UNIVERSIDAD TECNOLÓGICA ISRAEL

ESCUELA DE POSGRADOS "ESPOG"

MAESTRÍA EN ELECTRÓNICA Y AUTOMATIZACIÓN

Resolución: RPC-SO-09-No.265-2021

PROYECTO DE TITULACIÓN EN OPCIÓN AL GRADO DE MAGÍSTER

Título del proyecto:
PROCESAMIENTO DIGITAL DE IMÁGENES MEDIANTE INTELIGENCIA ARTIFICIAL PARA LA DETECCIÓN DE ACCIDENTES DE TRÁNSITO EN QUITO
Línea de Investigación:
Sistemas de Información e informática
Campo amplio de conocimiento:
Modelos y aplicaciones para Sistemas Inteligentes
Autor/a:
Juan Francisco Changotasig Yánez
Tutor/a:
Ing. Wilmer Fabian Albarracín Guarochico MBA

Quito – Ecuador

2023

APROBACIÓN DEL TUTOR

Universidad Israel

Yo, Wilmer Fabian Albarracín Guarochico con C.I: 1713341152 en mi calidad de Tutor del proyecto de investigación titulado *"PROCESAMIENTO DIGITAL DE IMÁGENES MEDIANTE INTELIGENCIA ARTIFICIAL PARA LA DETECCIÓN DE ACCIDENTES DE TRÁNSITO EN QUITO"*.

Elaborado por: Juan Francisco Changotasig Yanez, de C.I: 1723465041, estudiante de la Maestría: Electrónica y Automatización, de la **UNIVERSIDAD TECNOLÓGICA ISRAEL (UISRAEL)**, como parte de los requisitos sustanciales con fines de obtener el Título de Magister, me permito declarar que luego de haber orientado, analizado y revisado el trabajo de titulación, lo apruebo en todas sus partes.

Quito D.M., 15 de marzo de 2023

Firma

ESPOG | Escuela de Posgrados

Carta de declaración de autorización

Universidad Israel

DECLARACIÓN DE AUTORIZACIÓN

Yo, Juan Francisco Changotasig Yanez, portador/a de C.C. 1723465041, **autor**/a del trabajo de titulación:

Tema: PROCESAMIENTO DIGITAL DE IMÁGENES MEDIANTE INTELIGENCIA ARTIFICIAL PARA LA DETECCIÓN DE ACCIDENTES DE TRÁNSITO EN QUITO, previo a la obtención del título de **Magister en:** Electrónica y Automatización.

1. Declaro tener pleno conocimiento de la obligación que tienen las instituciones de educación superior, de conformidad con el Artículo 144 de la Ley Orgánica de Educación Superior, de difundir el respectivo trabajo de graduación para que sea integrado al Sistema Nacional de Información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor.
2. Manifiesto mi voluntad de ceder a la Universidad Tecnológica Israel los derechos patrimoniales consagrados en la Ley de Propiedad Intelectual del Ecuador, artículo 4,5 y 6, en calidad de autor/a del trabajo de titulación, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente. En concordancia suscribo este documento en el momento que hago entrega del trabajo final en el formato impreso y digital como parte del acervo bibliográfico de la Universidad Tecnológica Israel.
3. Autorizo a la SENESCYT a tener una copia del referido trabajo de graduación, con el propósito de generar un repositorio que democratice la información, respetando las políticas de prosperidad intelectual vigentes.

Quito, 15 de marzo de 2023

Juan Francisco Changotasig Yanez

Firma: _____

C.C. 1723465041

Tabla de contenidos

Índice de Figuras

INFORMACIÓN GENERAL

Contextualización del tema

Las aplicaciones de visión artificial mediante el procesamiento de imágenes permiten agilizar, mejorar, optimizar y reducir costos en los ámbitos industriales y de seguridad. El análisis de imágenes permite la automatización de procesos en una empresa y la identificación y clasificación de eventos dentro de la sociedad, parametrizando comportamientos ciudadanos. Mediante la clasificación de imágenes con la extracción de parámetros que identifican dentro de las imágenes, formas, colores, texturas, etc. Implementando algoritmos en la tarea de clasificación de imágenes basadas en formas y regiones de las imágenes de video, utiliza la técnica de extracción de histogramas en varias dimensiones, utilizando algoritmos de inteligencia artificial se efectúa la clasificación automática de imágenes (Adler, 2021).

La utilización de personas para el monitoreo de eventos relacionados con accidentes dentro de las vías del perímetro urbano de la ciudad de Quito incurre en tiempos de respuesta amplios y retardan el accionar del personal operativo para la asistencia e intervención de los organismos articulados de respuesta. (Departamento de Planificacion CACMQ, 2019).

Los Gobiernos autónomos descentralizados, apoyados en los organismos que coadyuvan en el mantenimiento de la seguridad y convivencia ciudadana, dentro de sus competencias, tienen el objetivo de mejorar, apoyar, asegurar y mantener la seguridad y convivencia ciudadana (COESCOP, 2020).

Los centros de monitoreo y control del espacio público de los Gobiernos Autónomos Descentralizados, interconectados con la red de emergencia ECU911, permiten monitorear las acciones, el comportamiento y la movilización, en tiempo real, de la sociedad que realiza sus actividades cotidianas dentro del perímetro urbano.

Existen cámaras de vigilancia ubicadas a lo largo del perímetro urbano de la ciudad de Quito. Parques, plazas, avenidas y demás puntos de concentración de personas, cuentan con cámaras de videovigilancia de alta resolución.

El centro de monitoreo de eventos del Cuerpo de Agentes de Control, ejecutados por el Centro de Mando y Comunicaciones del Cuerpo de Agentes de Control Quito, realiza actividades de monitoreo de plazas, avenidas, parques, instituciones educativas en coordinación con el ECU 911 y la secretaría de Seguridad y Gobernabilidad, para el manejo de eventos emergentes que implican el uso del espacio público, articulándose con las diferentes entidades municipales y nacionales para la atención integral de emergencias y eventos desarrollados en el espacio público donde se cuenta con equipos de videovigilancia.

Problema de investigación

El seguimiento eficaz de los eventos emergentes dentro del espacio público ha sido durante mucho tiempo uno de los objetivos principales de las entidades de control que coadyuvan con la seguridad y convivencia ciudadana. Una gran parte de los centros de monitoreo dependen de operadores humanos para el monitoreo de eventos en el espacio público mediante la notificación de incidentes que se desarrollan en la ciudad de Quito dentro de su perímetro urbano. Las actividades que implican la supervisión manual de cámaras de videovigilancia conllevan una gran cantidad de horas frente a monitores y la asignación de varias cámaras de videovigilancia al mismo tiempo para cada operador. Los operadores de cámaras de videovigilancia son propensos a falta de concentración y atención, además de estar sujetos a fatiga física y mental, lo que limita y vuelve ineficiente el monitoreo de eventos emergentes y por ende a demoras en los tiempos de notificación y respuesta de eventos que se desarrollan dentro del espacio público (Pacanchique, 2020).

Por lo tanto, es necesario el desarrollar herramientas automatizadas para el monitoreo de eventos y así disminuir la carga de trabajo de los operadores, aumentando la eficiencia y eficacia de los resultados.

Tomando en cuenta que el segundo motivo de causa de muerte en el ecuador es causado por accidentes de transporte terrestre, según El Instituto Ecuatoriano de Estadística y Censos INEC, los eventos emergentes que se desarrollan dentro del espacio público son principalmente accidentes de transporte terrestre que registraron un total de 1431 muertes para el 2019 (INEC, 2023).

Por otro lado, el aumento constante del parque automotor dentro de la ciudad de Quito, según la Secretaría de Movilidad de la ciudad de Quito, aumentó en 4.9% anualmente en promedio, introduciendo alrededor de 17.539 vehículos solo en la ciudad de Quito (COMERCIO, 2022).

Existen sistemas automáticos de seguimiento del tráfico mediante sistemas de cámaras basados en el monitoreo de tráfico. Sin embargo, estos sistemas de monitoreo son supervisados por humanos, lo que dificulta el seguimiento de la congestión y la detección de vehículos estacionados, el flujo de vehículos y el recuento de estos. La introducción de automatización en la gestión del tráfico mediante sistemas automatizados de vigilancia de vehículos que utilizan la Inteligencia Artificial permite gestionar el tráfico para prevenir y notificar situaciones emergentes en accidentes de tráfico.

Un sistema automatizado con Inteligencia artificial puede identificar vehículos, realizar seguimientos de su patrón de movimiento e identificar comportamientos de conducción peligrosos y erráticos en carretera. Los sistemas de videovigilancia basada en Inteligencia artificial permiten la detección de vehículos detenidos que impiden el flujo, causando congestión y dificultando la movilidad de vehículos (Benitez, 2014).

La demora en los tiempos de respuesta y la falta de atención a los eventos que se pasan por alto por parte de los operarios de sistema de monitoreo y videovigilancia, ocasionan una tardía coordinación con las instituciones encargadas del apoyo logístico y apoyo emergente frente a eventos que necesitan respuesta inmediata para precautelar la integridad y la vida de los ciudadanos que habitan dentro del Distrito Metropolitano de Quito.

De esta manera, es evidente la necesidad de detectar eventos emergentes mediante la utilización de algoritmos de inteligencia artificial que permita el monitoreo y detección de accidentes de forma autónoma, permitiendo notificar sobre las novedades al personal de monitoreo para una respuesta breve y oportuna.

Objetivo general

Desarrollar el algoritmo de procesamiento digital de imágenes mediante inteligencia artificial

Objetivos específicos

- Establecer técnicas de análisis de imágenes utilizando redes neuronales convolucionales.
- Aplicar técnicas de aprendizaje para la detección autónoma de vehículos mediante el entrenamiento de una red neuronal convolucional.
- Desarrollar código basado en inteligencia artificial que permita la identificación de vehículos.
- Implementar un algoritmo con datos de monitoreo de cámaras de videovigilancia del CACMQ para la detección de accidentes de tránsito.

Vinculación con la sociedad y beneficiarios directos:

La utilización de personas para el monitoreo de eventos relacionados con accidentes dentro de las vías del perímetro urbano de la ciudad de Quito incurre en tiempos de respuesta largos, lo que ocasiona la falta de coordinación oportuna y tiempos de coordinación y articulación extensos para la asistencia e intervención de los organismos de respuesta.

El seguimiento y notificación de eventos emergentes dentro del espacio público permiten la atención oportuna y la coordinación con instituciones especializadas que coadyuvan con la seguridad y convivencia ciudadana. Los centros de monitoreo dependen de operadores humanos para el monitoreo de eventos en el espacio público mediante la notificación de incidentes que se desarrollan en la ciudad de Quito dentro de su perímetro urbano. Las actividades que implican la supervisión manual de cámaras de videovigilancia conllevan una gran cantidad de horas frente a monitores y la asignación de varias cámaras de videovigilancia al mismo tiempo para cada operador.

El personal asignado para el monitoreo de cámaras de videovigilancia es propenso a falta de concentración y atención, además de estar sujetos a fatiga física y mental, lo que limita y vuelve ineficiente el monitoreo de eventos emergentes y por ende genera demoras en los tiempos de notificación y respuesta de eventos que se desarrollan dentro del espacio público (Pacanchique, 2020).

Es necesario el desarrollo de herramientas automatizadas para el monitoreo de eventos y así disminuir la carga de trabajo de los operadores, aumentando la eficiencia y eficacia de los resultados.

Los accidentes de transporte terrestre son el segundo motivo de causa de muerte de personas en el ecuador, según el Instituto Ecuatoriano de Estadística y Censos INEC en el 2019. Los eventos emergentes que se desarrollan dentro del espacio público son principalmente accidentes de transporte terrestre que registraron un total de 1431 y se incrementan cerca de 17.539 vehículos solo en la ciudad de Quito anualmente (INEC, 2023).

Existen sistemas automáticos de seguimiento del tráfico mediante sistemas de cámaras basados en el monitoreo de vehículos. Sin embargo, estos sistemas de monitoreo son supervisados por humanos, lo que dificulta el seguimiento de la congestión y la detección de vehículos estacionados, el flujo de vehículos y el recuento de estos. La introducción de automatización en la gestión del tráfico mediante sistemas automatizados de vigilancia de vehículos que utilizan la Inteligencia Artificial permite gestionar el tráfico para prevenir y notificar situaciones emergentes en accidentes de tráfico.

Un sistema automatizado con Inteligencia artificial puede identificar vehículos, realizar seguimientos de su patrón de movimiento e identificar comportamientos de conducción peligrosos y erráticos en carretera. Los sistemas de videovigilancia basada en Inteligencia artificial permiten la detección de vehículos detenidos que impiden el flujo, causando congestión y dificultando la movilidad de vehículos (Benitez, 2014).

CAPÍTULO I: DESCRIPCIÓN DEL PROYECTO

1.1. Contextualización general del estado del arte

Para dar solución a la problemática de aumento y optimización de los tiempos de respuesta a eventos relacionados con accidentes suscitados dentro del espacio público y específicamente en accidentes de tránsito, es necesaria la implementación de algoritmos que permitan el reconocimiento autónomo de eventos mediante el análisis de imágenes, tomando en cuenta aspectos como: la placa, el color y forma. (Rioja, 2022).

El software de programación MATLAB permite la implementación de algoritmos para el análisis de información y la generación de datos estadísticos que permitan comparar imágenes obtenidas de vídeos de monitoreo y compararlas con bases de datos de registros visuales existentes. Para lograrlo es necesario aplicar técnicas estadísticas que permitan discriminar la información.

Mediante la herramienta Image Processing Toolbox de Matlab se puede realizar el manejo de varios algoritmos que permiten procesar y analizar imágenes. Mediante la segmentación de imágenes, el mejoramiento, reducción y transformaciones de figuras geométricas insertas en las imágenes se puede lograr automatizar el procesamiento de imágenes.

Image Processing Toolbox permite la comparación del registro de imágenes mediante el procesamiento de información por lotes y datos relativamente grandes de forma efectiva. Permiten analizar imágenes con volúmenes 3D y vídeos configurando el contraste para obtener histogramas y manipular sitios de interés dentro de las imágenes procesadas (Matworks, 2022).

La integración de imágenes para la generación de visas compuestas permite mejorar el procesamiento y reducir el ruido, con esto se puede disponer de información de las imágenes para ser extraídas. Mediante la utilización de algoritmos de registro basados en características y algoritmos de registro basados en intensidad, se puede integrar imágenes de diferentes cámaras.

Los resultados arrojados por Juan Bravo P. (Pérez-Villar, 2015), en la investigación realizada mediante la utilización de algoritmos basados en Expectation-Maximization para la "*DETECCIÓN DE VEHÍCULOS MEDIANTE ANÁLISIS DE IMÁGENES*" permiten robustecer los sistemas de identificación vehicular en las zonas lejanas de imágenes donde la información pierde fidelidad.

El trabajo realizado por Luis Gil R. en la Universidad Autónoma de Madrid, muestran que los sistemas de clasificación de imágenes basados en estadísticas bayesianas estimados en función del movimiento permiten integrar sistemas IPM para obtener sistemas más robustos de identificación de vehículos y cambios de carril (Ramos, 2016).

Según los estudios realizados en la Universidad de La Rioja" Aplicación *de Teorías de Color e Imágenes Digitales*", realizadas mediante el análisis de algoritmos que utilizan sistemas de ecuaciones no lineales como el método Newton-Ramphson, se obtuvieron resultados alentadores en el desarrollo de algoritmos

1.2. Proceso investigativo metodológico

La investigación necesaria para la elaboración y el alcance del presente trabajo de titulación se obtendrá mediante la información obtenida de trabajos previos de análisis de imágenes, procesamiento digital de imágenes, utilización de APPS de software de procesamiento de video e imágenes, videos recolectados de la cámara de monitoreo recogidas de los videos del centro de monitoreo y demás trabajos que permitan recopilar procesos de análisis de imágenes vehiculares.

Para determinar el funcionamiento de las herramientas proporcionadas por MATLAB con respecto del procesamiento de imágenes que permitan la creación de algoritmos previamente entrenados para la identificación accidentes de tránsito y la notificación autónoma de incidentes en el Distrito Metropolitano de Quito, se realizará mediante consulta de fuentes propias de las páginas oficiales del software y trabajos de investigaciones afines, utilizados con el fin de generar respuestas rápidas de incidentes en comparación a los tiempos generados por la operación humana.

La información almacenada por las cámaras de videovigilancia dentro del Cuerpo de Agentes de Control permitirá el análisis de imágenes basado en los eventos de congestión, accidentalidad y flujo vehicular, registradas en los diferentes días de la semana. Esta información será procesada y analizada con el fin de utilizar los algoritmos de las herramientas de procesamiento de imágenes.

CAPÍTULO II: PROPUESTA

2.1 Fundamentos teóricos aplicados

Los fundamentos teóricos aplicados en el presente trabajo, necesarios para el desarrollo de este, se detallan a continuación:

Aprendizaje Autónomo

El aprendizaje autónomo, en lo referente al desarrollo y aplicación a la visión por computador como parte principal de la inteligencia artificial, permite el aprendizaje de equipos con capacidad de procesamiento, como a las computadoras, aprender de una forma independiente según la experiencia que van adquiriendo según los resultados arrojados por los algoritmos. Los algoritmos creados para el aprendizaje autónomo permiten obtener datos finales sobre varios escenarios posibles. El resultado es almacenado y el algoritmo conoce los resultados posibles de los datos analizados permitiendo el aprendizaje autónomo (Matworks, 2022). Mientras más datos se pueda procesar sobre ese caso particular, mayor será la probabilidad de acierto del algoritmo.

Visión por Computador

La visión por computador es la adquisición de información mediante la detección, seguimiento y procesamiento de imágenes obtenidas del mundo real con dispositivos de video para su análisis mediante software, con el propósito de transformar la información en datos que permitan ser almacenados y manipulados para diferentes fines (Adler, 2021). La visión por computador es utilizada como una herramienta que permite simular la capacidad visual de los seres vivos mediante la cual se percibe el entorno con la captura de imágenes y videos (Uribe, 2022).

Videovigilancia

La videovigilancia permite la captura de información mediante el registro de audio y video de dispositivos que utilizan lentes ópticos, ubicadas en edificaciones o estructuras con cierta altura para la captura de imágenes. Dependiendo de la capacidad de registro del medio óptico en la cámara y de su capacidad de procesamiento, las imágenes registradas permiten procesar imágenes a grandes distancias, además la estructura de comunicación permite la utilización de varios tipos de protocolos de comunicación para la comunicación y análisis de las imágenes y audios registrados. Permiten además el control y monitoreo de grandes entornos de manera continua, permitiendo aumentar la seguridad y protección de bienes en general (Ouahhadi, 2020).

Inteligencia Artificial

La inteligencia artificial consiste en emular las características humanas de inteligencia y razonamiento, basados en modelos matemáticos de análisis de datos y modelos computacionales (Vicente, 2022). Además, permite simular los sentidos de los seres vivos para captar su entorno, como la visión, el razonamiento, la experiencia, el aprendizaje, el movimiento, etc. Las aplicaciones desarrolladas en la actualidad sobre inteligencia artificial abarcan desde la asistencia para la conducción autónoma de vehículos hasta la detección de anomalías patógenas para diagnósticos médicos (Aguilera, 2021).

Etiquetado de Imágenes.

El etiquetado de imágenes permite identificar partes específicas dentro de las imágenes que se requieren ser analizadas y procesadas por un Deep Learning con el propósito de obtener un resultado particular sobre los datos de entrada de señales, regiones o puntos específicos de las imágenes.

Técnicas de Aprendizaje Autónomo

Existen varias técnicas que permiten el aprendizaje autónomo mediante el desarrollo de software, mismas que utilizan modelos matemáticos para el análisis de datos y la predicción de resultados.

Aprendizaje supervisado.

El Aprendizaje Supervisado es la utilización de una cantidad de datos previamente etiquetados dentro de un algoritmo donde ya se conocen los parámetros o atributos de los datos que permiten realizar el aprendizaje autónomo comparando datos nuevos. Dentro de los algoritmos utilizados para el aprendizaje supervisado se encuentran los algoritmos de regresión y algoritmos de clasificación. Se puede determinar un algoritmo como algoritmo de regresión cuando el resultado esperado de la predicción es un dato numérico (resultado numérico o estadístico) y se puede determinar un algoritmo como algoritmo de clasificación cuando el resultado esperado de la predicción es un dato categórico (número limitado de grupos o categorías) (Caparrini, 2022).

Aprendizaje no supervisado.

El Aprendizaje No Supervisado se refiere a la utilización de datos sin ningún tipo de identificación o etiqueta que permita conocer sus parámetros o atributos dentro del algoritmo. Los algoritmos son entrenados con datos sin etiquetar donde no se conoce un valor objetivo y requiere encontrar un grupo de datos similares mediante la agrupación para la identificación de datos analizados. Existen dos tipos principales de clustering o agrupamiento de datos en algoritmos dentro del aprendizaje no supervisado. El método jerárquico organiza jerárquicamente los datos de un grupo de datos formando niveles de

agrupación y el método particional que crea conjuntos de datos que no forman ningún tipo de nivel o agrupación (k-means) (Caparrini, 2022).

CNN (Redes Neuronales Convolucionales).

Las redes neuronales convolucionales o también conocidas como ConvNet son una estructura de red que permite el aprendizaje profundo mediante el análisis y procesamiento de datos sin necesidad de intermediarios u operarios. El análisis de datos esté en tipo de arquitectura de red, analiza patrones mediante la clasificación de imágenes y audio que permite la identificación de rostros, objetos y animales. El resultado del análisis siempre dependerá del tipo de dato analizado por la red. (Matworks, 2022).

Las redes neuronales convolucionales se definen como red neuronal por su estructura de capas, mediante la descomposición de las imágenes analizadas para transformarlas y darles diferentes resoluciones con el fin de filtrar sus características más simples, con lo que se obtiene como el brillo, borde y demás características particulares de cada imagen.

Deep Learning.

En la actualidad el Deep Learnig es utilizado para varios tipos de aplicaciones como el procesamiento de señales, la visión artificial, las comunicaciones inalámbricas, aplicaciones de rastreo y radares y demás tecnologías afines. El Deep Learning en una clase de Machine Learning que permite el entrenamiento de computadoras para realizar tareas autónomas, configurando parámetros básicos dentro de la computadora sobre los datos que son analizados para que el computador aprenda por su propia cuenta, permitiéndole reconocer patrones con la ayuda de muchas capas de procesamiento (SAS, 2022).

Transferencia de Aprendizaje CNN.

La transferencia de aprendizaje forma parte del Deep Learning y permite la utilización de modelos de algoritmos previamente entrenados para realizar tareas específicas. Además, la transferencia de aprendizaje de un modelo de red neuronal convolucional permite reducir el tiempo de entrenamiento de la red sin tener que realizar todo el proceso desde cero. En este tipo de transferencia de aprendizaje se toma en cuenta varios factores para su reutilización y funcionamiento óptimo en la obtención de resultados deseados, como el tamaño de los datos analizados, la precisión con la que realiza la predicción y la velocidad con la que realiza la misma (Matworks, 2022). La gran mayoría de arquitecturas basadas en CNN para la transferencia de aprendizaje tiene en común un número de capas superior a las 20 capas, realizando el reemplazo de capas, el reentrenamiento del modelo y evaluando los resultados de predicción se puede realizar una transferencia de aprendizaje óptima.

Python

Python es un lenguaje de programación de código abierto que permite escribir e integrar código de manera simple y rápida. Además, es un lenguaje interpretado de fácil aplicación multiparadigma orientada a objetos, programación interpretativa o programación funcional. Python permite el desarrollo de programas con una sintaxis fácil y simplificada. Permite escribir código con menos líneas en comparación a otros lenguajes y se puede escribir código desde cero. Además. Se puede utilizar a través de varios sistemas operativos. Este lenguaje de programación es utilizado en el desarrollo de redes neuronales convolucionales y Deep learning.

Figura 1

Logo lenguaje de programación Python.

Nota. Muestra el logo de Python, 2023, tomado de (https://www.python.org/static/img/python-logo@2x.png).

Anaconda

Anaconda es una de las distribuciones del lenguaje de programación Python que permite trabajar con aplicaciones de computación científica y aplicaciones de Learning Machine en el procesamiento de datos de gran volumen. Permite la simplificación y la gestión de paquetes en varias distribuciones y es compatible con muchos sistemas operativos. Permite trabajar de forma fácil en entornos sin la Necesidad de ejecutarlos dentro de un sistema operativo.

Figura 2

Logo distribución Python Anaconda

Nota. Muestra el logo de distribución Anaconda, tomado de (https://www.anaconda.com).

Jupyter Notebook

Es un interfaz abierto que permite manejar texto, audio video y escritura de código por el navegador utilizando múltiples lenguajes de programación. Esta interfaz (la interfaz web), permite una comunicación y un procesamiento a través de su kernel de cálculo permitiendo la utilización de varios núcleos de procesamiento como Julia, Rubí C++ Fortran, Java, Matlab entre otros, dentro de su interfaz de trabajo (Granado, 2023).

Figura 3

Logo interfaz web de programación Jupyter Notebook

Nota. Muestra el logo del gestor de lenguaje Python Junyper notebook, tomado de (https://jupyter.org/assets/logos/rectanglelogo-greytext-orangebody-greymoons.svg).

Mathlab

Mathlab es una plataforma que utiliza un lenguaje de programación basado en el cálculo y las matemáticas para el desarrollo de aplicaciones científicas de análisis y procesamiento de datos. Matlab es una herramienta que permite el desarrollo de aplicaciones de deep Learning y procesamiento de datos con visión artificial. Cuenta con varios aplicativos dentro de un mismo entorno de trabajo y es compatible con varios sistemas operativos. Realiza el análisis de datos, gráficas, desarrollo de algoritmos Apps y es

compatible e integrable a un sin número de lenguajes de programación como Python además maneja entornos de ejecución de scripts en la nube (Matworks, 2022).

Figura 4

Logo Mathlab

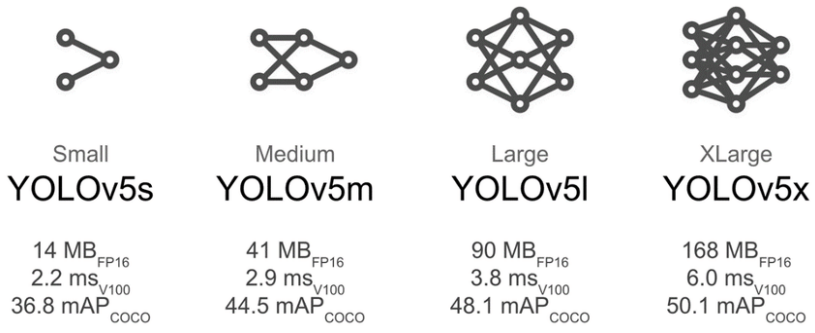
Nota. Muestra el logo de lenguaje de programacion Matlab, tomado de (https://1000marcas.net/wp-content/uploads/2020/02/logo-MATLAB.png).

Yolo

Yolo he es un algoritmo de procesamiento de imágenes basado en modelos de redes neuronales que permite la clasificación y procesamiento de imágenes de una manera rápida y eficiente. La velocidad de procesamiento y reconocimiento de imágenes mediante la predicción cuenta con varias versiones en función de la capacidad de procesamiento y la complejidad de su estructura y certeza en el resultado predicho (Chen, 2023). En la figura se muestran, de manera simplificada, las diferentes arquitecturas de Yolo.

Figura 5

Arquitectura de Yolov5 (Arquetipos).



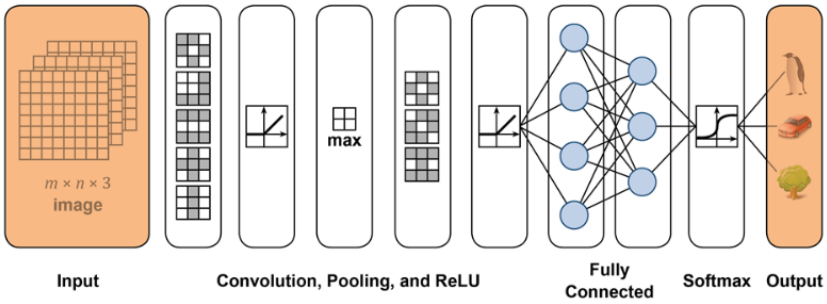
Nota. Muestra los diferentes arquetipos de la red neuronal Yolov5, tomado de (https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcRO4m-HMuA2J_pht3RXHFRsOw8lygslFxE-jw&usqp=CAU).

AlexNet.

AlexNet es un algoritmo desarrollado basándonos en una arquitectura de red CNN (Red Neuronal Convolucional) por Alex Krizhevsky, Ilya Sutskever y Geoffrey Hinton en el 2012. AlexNet implementa dentro de su estructura de red una serie de capas convolucionales, capas de asociación y capas de bloques de construcción básica que contienen alrededor de sesenta y dos millones de parámetros con ocho capas de profundidad, con datos tomados de la base de datos del proyecto ImageNet para su entrenamiento buscando clasificar imágenes en más de mil categorías de objetos contenidos en esta base de datos (ImageNet, 2022). AlexNet funciona mediante GPU's que permiten la utilización de varios núcleos que trabajan en conjunto para repartir el procesamiento de la información y logran obtener un rendimiento mucho más eficiente y rápido (Intel, 2023). Fue una de las primeras en usar ReLU (Unidad Lineal Rectificada) que permite la rectificación de funciones para aproximaciones analíticas, esquema que posteriormente fue adoptado por la mayoría de las arquitecturas de reconocimiento de imágenes, permitiendo mejoras en el reconocimiento de objetos, rostros y personas (DST, 2022).

Figura 6

Arquitectura de Alexnet.



Nota. Muestra el resumen gráfico de la arquitectura de Alexnet, tomado de (curso de certificación de Deep Learning Matwoks).

La estructura de trabajo de la red neuronal de AlexNet presenta un arreglo de capas donde los datos de entrada tienen un tamaño definido y mediante la utilización de clasificadores de régimen de entropía cruzada permiten obtener variables de salida que se pueden clasificar y catalogar basados en entrenamiento y porcentajes de certeza (latamt.ieeer9.org, 2022).

Figura 7

Número de capas de red neuronal profunda Alexnet.

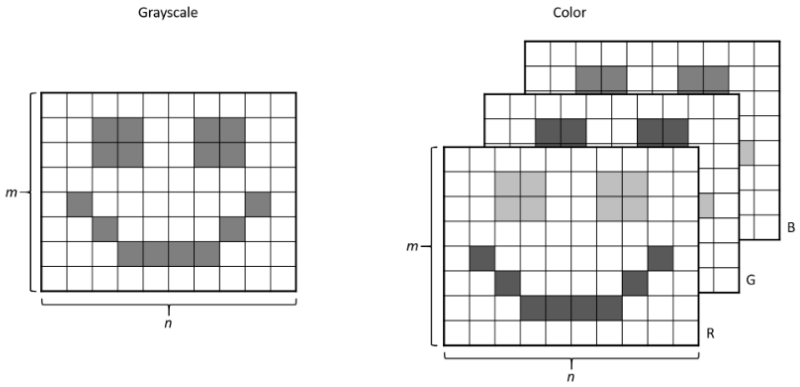
```
ly =
  25×1 Layer array with layers:

     1   'data'      Image Input
     2   'conv1'     Convolution
     3   'relu1'     ReLU
     4   'norm1'     Cross Channel Normalization
     5   'pool1'     Max Pooling
     6   'conv2'     Grouped Convolution
     7   'relu2'     ReLU
     8   'norm2'     Cross Channel Normalization
     9   'pool2'     Max Pooling
    10   'conv3'     Convolution
    11   'relu3'     ReLU
    12   'conv4'     Grouped Convolution
    13   'relu4'     ReLU
    14   'conv5'     Grouped Convolution
    15   'relu5'     ReLU
    16   'pool5'     Max Pooling
    17   'fc6'       Fully Connected
    18   'relu6'     ReLU
    19   'drop6'     Dropout
    20   'fc7'       Fully Connected
    21   'relu7'     ReLU
    22   'drop7'     Dropout
    23   'fc8'       Fully Connected
    24   'prob'      Softmax
    25   'output'    Classification Output
```

Nota. Muestra el número de capas que utiliza Alexnet, tomado de (curso de certificación de Deep Learning Matwoks).

Dimensiones de las Imágenes para AlexNet

Figura 8

Capas RGB red neuronal profunda Alexnet.



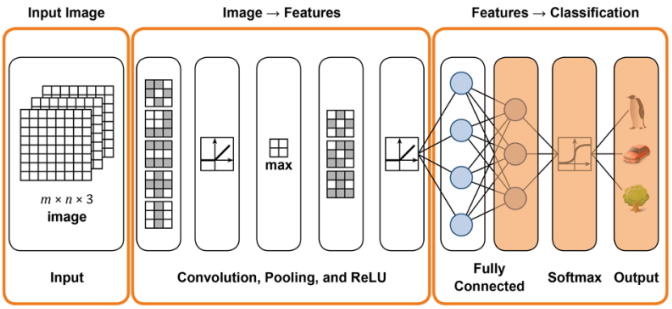
Nota. Muestra el procesamiento de Colores en Alexnet, tomado de (curso de certificación de Deep Learning Matwoks).

Las dimensiones de las imágenes de entrada requeridas por AlexNet son de 277 píxeles de alto x 277 píxeles de ancho x 3 colores RGB y representan un arreglo m por n, donde los elementos tienen un valor de intensidad en el píxel de la imagen. Las imágenes a color se representan como un arreglo m por n por 3, donde los tres planos m por n representan las intensidades de rojo, verde y azul.

Predicción

Figura 9

Clasificación de Imágenes red neuronal profunda Alexnet.

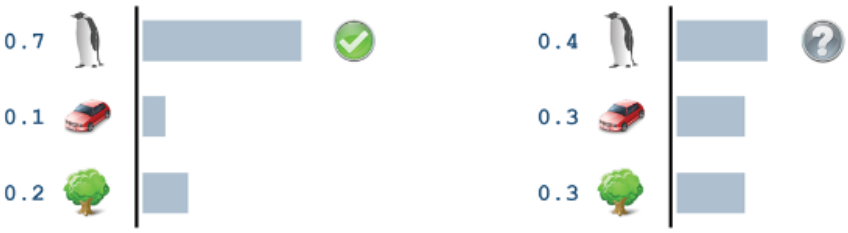


Nota. Muestra el procesamiento de imágenes con ReLu Alexnet, tomado de (curso de certificación de Deep Learning Matwoks).

AlexNet clasifica las imágenes mediante funciones y devuelve la clase predicha de la imagen de entrada con un grado de confiabilidad. Para clasificar las entradas en clases, la red neuronal utiliza una salida de "n" neuronas por cada clase. Con esto se puede obtener el cálculo numérico para cada neurona. Los valores numéricos representan la predicción de la red e indican la probabilidad de que esa entrada en particular pertenezca a una clase determinada.

Figura 10

Porcentajes de certeza de clasificación de imágenes.



.

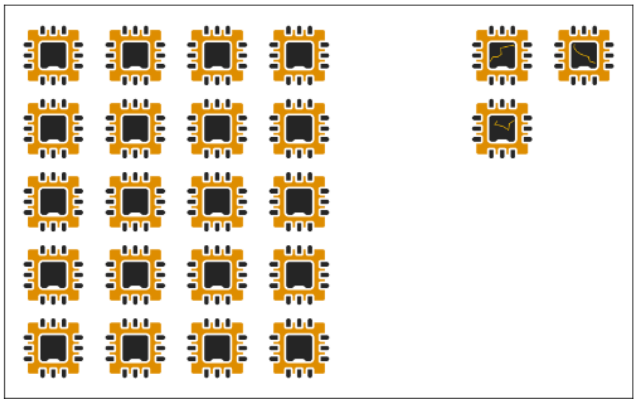
Nota. Muestra los resultados del análisis de imágenes de una red neuronal tomadas como acertadas sobre el 70% del total de images totales procesadas, tomado de (curso de certificación de Deep Learning Matwoks).

Datos de entrenamiento

Por lo general, al tener muchas más imágenes de una clase que de otra clase, se vuelve difícil tratar de identificar defectos dentro de las imágenes. Cuando AlexNet dentro de su estructura neuronal tiene como base imágenes perfectas, sin errores o imperfecciones, al intentar predecir la clase de las imágenes con defectos o imperfectas, falla debido a que es mucho más difícil obtener imágenes con defectos dentro de la clase que debe predecir (Matworks, 2022).

Figura 11

Sesgo existente respecto a la calidad de las imágenes.



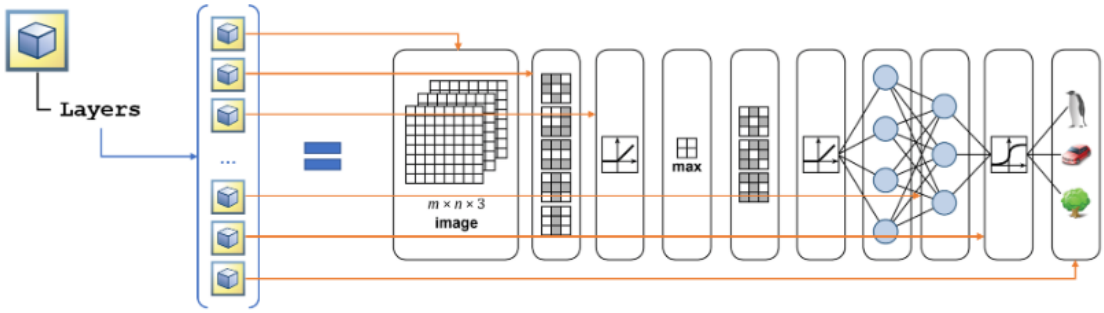
Nota. Muestra el Sesgo de identificación de imágenes imperfectas o desenfocadas, tomado de (curso de certificación de Deep Learning Matwoks).

Al momento de realizar el entrenamiento de la red, los datos proporcionados determinarán el porcentaje de predicciones y similitud de las diferentes clases, lo que puede dar como resultado un entrenamiento basado en imágenes sin defectos o con defectos dependiendo de la base de datos que usemos. Con esto se evidencia que la información de salida de la red neuronal puede quedar sesgada según el entrenamiento dado, mostrando porcentajes de clasificación no apegados a la realidad. En el caso de entrenar la red con datos sin errores o defectos, dará como resultado que la red recibirá entrenamiento fundamentalmente con imágenes sin defectos, descartando las imágenes que sí corresponden a la clase por contener errores o defectos (Matworks, 2022).

Para evitar este problema es necesario dividir los datos de manera que las imágenes de entrenamiento tengan el mismo número de cada clase.

Figura 12

Estructura de las capas de una red neuronal.

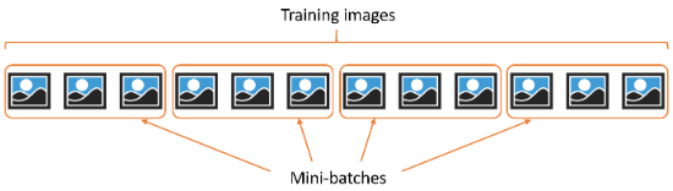


Nota. Muestra la estructura de una red neuronal, tomado de (curso de certificación de Deep Learning Matwoks).

Figura 13

"Minilotes" o conjunto de imágenes de entrenamiento.

```
Training on single GPU.
Initializing image normalization.
|=========================================================================================|
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Mini-batch  |  Base Learning  |
|         |             |   (seconds)    |     Loss     |   Accuracy   |      Rate       |
|=========================================================================================|
|       1 |           1 |           0.47 |       3.5061 |        7.81% |          0.0010 |
|       3 |          10 |          10.31 |       0.7686 |       75.00% |          0.0010 |
```

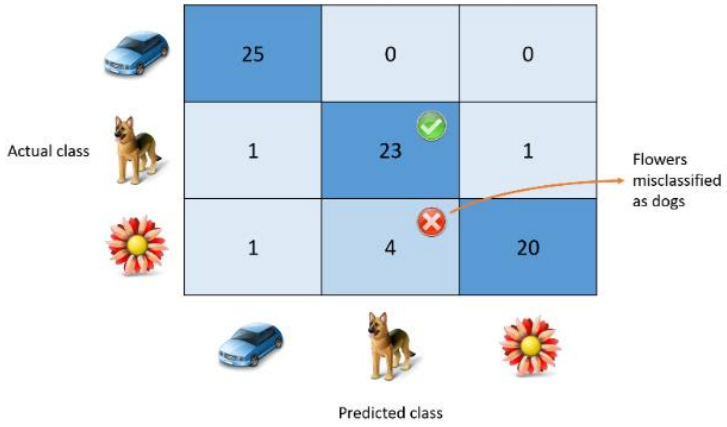


Nota. Muestra el esquema de procesamiento y modificación de capas con épocas, perdidas de datos y certeza de la red neuronal, tomado de (curso de certificación de Deep Learning Matwoks).

La utilización de subconjuntos de imágenes de entrenamiento llamados "mini lotes" permiten actualizar los pesos de entrenamiento en cada iteración para completar una época. El número de épocas y "mini lotes" se pueden establecer dependiendo de las capacidades y características del algoritmo de entrenamiento que se desea desarrollar o la cantidad de datos y certeza que se desea obtener de la red. La pérdida y la precisión determinan la eficiencia y el rendimiento de la red neuronal convolucional (CNN).

Figura 14

Matriz de predicción de clases Alexnet.



Nota. Comparación de clases existentes y la predicción de la red en una matriz simple, tomado de (curso de certificación de Deep Learning Matwoks).

2.2 Descripción de la propuesta

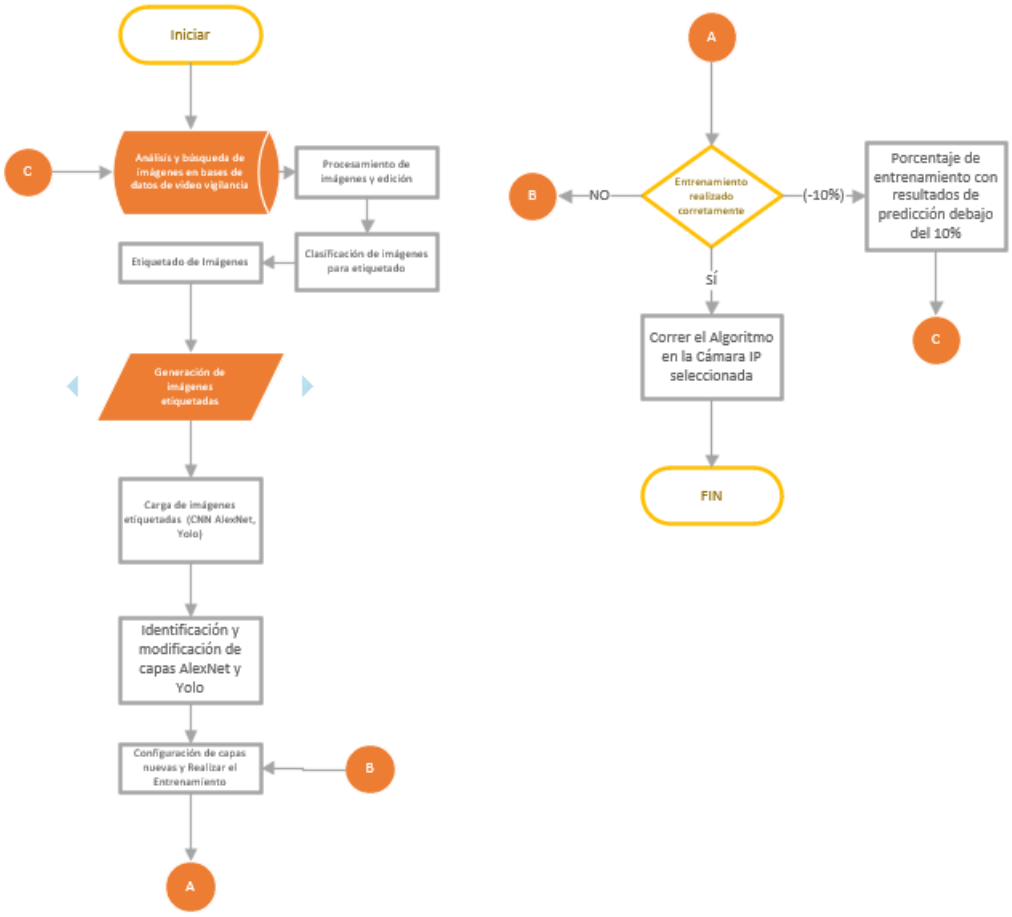
En la actualidad, las empresas y entidades públicas cuentan con centros de vigilancia y monitoreo de eventos en el Distrito Metropolitano de Quito. El empleo de recursos económicos y talento humano para el monitoreo eficaz y eficiente de los eventos relacionados con la seguridad pública aumentan constantemente en relación con el creciente número de accidentes y eventos relacionados con la seguridad ciudadana en el espacio público del Distrito Metropolitano de Quito. La presente propuesta tiene por objeto el desarrollo e implementación de una Red Neuronal Convolucional previamente entrenada, para la detección de accidentes de tránsito utilizando cámaras de video IP de la red de monitoreo del Cuerpo de Agentes de Control Metropolitano de Quito.

a. Estructura general

La estructura general del Proyecto se muestra a continuación:

Figura 15

Diagrama de Procesos.



Nota. Diagrama de flujo aplicada al proceso de desarrollo del algoritmo de detección de incidentes, Fuente Propia.

b. Explicación del aporte

El funcionamiento del algoritmo de detección de accidentes de tránsito consiste analizar las imágenes de una de las cámaras de videovigilancia que funcionan dentro de la red de cámaras de video del Centro de Mando y Comunicaciones del CACMQ. Mediante la conexión con protocolos RSTP (Rapid Spanning Tree Protocol) que permite una conexión segura y directa con las cámaras de videovigilancia, se ingresa la IP asignada a la cámara con las autenticaciones correspondientes al equipo (usuario y contraseña) y se observa en la pantalla la conexión con el algoritmo de MATLAB. Una vez conectado a la cámara se realiza la inicialización del algoritmo para que tome los datos de entrada de la cámara seleccionada y capture fotogramas para realizar la predicción, obteniendo un mensaje dentro del panel que muestra el estado actual de las imágenes procesadas (CISCO, 2023).

Recursos utilizados:

Acceso a información requerida para el procesamiento y predicción

La información requerida para el desarrollo del código que permita la detección autónoma de accidentes de tránsito será extraída de los servidores de video donde se almacenan los datos de las cámaras de videovigilancia instaladas en el Distrito Metropolitano de Quito. Mediante el acceso de las credenciales a los servidores, la información está disponible para el procesamiento.

Figura 16

Servidor de almacenamiento de video del CACMQ.

Nota. Servidor de video de cámaras de video del CACMQ, Fuente Propia.

Creación de base datos (Imágenes para entrenamiento)

Mediante la creación de una carpeta principal, llamada “raíz” se realizó la recopilación de 210 imágenes específicas de vehículos que permiten trabajar la red neuronal para el proceso de entrenamiento. Las imágenes cumplen con las características necesarias para ser utilizados como datos de entrada en la red neuronal con una resolución de entrada de 227×227 píxeles RGB a color. A continuación, se muestran las propiedades de la capa de entrada “data” de la red neuronal convolucional.

Figura 17

Formato de imágenes requeridas (alexnet).

```
ImageInputLayer with properties:

                    Name: 'data'
               InputSize: [227 227 3]
      SplitComplexInputs: 0

 Hyperparameters
        DataAugmentation: 'none'
           Normalization: 'zerocenter'
  NormalizationDimension: 'auto'
                    Mean: [227×227×3 single]
```

Nota. Muestra los requisitos que debe cumplir un lote de imágenes para el entrenamiento de la red neuronal, Matlab R2022b.

Una vez establecidas las características de las imágenes necesarias, se realizó la clasificación de las carpetas que contienen la información de las diferentes imágenes obtenidas. Se utilizaron únicamente dos tipos diferentes de categorías de datos, para poder realizar el desarrollo y configuración de la red neuronal.

Estrategias y/o técnicas

Librerías Python

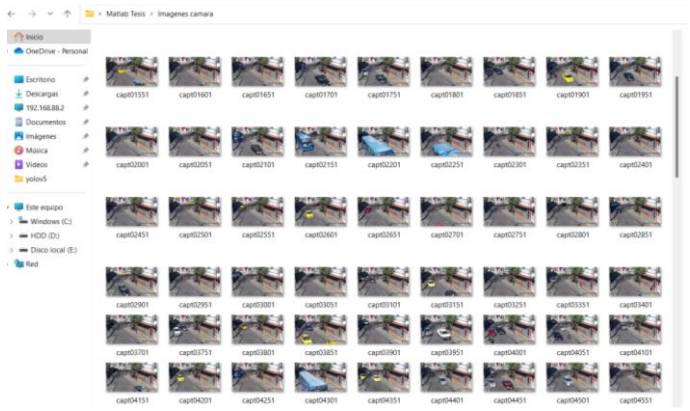
Las librerías de Python son un conjunto de repositorios donde se alojan una gran cantidad de código desarrollado con anterioridad para ser reutilizado en nuevos desarrollos de scripts. Las librerías son utilizadas como módulos desarrollados y almacenados en la web. La importación de estos contenedores de código depende de la distribución de Python que se esté usando. para el desarrollo de los objetivos del presente proyecto se utilizaron las siguientes librerías; Gitpython, matplotlib, numpy, opencv-python, torch, torchvision, pandas, nvidia-pyindex, ultralytics.

Datos

Los datos se clasificaron para realizar el entrenamiento de la red neuronal en dos categorías. La primera para la detección e identificación de vehículos sobre la vía y la segunda para la detección de accidentes de tránsito en la vía. Para la detección de vehículos en la vía fue necesaria la utilización de 619 imágenes (569 imágenes de entrenamiento y 50 imágenes para validación) y, para la detección de accidentes de tránsito se utilizaron 856 imágenes (806 para entrenamiento y 50 para validación).

Figura 18

Datos recolectados de servidores de video.



Nota. Muestra la carpeta de imágenes utilizadas para la modificación de la red neuronal Yolov5, Fuente propia.

Datos de entrenamiento

Para los datos de entrenamiento se utilizó el 90% del total de las imágenes obtenidas para el etiquetado.

Los datos de entrenamiento requieren un etiquetado de las imágenes que permita la identificación de las zonas de la imagen que contienen una acción determinada que se busca identificar. Los formatos de salida de las etiquetas varían dependiendo de la red neuronal que se utilice. En el etiquetado se usaron dos tipos de herramientas Matlab Labeler y makesense.ai.

Etiquetado de imágenes

Makesense.ia

Es una plataforma de procesamiento y etiquetado de imágenes en la nube que permite realizar el etiquetado de forma rápida y segura. No requiere realizar registros dentro de su plataforma y trabaja con varios tipos de formatos de salida. Es de código abierto y de uso gratuito bajo licencia GPLV3, y permite el etiquetado en formatos para YOLO, VOC XML, VGG JSON, CSV.

Figura 19

Imagen de Makesense.ia.

Nota. Aplicación de etiquetado de imágenes de software libre, tomado de(https://www.makesense.ai/).

Matlab Image Labeler

Es una aplicación desarrollada por Matlab dentro de una gran variedad de aplicaciones creadas para el procesamiento de imágenes para aplicaciones de visión por computador e inteligencia artificial en el entorno desarrollado por Matlab. Dentro de esta aplicación se encuentran varias funciones de etiquetado de imágenes. Los archivos generados por la aplicación solo se pueden utilizar en Matlab, además esta aplicación tiene un costo adicional si se quiere instalar dentro de Matlab.

Figura 20

Matlab Image Labeler.

Nota. Aplicación de etiquetado de imágenes de Matlab, tomado de(https://la.mathworks.com/help/vision/ref/image_labeler_refpage_screenshot.png).

Datos de validación

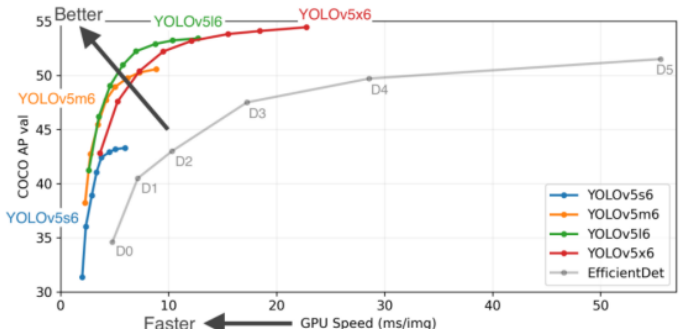
Para los datos de validación se utilizó el 10% del total de las imágenes obtenidas de las cámaras de videovigilancia de la red de cámaras del Distrito Metropolitano de Quito.

Precisión de la red neuronal

La precisión en la predicción de la red mediante el procesamiento de las imágenes recolectadas y etiquetadas depende de la calidad y de la identificación del etiquetado dentro de la imagen.

Figura 21

Velocidad de procesamiento vs arquetipo Yolov5.



Nota. Comparación de precisión entre arquitecturas de Yolov5, tomado de (https://github.com/ultralytics/yolov5/releases/download/v1.0/model_plot.png).

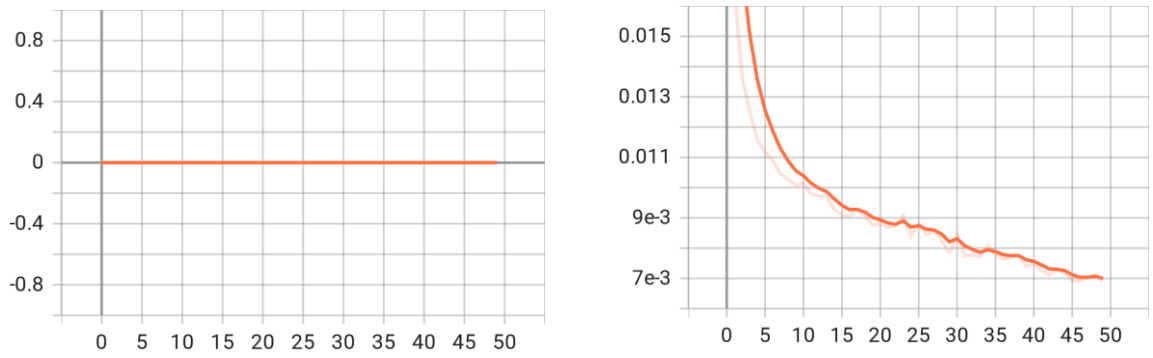
La precisión de la red utilizada en este caso es el arquetipo YOLOV5x que muestra la mayor precisión de entre las varias versiones de YOLOv5. Dentro de las redes la que mejor responde a una baja incidencia de falsos positivos es la antes mencionada. Se utilizo un número de 50 ciclos completos de paso del algoritmo (épocas) para el entrenamiento. Cabe aclarar que para el entrenamiento es necesario fue necesario tener en cuentas lo siguiente:

- Redefinir el tamaño de las imágenes de entrada.
- Determinar el tamaño del lote de imágenes utilizadas.
- Determinar el número de épocas de entrenamiento para los datos de entrada.
- La ubicación de los datos con los respectivos directorios y subdirectorios ordenados por datos de entrenamiento y datos de validación para cada tipo.
- Especificar los pesos de entrenamiento que en este caso corresponde al arquetipo de YOLOV5x.

Las gráficas obtenidas del entrenamiento de la red neuronal con el arquetipo especificado son las siguientes:

Figura 22

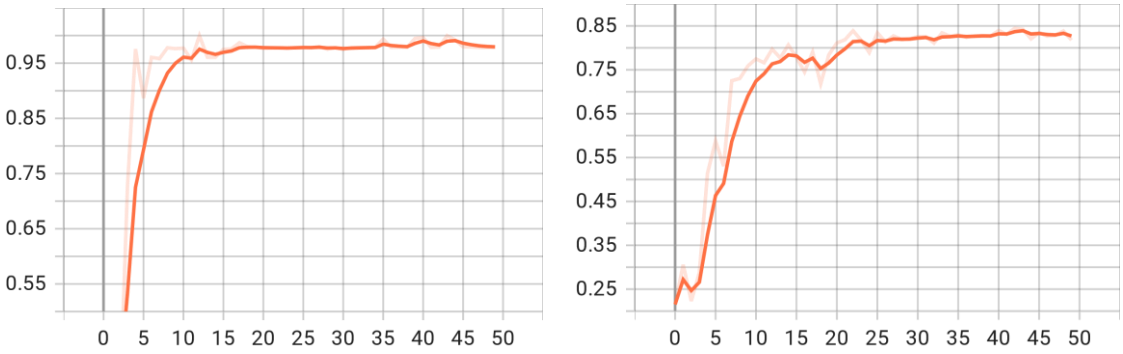
Perdida de datos durante el entrenamiento vs número de épocas del algoritmo.



Nota. Se muestra los porcentajes de pérdida de datos (eje horizontal) después de 50 épocas de entrenamiento (eje vertical) con Yolov5x, Fuente Propia.

Figura 23

Precisión del entrenamiento vs número de épocas del algoritmo.



Nota. Se muestra los porcentajes de certeza después del entrenamiento de la red neuronal Yolov5x (eje vertical) después de 50 épocas de entrenamiento (eje horizontal) con Yolov5x, Fuente Propia.

Utilización de CPU

Para el procesamiento de las imágenes se utiliza por defecto el CPU del computador, por ende, el procesamiento de las imágenes mediante captura de frames desde las cámaras que utilizan protocolos RSTP de transmisión dependerá de la capacidad de procesamiento de las imágenes del equipo donde se procesan.

Las imágenes obtenidas con el procesamiento del CPU del computador donde se realizan las pruebas (Intel(R) Core (TM) i7-8550U CPU @ 1.80GHz 1.99 GHz de octava generación), muestran una latencia mayor sobre las que utilizan GPU (Intel, 2023).

Utilización de GPU

Para un procesamiento más veloz y adecuado en la detección de incidentes es necesaria la utilización de procesadores especializados en el control de gráficos y efectos de video específicamente. Para esta configuración fue necesaria la identificación de las librerías de procesamiento de imágenes de Python y la compatibilidad de Pytorch usando GPU NVIDIA. Se utilizo CUDA 11.6 para Windows 11 y las librerías de `pytorch, torchvision, cudatoolkit.`

Figura 24

Integración de GPU en procesamiento de datos.

```
YOLOv5  v7.0-120-g3e55763 Python-3.9.13 torch-1.12.1+cu116 CUDA:0 (NVIDIA GeForce MX150, 4096MiB)
Setup complete  (8 CPUs, 15.9 GB RAM, 403.2/464.4 GB disk)
```

Nota. Se muestra la integración de la GPU del computador mediante CUDA 11.6 de NVIDA, Fuente Propia.

Una vez realizadas probada la funcionalidad y compatibilidad con el equipo se implementará en la red interna de video vigilancia del Cuerpo de Agentes de Control Quito.

2.3 Validación de la propuesta

La validación de la propuesta se puede demostrar una vez desarrollado el script en lenguaje Python mediante la interfaz web jupyter notebook donde se utiliza un kernel específico donde se ejecuta el código para comprobar su funcionamiento mediante la conexión y detección de eventos relacionados con automóviles en la vía.

Procesamiento de datos e imágenes

La comunicación con las cámaras de la red de videovigilancia se realizó mediante protocolos RTSP que permiten compartir video en tiempo real. Las cámaras al contar con controles de acceso y seguridades limitan su acceso por lo que las pruebas se las realizo con videos generados por las cámaras de video.

Detección

La detección se realizó en las cámaras de video instaladas en el DMQ, en la zona norte, centro y sur de la cuidad como se muestra en las siguientes imágenes:

Sector Centro de la Cuidad de Quito

Figura 25

Prueba realizada sobre la cámara ubicada en las avenidas 24 de Mayo y Mariscal Sucre (Los Túneles de San Roque).

Nota. Se muestra la detección de automóviles con una precisión del 90%, Fuente propia.

Figura 26

Prueba realizada sobre la cámara ubicada en las avenidas Velasco Ibarra y Pichincha (EL Trébol).

Nota. Detección de automóviles al 91%, (detección realizada en vehículos), Fuente propia.

Sector Norte de la Cuidad de Quito

Figura 27

Prueba realizada sobre la cámara ubicada en las avenidas Amazonas y Cristóbal Colón (La Mariscal).

Nota. Detección de automóviles al 93%, (detección realizada en vehículos), Fuente propia.

Figura 28

Detección de automóviles al 83%, (detección realizada en vehículos).

Nota. Prueba realizada sobre la cámara ubicada en las avenidas Rio Coca y 6 de Diciembre, Fuente propia.

Figura 29

Detección de un posible accidente de tránsito.

Nota. Prueba realizada sobre la cámara ubicada en las avenidas Rio Coca y 6 de Diciembre, Fuente propia.

Sector Sur de la Cuidad de Quito

Figura 30

Detección de un posible accidente de tránsito.

Nota. Prueba realizada sobre la cámara ubicada en las avenidas Quitumbeñan y Moran Valverde, Fuente propia.

Figura 31

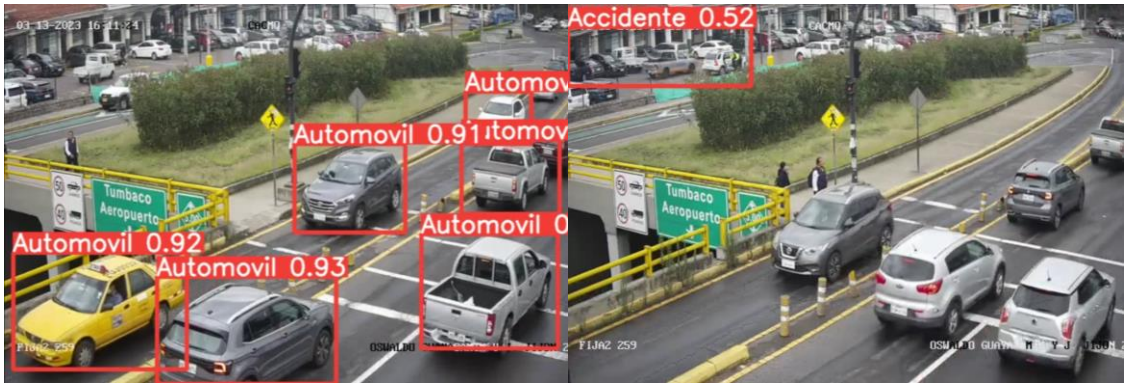
Detección de vehículos sobre reasfaltado de vía.

Nota. Prueba realizada sobre la cámara ubicada en las avenidas Moran Valverde y Manuel Coronado (Chillogallo), Fuente propia.

Sector Valle de Tumbaco

Figura 32

Detección de vehículos 93% de precisión.



Nota. Prueba realizada sobre la cámara ubicada en las avenidas Oswaldo Guayasamín y J. Jijón (Valle de Tumaco), Fuente propia.

Descripción de perfil de validadores

Tabla 1

Descripción de perfil de validadores.

Nombres y Apellidos	Años de experiencia	Titulación Académica	Cargo
Wilson Iván Rea Chicaiza	9 (nueve años)	Ingeniero en Electrónica y Telecomunicaciones	Responsable de Desarrollo de Proyectos del Centro de Mando y Comunicaciones del CACMQ
Wilson Honorio Ponce	29 (veinte y nueve años)	Licenciado en Ciencias de la Educación Mención Física y Matemáticas	Responsable del Centro de Mando y Comunicaciones y ECU 911 del CACMQ
Luis Orlando Catota Puruncaja	19 (diez y nueve años)	Magister en Gestión de Proyectos	Coordinador del Área Técnica y de Proyectos del Centro de Mando y Comunicaciones del CACMQ

Nota. Se muestra la descripción de los tres profesionales validadores del presente proyecto de titulación.

2.4 Matriz de articulación de la propuesta

En la presente matriz se sintetiza la articulación del producto realizado con los sustentos teóricos, metodológicos, estratégicos-técnicos y tecnológicos empleados.

Tabla 2

Matriz de articulación

Ejes o partes principales del proyecto		Breve descripción de los resultados de cada parte	Sustento teórico que se aplicó en la construcción del proyecto	Metodologías, herramientas técnicas y tecnológicas que se emplearon
1	El proyecto fue desarrollado mediante la aplicación de lenguaje de programación Python y Matlab. Análisis de métodos de procesamiento de imágenes y complejidad en el desarrollo del código de programación en los diferentes lenguajes utilizados.	Verificación de precisión de identificación de vehículos sobre el 90% posterior al entrenamiento de la red neuronal. Verificación de identificación de accidentes de tránsito sobre el 80% en los videos extraídos de las cámaras de video vigilancia dentro del perímetro urbano de la cuidad. Los costos de adquisición de software licenciado por adquisición de la licencia de Matlab.	Inteligencia Artificial: NumPy, SciPy, Matplotlib, Pip, Pandas. Visión por Computador: Seaborn Machine Learning: scikit-learn. Deep Learning: TensorFlow, Keras, PyTorch, SHAP.	Matlab: Red neuronal: Alexnet Procesamiento de Datos: Matlab Image Labeler Python: Yolov5 Xl GPL-3.0 Makesense.ia

		Identificación de incidentes sobre el espacio público para la asistencia oportuna de accidentes y afines.	Procesamiento de Lenguaje Python: Anaconda Navigator, Jupyter Notebook, Anaconda mini	
2	Recolección de información mediante la adquisición de imágenes de cámaras de seguridad instaladas en el distrito metropolitano. La clasificación de imágenes para el entrenamiento y finalmente la precisión obtenida posterior al análisis del algoritmo. 619 imágenes para reconocimiento de vehículos 856 imágenes para identificación de accidentes de transito	2.1. Aplicación de Python y librerías. 2.2 Aplicación Matlab y Matwork app's.	Programación en Matlab y Python.	Relacionar las herramientas que se aplicaron para cada parte y por qué se aplicaron las mismas

3	Se realizo la implementación con el procesamiento de videos extraídos de la base de datos de las cámaras de video vigilancia en el distrito Metropolitano de Quito, observado la identificación en las imágenes mostradas después de su procesamiento.	3.1. Adquisición de cámaras IP con protocolos de comunicación Onvif y Hikvsion (Cámara IP marca IPCHD de 2MP PTZ, y cámara Hikvision modelo DS-2CD1321-I). 3.2. distribución de Python Anaconda 3 y Minianaconda y Jupyter Notebook	Protocolos de comunicación RTSP, HTTPS Y ONVIF soporte de software para uso de procesamiento especializado de gráficos de múltiples núcleos mediante Toolkit CUDA11.6 para desarrolladores (NVIDIA, 2023).	Computador (Intel(R) Core (TM) i7-8550U CPU @ 1.80GHz 1.99 GHz de octava generación 16 GB de memoria Ram y 4096 MiB de GPU tarjeta de video NVIDIA Geforce MX150).

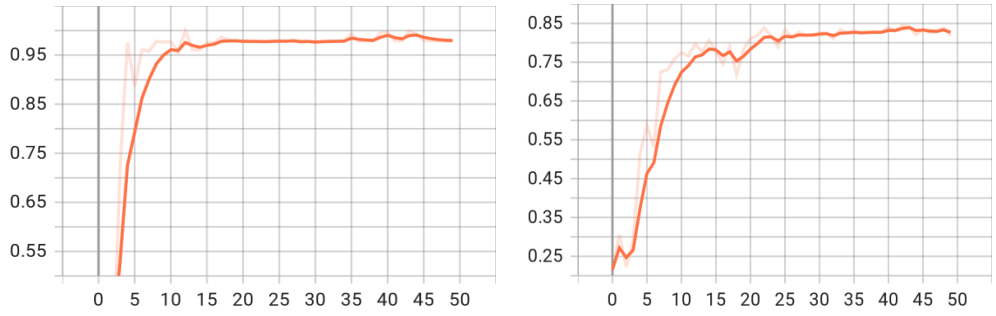
Nota. Se muestra la Matriz de articulación del proyecto de titulación.

2.5 Análisis de resultados. Presentación y discusión.

Los resultados obtenidos sobre el desarrollo y aplicación de algoritmos de aprendizaje autónomo muestran que la red neuronal depende directamente de la calidad de imágenes y la cantidad de elementos considerados para la modificación de las capas de la red neuronal. Mientras más datos se utilizó en el entrenamiento y más número de épocas se configuraron, los resultados arrojaron un menor número de falsos positivos dentro de las imágenes de video de las cámaras de videovigilancia.

Figura 33

Precisión en la detección de vehículos.



Nota. Se muestra los porcentajes de certeza (eje vertical) versus las 50 épocas de entrenamiento (eje horizontal) con Yolov5x, Fuente Propia.

Los protocolos de comunicación utilizados por los fabricantes de cámaras de propiedad de Municipio de Quito para el monitoreo de incidentes dentro el espacio público no permite una integración con softwares de código abierto. La identificación de eventos posterior a la implementación se volvió autónoma como se muestra en las figuras obtenidas posterior al procesamiento de los videos obtenidos de las cámaras de videovigilancia.

Figura 34

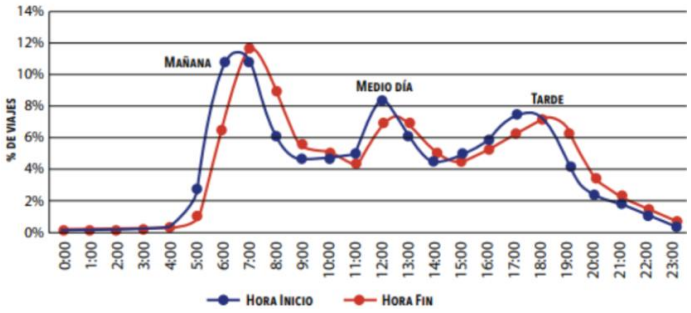
Resultados de la detección sobre vías del sur y norte de la cuidad de Quito.

Nota. Se muestra la detección de un posible accidente de tránsito, Fuente Propia.

Los resultados con respecto al problema de investigación arrojaron que en la cuidad de Quito existen altos niveles de congestión vehicular, especialmente en las horas pico. La mayor parte de las cámaras de videovigilancia se encuentran en sectores con alta afluencia vehicular. Uno de los objetivos del presente trabajo es la detección de accidentes de tránsito y aclarando que una de las características principales de un accidente de tránsito es el congestionamiento de vehículos y la detención del flujo vehicular, dando como resultado que la aplicación del algoritmo es más efectiva en cámaras de video que monitorean vías de alta velocidad ubicadas en el perímetro de la cuidad de Quito (Gestion-Digital, 2023).

Figura 35

Concentración de viajes por diarios por horario en la cuidad de Quito 2022.



Nota. Se muestra la densidad de tráfico en los horarios del día con respecto al porcentaje de viajes que se realizan al día en la cuidad de Quito, tomada de (https://revistagestion.ec/sites/default/files/inline-images/Captura%20de%20Pantalla%202023-02-07%20a%20la%28s%29%2014.01.07.png).

CONCLUSIONES

Las técnicas utilizadas para el análisis de datos por visión artificial fueron establecidas con el uso de dos softwares, MATLAB y Python. El primero (Matlab) utiliza técnicas de aprendizaje supervisado basado en algoritmos de regresión y el segundo (Python) con técnicas de aprendizaje supervisado en algoritmos de clasificación. Se utilizaron para lograr un entrenamiento de una red neuronal cambiando sus capas de aprendizaje y modificando los datos de entrada. Se tuvo éxito únicamente en la conexión y procesamiento de imágenes con Python y ya que Matlab en su última actualización (2022b) no permite la comunicación por protocolos RTSP.

Se establecieron técnicas de análisis de video utilizando técnicas de visión artificial basadas en aprendizaje autónomo supervisado de código abierto Yolov5 que permitieron la clasificación de imágenes por categorías para la identificación de automóviles y accidentes con una precisión de más del 90% en la identificación de las mencionadas categorías (automóviles y posibles accidentes de tránsito).

El código desarrollado para la aplicación de la red neuronal integra librerías de visión por computador y análisis de datos de ultralitycs de Python y código desarrollado en Matlab utilizando Alexnet.

La implementación se realizó con cámaras IP dentro de la red Interna de CACMQ que manejan autenticación básica y protocolos abiertos onvif de integración, además de videos extraídos de las cámaras de video del DMQ.

RECOMENDACIONES

Sobre las técnicas utilizadas para el análisis de datos del presente trabajo de titulación. Es necesario aclarar que existe una gran variedad de software que manejan los mismos conceptos. Es necesaria para futuras investigaciones tomar en cuenta las compatibilidades que existen entre los protocolos de comunicación de cámaras IP. Los protocolos muchas veces no son compatibles con la tecnología de código abierto. Las seguridades de empleadas en la transmisión y las codificaciones que utilizan para el acceso a esas cámaras no permiten. Capturar el vídeo en vivo Motivo por el cual, En ciertos casos no es posible una conexión en vista en directo de los incidentes que se registran en las cámaras de videovigilancia.

En lo referente a las técnicas utilizadas para el procesamiento de imágenes en el presente trabajo, me permito destacar la dificultad de manejar un software licenciado (Matlab) que, aunque ejecuta líneas de código de Python, la programación y el desarrollo es mucho más complejo. Es recomendable usar lenguajes de programación de código abierto disponen de una gran cantidad de información y librerías que pueden ser usadas en el desarrollo de nuevos scripts sin tener que desarrollar desde cero un algoritmo de aprendizaje autónomo desde cero.

El código desarrollado en el presente trabajo utiliza una gran variedad de librerías para el procesamiento de imágenes por computadora y aprendizaje autónomo. Si bien se alcanzó el objetivo del presente proyecto es necesario recomendar la utilización de equipos que soporten en el procesamiento de las imágenes. En el desarrollo del presente proyecto se encontraron varios inconvenientes sobre los requisitos de hardware y utilización del GPU y CPU para el procesamiento de imágenes y vídeos.

En sesión del del código desarrollado dentro del lenguaje de programación Python hubo varios inconvenientes con respecto a la conexión de las cámaras de streaming en vivo de la marca Hikvision. Los fabricantes de las cámaras de propiedad del municipio de Quito cuentan con licencias de funcionamiento y seguridades de autenticación. Se recomienda que para próximas implementaciones se utilice cámaras que permitan una comunicación directa con el software libre.

BIBLIOGRAFÍA

A. Corrales, R. R. (02 de 02 de 2022). *Roboticslab Universidad Carlos III de Madrid.* http://roboticslab.uc3m.es/publications/1952-RFIDSkill.pdf

Adler, Q. V. (12 de 08 de 2021). *Google Academico.* https://repositorio.uss.edu.pe/bitstream/handle/20.500.12802/8392/Quiroz%20Valencia%20Adler%20Luis.pdf?sequence=1&isAllowed=y

Aguilera, D. M. (12 de 08 de 2021). *oa.upm.es.* https://oa.upm.es/66290/1/TFG_DANIEL_MORA_AGUILERA.pdf

Alarcon, N. C. (09 de Junio de 2020). *SYSCOM*. SYSCOM: https://soporte.syscom.mx/es/articles/2259977-mikrotik-caracteristicas-principales-de-routeros

Alberto Gómez-Gómez, B. E.-R. (02 de 02 de 2022). *revista.profesionaldelainformacion.com.* https://revista.profesionaldelainformacion.com/index.php/EPI/article/view/epi.2007.jul.05/31641

Algoritmos. (06 de Mayo de 2020). *profesores.ar*. (Profesores.ar) Retrieved 06 de Mayo de 2020, from http://www.profesores.frc.utn.edu.ar/sistemas/ingsanchez/Redes/Archivos/AlgoritmosCCong.asp

Alonso, N. O. (01 de 01 de 2013). *books.google.com.* Retrieved 05 de 02 de 2022, from https://books.google.com.ec/books?id=4TKJ9IpMSJEC&pg=PT263&lpg=PT263&dq=%E2%80%A2+JB+(Junction+Box):+permite+hasta+dos+accesos+para+derivaciones.&source=bl&ots=gUCuxAGszo&sig=ACfU3U1lnq-3NwLGLqOeUj_nouz8ReOTzg&hl=es-419&sa=X&ved=2ahUKEwjJ8JL88u71AhUGRTABH

ALONSO, N. O. (14 de 02 de 2022). *EBOOKS.* https://books.google.es/books?hl=es&lr=&id=4TKJ9IpMSJEC&oi=fnd&pg=PT214&dq=protocolo+WORLDFIP&ots=gUCvvsOtzm&sig=V_6WibZSbrSHGnMPWC2BC2gRXDk#v=onepage&q=protocolo%20WORLDFIP&f=false

Archilinux. (20 de Junio de 2020). *Archilinux*. Archilinux: https://wiki.archlinux.org/index.php/QEMU

Arévalo, F. (2020). Importancia de la seguridad de los sistemas de información frente el abuso, error y hurto de información. *Revista cientifica dominio de la ciencia*, 841.

Avila, H. (20 de Abril de 2020). *eumed.net*. eumed.net: http://www.eumed.net/libros-gratis/2006c/203/1s.htm

Bellido, E. (03 de 10 de 2021). http://repositorio.untels.edu.pe//handle/123456789/355

Benitez, R. (01 de 09 de 2014). *Google academico.* https://books.google.es/books?id=eT7ABAAAQBAJ&lpg=PT4&ots=9xcHj34EEo&dq=analisis%20de%20imagenes%20inteligencia%20artificial&lr&hl=es&pg=PT8#v=onepage&q=analisis%20de%20imagenes%20inteligencia%20artificial&f=false

Bolaños, J. (2014). Modelo de Gestión del Circuito Centenario del Distrito Sur de la Zona 8 de la Policia Nacional del Ecuador . *Universidad San Francisco de Quito*, 13. http://repositorio.usfq.edu.ec/bitstream/23000/4558/1/120497.pdf

BONILLA, P. (2015). http://repositorio.espe.edu.ec/bitstream/21000/11237/1/T-ESPE-049435.pdf

Caparrini, F. S. (20 de 08 de 2022). *www.cs.us.es*. http://www.cs.us.es/~fsancho/?e=77

Carrion, G. (10 de Noviembre de 2014). *Universidad Nacional de Loja*. Universidad Nacional de Loja: https://dspace.unl.edu.ec/jspui/bitstream/123456789/11423/1/Carri%C3%B3n%20Matamoros%2C%20Marlon%20Gabriel%20.pdf

Cartagena, J. (09 de Noviembre de 2019). *profesores.elo.utfsm.cl.* profesores.elo.utfsm.cl: http://profesores.elo.utfsm.cl/~agv/elo322/1s08/project/JuanCartajena.pdf

Castillo, Á. M. (08 de Enero de 2020). *Guía para proyectos de investigación*. Universitas: file:///C:/Users/DET-PC/Downloads/105-Texto%20del%20art%C3%ADculo-202-1-10-20150605.pdf

Ceballos Bejarano Edison Wernher. (s.f.). http://athenea.autanabooks.com/index.php/revista/article/view/8/15

CGE. (08 de 03 de 2021). https://www.contraloria.gob.ec/Normatividad/BaseLegal

CGE. (29 de abril de 2021). Norma de Control Interno. Quito, Pichincha, Ecuador: LEXIS.

Chen, Z. (12 de 02 de 2023). *MDPI*. MDPI: https://www.mdpi.com/2073-4395/12/2/365

Chust, A. P. (20 de Abril de 2020). *Universidad Politecnica de Valencia*. Universidad Politecnica de Valencia: http://www.upv.es/visorv/media/a062b7cb-cb84-954c-a5fe-1247082dc09f/v

Cillero, M. (10 de Junio de 2020). *Manuel Cillero*. Manuel Cillero: https://manuel.cillero.es/doc/metrica-3/tecnicas/pruebas/sistema/

CISCO. (01 de 03 de 2023). *cisco.com*. https://www.cisco.com/c/en/us/support/docs/lan-switching/spanning-tree-protocol/24062-146.html

CODIGO MUNICIPAL, M. D. (28 de abril de 2019). *Portal Municipal*. Portal Municipal: http://www.epmrq.gob.ec/images/servicios/Codigo_Municipal.pdf

COESCOP. (2020). CODIGO ORGANICO DE ENTIDADES COMPLEMENTARIAS DE SEGURIDAD. En A. N. ECUADOR, *Entidades Complementarias de Seguridad de los Gobiernos Autonomos decentralizados municipales o metropolitanos* (pág. 51). quito.

COMERCIO, E. (15 de 12 de 2022). *EL COMERCIO*. https://www.elcomercio.com/actualidad/quito/suman-vehiculos-nuevos-quito-2022.html

Departamento de Planificacion CACMQ. (05 de Julio de 2019). Direccionamiento Estratégico para el Desarrollo Institucional 2019-2023 CACMQ. *Direccionamiento Estratégico para el Desarrollo Institucional 2019-2023 CACMQ*. Quito, Pichincha, Ecuador: s/n.

DST. (20 de 08 de 2022). *datascience.eu*. https://datascience.eu/es/aprendizaje-automatico/funcion-de-activacion-relu/

Gestion-Digital. (12 de 02 de 2023). *Gestion-Digital*. Gestion-Digital: https://www.revistagestion.ec/analisis-sociedad/el-trafico-y-la-movilidad-el-gran-desafio-de-la-nueva-alcaldia-de-quito#:~:text=De%20acuerdo%20con%20INRIX%202022,del%20centro%20de%20la%20ciudad.

Granado, E. C. (20 de 02 de 2023). *Google academico*. Google academico: https://eprints.ucm.es/id/eprint/48304/1/ManualJupyter.pdf

ImageNet. (20 de 08 de 2022). *www.image-net.org*. https://www.image-net.org/

INEC. (22 de 02 de 2023). *INEC*. INEC: https://app.powerbi.com/view?r=eyJrIjoiYmM4NWZjNTktNGRlZi00NDkxLWEzOWUtYmEwNDg3NTYwMjI1IiwidCI6ImYxNThhMmU4LWNhZWMtNDQwNi1iMGFiLWY1ZTI1OWJkYTExMiJ9

Intel. (23 de 01 de 2023). *Intel*. Intel: https://www.intel.com/content/www/us/en/products/docs/processors/what-is-a-gpu.html

latamt.ieeer9.org. (20 de 08 de 2022). *latamt.ieeer9.org*. https://latamt.ieeer9.org/index.php/transactions/article/download/4336/961/63509#:~:text=La%20funci%C3%B3n%20softmax%20toma%20como,n%C3%BAmero%20de%20entradas%20(clases).&text=Una%20importante%20caracter%C3%ADstica%20de%20AlexNet,ReLu%20(Rectified%20Linea

Matworks. (15 de 03 de 2022). *Matworks*. Matworks: https://la.mathworks.com/products/image.html

NVIDIA. (06 de 02 de 2023). *NVIDIA.DEVELOPER*. NVIDIA.DEVELOPER: https://developer.nvidia.com/cuda-11-6-0-download-archive

Ouahhadi, E. F. (06 de 10 de 2020). *Fátima Ouahhadi*. https://academica-e.unavarra.es/bitstream/handle/2454/37597/116975ouahhadiTFG.pdf?sequence=1&isAllowed=y

Pacanchique, E. (12 de 11 de 2020). *Universidad Catolica de Colombia*. https://repository.ucatolica.edu.co/bitstream/10983/25158/1/DocumentoTesis%20%281%29.pdf

Pérez-Villar, J. I. (15 de 06 de 2015). *repositorio.uam.es*. repositorio.uam.es: https://repositorio.uam.es/bitstream/handle/10486/667858/Bravo_Perez_Villar_JuanIgnacio.tfg.pdf?sequence=1&isAllowed=y

Ramos, L. M. (12 de 07 de 2016). *repositorio.uam.es*. repositorio.uam.es: https://repositorio.uam.es/bitstream/handle/10486/675526/Gil_Ramos_LuisMiguel_tfg.pdf?sequence=1&isAllowed=y

Rioja, U. d. (05 de 11 de 2022). *Rioja*. https://reunir.unir.net/bitstream/handle/123456789/12124/Tinajero%20Le%c3%b3n%2c%20Jose%20Luis%3b%20Fern%c3%a1ndez%20Ortiz%2c%20Eduardo%3b%20Pa%c3%b1i%20Pizarro%2c%20Jose%20Luis.pdf?sequence=1&isAllowed=y

SAS, I. (07 de 04 de 2022). *SAS, Institute*. SAS, Institute: https://www.sas.com/es_ar/insights/analytics/deep-learning.html

Tigrero, W. F. (14 de 02 de 2022). *Repositorio Institucional de la Universidad Politécnica Salesiana* . Repositorio Institucional de la Universidad Politécnica Salesiana : http://dspace.ups.edu.ec/handle/123456789/20649

Uribe, J. A. (10 de 08 de 2022). *Dialnet*. https://dialnet.unirioja.es/descarga/articulo/4797311.pdf

XYZ, D. (10 de Junio de 2020). *Definicion XYZ*. Definicion XYZ: https://www.definicion.xyz/2018/03/analisis-de-costo.html

ANEXOS

ANEXO 1

CÓDIGO PARA DETECCIÓN DE ACCIDENTES DE TRÁNSITO

```
url="rtsp://192.168.1.50:8554/profile0"
cap = cv2.VideoCapture(url)
#cap=cv2.resize(cam,(500,500))
fgbg = cv2.bgsegm.createBackgroundSubtractorMOG()
kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(3,3))
while True:

    ret,frame=cap.read()

    if ret == False: break
    grayy = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    # Dibujamos un rectángulo en frame, para señalar el estado
    # del área en análisis (movimiento detectado o no detectado)
    cv2.rectangle(frame,(0,0),(frame.shape[1],40),(0,0,0),-1)
    color = (0, 255, 0)
    cuadro_dialogo = "Estado: No se ha detectado movimiento"

    area_pts = np.array([[150,320], [300,100], [620,frame.shape[0]],
[50,frame.shape[0]]])

    imAux = np.zeros(shape=(frame.shape[:2]), dtype=np.uint8)
    imAux = cv2.drawContours(imAux, [area_pts], -1, (255), -1)
    image_area = cv2.bitwise_and(gray, gray, mask=imAux)
    print('numero de objetos: ', len(frame))


    fgmask = fgbg.apply(image_area)
    fgmask = cv2.morphologyEx(fgmask, cv2.MORPH_OPEN, kernel)
    fgmask = cv2.dilate(fgmask, None, iterations=2)



    cnts = cv2.findContours(fgmask, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)[0]
    for cnt in cnts:
        if cv2.contourArea(cnt) > 500:
            x, y, w, h = cv2.boundingRect(cnt)
            cv2.rectangle(frame, (x,y), (x+w, y+h),(0,255,0), 2)
            cuadro_dialogo = "Estado: Alerta Movimiento Detectado!"
            color = (120, 090, 199)


    cv2.drawContours(frame, [area_pts], -3, color, 1)

    cv2.putText(frame, cuadro_dialogo, (6, 20),
                cv2.FONT_HERSHEY_SIMPLEX,2, color,1)
```

```
    cv2.imshow('imagen',frame)



    k = cv2.waitKey(10) & 0xFF
    if k == 27:
        break


sourcc = str(sourcc)
gimg = not nosave and not sourcc.endswith('.txt')  # extencion de archivos
etiquetas inferencia
archi = Path(sourcc).suffix[1:] in (IMG_FORMATS + VID_FORMATS)
is_url = sourcc.lower().startswith(('rtsp://'))
camweb = sourcc.isnumeric() or sourcc.endswith('.streams') or (is_url and not
archi)
screenshot = sourcc.lower().startswith('screen')
if is_url and archi:
    sourcc = check_file(sourcc)  # descarga de archivos

# Directorios de guardado de descargas de archivos
s_dirct = aumnt_path(Path(project) / name, exist_ok=exist_ok)  #
especificacion de nobre de proyecto y constancia de existencia
(s_dirct / 'etqs' if g_tx else s_dirct).mkdir(parents=False, exist_ok=False)
# guardado automatico

# subida de archivos del modeloooo y extenciones que se edmite
dispoo = select_dispoo(dispoo)
modelo = DetectMultiBackend(weights, dispo=dispo, dpn=dpn, data=data,
fp16=half)
stride, names, pt = modelooo.stride, modelooo.names, modelooo.pt
imgsz = check_img_size(imgsz, s=stride)  # revisa el tamaño de la imagen

# carga de datos
bs = 1  # peso de archivos por cantidad
if camweb:
    view_img = check_imshow(warn=True)
    dataset = LoadStreams(sourcc, img_size=imgsz, stride=stride, auto=pt,
vid_stride=vid_stride)
    bs = len(dataset)
elif screenshot:
    dataset = LoadScreenshots(sourcc, img_size=imgsz, stride=stride, auto=pt)
else:
    dataset = LoadImages(sourcc, img_size=imgsz, stride=stride, auto=pt,
vid_stride=vid_stride)
vid_path, vid_writer = [None] * bs, [None] * bs

# Inferencia identificacion de objetos
modelo.warmup(imgsz=(1 if pt or modelo.triton else bs, 3, *imgsz))  #
ver, windows, dt = 0, [], (Profile(), Profile(), Profile())
for path, im, imms, vid_cap, s in dataset:
    with dt[0]:
        img = torch.from_numpy(im).to(modelo.dispo)
        img = im.half() if modelo.fp16 else im.float()
        img /= 255
```

```
        if len(im.shape) == 3:
            img = im[None]

    # Identificacion de objetos en tiempo real
    with dt[1]:
        visual = aumnt_path(s_dirct / Path(path).stem, mkdir=True) if visual
else False
        prdict = modelo(im, augment=augment, visual=visual)

    # NMS supresion de deteccion del cuadro al mayor porcentaje de pixeles
    with dt[2]:
        prdict = non_max_suppression(prdict, conf_thres, iou_thres, classes,
agnostic_nms, max_det=max_det)


    # Prdicticciones del algoritmo
    for i, det in enumerate(prdict):  # por cada frame
        ver += 1
        if camweb:  # cambio de tamaño de la imagen
            p, imm, frame = path[i], imms[i].copy(), dataset.count
            s += f'{i}: '
        else:
            p, imm, frame = path, imms.copy(), getattr(dataset, 'imagen', 0)

        p = Path(p)  # seleccion de archivo de imagen dentro de la carpeta
        g_dir = str(s_dirct / p.name)  # im.jpg
        t_dir = str(s_dirct / 'etqs' / p.stem) + ('' if dataset.mode ==
'image' else f'_{frame}')  # im.txt
        s += '%gx%g ' % im.shape[2:]
        gn = torch.tensor(imm.shape)[[1, 0, 1, 0]]
        imc = imm.copy() if save_crop else imm
        anotar = Anotar(imm, linea_width=linea_thickness, ejemp=str(names))
        if len(det):
            # modificacion de tamaño de la img_size a imm size
            det[:, :8] = scale_boxes(im.shape[4:], det[:, :6],
imm.shape).round()

            # Muestra resultados de la inferenica
            for c in det[:, 5].unique():
                n = (det[:, 5] == c).sum()  # deteccion de la clase
                s += f"{n} {names[int(c)]}{'s' * (n > 1)}, "

            # impresion de resultados
            for *xyxy, conf, cls in reversed(det):
                if g_tx:  # creacion de archivo txt
                    xywh = (xyxy2xywh(torch.tensor(xyxy).view(1, 4)) /
gn).view(-1).tolist()  # establecimietno de anchos y centros
                    linea = (cls, *xywh, conf) if save_conf else (cls, *xywh)
# formato de la etiqueta
                    with open(f'{t_dir}.txt', 'a') as f:
                        f.write(('%g ' * len(linea)).rstrip() % linea + '\n')

                if gimg or save_crop or view_img:  # se añade la etiqueta a la
imagen
                    c = int(cls)
                    etq = None if hide_etqs else (names[c] if hide_conf else
f'{names[c]} {conf:.2f}')
```

```
                    anotar.box_etq(xyxy, etq, color=colors(c, True))
                if save_crop:
                    save_one_box(xyxy, imc, file=s_dirct / 'crops' / names[c]
/ f'{p.stem}.jpg', BGR=True)

        # muestra resultados inferencia de imagenes
        imm = anotar.result()
        if view_img:
            if platform.system() == 'Linux' and p not in windows:
                windows.append(p)
                cv2.namedWindow(str(p), cv2.WINDOW_NORMAL |
cv2.WINDOW_KEEPRATIO)  # reacondicionamiento de ventana en linux
                cv2.resizeWindow(str(p), imm.shape[1], imm.shape[0])
            cv2.imshow(str(p), imm)
            cv2.waitKey(1)

        # guardado de imagenes con el respectivo resultado
        if gimg:
            if dataset.mode == 'image':
                cv2.imwrite(g_dir, imm)
            else:  # procesamiento de video
                if vid_path[i] != g_dir:  # video nuevo
                    vid_path[i] = g_dir
                    if isinstance(vid_writer[i], cv2.VideoWriter):
                        vid_writer[i].release()
                    if vid_cap:  # captura de video
                        fps = vid_cap.get(cv2.CAP_PROP_FPS)
                        w = int(vid_cap.get(cv2.CAP_PROP_FRAME_WIDTH))
                        h = int(vid_cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
                    else:  # fromato de video de salida
                        fps, w, h = 30, imm.shape[1], imm.shape[0]
                    g_dir = str(Path(g_dir).with_suffix('.mp4'))  # graba el
nuevo video en formato MP4
                    vid_writer[i] = cv2.VideoWriter(g_dir,
cv2.VideoWriter_fourcc(*'mp4v'), fps, (w, h))
                vid_writer[i].write(imm)

    # tiempo que tomo prosesar la inferencia
    LOGGER.info(f"{s}{'' if len(det) else '(no detections), '}{dt[1].dt *
1E3:.1f}ms")

# muestra de resultados
t = tuple(x.t / ver * 1E3 for x in dt)  # rapidez de prosesamiento de imagenes
LOGGER.info(f'Speed: %.1fms pre-process, %.1fms inference, %.1fms NMS per
image at shape {(1, 3, *imgsz)}' % t)
if g_tx or gimg:
    s = f"\n{len(list(s_dirct.glob('etqs/*.txt')))} etqs saved to {s_dirct /
'etqs'}" if g_tx else ''
    LOGGER.info(f"Results saved to {colorstr('bold', s_dirct)}{s}")
if update:
    strip_optimizer(weights[0])  #actualizacion de pesos
```