alimentar la red neuronal para la inferencia de datos. El valor se puede aumentar en un valor de 32, y su valor recomendable es 416. Se establecieron en valores iguales a 608x608 con el objetivo de aumentar la precisión de inferencia de la red.
- **channels:** Número de canales especificados para la red, por lo que todas las imágenes de entrada tendrán una conversión para cumplir este requisito. Se realizará una reorganización de los pixeles de la imagen en 3 canales diferentes, en donde cada uno tendrá diferentes pixeles adyacentes. De forma estándar se establece el valor a **channels = 3**.

Adicionalmente, en la especificación de la estructura de la red, fue posible fijar parámetros que permitan un aumento de datos conveniente para mejorar la precisión del modelo, durante el entrenamiento de la red; estos se describen a continuación:

- **saturation:** Permite cambiar aleatoriamente la saturación de las imágenes durante el entrenamiento. Se estableció en un valor de **saturation = 1.5** para un aumento del 50% de la saturación a una imagen seleccionada aleatoriamente.
- **random:** Señala si las imágenes serán dimensionadas aleatoriamente durante el entrenamiento para una mejora en la precisión. Se estableció en un valor de **random = 1** dado que el rango admitido es de 0 o 1, donde 0 indica **falso** y 1 indica **verdadero**.
- **exposure**: Permite cambiar aleatoriamente el brillo de las imágenes durante el entrenamiento de la red neuronal. Su valor establecido fue de **exposure = 1.5** para un aumento del 50% del brillo a una imagen seleccionada aleatoriamente.
- **hue:** Permite cambiar aleatoriamente el matiz de las imágenes durante el entrenamiento de la red neuronal. Su cuantificación se encuentra normalizada en un rango establecido de 0 a 1. Para el presente caso, se fijó un valor de **hue = .1** para cambios de un 10% en el matiz de una imagen aleatoria.

En la Figura 16 se presenta la gráfica de los datos obtenidos del proceso de entrenamiento realizado con el conjunto de datos obtenido de OID, compuesto por un total de 1800 imágenes (1500 para entrenamiento y 300 para validación), en donde se observan los valores mAP obtenidos a lo largo de las iteraciones completadas. El valor máximo de mAP que se pudo obtener fue de 26%, por lo que el uso de dicho modelo se descartó para la implementación en el detector de intrusos.

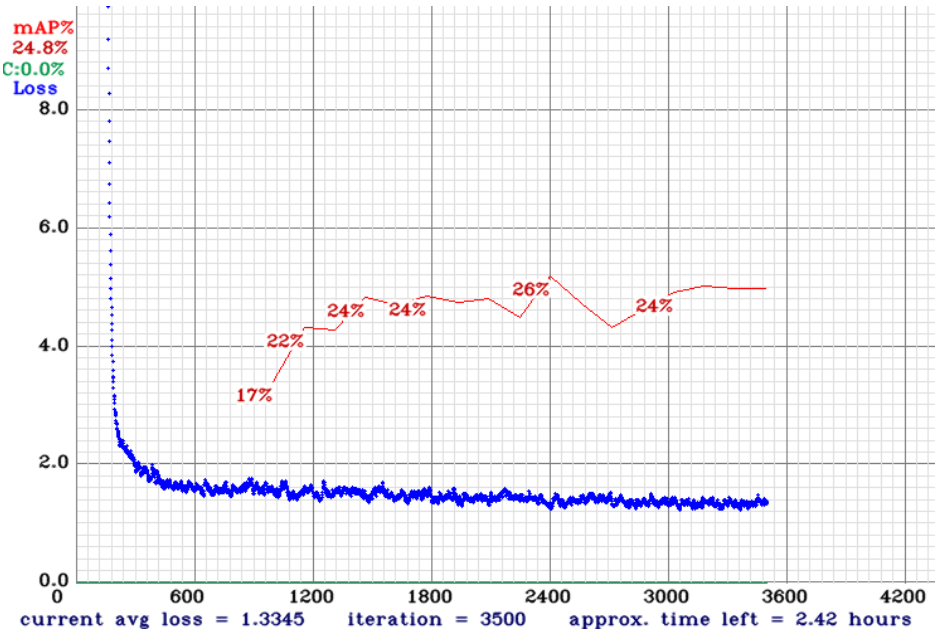


Figura 16. Proceso de entrenamiento con imágenes obtenidas de Open Image Dataset v2
Fuente: Elaborado por el autor.

Por consiguiente, se procedió a aplicar el conjunto de datos generado a partir de recursos videográficos, con un total de 1.176 imágenes (918 para entrenamiento y 258 para validación), en donde se pudo obtener un considerable aumento en la precisión del modelo, lo cual se encuentra evidenciado en la Figura 17. El valor mAP máximo obtenido fue de 88,59%.

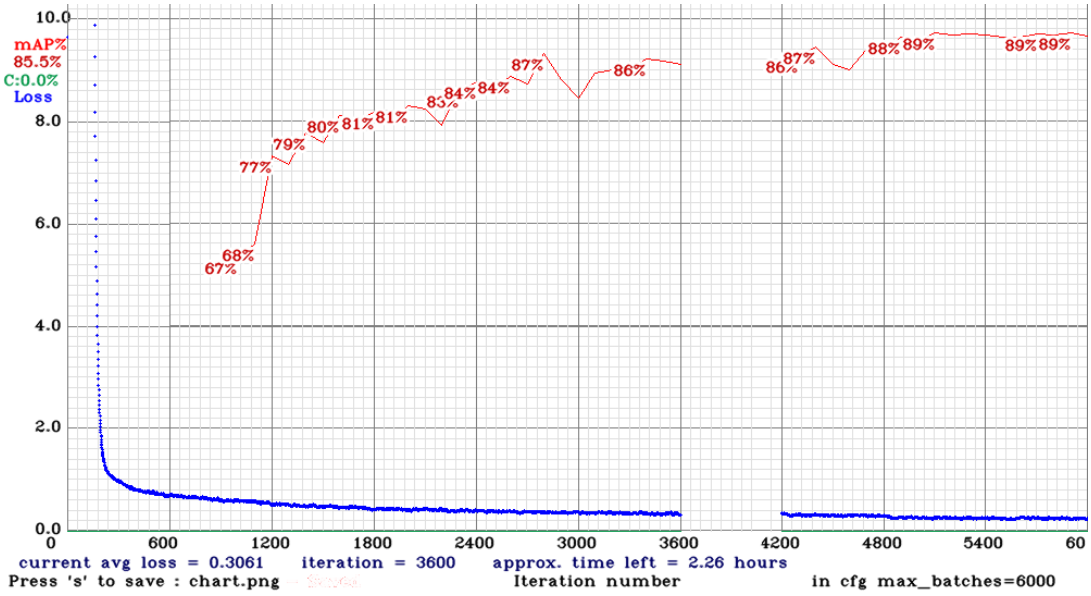


Figura 17. Proceso de entrenamiento con imágenes generadas desde fuentes de vídeo (1176 imágenes).
Fuente: Elaborado por el autor.

El valor correspondiente mAP representa la precisión del modelo en base al cálculo del valor medio de los 11 puntos correspondientes a la curva PR, por cada umbral posible (rango de 0 a 1, espaciados equitativamente [0, 0.1, …, 1]) (Everingham et al., 2009).

El valor AP se calcula a partir de la fórmula (1) el cual indica el valor promedio de la precisión a partir de la curva PR (Precision-Recall).

$$AP = \frac{1}{11} \sum_{r \in \{0, 0.1, \ldots, 1\}} p_{interp}(r) \quad (1)$$

Donde $p_{interp}(r)$ se especifica a partir de la fórmula (2). Dicha fórmula define el nivel de Precisión máximo dado un nivel de Recall r.

$$p_{interp}(r) = \max_{\tilde{r}:\tilde{r} \geq r} p(\tilde{r}) \quad (2)$$

Para el caso de la detección de objetos, el umbral que se establece para determinar cuándo una muestra corresponde a un ejemplo positivo clasificado correctamente o a un ejemplo negativo clasificado correctamente, se deriva del porcentaje de intersección que existe entre el verdadero cuadro delimitador de la muestra comparada con la predicción realizada,

mejor conocido como Intersection Over Union (IOU). Para la evaluación de estos parámetros, Bochkovskiy et al. (2020) establecen un valor de IOU de 50%, el cual deriva del estándar establecido en las mayores competencias de modelos de detección de objetos (PascalVOC, MS COCO, entre otras).

6.1.3. Evaluación del modelo de detección.

Bochkovskiy et al. (2020) proveen el medio necesario para evaluar la precisión del modelo, en base a los parámetros descritos anteriormente. Esto se realizó ejecutando el comando *darknet.exe detector test data/obj.data <ruta/a/conf.cfg> <ruta/a/weights.weights>*, el cual brindó los resultados que se pueden observar en la Figura 18, estableciendo un mAP@0.5 igual a 88,59% para el modelo entrenado, derivado del cálculo de la cantidad de Verdaderos Positivos, Falsos Positivos y Falsos Negativos. De igual forma, se analiza que el IOU promedio calculado a partir de las detecciones realizadas en el conjunto de datos destinado a validación, es de 69,24%; superando los 50% establecidos como umbral por estándar. El detalle de los datos obtenidos se registra en la Tabla 5.

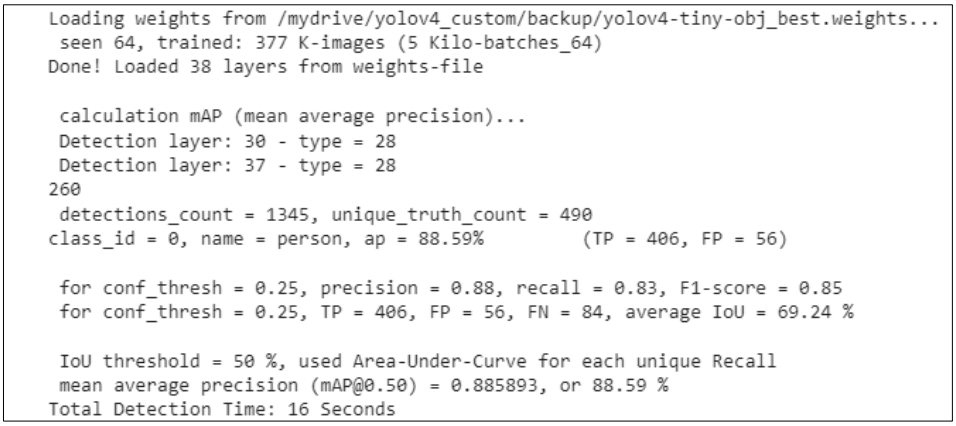


Figura 18. Convergencia del modelo de detección personalizado YOLOv4-Tiny
Fuente: Elaborado por el autor.

Tabla 5. Resultados de evaluación del modelo de detección personalizado YOLOv4-Tiny.

Parámetro	TP	FP	FN	Av. IoU	mAP
Valor	406	56	84	69,24%	88,59&

Positivos Verdaderos (TP); Falsos Positivos (FP); Falsos Negativos (FN), Promedio de la Intersección sobre la Unión (Av. IoU); Valor medio AP (mAP). **Fuente:** Elaborado por el autor.

A partir de ello se procedió a establecer un ambiente de pruebas, donde se instaló un sistema de CCTV para verificar que el modelo esté trabajando adecuadamente. El resultado se puede visualizar en la Figura 19.

Figura 19. Comparación entre modelos de detección: Personalizado vs Preentrenado.
Fuente: Elaborado por el autor.

6.2. Objetivo 2: Desarrollar un aplicativo móvil para el despliegue del servicio al usuario, en donde se presente la imagen obtenida por el sistema de videovigilancia y el estado de la situación del sitio vigilado.

En la presente sección se presentan las arquitecturas generadas, correspondientes al sistema completo propuesto, el cual consta de los recursos que permiten la detección de objetos, el servidor en donde se establece un medio de alojamiento y ejecución de dichos recursos que permiten el acceso al servicio a la aplicación móvil, admitiendo la transmisión de información de estado de los escenarios vigilados.

6.2.1. API Web

Antes del desarrollo de la aplicación, fue pertinente establecer la API Web con el fin de establecer un medio de comunicación entre el sistema de detección y la aplicación móvil. El funcionamiento del producto final se resume en la Figura 20.

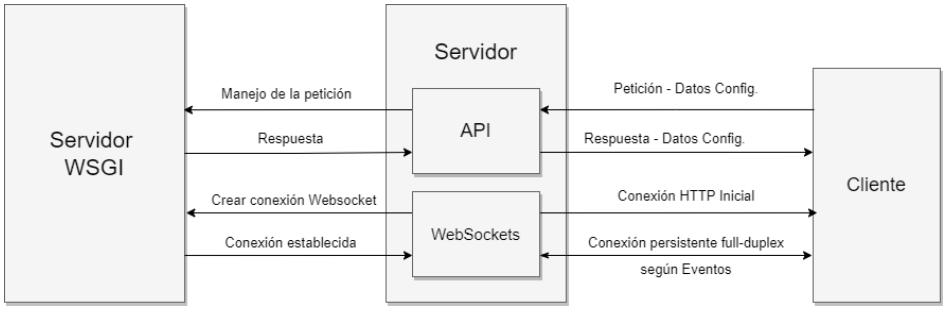


Figura 20. Estructura de la aplicación web
Fuente: Elaborado por el autor.

Para el registro de la configuración de los sistemas a analizar además de las detecciones desencadenadas, se utilizó la base de datos relacional PostgreSQL, debido a la capacidad multiproceso en las tareas de lectura y escritura de registro por conexión, adecuada para el

contexto y requerimientos de procesamiento presentados. La estructura de las tablas para los registros a almacenar se describe en la Figura 21.

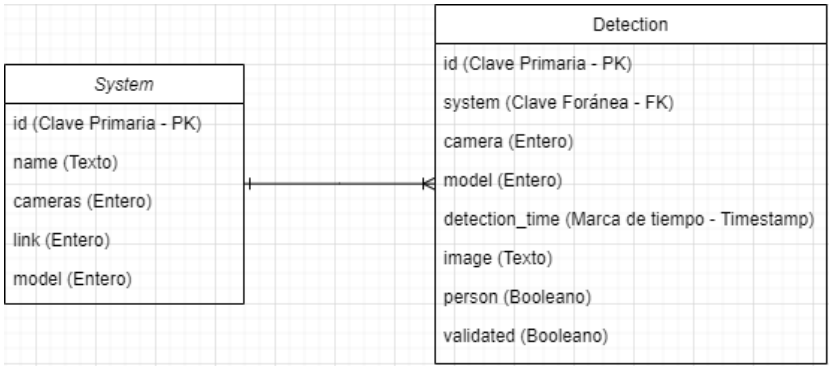


Figura 21. Estructura de la base de datos implementada.
Fuente: Elaborado por el autor

Como protocolo de conexión, desde la aplicación móvil al servidor, se utilizaron Websockets; esta implementación dio como resultado una comunicación síncrona con todos los clientes que realizan una petición de acceso al sistema de detección alojado en el servidor, permitiendo así, que las alertas y la información pertinente que se transmite, se presente de forma simultánea para todas las terminales.

Este tipo de conexión fue necesario, dado que al realizarlo con constantes peticiones HTTP para consultar el estado de las detecciones, añade una sobrecarga en el rendimiento del servidor cuando varios clientes se encuentren de forma simultánea en una sesión hacia el sistema.

Es por ello que, para lograr la transmisión de vídeo e información de estado del sistema hacia la aplicación móvil, se utilizó el framework SocketIO, el cual permitió segmentar cada uno de los datos a enviar en "eventos", admitiendo el envío simultáneo de la información del estado de alerta, sin la necesidad de compartir canal con otros ambientes vigilados. Los eventos se describen en a detalle en la Tabla 6.

Tabla 6. Eventos Websocket implementados en la estructura de la API Web.

Evento	**Descripción**
detection{n}	Permite enviar los datos del estado actual del sistema. Indica la existencia de movimiento o intrusos.
video{n}	Envía la información de vídeo en tiempo real para su presentación en la aplicación móvil.
detected	Envía la alerta en caso de darse un evento de detección de intrusos.

Número de cámara o sistema en análisis **{n}**. **Fuente:** Elaborado por el autor.

Una vez especificada la estructura de cada evento, se procedió a realizar las pruebas de transmisión, a través del establecimiento de un cliente de prueba cuya configuración se especificó de modo que al recibir los bytes de la imagen correctamente imprima las dimensiones de la misma como medio de comprobación. El resultado del proceso anteriormente descrito se puede apreciar a través de la Figura 22.

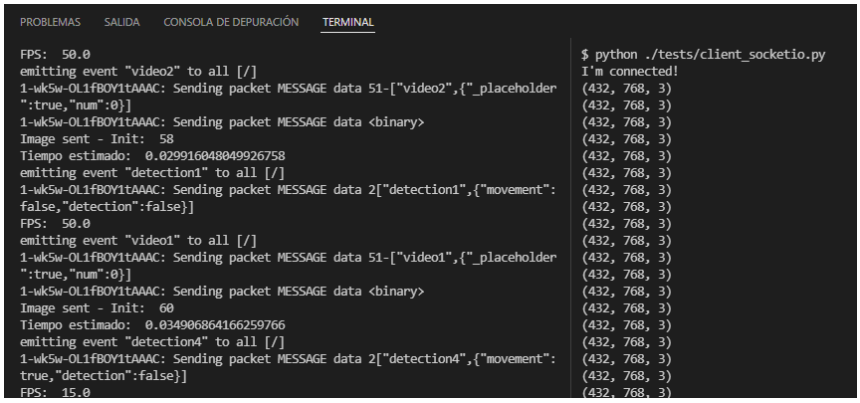


Figura 22. Proceso de envío y recepción de datos a través de WebSockets
Fuente: Elaborado por el autor.

Adicionalmente se estableció una API Web, con el objetivo de facilitar la adición y consulta de fuentes de vídeo para el análisis y la detección. Se añadieron 2 rutas que cumplen las funciones antes mencionadas las cuales se enlistan a continuación. El uso de esta API Rest y el formato de datos utilizado se detallan en el Anexo 1.

- Añadir un sistema: ***http://<direccion_web>/api/add_system***
- Editar un sistema: ***http://<direccion_web>/api/update_system***

Como medio de optimización se estableció un algoritmo de detección de movimiento, debido a la carga computacional que supone el modelo de detección que, aunque es adecuada para aplicaciones en tiempo real, es un uso inadecuado de recursos el ejecutarlo de forma constante, cuando la mayoría de tiempo el área vigilada se encuentra en un estado normal y sin anomalías.

Es por ello que se utilizó el algoritmo KNN para la extracción de fondo que sirva como base para detectar cambios significativos en el ambiente y así dar paso a la ejecución del modelo de detección para su respectivo reconocimiento y ubicación dentro de la imagen en caso de tratarse del movimiento de una persona. En la Figura 23 se puede apreciar el algoritmo en funcionamiento.

El método obtiene una determinada cantidad de cuadros de vídeo para definir el modelo de fondo base, y a partir del cálculo de la distancia Euclidiana entre dos puntos (como se define

en la ecuación 3) se obtiene la imagen con el objeto en movimiento aislado una vez superado un umbral establecido, el cual señala el límite del valor que se toma como un cambio significativo en el ambiente.

$$d(A,B) = \sqrt{\sum_{n=1}^{\infty} (a_i - b_i)^2} \qquad (3)$$

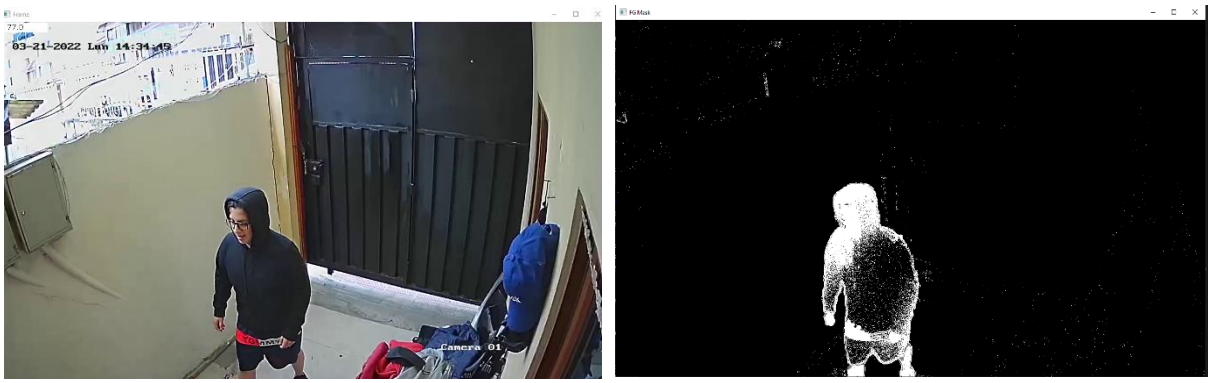


Figura 23. Detección de movimiento a través del algoritmo K-Nearest Neighbor
Fuente: Elaborado por el autor.

6.2.2. Aplicación Móvil

Con respecto a la aplicación móvil, se estableció el uso de recursos tales como el lenguaje de programación Kotlin, en conjunto con el entorno de desarrollo Android Studio, tanto para la escritura del sistema como para el despliegue y construcción del producto final.

La selección del lenguaje de programación de Kotlin para la construcción de la aplicación, se basó en las características y ventajas presentadas en el marco teórico, además de la compatibilidad con Websockets incluido en el conjunto de herramientas que ofrece.

Como primer punto se estableció una fase de análisis para el desarrollo de la aplicación, con el propósito de generar una compatibilidad de funcionamiento con la API Web generada. Por lo tanto, se establecieron 3 secciones distribuidas uniformemente en la vista principal de la aplicación, tal como se observa en la Figura 24, cuyo designio es descrito a continuación:

- Vídeo en vivo extraído de las cámaras en servicio, del sistema de CCTV.
- Zona de detecciones, en donde se fijó un espacio dedicado a la imagen correspondiente a la última detección de intrusos realizada por el sistema.
- Estado de la situación en vivo, que permite monitorear la actividad en cada una de las cámaras que se estén analizando en el momento.

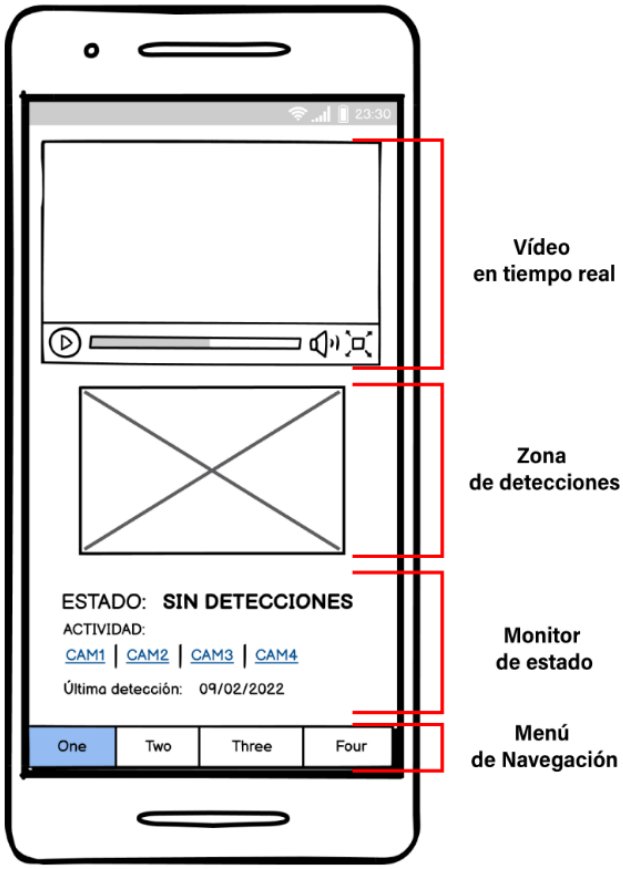


Figura 24. Estructura de la aplicación en fase de planificación.
Fuente: Elaborado por el autor.

A partir del anterior análisis y los requerimientos de funcionamiento, se realizó la construcción del aplicativo móvil, poniendo énfasis en la correcta transmisión de los datos. Es por ello que se realizaron pruebas de rendimiento, lo que permitió evidenciar que la aplicación tenga un funcionamiento óptimo. Los datos obtenidos se describen en la Tabla 7, y la gráfica generada a través de la herramienta Android Studio para el caso, se puede visualizar en la Figura 25.

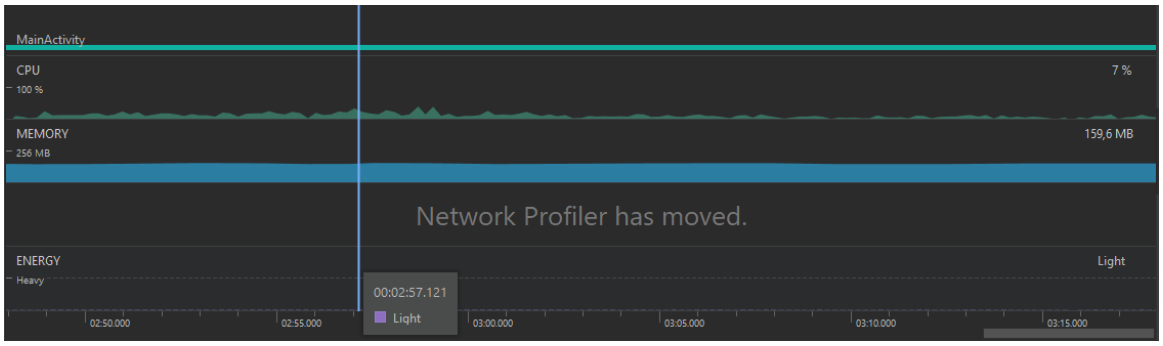


Figura 25. Medidas de rendimiento de la aplicación móvil
Fuente: Elaborado por el autor.

Tabla 7. Prueba de rendimiento de la aplicación móvil

Recurso en uso	Estado Estable	Inicio
CPU	7%	100%
Memoria RAM	159,6 MB	256 MB
Energía (Batería)	Light (Uso Ligero)	Heavy (Alto)

Tiempo de uso para la medida de las métricas: 25min. **Fuente:** Elaborado por el autor.

Los procesos que sigue la aplicación para iniciar y ejecutar las funciones requeridas para cumplir con el objetivo planteado, se resumen a continuación:

a. Obtener la información de los sistemas registrados en la base de datos del servidor.
b. Generar los contenedores respectivos para la transmisión de vídeo según el número de sistemas y cámaras establecido.
c. Establecer la conexión con el WebSocket para obtener la información de vídeo y estado de detecciones.
d. Asignar cada uno de los eventos de la conexión WebSocket, a los respectivos a los contenedores o widgets establecidos en la aplicación, con el fin de cumplir el propósito de informar la situación actual del lugar vigilado.
e. Una vez terminados los procesos anteriores, comienza la etapa de transmisión y actualización de la interfaz según sea necesario. En caso de obtener un evento de intrusión, se situará una captura del cuadro de vídeo correspondiente en la Zona de Detecciones y se señalará en el Monitor de Estado en qué cámara o entorno se ha suscitado tal evento.

El producto final se puede observar en la Figura 26, en donde se presenta capturas del funcionamiento de la aplicación, en conjunto con un sistema de CCTV instalado en una ubicación específica, con el fin de observar su desempeño aplicado en un escenario real.

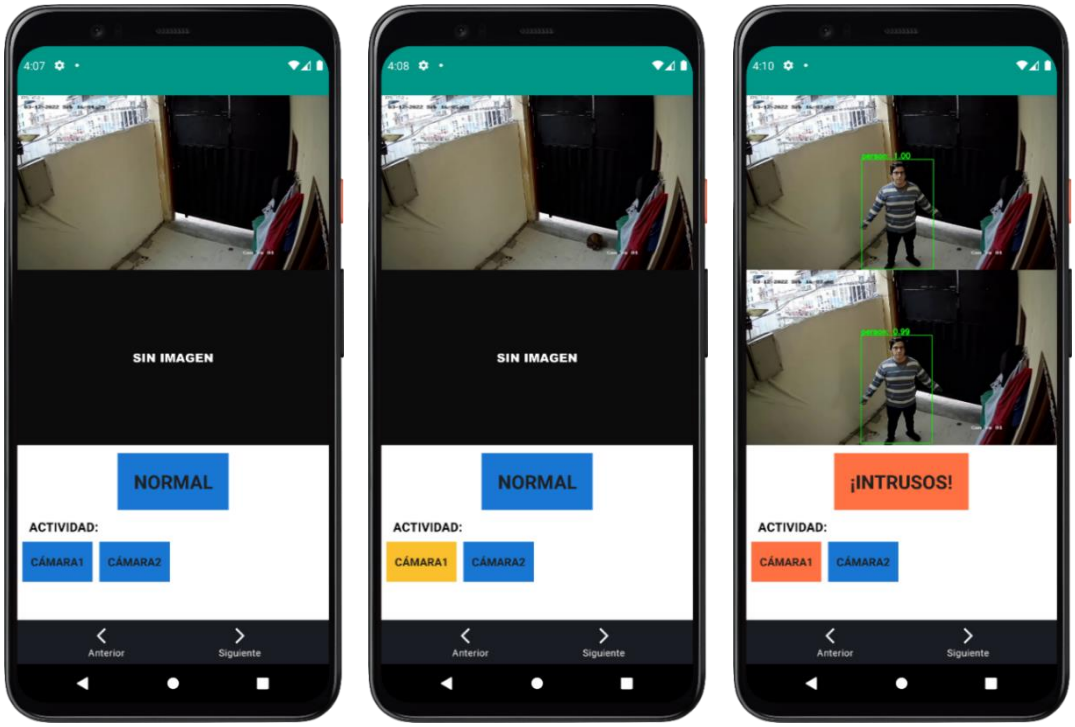


Figura 26. Aplicación móvil en funcionamiento.
Nota: Elaborado por el autor.

6.3. Objetivo 3: Evaluar estadísticamente la eficiencia del sistema en la detección y reconocimiento de intrusos, sometiéndolo a distintos ambientes de prueba.

Para medir la eficiencia del sistema, se establecieron 4 entornos diferentes en donde se obtuvieron de forma aleatoria, imágenes correspondientes a detecciones tanto de movimiento como de intrusiones; y cuyo análisis parte de la cantidad de detecciones realizadas exitosamente y los fracasos que existieron en el proceso.

Para lograr el acometido, se estableció el registro de los eventos antes mencionados, a través del sistema web en funcionamiento, por medio del uso de una base de datos PostgreSQL. Esto con el objetivo de poder registrar la hora exacta, la imagen obtenida, el tipo de modelo utilizado, el número de la cámara que hizo la detección y el sistema al que corresponde dicha cámara. Dicha estructura se puede observar en la Figura 27.

```
unlobjdet=# select * from detections;
 id | system | camera |       model        |       detection_time       |                                    image                                    | movement | person | validated
----+--------+--------+--------------------+----------------------------+-----------------------------------------------------------------------------+----------+--------+-----------
  1 |      1 |      1 | CUSTOM_YOLOV4_TINY | 2022-03-10 20:50:32.201231 | 2022-03-10_20-50-32-201231_HiLook-CCTV_camera1.png | t        | f      | f
  2 |      1 |      2 | CUSTOM_YOLOV4_TINY | 2022-03-10 20:50:37.709737 | 2022-03-10_20-50-37-709737_HiLook-CCTV_camera2.png | t        | f      | f
  3 |      1 |      2 | CUSTOM_YOLOV4_TINY | 2022-03-10 20:55:52.300495 | 2022-03-10_20-55-52-300495_HiLook-CCTV_camera2.png | t        | f      | f
  4 |      1 |      1 | CUSTOM_YOLOV4_TINY | 2022-03-10 21:00:53.983979 | 2022-03-10_21-00-53-983979_HiLook-CCTV_camera1.png | t        | f      | f
  5 |      1 |      2 | CUSTOM_YOLOV4_TINY | 2022-03-10 21:05:41.83686  | 2022-03-10_21-05-41-836860_HiLook-CCTV_camera2.png | t        | f      | f
  6 |      1 |      1 | CUSTOM_YOLOV4_TINY | 2022-03-10 21:06:43.148042 | 2022-03-10_21-06-43-148042_HiLook-CCTV_camera1.png | t        | f      | f
  7 |      1 |      2 | CUSTOM_YOLOV4_TINY | 2022-03-10 21:06:48.692449 | 2022-03-10_21-06-48-692449_HiLook-CCTV_camera2.png | t        | f      | f
```

Figura 27. Estructura de la tabla Detecciones de la base de datos
Fuente: Elaborado por el autor.

En la Figura 28 se presenta una muestra de los resultados obtenidos de las pruebas, donde se puede evidenciar algunas de las detecciones realizadas en distintas horas del día, aplicando énfasis a las aplicadas en el horario nocturno, cuándo la iluminación no es la óptima.

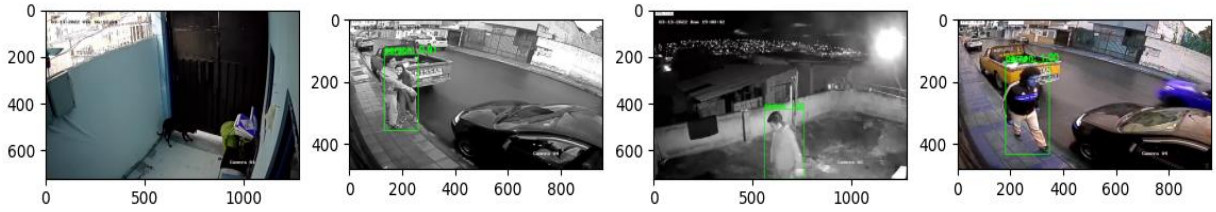


Figura 28. Imágenes tomadas de las pruebas de rendimiento realizadas al sistema
Fuente: Elaborado por el autor.

En base al modelo de detección YOLOv4-Tiny generado a partir del entrenamiento con un conjunto de datos, cuyos resultados se describen en la sección 6.1, se procedió a registrar cada una de las detecciones tomando como criterio de éxito que en cada uno de los frames captados, al menos exista una sola detección exitosa en un mismo entorno. Las estadísticas correspondientes se registran en la Tabla 8.

Tabla 8. Evaluación del sistema en general.

Entorno	# Muestras	Detecciones con éxito	Detecciones incorrectas	Precisión detección de Intrusiones	Confiabilidad de las detecciones
Entorno 1	144	136	08	0,9444	0,8987
Entorno 2	305	293	12	0,9607	0,8682
Entorno 3	396	382	14	0,9646	0,8813
Entorno 4	306	294	12	0,9608	0,8585
TOTAL	1151	1105	46	0,9600	0,8767

Los cuadros por segundo (FPS) en promedio registrados, fuero de aproximadamente 21 FPS. Se utilizó una computadora estándar con una arquitectura similar a los servidores web. **Fuente:** Elaborado por el autor.

Adicionalmente, fue preciso y pertinente realizar una comparativa de costos de la instalación de un nuevo sistema de CCTV que incluya la detección de intrusos en su sistema, contrastado a la implementación del sistema desarrollado en el presente trabajo de titulación, destacando que su aplicación puede suscitarse en la mayoría de dispositivos de grabación (DVR o NVR), siempre y cuando incluyan un acceso al material videográfico obtenido por las cámaras utilizadas.

Tomando como referencia a la proforma incluida en el Anexo 3, se establecieron los valores expuestos en la Tabla 9, lo cual resulta en un valor total de $656,92; correspondiente a

la instalación de un nuevo sistema de CCTV que incluya la detección de intrusos en los entornos vigilados, y, de ser el caso, ocasionar el posible reemplazo de los equipos que se encuentren instalado.

Tabla 9. Valor de costo de instalación de un sistema de CCTV con detección de personas

Descripción	Cantidad	Precio Unitario	Total
Hikvision – DVR iDS-7204HQHI-K1/2S	1	172,00	172,00
Cámara Hikvision tipo Bullet HWT-B223-M	4	58,00	232,00
Cable coaxial RG59 40 metros – RG59 + DC + Conectores BNC Hembra y Macho	8	18,00	144,00
Mano de obra especializada – Instalación por punto	4	27,23	108,92
		TOTAL	**656,92**

Costos de la implementación de un sistema de CCTV, a partir de los datos presentados en el **Anexo 3** y **Anexo 5**. **Fuente:** Elaborado por el autor.

Por último, en base a los Anexos 4 y 5 se establecieron los valores de las opciones de implementación, expuestos en las tablas 10 y 11, correspondientes a los gastos previstos para la instalación del sistema de detección desarrollado en el presente trabajo de titulación. Se puede observar que el valor total de los costos es menor a la prevista en la Tabla 9, facilitando la accesibilidad a la mayoría de usuarios que requieran una solución eficiente y de bajo costo.

Tabla 10. Cotización Servidor Web Remoto

Descripción	Cantidad	Precio Unitario	Total
Instancia en Public Cloud – 16GB RAM; 6v CPU Core; Almacenamiento SSD 400GB	12	11,99	143,88
Certificado SSL	1	89,99	89,99
Mano de obra especializada (JORNAL/HR)	10	2,44	24,43
		TOTAL	**258,30**

Cotización de la implementación de un servidor web a partir de la información presentada en los **Anexos 4** y **5**. **Fuente:** Elaborado por el autor.

Tabla 11. Cotización Servidor Web Local

Descripción	Cantidad	Precio Unitario	Total
NVIDIA Jetson Nano 2GB – GPU 128-core Maxwell	1	132,98	132,98
Mano de obra especializada (JORNAL/HR)	10	2,44	24,43
		TOTAL	**157,41**

Información de la implementación de un servidor local, elaborado a partir de los **Anexos 5** y **6**. **Fuente:** Elaborado por el autor.

7. Discusión

El proceso de entrenamiento de una red neuronal es una actividad que requiere una cantidad de tiempo considerable, si lo que se requiere es un modelo de detección con alta confiabilidad. El tiempo requerido para la preparación del modelo puede variar dependiendo de las características de la red neuronal, el hardware a disposición y la cantidad de información que se requiera para la construcción del conjunto de datos necesario para el aprendizaje y evaluación.

En ocasiones, la calidad impera por sobre la cantidad; es decir, no siempre la construcción de un conjunto de datos con una gran cantidad de imágenes es mejor; sino que al contrario, suele ser contraproducente en el proceso de entrenamiento de la red neuronal, dado que, si no se seleccionan adecuadamente, puede dar paso a efectos adversos tales como el sobreajuste a los datos (overfitting) ocasionando que la red no pueda generar nuevos datos, limitando su desempeño solamente a la información utilizada para el aprendizaje, tal como se presentó en los resultados del objetivo 1 con el conjunto de datos adquirido de OID.

Para el presente proyecto, el sistema generado basado en un modelo de detección de objetos YOLOv4, se logró alcanzar una precisión del 96% en los eventos de intrusión recolectados en las pruebas; incluyendo imágenes adquiridas donde no existe la presencia de una persona.

La versión YOLOv4-Tiny posee una baja precisión en lo que respecta a la detección, a comparación con su versión estándar, el cual contiene cada una de las capas convolucionales para establecer una detección con un alto grado de precisión. Su uso fue necesario, dado que el objetivo es alcanzar un algoritmo que sea capaz de trabajar de forma adecuada, sin exigir requerimientos de hardware estrictos para su correcto funcionamiento. Es por ello que la premisa de éxito en cada intrusión, no se fijó en una condición de detección continua de la persona; sino que, al contrario, se estableció que el sistema ha detectado satisfactoriamente la intrusión cuando se dispara el evento de alarma; lo cual se registró en la base de datos como un caso positivo en cada prueba.

A pesar de las limitaciones que contiene la versión ligera de YOLOv4, los datos demuestran que, con un conjunto de datos adecuado, se puede obtener un modelo de detección con un índice de precisión por encima de soluciones clásicas, tales como los sensores de movimiento o presencia.

Otras investigaciones abarcan un enfoque similar al presente trabajo de titulación, adquiriendo cierta similitud con los resultados obtenidos actualmente. Sabri & Li (2021) exponen una solución aplicada a sistemas de videovigilancia, donde los resultados que presentan, describen una precisión de un 99,4% para la detección de personas utilizando Fast R-CNN + Mobilenet, en donde se denota que la precisión impera por sobre la velocidad de reconocimiento, dado que la velocidad de detección se registra en un tiempo de 0,9 segundos (1 FPS aproximadamente). En otra aproximación, Kumar et al. (2020) exponen una comparación de los distintos modelos de detección utilizando conjuntos de datos de MS COCO y PascalVOC, cuyos resultados adquiridos con respecto a YOLO fueron de un mAP de 66,4% a 155 FPS y un 63,4% a 45 FPS respectivamente. Esto demuestra que existen investigaciones con distintos enfoques, en donde el equilibrio entre velocidad y precisión no es una tarea sencilla de lograr, por lo que generalmente las soluciones suelen ir en conjunto con hardware especializado para lograr un rendimiento óptimo en el ejercicio de sus funciones.

Otro de los puntos importantes que se cubrieron para lograr una optimización de uso de recursos computacionales, fue la forma de transmisión de los datos desde el servidor a la aplicación móvil, dado que comúnmente en este tipo de contexto se suele utilizar las peticiones HTTP clásicas. Para generar una solución de bajo costo, se requirió que exista una sola instancia de ejecución del modelo de detección por cada entorno vigilado, lo cual, al aplicarlo con peticiones HTTP significaría que por cada usuario conectado existan N instancias ejecutándose para N entornos vigilados, lo que supone una carga computacional innecesaria para el servidor que aloja el sistema de detección. Añadido a esto, sería necesario consultar cada cierto tiempo para poder obtener una actualización de estado de los sitios vigilados, que, para aplicaciones en tiempo real, significaría pérdida de información y aumentaría el índice de fallos en la detección de intrusiones.

Es por ello que se la implementación de WebSockets permitió superar los inconvenientes antes mencionados, cumpliendo los requerimientos de optimización planteados, manteniendo el enfoque de generar una solución de bajo costo y con un rendimiento óptimo.

8. Conclusiones

- En el presente trabajo de titulación se desarrolló un sistema de detección y reconocimiento de intrusos, basado en el procesamiento digital de señales, para la aplicación en sistemas de videovigilancia.
- La construcción de un conjunto de datos adecuado es primordial en la fase de entrenamiento de una red neuronal, dado que tiene un impacto significativo en la precisión y desempeño de la misma.
- La alteración aleatoria de las imágenes del conjunto de datos aplicada a la red neuronal durante el entrenamiento, establece un aumento de la precisión de inferencia dado que se cada vez se presenta un escenario distinto al anterior envuelto en la imagen, fortaleciendo el proceso de aprendizaje.
- La implementación de Websockets en el sistema desarrollado, permitió establecer una comunicación bidireccional entre la aplicación móvil y el servidor, dando paso a la utilización de una sola instancia de ejecución del modelo de detección por escenario vigilado, ahorrando de forma significativa recursos computacionales durante el despliegue del servicio.
- La aplicación del lenguaje de programación Kotlin para el desarrollo de la aplicación móvil, permitió un entorno versátil para la configuración de la transmisión de los datos, obteniendo así, una solución ligera para un funcionamiento óptimo en la mayoría de teléfonos inteligentes existentes en la actualidad.

9. Recomendaciones

- Para obtener resultados precisos, se recomienda utilizar un conjunto de datos compuesto por imágenes tomadas precisamente de sistemas de videovigilancia, dado que, bajo esas circunstancias, las imágenes no suelen tener una definición alta en comparación con imágenes obtenidas por cámaras especializadas, por ejemplo, de teléfonos inteligentes.
- En caso de poseer los recursos necesarios, se recomienda el uso de Google Colab para el entrenamiento de la red neuronal, dado que provee un espacio de procesamiento con GPU disponible, adecuado para este tipo de situaciones.
- En caso de requerirse, se recomienda utilizar imágenes que contengan distintos tipos de iluminación, tamaño y ángulos de rotación dado que permite aumentar la información de aprendizaje necesaria para un entrenamiento óptimo de la red neuronal.
- En caso de requerir una precisión estrictamente superior, y el presupuesto disponible es el suficiente, se recomienda utilizar la versión clásica de YOLOv4 en conjunto de la minicomputadora Jetson Nano, destinada específicamente para la ejecución de redes neuronales, a partir de la GPU integrada en su arquitectura.
- Se recomienda el uso de Kotlin por sobre Flutter para aplicaciones que requieran actualización constante de sus componentes, dado que presenta una arquitectura versátil para la implementación de procesos complejos.
- Para establecer una conexión segura en la transmisión de datos a través de WebSockets, es recomendable habilitar certificados SSL, que permiten el cifrado de los datos en ambas direcciones de la comunicación.

10. Bibliografía

Alvarado Moya, J. P. (2012). *Procesamiento y Análisis de Imágenes Digitales*. http://www.ie.tec.ac.cr/palvarado/PAID/paid.pdf

Barot, S. (2021). *Kotlin vs Flutter – Comparision of Popularity*. Glowid. https://aglowiditsolutions.com/blog/kotlin-vs-flutter/

Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). *YOLOv4: Optimal Speed and Accuracy of Object Detection*. https://arxiv.org/abs/2004.10934v1

Datta, P. (2020, September). *Índice de Similitud Estructural (SSIM)*. https://medium.com/srm-mic/all-about-structural-similarity-index-ssim-theory-code-in-pytorch-6551b455541e

Dirección de Estadística y Sistemas de Información - FGE. (2021). *Ecuador: Las cifras de robos*. Fiscalía General Del Estado FGE Ecuador. https://www.fiscalia.gob.ec/estadisticas-de-robos/

Everingham, M., Van Gool, L., Williams, C., Winn, J., & Andrew, Z. (2009). The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Networks and Communications Security*, *2*, 1–34. https://homepages.inf.ed.ac.uk/ckiw/postscript/ijcv_voc09.pdf

Fette, I., & Melnikov, A. (2011). *RFC 6455 The WebSocket Protocol*. https://datatracker.ietf.org/doc/html/rfc6455

Gandhi, R. (2018). *R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms*. Towards Data Science. https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. https://doi.org/10.1109/CVPR.2014.81

Guenther, N., & Schonlau, M. (2016). Support Vector Machine. *The Stata Journal*, *16*, 917–937. https://doi.org/10.1177/1536867x1601600407

Guevara Patiño, R. (2016). El estado del arte en la investigación: ¿análisis de los conocimientos acumulados o indagación por nuevos sentidos? *Folios*, *1*(44), 165–179.

https://doi.org/10.17227/01234870.44FOLIOS165.179

Imatest. (2021). *SSIM: Structural Similarity Index | imatest*. https://www.imatest.com/docs/ssim/

InVerita. (2020). *Flutter vs React Native vs Native: Deep Performance Comparison.* https://medium.com/swlh/flutter-vs-react-native-vs-native-deep-performance-comparison-990b90c11433

Isasi Viñuela, P., & León Galván, I. (2004). *Redes de Neuronas Artificiales. Un enfoque práctico* (Primera). PEARSON Prentice Hall.

Karmarkar, T. (2018). *Region Proposal Network (RPN) — Backbone of Faster R-CNN*. Egen Engineering & Beyond. https://medium.com/egen/region-proposal-network-rpn-backbone-of-faster-r-cnn-4a744a38d7f9

Khan, P. W., Byun, Y. C., & Park, N. (2020). A data verification system for cctv surveillance cameras using blockchain technology in smart cities. *Electronics (Switzerland)*, *9*(3). https://doi.org/10.3390/electronics9030484

Kitamura, E., & Ubl, M. (2010). *El problema: conexiones de baja latencia cliente-servidor y servidor-cliente*. Introducción a Los WebSockets. https://www.html5rocks.com/es/tutorials/websockets/basics/

Kumar, A., Zhang, Z. J., & Lyu, H. (2020). Object detection in real time based on improved single shot multi-box detector algorithm. *Eurasip Journal on Wireless Communications and Networking*, *2020*(1). https://doi.org/10.1186/s13638-020-01826-x

Lendave, V. (2021, August). *Using Background Subtraction Methods in Image Processing*. Developers Corner. https://analyticsindiamag.com/using-background-subtraction-methods-in-image-processing/

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *9905 LNCS*, 21–37. https://doi.org/10.1007/978-3-319-46448-0_2

Lopera, J., Ramírez, C., Zuluaga, M., & Ortiz, J. (2010). El método analítico como método natural. *Nómadas. Critical Journal of Social and Juridical Sciences*. https://www.redalyc.org/articulo.oa?id=18112179017

Masdiyasa, I. G. S., Budiwitjaksono, G. S., M, H. A., Sampurno, I. A. W., & Mandenni, N. M. I. M. (2020). Graph-QL Responsibility Analysis at Integrated Competency Certification Test System Base on Web Service. *Lontar Komputer : Jurnal Ilmiah Teknologi Informasi*, *11*(2). https://doi.org/10.24843/lkjiti.2020.v11.i02.p05

Massiris, M., Delrieux, C., & Fernández, J. Á. (2020). *Detección de equipos de protección personal mediante red neuronal convolucional YOLO*. https://doi.org/10.17979/spudc.9788497497565.1022

Mateus, B. G., & Martinez, M. (2018). An Empirical Study on Quality of Android Applications written in Kotlin language. *Empirical Software Engineering*, *24*(6), 3356–3393. https://doi.org/10.1007/s10664-019-09727-4

Mejía Vilet, J. R. (2005). *Procesamiento Digital de Imágenes*. http://laurence.com.ar/artes/comun/Apuntes procesamiento digital de imagenes.pdf

Miao, X., Liu, X., Chen, J., Zhuang, S., Fan, J., & Jiang, H. (2019). Insulator detection in aerial images for transmission line inspection using single shot multibox detector. *IEEE Access*, *7*. https://doi.org/10.1109/ACCESS.2019.2891123

Monje, C. (2011). *Metodología de la Investigación Cuantitativa y Cualitativa - Guía Didáctica*. https://www.uv.mx/rmipe/files/2017/02/Guia-didactica-metodologia-de-la-investigacion.pdf

Moreira, D. (2021). *Aplicación de un modelo de reconocimiento de objetos utilizando YOLO* [Universidad Estatal Península de Santa Elena]. https://repositorio.upse.edu.ec/bitstream/46000/5755/1/UPSE-TTI-2021-0008.pdf

Moreno, A. (2019). Clasificación de imágenes usando redes neuronales convolucionales en Python. *Universidad de Sevilla*, 80. https://idus.us.es/bitstream/handle/11441/89506/TFG-2402-ARTOLA.pdf

Nalepa, J., & Kawulok, M. (2019). Selecting training sets for support vector machines: a review. In *Artificial Intelligence Review* (Vol. 52, Issue 2). https://doi.org/10.1007/s10462-017-9611-1

Olveres, J., González, G., Torres, F., Moreno-Tagle, J. C., Carbajal-Degante, E., Valencia-Rodríguez, A., Méndez-Sánchez, N., & Escalante- Ramírez, B. (2021). What is new in computer vision and artificial intelligence in medical image analysis applications. In *Quantitative Imaging in Medicine and Surgery* (Vol. 11, Issue 8).

https://doi.org/10.21037/qims-20-1151

Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(6). https://doi.org/10.1109/TPAMI.2016.2577031

Sabri, Z. S., & Li, Z. (2021). Low-cost intelligent surveillance system based on fast CNN. *PeerJ Computer Science*, *7*, 1–23. https://doi.org/10.7717/PEERJ-CS.402

Soewito, B., Christian, Gunawan, F. E., Diana, & Gede Putra Kusuma, I. (2019). Websocket to support real time smart home applications. *Procedia Computer Science*, *157*. https://doi.org/10.1016/j.procs.2019.09.014

Trnovszký, T., Sýkora, P., & Hudec, R. (2017). Comparison of Background Subtraction Methods on Near Infra-Red Spectrum Video Sequences. *Procedia Engineering*, *192*, 887–892. https://doi.org/10.1016/J.PROENG.2017.06.153

Vigren, A. (2020). *Multiple methods for motion detection*. Axis Communications. https://www.axis.com/blog/secure-insights/motion-detection/

Wilfred, T. (2020). *Flutter vs React Native*. https://dev.to/thomaswilfred15/flutter-vs-react-native-535j

Wu, W. (2018). React Native vs Flutter, cross-platform mobile application frameworks. *Metropolia University*, *March*. https://www.theseus.fi/bitstream/handle/10024/146232/thesis.pdf

Yohanandan, S. (2020). *mAP (mean Average Precision)*. https://towardsdatascience.com/map-mean-average-precision-might-confuse-you-5956f1bfa9e2

11. Anexos

Anexo 1: Manual del programador y uso de la API web

UNIVERSIDAD NACIONAL DE LOJA

CARRERA DE INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

Manual Técnico: API Web

Proyecto:

Sistema de detección y reconocimiento de intrusos de bajo costo basado en visión artificial, aplicado a sistemas de videovigilancia

---

AUTOR:

Jorge Enrique Ortega Jaramillo

Estudiante CIEYT

1. Introducción

El presente documento contiene información relacionada a la instalación y uso del sistema web, y los procesos relacionados con el manejo de consultas tanto a la API como a la base de datos, con el propósito de establecer una guía en el caso de una implementación en conjunto con un sistema de CCTV.

1.1. Objetivo

Detallar los procesos de uso e instalación del sistema web, para su despliegue y uso en conjunto con un sistema de videovigilancia y la aplicación móvil.

1.2. Requisitos previos

Como requisito principal, se requiere obtener el sistema web a través del repositorio digital alojado en Github.

Existen 2 formas de obtener el repositorio. Una manera es ejecutando el siguiente comando a través de la consola de Windows, Linux o Git (previamente se requiere instalar Git, lo cual se puede realizar descargándolo de la página web *https://git-scm.com/downloads*):

```
git clone https://github.com/PotatoBhk/server_tesis_unl.git
```

Otra forma es accediendo directamente al repositorio a través del enlace *https://github.com/PotatoBhk/server_tesis_unl* y descargando como se indica en la Figura 29.

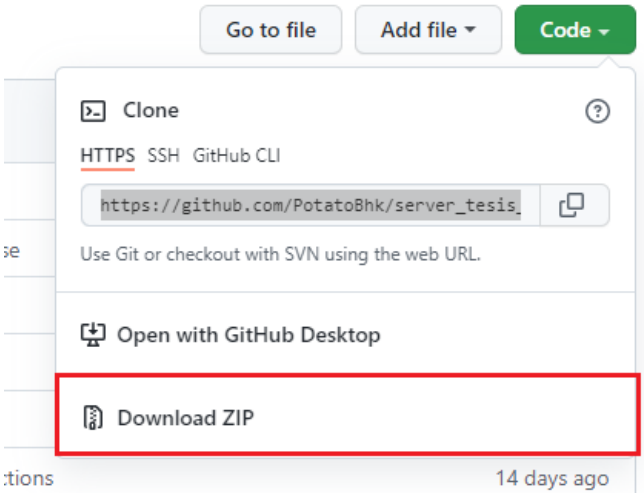


Figura 29. Descarga del repositorio de Github
Fuente: Elaborado por el autor.

Los requisitos necesarios para la ejecución de los entornos y scripts se enlistan a continuación:

- PostgreSQL 11.0
- Python 3.9 o superior
- opencv-python 4.5 o superior

- imutils 0.5.4 o superior
- numpy 1.22 o superior
- scikit-image 0.19.2 o superior
- enum 0.4.7 o superior
- functools 0.5 o superior
- Flask-SocketIO 5.1.1
- gevent 21.12
- eventlet 0.33.0.0
- psycopg2 2.9.3

Todos los requisitos listados anteriormente se encuentran especificados en el documento ***requerimientos.txt*** (exceptuando PostgreSQL) ubicado en el repositorio del proyecto en cuestión, y se pueden instalar a través del siguiente comando:

```
pip install -r requerimientos.txt
```

O en su defecto, si requiere una instalación individual de cada una de las librerías, puede ejecutar el siguiente comando, reemplazando ***librería*** por el nombre del paquete requerido:

```
pip install libreria
```

Finalmente, otro de los requerimientos importantes es copiar los archivos del modelo entrenado YOLOv4. Los archivos necesarios para la detección de intrusos (reconocimiento de personas), se encuentran incluidos dentro del repositorio digital.

2. Instalación PostgreSQL y creación de la base de datos.

Uno de los requerimientos primordiales para el correcto funcionamiento de la aplicación web es tener instalado el gestor de base de datos PostgreSQL. La instalación se la realiza dependiendo del sistema operativo del servidor en donde se vaya a alojar la aplicación; por lo que para Linux se deben ejecutar los siguientes comandos:

```
$ sudo apt-get update

$ sudo apt-get -y install postgresql
```

Después de la instalación, el servicio debe estar habilitado y ejecutándose. Esto se puede verificar ingresando el siguiente comando:

```
$ sudo systemctl status postgresql.service
```

Se debe presentar un mensaje de estado similar al siguiente ejemplo:

```
$ sudo systemctl status postgresql.service
● postgresql.service - PostgreSQL RDBMS
     Loaded: loaded (/lib/systemd/system/postgresql.service; enabled; vendor
preset: enabled)
```

```
     Active: active (exited) since Sun 2021-06-06 16:06:45 UTC; 46s ago
   Main PID: 3364 (code=exited, status=0/SUCCESS)
      Tasks: 0 (limit: 1113)
     Memory: 0B
     CGroup: /system.slice/postgresql.service

Jun 06 16:06:45 ubuntu20cloud systemd[1]: Starting PostgreSQL RDBMS...
Jun 06 16:06:45 ubuntu20cloud systemd[1]: Finished PostgreSQL RDBMS.
$
```

Entonces para acceder al contexto de PostgreSQL y comenzar con la creación de la base de datos, se procede a ejecutar el siguiente comando:

```
$ sudo su - postgre
```

Por otro lado, la instalación en Windows se puede realizar de varias maneras, una de ellas y la más sencilla es hacerlo mediante Laragon. El proceso se detalla a continuación:

a. Descargar el instalador desde la página web de Laragon (***https://laragon.org/download/index.html***).
b. Al terminar la descarga, ejecutar el archivo y seguir los pasos de instalación establecidos en el siguiente enlace: ***https://laragon.org/docs/install.html***.
c. Una terminada la instalación de Laragon, se ejecuta el programa.
d. Se procede a instalar PostgreSQL a través del menú contextual siguiendo la ruta **Menú > Herramientas > Quick add > postgresql-11** tal como se muestra en la Figura 30.

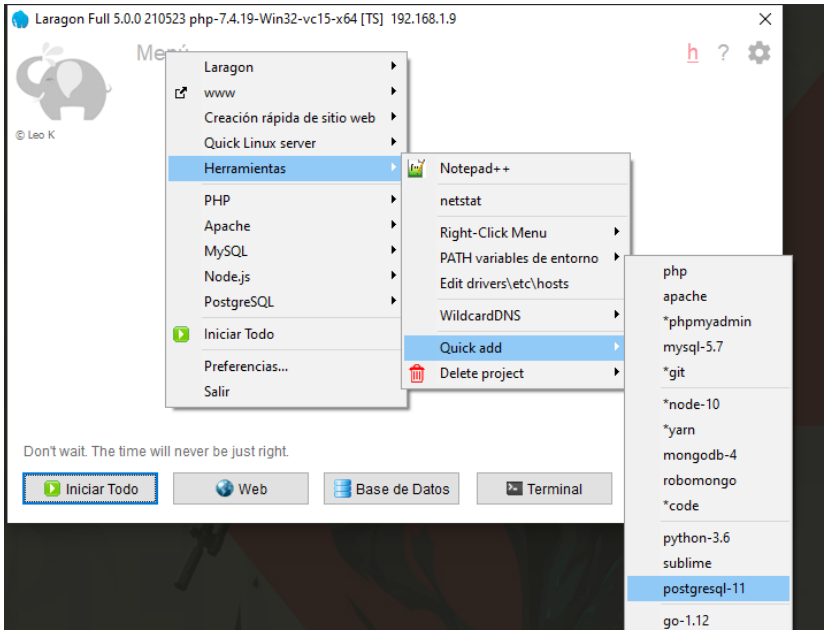


Figura 30. Instalación de PostgreSQL a través de Laragon.
Fuente: Elaborado por el autor.

e. Una vez terminada la instalación, se inician los servicios y se ejecuta la terminal de Laragon para continuar con la configuración y administración de la base de datos.

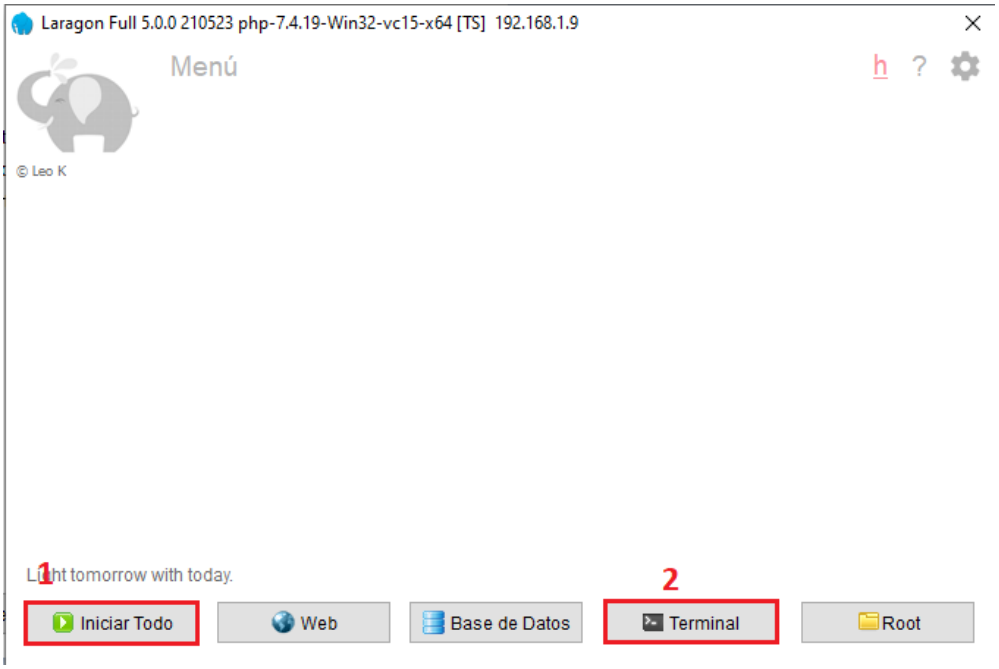


Figura 31. Ejecución de los servicios y terminal de Laragon
Fuente: Elaborado por el autor.

2.1. Creación de la base de datos

Una vez instalado PostgreSQL, dentro de la consola, ya sea de Linux o de Laragon, se procede a ejecutar los siguientes comandos:

```
$ psql
$ CREATE USER cieyttesis WITH ENCRYPTED PASSWORD 'mpassword';
$ CREATE DATABASE unlobjdet OWNER cieyttesis;
$ GRANT ALL PRIVILEGES ON DATABASE unlobjdet TO cieyttesis;
```

Una vez ejecutado, se verificará que la base de datos esté correctamente creada, estableciendo una conexión en la misma ventana de comandos tal como se indica:

```
$ \c unlobjdet
```

Por lo que debería mostrarse el siguiente mensaje, si todo está correctamente instalado:

```
Ahora está conectado a la base de datos «unlobjdet» con el usuario «user».
unlobjdet=#
```

3. Ejecución y uso de la aplicación web.

3.1. Ejecución de la API Web

Finalmente, para ejecutar la aplicación web, si se lo realiza en Linux, se debe ejecutar el siguiente comando a través de la consola:

```
$ nohup python main.py > log.out 2>&1 &
```

Este comando permitirá ejecutar en segundo plano la aplicación, además de redirigir todas las salidas correspondientes al archivo ***log.out***.

Para observar los cambios que se realicen en el archivo ***log.out*** se puede ejecutar el siguiente comando:

```
$ tail -f log.out
```

Por otra parte, en Windows, se puede ejecutar el programa a través de la consola del propio sistema operativo, a través del siguiente comando:

```
$ python main.py
```

3.2. Uso de la API Web

Se han especificado 2 endpoints para la facilitar la adición y configuración de sistemas de videovigilancia, en donde se requiera aplicar la solución.

Cada uno de estos endpoints se lo debe realizar a partir de una petición de tipo POST, cuyo cuerpo debe contener un archivo JSON con la estructura especificada en la Figura 32.

Figura 32. Estructura del cuerpo de la petición para Añadir/Editar un sistema de videovigilancia a la aplicación web.
Fuente: Elaborado por el autor.

El parámetro ***link*** hace referencia al enlace de donde se va a obtener la fuente de vídeo, cuya opción **{ch}** se debe colocar justo en donde iría especificado el número de canal correspondiente a una cámara. Esto con el fin de que, según el número de cámaras especificadas, ese parámetro vaya variando y pueda obtener todas las fuentes de vídeo disponibles en el sistema.

El parámetro ***model*** hace referencia al tipo de modelo de detección YOLOv4 se va a utilizar. Las opciones disponibles son:

```
PRETRAINED_YOLOV4 = 1
PRETRAINED_YOLOV4_TINY = 2
CUSTOM_YOLOV4_TINY = 3
```

Si se añadir archivos adicionales, es posible modificando la línea 18 a 21 del archivo ***video_streaming.py***. Cada opción debe referenciar al nombre de los archivos del modelo YOLOv4 que se esté utilizando. Por ejemplo, ***CUSTOM_YOLOV4_TINY*** hacer referencia a

los archivos con el mismo nombre ***CUSTOM_YOLOV4_TINY.weights*** y ***CUSTOM_YOLOV4_TINY.cfg*** respectivamente.

El listado de endpoints se especifican a continuación:

- Agregar un sistema: ***http://<direccion_web>/api/add_system***
- Editar un sistema: ***http://<direccion_web>/api/update_system***

Todos con la misma estructura especificada en la Figura 32, y envueltos en una petición de tipo POST.

Anexo 2: Proceso de entrenamiento del modelo yolov4

UNIVERSIDAD NACIONAL DE LOJA

CARRERA DE INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

Proceso de entrenamiento del modelo YOLOv4

Proyecto:

Sistema de detección y reconocimiento de intrusos de bajo costo basado en visión artificial, aplicado a sistemas de videovigilancia

---

AUTOR:

Jorge Enrique Ortega Jaramillo

Estudiante CIEYT

1. Introducción

El presente documento contiene información relacionada al uso de los modelos de detección YOLOv4 y los procesos necesarios a seguir para llevar a cabo el correcto entrenamiento de los mismos, para generar soluciones acordes a las necesidades del usuario; con el fin de proveer un medio por el cual, el investigador que lo requiera, tenga presente una base en la que se pueda basar para desarrollar nuevas soluciones basadas en visión artificial.

1.1. Objetivo

El propósito del presente documento es detallar el proceso a seguir para realizar el entrenamiento de un modelo de reconocimiento de objetos YOLOv4, así como las distintas actividades que permitan la recolección adecuada de un conjunto de datos que sirvan información de entrada para el aprendizaje de la red neuronal.

1.2. Alcance

Este documento va dirigido a investigadores, estudiantes de la Universidad Nacional de Loja y público en general, con el propósito de proveer un instructivo que permita establecer un punto de partida hacia una investigación y/o aplicaciones distintas a las expuestas en el presente proyecto de Titulación.

1.3. Requisitos previos

Se requiere como primer punto, clonar el repositorio para obtener los diferentes algoritmos necesarios para la construcción del conjunto de datos y la evaluación del modelo generado posterior al proceso de entrenamiento.

Existen 2 formas de clonar el repositorio, una manera es ejecutando el siguiente comando a través de la consola de Windows, Linux o Git (previamente se requiere instalar Git, lo cual se puede realizar descargándolo de la página web ***https://git-scm.com/downloads***):

```
git clone https://github.com/PotatoBhk/ia_tesis_unl.git
```

Otra forma es accediendo directamente al repositorio a través del enlace ***https://github.com/PotatoBhk/ia_tesis_unl*** y descargando como se indica en la Figura 33.

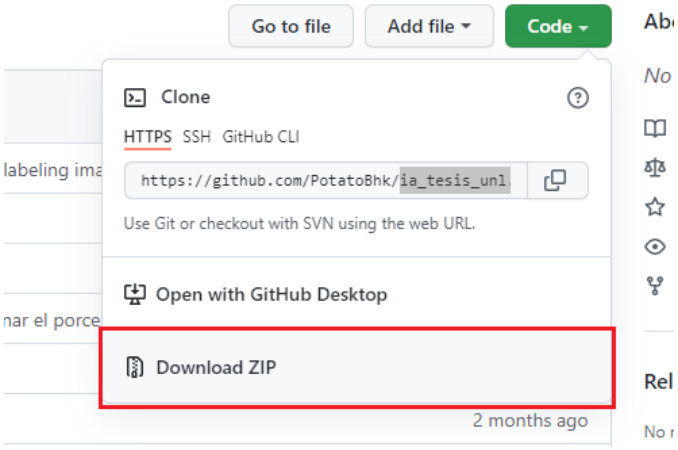


Figura 33. Descarga del repositorio de Github
Fuente: Elaborado por el autor.

Los requisitos necesarios para la ejecución de los diferentes scripts y entornos se enlistan a continuación:

- Python 3.9
- opencv-python 4.5 o superior
- imutils 0.5.4 o superior
- tqdm 4.63 o superior
- numpy 1.22 o superior
- matplotlib 3.5.1 o superior
- argparse 1.4.0 o superior
- scikit-image 0.19.2 o superior
- Pillow 9.0.1 o superior
- functools 0.5 o superior

Todos los requisitos listados anteriormente se encuentran especificados en el documento ***requerimientos.txt*** ubicado en el repositorio del proyecto en cuestión, y se pueden instalar a través del siguiente comando:

```
pip install -r requerimientos.txt
```

O en su defecto, si requiere una instalación individual de cada una de las librerías, puede ejecutar el siguiente comando, reemplazando ***librería*** por el nombre del paquete requerido:

```
pip install libreria
```

2. Generación del conjunto de datos

2.1. Consideraciones generales

Para generar el conjunto de datos es necesario tener las siguientes consideraciones:

- Obtener imágenes en donde cada uno de los objetos que se desea detectar, esté adecuadamente etiquetado. Ningún objeto en su **conjunto de datos** debe estar sin etiqueta. El proceso de etiquetación se explicará a detalle más adelante.
- En la mayoría de los problemas que surgen al entrenar un modelo de detección, se debe a que existen etiquetas incorrectas en el **conjunto de datos**. Una vez seguido el proceso de etiquetado de la sección 2.2.2, es recomendable verificar el **conjunto de datos** utilizando la herramienta **labelImg**.
- Para cada uno de los objetos a detectar, debe haber al menos un objeto similar en el **conjunto de datos** de entrenamiento, así mismo estos deben incluir características variadas en cada escenario presentado en las imágenes, tales como: diferentes escalas, rotaciones, iluminaciones, desde diferentes lados, en diferentes fondos.
- Es recomendable que por cada clase de objeto se obtengan al menos 2000 imágenes diferentes o más. Este criterio puede diferir según la calidad de información que reúna el **conjunto de datos**, dado que se pueden utilizar una cantidad inferior de imágenes, siempre y cuando exista la información suficiente para que el aprendizaje se pueda dar de la mejor forma posible.
- El **conjunto de datos** puede incluir imágenes con objetos no etiquetados que no se desea detectar. Estos objetos pueden estar combinados con clases que sí se desea detectar para aumentar la información de aprendizaje, o en su defecto, pueden ser imágenes sin etiquetar (con archivos **.txt** vacíos) que contengan dichas clases no deseadas. Este tipo de imágenes se denominan como ***muestras negativas***. Se pueden utilizar tantas muestras negativas como sea posible, siempre y cuando también existan en una proporción mayor, imágenes que contengan las clases que se desee detectar.
- Si se desea etiquetar las imágenes utilizando solamente la herramienta de **labelImg** debe tener en cuenta que debe marcar el objeto dentro de la imagen como desea que se detecte en el sistema (pudiendo ser solo la parte visible del objeto, como la parte visible y superpuesta de dicho objeto).

2.2. Procedimiento para generar el conjunto de datos.

2.2.1. Adquisición de imágenes

Existen varias formas de obtener las imágenes necesarias para la construcción del conjunto de datos. Una de las formas es descargarlas directamente de la colección de imágenes

seleccionadas por las distintas organizaciones que se dedican a la investigación y desarrollo de modelos de clasificación, reconocimiento y segmentación de objetos. Los más populares son:

- MS COCO (***https://cocodataset.org/***)
- Open Images Dataset (***https://storage.googleapis.com/openimages/web/index.html***)
- ImageNet (***https://image-net.org/***)
- PASCAL Visual Object Classes (VOC) (***http://host.robots.ox.ac.uk/pascal/VOC/***)

Proveen variedad de imágenes que pueden ser utilizadas para entrenar cualquier modelo, con distintos tipos de clases, según sea necesario.

Por otro lado, si se requiere construir un conjunto de datos conformado por imágenes obtenidas por recursos multimedia, tales como vídeo, para obtener una mejor personalización en el aprendizaje de la red neuronal, es necesario seguir los siguientes pasos:

a. Descargar los videos de donde se van a obtener las imágenes y ubicarlos dentro de la carpeta ***~/ia_tesis_unl/training/vids/*** y ejecutar el script ***get_dataset.py***. Al ejecutar se observará un indicador de proceso tal como se muestra la Figura 34. El algoritmo se encargará de filtrar imágenes en donde se encuentre la presencia de los objetos a detectar. El procedimiento puede tardar varias horas, dependiendo de la resolución de los vídeos a analizar y la capacidad de procesamiento del hardware en donde se esté ejecutando.

   El algoritmo está programado para comparar cada uno de los cuadros de imagen del vídeo analizado, para guardar los que tengan una diferencia entre el actual y el anterior de un 75%. Esto funciona perfectamente con imágenes que tengan un escenario estático; para escenarios en donde la cámara esté móvil, se puede calibrar este condicional modificando la línea 74 del archivo en cuestión, fijando valores en un rango de 0.0 a 1.0. Por otro lado, para establecer una clase diferente a la de ***persona***, se debe modificar la línea 51 en base al número correspondiente al índice de la clase deseada establecido en el archivo ***~/ia_tesis_unl/sources/yolo/coco.names*** en un rango de 0 a 79.

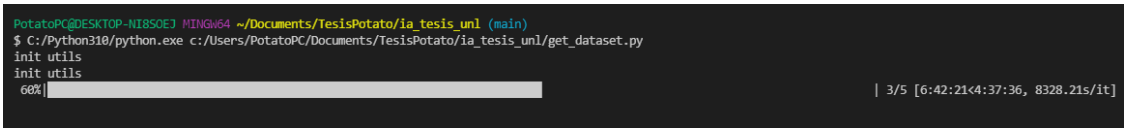


Figura 34. Proceso de ejecución del script get_dataset.py
Fuente: Elaborado por el autor.

b. Una vez terminado el proceso, lo siguiente es etiquetar cada uno de los objetos que se desea etiquetar, en cada una de las imágenes adquiridas; si se desea etiquetar cada

imagen de forma manual, se recomienda utilizar la herramienta **labelImg**, cuyas instrucciones de uso se especifican en la sección 2.2.2. Por otro lado, si se desea realizar este proceso de forma automática, es posible con la ejecución del script ***labelling_1.5.py***. En la Figura 35 se observa un ejemplo de ejecución en conjunto con los indicadores del procedimiento. Este algoritmo detectará en cada una de las imágenes los objetos de la clase requerida para el entrenamiento del modelo dentro de las imágenes seleccionadas para su análisis.

De la misma forma, para detectar una clase diferente a la de ***persona*** se debe modificar la línea 44 con un valor en el rango de 0 a 79, según la clase requerida de las disponibles en el archivo ***~/ia_tesis_unl/sources/yolo/coco.names***.

Figura 35. Generación de etiquetas siguiendo el formato establecido en YOLOv4.
Fuente: Elaborado por el autor.

c. Al final de este proceso se podrá observar todas las imágenes generadas en conjunto con las anotaciones en el formato que se establece para YOLOv4 en la carpeta de destino ***~/ia_tesis_unl/training/result/***. Se habrán creado dos subcarpetas que contienen un conjunto de datos tanto para entrenamiento (carpeta ***outs***) como para validación (carpeta ***tests***). Dicho formato se almacenará en un archivo **.txt** con el mismo nombre de la imagen a la que pertenecen los objetos a detectar, el cual, describe de la siguiente forma:

`<object-class> <x_center> <y_center> <width> <height>`

Donde, `<x_center>` y `<y_center>` hacen referencia al punto central del objeto y; `<width>` y `<height>` representan las dimensiones del objeto dentro de la imagen en cuestión. Todos estos valores se encuentran normalizados a un valor en el rango de 0 a 1.0.

Como recomendación adicional, para establecer un orden adecuado cuando se requiere detectar una o más clases, se puede modificar el archivo ***coco.names*** dejando solamente las que se requieran; permitiendo así establecer un índice de 0 a N – 1, donde N sería el número de clases a detectar.

Una vez generado completamente el conjunto de datos, se procede con la validación del mismo siguiendo el procedimiento de la siguiente sección, según sea necesario.

2.2.2. Validación del conjunto de datos

Dado que es necesario que todos los objetos dentro del conjunto de datos estén correctamente etiquetados, se requiere realizar una verificación por cada una de las imágenes, revisando que todo esté en orden. Debido a que el proceso de etiquetación descrito en la sección anterior no es perfecto, suelen incurrir algunas fallas tales como la ausencia de etiqueta de objetos no detectados pero que requieren tener esta característica.

Para validar que todas las etiquetas se encuentren correctamente generadas, se utilizará la herramienta **labelImg**, cuyas instrucciones de uso se exponen a continuación:

a. Agregar un archivo .txt en cada una de las carpetas generadas del conjunto de datos, especificando las clases en orden adecuado de acuerdo al índice establecido en la sección anterior. En la Figura 36 se presenta un ejemplo del contenido del archivo .txt.

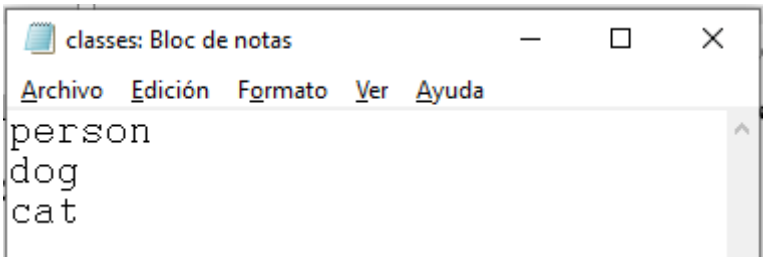


Figura 36. Clases a especificar previo a la ejecución de labelImg.
Fuente: Elaborado por el autor.

b. Como siguiente paso, se procede a ejecutar el siguiente comando a través de la consola de Windows o de Linux:

```
labelImg ./ classes.txt
```

Cabe destacar que la consola debe estar apuntando a la carpeta en donde se encuentren las imágenes a analizar.

c. Se presentará una ventana similar a la de la Figura 37, en donde, de ser el caso se presionará el botón que se encuentra en el menú izquierdo, arriba del botón **Create RectBox**, hasta que el formato de anotaciones se haya fijado en **YOLO**.

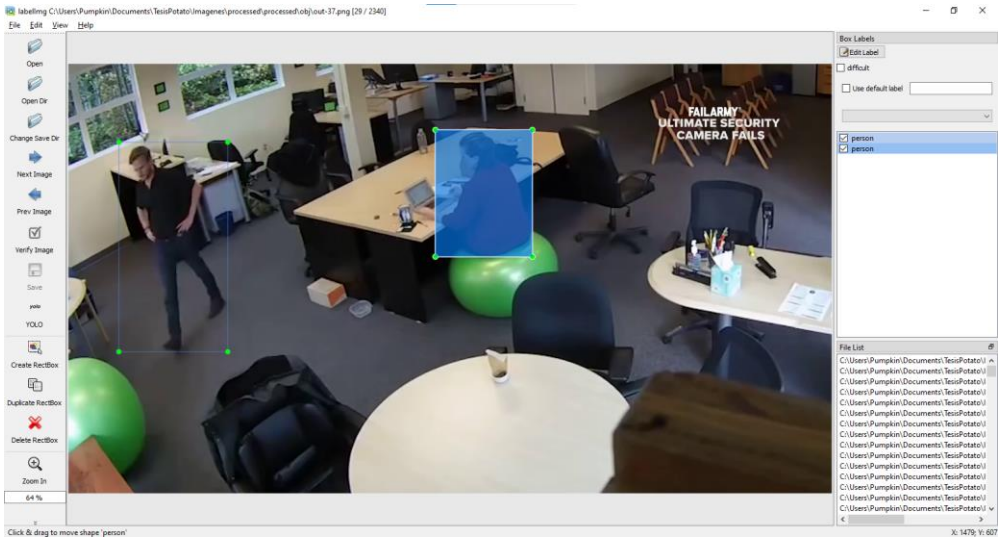


Figura 37. Interfaz gráfica de la herramienta labelImg.
Fuente: Elaborado por el autor.

d. Se procederán a hacer los ajustes que se requieran, de ser necesario, en los cuadros delimitadores de los objetos detectados en las imágenes, y, en caso de existir la ausencia de alguna etiqueta, crearla seleccionando la opción **Create RectBox**.

e. Luego de realizar los ajustes pertinentes, se podrá navegar a través de las imágenes mediante los botones **Next Image** y **Prev Image**, hasta que se verifique que todas las imágenes estén correctamente etiquetadas.

3. Entrenamiento del modelo YOLOv4

Dentro del proceso de entrenamiento de YOLOv4, existe la posibilidad de seleccionar uno de los dos tipos de modelos disponibles: la versión full o completa, o la versión ligera (YOLOv4-Tiny).

Para el caso de aplicaciones en tiempo real y de bajo costo, se recomienda seleccionar la versión YOLOv4-Tiny dado que el modelo tiene un rendimiento óptimo en arquitecturas que no cuenten con un GPU dedicado para la inferencia de datos.

El requisito previo para poder continuar con el proceso de entrenamiento es el haber compilado e instalado los archivos necesarios para la generación de las redes neuronales para la detección de objetos cuyo enlace es ***https://github.com/AlexeyAB/darknet*** (Bochkovskiy et al., 2020)

La instalación se detalla en las secciones:

- Para Windows: ***https://github.com/AlexeyAB/darknet#how-to-compile-on-windows-using-cmake***.
- Para Linux y macOS: ***https://github.com/AlexeyAB/darknet#how-to-compile-on-linuxmacos-using-cmake***.

Bochkovskiy et al., (2020) detallan el proceso de entrenamiento a través de su repositorio de Github, como se explica a continuación:

a. Se renombran las carpetas del conjunto de datos a ***obj*** para las imágenes destinadas para el entrenamiento y a ***test*** para las imágenes destinadas para pruebas y validación.
b. Se copia las carpetas del conjunto de datos a la carpeta ***~/darknet/data/***.
c. Como requerimiento adicional, se deben generar los archivos **train.txt** y **test.txt**, los cuales tienen el listado de todas las imágenes que conforman el conjunto de datos. Para ello, se debe copiar los archivos ***generate_traintxt.py*** y ***generate_testtxt.py*** ubicados en la carpeta ***~/ia_tesis_unl/training/tests/*** en ***~/darknet/*** y ejecutarlos.
d. Luego, se deben crear los archivos **obj.data** y **obj.names** dentro del directorio ***~/darknet/data***, como se indica en la Figura 38, modificando los parámetros **classes** y **backup** según corresponda, y agregando el nombre de las clases a detectar respectivamente.

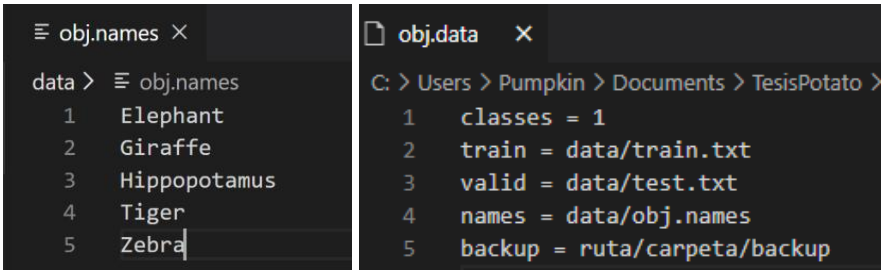


Figura 38. Ejemplo de configuración de archivos obj.names y obj.data
Fuente: Adaptado de *YOLOv4: Optimal Speed and Accuracy of Object Detection*, de A. Bochkovskiy & H. Mark Liao, 2020.

e. Se procede a crear un nuevo archivo de configuración. Esto se puede simplificar copiando el archivo ***~/darknet/cfg/yolov4-custom.cfg*** en el caso de YOLOv4, o ***~/darknet/cfg/yolov4-tiny-custom.cfg*** en el caso de YOLOv4-Tiny en el mismo directorio y se renombra el archivo a ***yolov4-obj.cfg***.
f. Como siguiente paso, se modifica el archivo de configuración estableciendo los valores de **batch = 64**, **subdivisions = 16**. Se establece el valor de **max_batches** siguiendo la fórmula **max_batches = classes * 2000** (cuyo valor mínimo es 6000), el valor de **steps** a partir del 80% y 90% de **max_batches** respectivamente (para 6000 sería **steps = 4800, 5400**), el valor de **classes** de acuerdo al número de clases que se deseen detectar, y por último el parámetro **filters** se establece con la fórmula **filters = (classes + 5) x 3**. Este último parámetro se debe modificar solamente en cada una de las capas **[convolutional]** ubicadas antes de cada capa **[yolo]**.

g. Se procede a descargar el archivo **.weigths** pre-entrenado según el tipo de modelo que se esté aplicando:

   - ***https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v4_pre/yolov4-tiny.conv.29*** para YOLOv4-Tiny
   - ***https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v3_optimal/yolov4.conv.137*** para YOLOv4 Full

   Es opcional el uso de estos archivos, pero es recomendable, dado que agilizan de sobremanera el entrenamiento del modelo en cuestión.

h. Finalmente se ejecuta el siguiente comando para comenzar el entrenamiento de la red neuronal:

```
./darknet detector train data/obj.data cfg/yolov4-obj.cfg yolov4.conv.137 -dont_show -map
```

   Este proceso puede tomar varias horas, dependiendo de las iteraciones que se hayan configurado en el archivo ***yolov4-obj.cfg***. El parámetro **-map** permitirá generar una gráfica con los valores **mAP** generados cada cierta iteración, con el fin de poder monitorear el estado de aprendizaje de la red.

Al finalizar las iteraciones, se pueden apreciar que en la carpeta de **backup**, cuya ruta se encuentra especificada en **obj.data**, se habrán generado varios archivos **.weight**. Esto permite elegir el archivo del modelo adecuado, según la precisión que se haya generado cada cierta iteración en algún punto del entrenamiento. Por lo general, se selecciona el archivo generado **yolov4-obj_best.weigths**, dado que registra el valor pico de **mAP** registrado.

Para más información, se puede visitar el enlace al repositorio **Darknet** y acceder al **paper** para una lectura y aprendizaje adicional acerca del modelo de detección YOLOv4.

Anexo 3: Listado de precios CCTV

JM SYSTEMS COM SL

Parque Tecnológico

Ronda Isaac Peral, 9

46980 Paterna - Valencia - Spain

Tel. 96 136 60 82 - E-mail: pedidos@jmsystems.es

Listado de precios 01/08/2021

* Precios válidos salvo error u omisión
* Estos precios pueden variar sin previo aviso
* Confirmar/Validar precios en la web
* Es necesario registrarse en la web para visualizar los PVP
* Esta lista de precios anula a todas las anteriores

ANALÓGICO HD 4N1 - CÁMARAS BULLET			
Ref.		**Descripción**	**P.V.P.**
CV581KW-F4N1		Cámara Bullet HDTVI, HDCVI, AHD y Analógica - Gama PRO - 1/3" Panasonic© 2.1 Megapixel - MN34227PL+EN771E - HD 1080P (1920x1080) - Salida 4 en 1 - Soporta lentes manuales y DC (no incluidas) - Color 0.1 Lux / B/W 0.01 Lux - WDR (12FPS) - SenseUp - 3DNR - ATR - IR CUT	70,00
B581SW-5U4N1		Camara BOX HDCVI HDTVI AHD y CVBS - 1/2.8" Sony Progressive CMOS - 5 Megapixel (2592x1944) - 0 Lux IR - Soporta Starlight - Sin lente - Salida 4N1 - WDR - IR CUT - 3D DNR	90,00
B032-2E4N1-WIDE		Cámara Bullet HDTVI, HDCVI, AHD y Analógica - Gama ECO - 1/3" SOI 2.0Mpx - F23+8536H - 1080P (1920x1080) - Salida 4 en 1 - Lente 1.8 mm - 0 lux - IR Alcance 30 m - 2DNR - Uso exterior IP66 - Dewarping: Gran angular sin deformación esférica	42,00
SF-B023-WIDE-5P4N1		Safire - Cámara Bullet HDTVI, HDCVI, AHD y Analógica - Gama PRO - High Performance CMOS - 5 Megapixel (2560x1944) - Salida 4 en 1 - Lente 2.4 mm - Gran Angular 110º - 0 Lux - IR Matrix Alcance 20 m - IR CUT - Impermeable IP67	50,00
CV020-F4N1		Cámara Bullet HDTVI, HDCVI, AHD y Analógica - Gama ECO - Sensor 1/2.7" 2.0Mpx CMOS - SC2235+FM8536E - HD 1080P (1920x1080) - Salida 4 en 1 - Lente 2.8mm - 0 Lux - IR SMD Alcance 30 m - IR CUT - Impermeable IP66	30,00
SF-CV022IB-F4N1		Safire - Cámara Bullet HDTVI, HDCVI, AHD y Analógica - Gama ECO - High Performance CMOS - HD 1080P (1920x1080) - Salida 4 en 1 - Lente 2.8 mm - 0 Lux - IR Array Alcance 20 m - 2DNR - IR CUT - Impermeable IP66	38,00

SF-B022A-2E4N1		Safire - Cámara Bullet HDTVI, HDCVI, AHD y Analógica - Gama ECO - High Performance CMOS - HD 1080p (1920x1080) - Salida 4 en 1 - Lente 2.8 mm - 0 Lux - IR Matrix Alcance 30 m - Audio sobre Coaxial - 3D DNR - IR CUT - Impermeable IP67	38,00
SF-B024-2E4N1		Safire - Cámara Bullet HDTVI, HDCVI, AHD y Analógica - Gama ECO - 1/3" SONY 2.0 Mpx - HD 1080p (1920x1080) - Salida 4 en 1 - Lente 2.8 mm - 0 Lux - SMD LED Alcance 20 m - 3D DNR - IR CUT - Impermeable IP66	30,00
XS-B201A-2E4N1		X-Security - Cámara Bullet HDTVI, HDCVI, AHD y Analógica - Gama ECO - 1/2.7 CMOS - HD 1080p (1920x1080) - Salida 4 en 1 - Lente 2.8 mm - 0 Lux - IR Matrix Alcance 80 m - Audio sobre Coaxial en HDCVI - Micrófono integrado - IR CUT - Impermeable IP67	44,00
SF-B035-PIRS-2E4N1		Safire - Cámara Bullet HDTVI, HDCVI, AHD y Analógica - Gama ECO - High Performance CMOS - 2 Mpx (1920x1080) - Salida 4 en 1 - Lente 2.8 mm - Color 0.01 Lux - Alcance IR 40 m - Luz disuasoria LED blanco - Sonido de alarma y sirena integrada - Luz flash roja/azul - PIR Real - DWDR - 2DNR - Impermeable IP67	80,00
HWT-B120-M		Hikvision HiWatch Series - Cámara Bullet HDTVI, HDCVI, AHD y Analógica - Gama ECO - High Performance CMOS - HD 1080P (1920x1080) - Salida 4 en 1 - Lente Fija 2.8 mm - 0 Lux - EXIR 2.0 IR Alcance 20 m - 2DNR - IR CUT - IP66	36,00
HWT-B120-MS		Hikvision HiWatch Series - Cámara Bullet HDTVI, HDCVI, AHD y Analógica - High Performance CMOS - HD 1080p (1920x1080) - Salida 4 en 1 - Lente Fija 2.8 mm - 3 Axis - 0 Lux - EXIR 2.0 IR Alcance 30 m - DNR - IR CUT - Micrófono integrado - Apto para exterior IP66	56,00
HWT-B123-M		Hikvision HiWatch Series - Cámara Bullet HDTVI, HDCVI, AHD y Analógica - Gama PRO - 2 Megapixel High Performance CMOS - Ultra Low Light - HD 1080P (1920x1080) - Salida 4 en 1 - Lente 2.8mm - 0 Lux - IR EXIR Alcance 30 m - IR CUT - WDR - Impermeable IP66	52,00
HWT-B220-M		Hikvision HiWatch Series - Cámara Bullet HDTVI, HDCVI, AHD y Analógica - Gama ECO - High Performance CMOS - HD 1080P (1920x1080) - Salida 4 en 1 - Lente Fija 2.8 mm - 0 Lux - EXIR 2.0 IR Alcance 40 m - 2DNR - IR CUT - IP66	45,00
HWT-B223-M		Hikvision HiWatch Series - Cámara Bullet HDTVI, HDCVI, AHD y Analógica - Gama PRO - 2 Megapixel High Performance CMOS - Ultra Low Light - HD 1080P (1920x1080) - Salida 4 en 1 - Lente 2.8mm - 0 Lux - IR EXIR Alcance 50 m - IR CUT - WDR - Impermeable IP66	58,00

SF-HTVR8216AP-HEVC		Safire - DVR 5n1 - 16 CH vídeo HDTVI/HDCVI/AHD/CVBS / 8 IP (extra) / 4 CH audio - H.265+/H.265/H.264+/H.264 - Resolución por canal: 8Mpx (8FPS) o 5Mpx (10FPS) o 4Mpx (15FPS) o 3Mpx (18FPS) o 1080p (25FPS) - Control PTZ (RS485/Coaxial) - Alarmas - Salida HDMI 4K, VGA y CVBS - Power Over Coaxial (PoC) - Mando a distancia - Ratón - Espacio para 2 HDD	660,00
SF-XVR8116AS-4KL		Safire - DVR 5n1 - 16 CH vídeo HDTVI/HDCVI/AHD/CVBS / 16+16 IP / 1 CH audio - H.265Pro+/H.265Pro/H.264+/H.264 - Resolución 8Mpx (8FPS) o 1080P/720P (25FPS) - Audio sobre cable coaxial HDTVI - Salida HDMI 4K y VGA - Ratón - Alarmas - Espacio para 1 HDD - Almacenamiento en la nube para substream (OneDrive, Dropbox...)	490,00
SF-XVR8216AS-4KL		Safire - DVR 5n1 - 16 CH vídeo HDTVI/HDCVI/AHD/CVBS / 16+16 IP / 1 CH audio - H.265Pro+/H.265Pro/H.264+/H.264 - Resolución 8Mpx (8FPS) o 1080P/720P (25FPS) - Audio sobre cable coaxial HDTVI - Salida HDMI 4K y VGA - Ratón - Alarmas - Espacio para 2 HDD - Almacenamiento en la nube para substream (OneDrive, Dropbox...)	470,00
SF-XVR8416A-4K		Safire - DVR 5n1 - 16 CH vídeo HDTVI/HDCVI/AHD/CVBS / 16+16 IP / 4 CH audio - H.265+/H.265/H.264+/H.264 - Resolución por canal: 8Mpx, 5Mpx, 4Mpx, 3Mpx o 1080p (25FPS) - Control PTZ (RS485/Coaxial) - Alarmas - Salida HDMI 4K, HDMI Full HD y VGA - Mando a distancia - Ratón - Espacio para 4 HDD	1.350,00
SF-HTVR6232-HEVC		Safire - DVR 5n1 - 32 CH vídeo HDTVI/HDCVI/AHD/CVBS / 2 IP (extra) / 1 CH audio - H.265+/H.265/H.264+/H.264 - Resolución 4MpxLite/1080p (15fps) o 1080p Lite/720P (25fps) - Control PTZ (RS485/Coaxial) - Salida HDMI 4K, VGA y CVBS - Mando a distancia - Ratón - Espacio para 2 HDD	450,00
SF-HTVR8432A-HEVC		Safire - DVR 5n1 - 32 CH vídeo HDTVI/HDCVI/AHD/CVBS / 8 IP (extra) / 4 CH audio - H.265 Pro+/H.265 Pro/H.265/H.264+/H.264 - Resolución por canal: 8Mpx (8fps), 5Mpx (12fps), 4Mpx (15fps), 3Mpx (18fps) ó 1080p (25fps) - Control PTZ (RS485/Coaxial) - Alarmas - Salida HDMI 4K, VGA y CVBS - Mando a distancia - Ratón - Espacio para 4 HDD	1.300,00

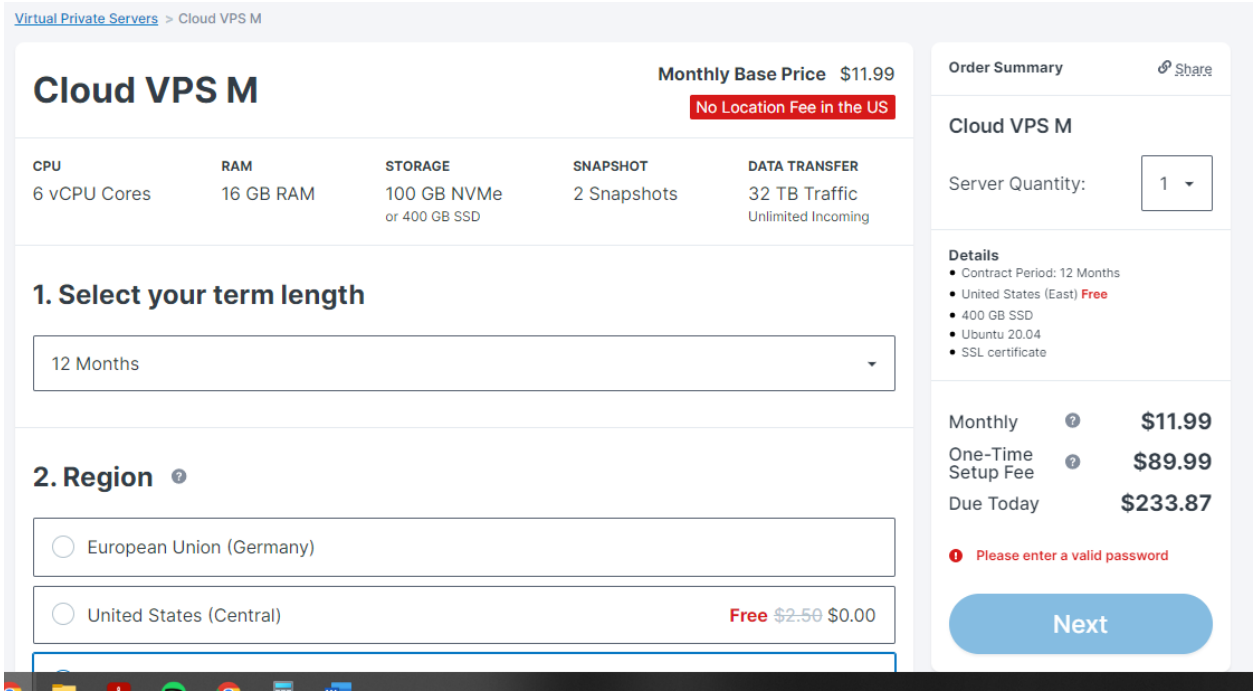
5n1 - DVR HIKVISION			
Ref.		**Descripción**	**P.V.P.**
HWD-5104S		Hikvision HiWatch Series - DVR 5n1 - 4 CH vídeo HDTVI/HDCVI/AHD/CVBS / 1 IP / 4 CH audio - H.264+/H.264 - Resolución por canal: 1080PLite/720P (25FPS) - Audio sobre cable coaxial - Control PTZ (Coaxial) - Salida HDMI Full HD y VGA - Ratón - Espacio para 1 HDD	74,00

HWD-5104MS		Hikvision HiWatch Series - DVR 5n1 - 4 CH vídeo HDTVI/HDCVI/AHD/CVBS / 1 IP / 4 CH audio - H.264+/H.264 - Resolución por canal: 1080PLite/720P (25FPS) - Audio sobre cable coaxial - Control PTZ (Coaxial) - Salida HDMI Full HD y VGA - Ratón - Espacio para 1 HDD	74,00
HWD-6104MH-G2AS		Hikvision HiWatch Series - DVR 5n1 - 4 CH vídeo HDTVI/HDCVI/AHD/CVBS / 2 IP / C26:C3365Pro+/H.265 - Resolución por canal: 4 Mpx (8FPS) o 4Mpx Lite (15FPS) - Audio sobre cable coaxial - Alarmas - Control PTZ (Coaxial) - Salida HDMI Full HD y VGA - Ratón - Espacio para 1 HDD	120,00
HWD-7104MH-G2S		Hikvision HiWatch Series - DVR 5n1 - 4 CH vídeo HDTVI/HDCVI/AHD/CVBS / 4 IP / 4 CH audio - H.265 Pro+/H.265 Pro - Resolución por canal: 8 Mpx (8FPS) o 8Mpx Lite (15FPS) - Audio sobre cable coaxial - Control PTZ (Coaxial) - Salida HDMI Full HD y VGA - Ratón - Espacio para 1 HDD	158,00
HWD-5108MS		Hikvision - DVR 5n1 - 8 CH vídeo HDTVI/HDCVI/AHD/CVBS / 2 IP / 4 CH audio - H.264+/H.264 - Resolución por canal: 1080PLite/720P (12FPS) - Audio sobre Cable Coaxial - Control PTZ (Coaxial) - Salida HDMI Full HD y VGA - Ratón - Espacio para 1 HDD	96,00
HWD-5108S		Hikvision - DVR 5n1 - 8 CH vídeo HDTVI/HDCVI/AHD/CVBS / 2 IP / 4 CH audio - H.264+/H.264 - Resolución por canal: 1080PLite/720P (12FPS) - Audio sobre Cable Coaxial - Control PTZ (Coaxial) - Salida HDMI Full HD y VGA - Ratón - Espacio para 1 HDD	96,00
iDS-7204HQHI-K1/2S		Hikvision - DVR 5n1 - 4 CH vídeo HDTVI/HDCVI/AHD/CVBS / 6 IP / 1 CH audio - H.265 Pro+/H.265+/H.265 - Resolución por canal: 4Mpx Lite / 3Mpx / 1080p (15FPS) - TrueSense, filtro de falsa alarma para vehículos y personas (2Ch) - Salida HDMI Full HD, VGA y CVBS - Control PTZ (RS485/Coaxial) - Mando a distancia - Ratón - Espacio para 1 HDD	172,00
HWD-6108MH-G2S		Hikvision HiWatch Series - DVR 5n1 - 8 CH vídeo HDTVI/HDCVI/AHD/CVBS / 4 IP / 8 CH audio - H.265 Pro+/H.265 Pro - Resolución por canal: 4 Mpx (8FPS) o 4Mpx Lite (15FPS) - Audio sobre cable coaxial - Control PTZ (Coaxial) - Salida HDMI Full HD y VGA - Ratón - Espacio para 1 HDD	178,00
HWD-6108MH-G2AS		Hikvision HiWatch Series - DVR 5n1 - 8 CH vídeo HDTVI/HDCVI/AHD/CVBS / 8 IP / 8 CH audio - H.265 Pro+/H.265 Pro - Resolución por canal: 4Mpx Lite (15FPS) o 4 Mpx (8FPS) - Audio sobre cable coaxial - Control PTZ (Coaxial) - Salida HDMI Full HD y VGA - Alarmas - Ratón - Espacio para 1 HDD	188,00

Anexo 4: Cotización servidor web

Dirección web: *https://contabo.com/en/*

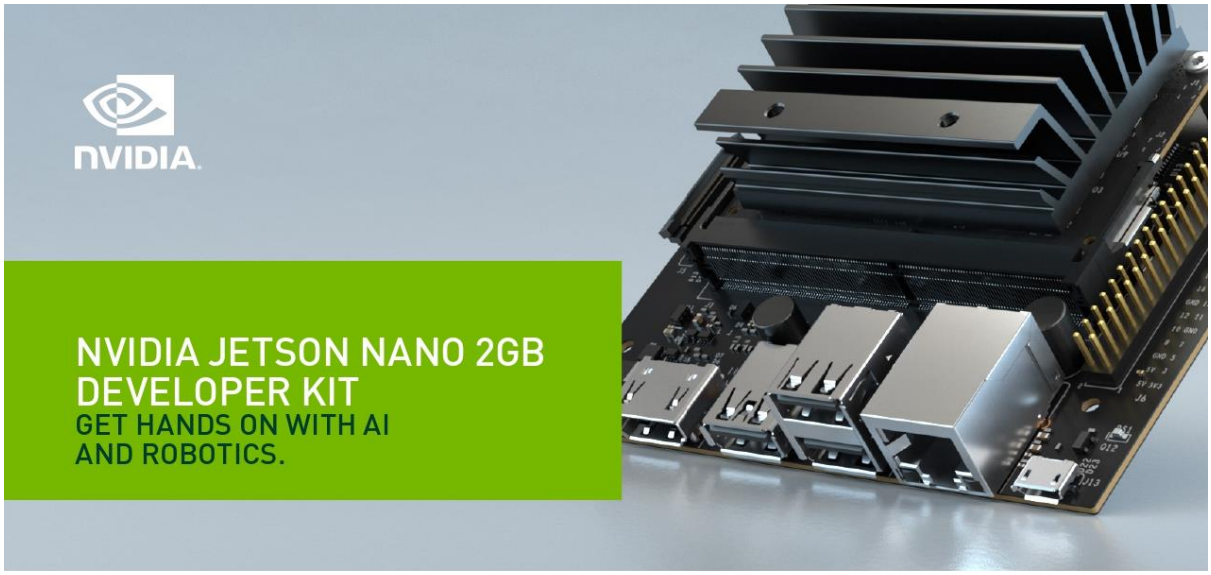
Característica	Descripción
Nombre	Cloud VPS M
CPU	6 Cores
RAM	16 GB
Almacenamiento	400 GB SSD
Transferencia de datos	32 TB de Tráfico
Precio Base Mensual	$11,99



Anexo 5: Sueldos y salarios mínimos – extracto TAXFINCORP CÍA. Ltda 2020

SALARIOS	Salario mensual	JORNAL/HR
Director de telecomunicaciones/Jefe de área	431,80	2,4534
Supervisor general de telecomunicaciones	431,20	2,4500
Supervisor de sistemas, desarrollo, tecnología y proyectos	431,20	2,4500
Arquitecto y usabilidad de software	431,20	2,4500
Supervisor de diseño de software	430,60	2,4466
Administrador de base de datos	430,60	2,4466
Ingeniero electrónico especialista en mantenimiento	430,60	2,4466
Analista de investigación y desarrollo de hardware y software	430,60	2,4466
Analista/Controller de calidad de software	430,60	2,4466
Especialista de telecomunicaciones	430,01	2,4432
Supervisor de plataformas/equipo de voz y datos	430,01	2,4432
Supervisor de planta externa/seguridad electrónica/cableado estructurado	430,01	2,4432
Programador en telecomunicaciones	429,41	2,4398
Analista de software	429,41	2,4398
Tester de software	429,41	2,4398
Programador y diseñador multimedia/web	429,41	2,4398
Técnico instalador de servicios agregados	429,41	2,4398
Técnico de fibra óptica/cobre/empalmador	429,41	2,4398
Analista de redes	429,41	2,4398
Analista de sistemas/telecomunicaciones	429,41	2,4398
Programador semi senior de software	429,41	2,4398
Técnico de redes de datos	427,81	2,4307
Técnico en cableado estructurado	427,81	2,4307
Técnico de planta externa/cablista/instalador	423,01	2,4035
Asistente/ayudante/instalador auxiliar de telecomunicaciones	423,01	2,4035
Técnico en ensamblaje y mantenimiento de equipos de computación y electrónicos	423,01	2,4035
Asistente/ayudante/auxiliar/instalador de sistemas	423,01	2,4035
Ingeniero Eléctrico	465,51	2,6449
Inspector de obra	465,51	2,6449
Maestro eléctrico / liniero / subestaciones	463,52	2,6336
Maestro de obra	439,95	2,4997
Ingeniero Electrónico	430,60	2,4466
Electricista	405,24	2,3025
Técnico liniero eléctrico	415,75	2,3622

Anexo 6: Especificaciones técnicas JETSON NANO



Small Size. Small Price. Big AI Discoveries.

Discover the power of AI and robotics with NVIDIA® Jetson Nano™ 2GB Developer Kit. It's small, powerful, and priced for everyone at $59*. This means educators, students, and other enthusiasts can now easily create projects with fast and efficient AI using the entire GPU-accelerated NVIDIA software stack.

Learning by doing is key for anyone new to AI and robotics, and the Jetson Nano 2GB Developer Kit is ideal for hands-on teaching and learning. Unlike online-only learning, you'll see your work on the developer kit perceive and interact with the world around you in real time.

Thousands of Jetson Nano developers actively contribute videos, how-tos, and open-source projects in addition to the free and comprehensive tutorials offered by NVIDIA. These start with an introductory "Hello AI World," continue to robotics projects such as the open-source NVIDIA JetBot AI robot platform, and lead to the next level of robotics development with NVIDIA Isaac™.

All these resources are enabled by NVIDIA JetPack™, which brings to each Jetson developer the same CUDA-X™ software and tools used by professionals around the world. JetPack includes a familiar Linux environment and simplifies the development process with support for cloud-native technologies such as containerization and orchestration.

The Jetson Nano 2GB Developer Kit delivers incredible AI performance at a low price. It makes the world of AI and robotics accessible to everyone with the exact same software and tools used to create breakthrough AI products across all industries. There's no better way to start.

KEY FEATURES

NVIDIA Jetson Nano 2GB Developer Kit
- 128-core NVIDIA Maxwell™ GPU
- Quad-core ARM® A57 CPU
- 2 GB 64-bit LPDDR4 | 25.6 GB/s

Power options
- USB-C 5V 3A

I/O
- USB 3.0 Type A
- USB 2.0 Type A
- USB 2.0 Micro-B
- HDMI
- Gigabit Ethernet
- GPIOs, I2C, I2S, SPI, UART
- MIPI camera connector
- UART header
- Fan connector**
- Button Header**

Kit Contents
- NVIDIA Jetson Nano 2GB Developer Kit
- Printed Quick Start Guide and Support Guide
- Wi-Fi module and cable***

* Local pricing will vary due to currency and taxes | ** not stuffed | *** for Wi-Fi bundle SKU only

NVIDIA JETSON NANO 2GB DEVELOPER KIT
TECHNICAL SPECIFICATIONS

GPU	**128-core Maxwell**
CPU	**Quad-core ARM A57 @ 1.43 GHz**
Memory	**2 GB 64-bit LPDDR4 25.6 GB/s**
Storage	**microSD (card not included)**
Video Encode	**4K @ 30 | 4x 1080p @ 30 | 9x 720p @ 30 (H.264/H.265)**
Video Decode	**4K @ 60 | 2x 4K @ 30 | 8x 1080p @ 30 | 18x 720p @ 30 (H.264/H.265)**
Camera	**1x MIPI CSI-2 D-PHY lanes**
Connectivity	**Gigabit Ethernet, Wi-Fi (SKU 0 only)**
Display	**HDMI**
USB	**1x USB 3.0 Type A, 2x USB 2.0 Type A, USB 2.0 Micro-B**
Others	**GPIO, I²C, I²S, SPI, UART**
Mechanical	**100 mm x 80 mm x 29 mm**

Learn more at **www.nvidia.com/JetsonNano2GB**

© 2020 NVIDIA Corporation. All rights reserved. NVIDIA, the NVIDIA logo, CUDA-X, Jetson, Jetson Nano, NVIDIA Isaac, NVIDIA JetPack, and NVIDIA Maxwell are trademarks and/or registered trademarks of NVIDIA Corporation in the U.S. and other countries. Other company and product names may be trademarks of the respective companies with which they are associated. ARM, AMBA and ARM Powered are registered trademarks of ARM Limited. Cortex, MPCore and Mali are trademarks of ARM Limited. All other brands or product names are the property of their respective holders. "ARM" is used to represent ARM Holdings plc; its operating company ARM Limited; and the regional subsidiaries ARM Inc.; ARM KK; ARM Korea Limited.; ARM Taiwan Limited; ARM France SAS; ARM Consulting (Shanghai) Co. Ltd.; ARM Germany GmbH; ARM Embedded Technologies Pvt. Ltd.; ARM Norway, AS and ARM Sweden AB. SEP20

Anexo 7: Certificado de Traducción

English Speak Up Center

Nosotros "English Speak Up Center"

CERTIFICAMOS que

La traducción del documento adjunto solicitado por el señor **Jorge Enrique Ortega Jaramillo** con cédula de ciudadanía número **1105335002** cuyo tema de investigación se titula: **"SISTEMA DE DETECCIÓN Y RECONOCIMIENTO DE INTRUSOS DE BAJO COSTO BASADO EN VISIÓN ARTIFICIAL, APLICADO A SISTEMAS DE VIDEOVIGILANCIA"**, ha sido realizada por el Centro Particular de Enseñanza de Idiomas "English Speak Up Center".

Esta es una traducción textual del documento adjunto, y el traductor es competente para realizar traducciones.

Loja, 12 de julio de 2022

Mg. Sc. Elizabeth Sánchez Burneo
DIRECTORA ACADÉMICA

DIRECCION: SUCRE 207-46 ENTRE AZUAY Y MIGUEL RIOFRIO | TELF: 099 5263 264