UNIVERSIDAD RICARDO PALMA

FACULTAD DE INGENIERÍA

ESCUELA PROFESIONAL DE INGENIERÍA ELECTRÓNICA

Implementación de una red neuronal artificial convolucional para la detección y conteo de vehículos en una sección del estacionamiento de la Universidad Ricardo Palma

TESIS

Para optar el título profesional de Ingeniero Electrónico

AUTORES

Gonzales del Valle Romero, Gian Franco
ORCID: 0000-0001-6532-7088

Neyra Espinoza, Walter Jesus
ORCID: 0000-0002-3328-2855

ASESOR

Huamaní Navarrete, Pedro Freddy
ORCID: 0000-0002-3753-9777

Lima, Perú

2022

Metadatos Complementarios

Datos del autor(es)

Gonzales del Valle Romero, Gian Franco
DNI: 73242037
Neyra Espinoza, Walter Jesus
DNI: 72501192

Datos de asesor

Huamaní Navarrete, Pedro Freddy
DNI: 10032682

Datos del jurado

JURADO 1
González Prado, Julio Cesar
DNI: 07702235
ORCID: 0000-0003-0384-7015

JURADO 2
López Córdova, Jorge Luis
DNI: 09638009
ORCID: 0000-0002-3817-6859

JURADO 3
Sánchez Bravo, Miguel Angel
DNI: 08443357
ORCID: 0000-0001-9384-1391

Datos de la investigación

Campo del conocimiento OCDE: 2.02.01
Código del Programa: 712026

DEDICATORIA

A Dios, a mis padres Emma y Walter, a mi hermana Angélica y a mis familiares que me brindaron el apoyo y los consejos necesarios para superar y sobreponerme en mis momentos más difíciles de mi vida y los de mi etapa universitaria, de los cuales estoy profundamente agradecido.

Walter Jesus Neyra Espinoza

A Dios, a mis padres Rossana y Miguel, quienes me apoyaron a seguir adelante en mis estudios y darme la oportunidad de forjarme un futuro como profesional; a mi hermano Angello y personas muy cercanas quienes siempre creyeron en mí y de las cuales siempre estaré agradecido por todo su apoyo.

Gian Franco Gonzales del Valle Romero

AGRADECIMIENTO

A nuestra alma máter la Universidad Ricardo Palma por brindarnos los conocimientos necesarios para nuestro desarrollo profesional, a nuestro asesor el ingeniero Huamaní y a nuestra profesora la ingeniera Terukina por su apoyo durante nuestros años en la universidad y a nuestros amigos y compañeros Leonardo, Bill y Pedro, por todos los buenos momentos que pasamos en la universidad.

Walter Jesus Neyra Espinoza
Gian Franco Gonzales del Valle Romero

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

En esta tesis se implementó un sistema de detección y conteo de vehículos en una sección del estacionamiento de la Universidad Ricardo Palma, con el objetivo de obtener el número de espacios disponibles y reducir el tiempo de consulta de espacios para el personal de seguridad; para esto, se propuso dos redes neuronales convolucionales (CNN): You Only Look Once v5 (YOLO v5) y Faster Region-based Convolutional Neural Network (Faster RCNN). Además, se optó por aplicar Transfer Learning a dichas CNN. Para ello se construyó un dataset de 460 imágenes conformado por imágenes de internet y autos estacionados en la universidad. Asimismo, para la operación del conteo de autos se utilizaron los métodos de limitación de espacios, por línea de interés (LOI) y región de interés (ROI), optando por el segundo método debido a ser el más adecuado para delimitar espacios de estacionamiento y evitar resultados erróneos al contabilizar.

Para la interfaz gráfica de usuario (GUI), se utilizó la librería Tkinter, creando una ventana interactiva, visualizando numéricamente los autos detectados y espacios disponibles. Finalmente, para la implementación del sistema en tiempo real, se eligió la red YOLO, después de evaluar ciertos parámetros (mAP, FPS, porcentaje de error y falsos positivos) en ambas CNN, modificándose los umbrales de detección y las ROI a través de una tabla comparativa. Asimismo, se instaló un teléfono móvil sobre la garita de la puerta 3 de la Universidad Ricardo Palma, utilizando la red de internet de la propia universidad, el cual se conectó a la laptop con la red YOLO, concluyendo la efectividad de la red seleccionada, la implementación del contador y la interfaz GUI.

Palabras claves: YOLO v5, Faster RCNN, línea de interés, región de interés, interfaz gráfica de usuario, detección y conteo de vehículos.

ABSTRACT

In this thesis, a vehicle detection and counting system was implemented in a section of the Ricardo Palma University parking lot, with the aim of obtaining the number of available spaces and reducing the space query time for security personnel; For this, two convolutional neural networks (CNN) were proposed: You Only Look Once v5 (YOLO v5) and Faster Region-based Convolutional Neural Network (Faster RCNN). In addition, it was decided to apply Transfer Learning to these CNNs. For this, a dataset of 460 images was built, made up of images from the internet and cars parked at the university. Likewise, for the car counting operation, the space limitation methods were used, by line of interest (LOI) and region of interest (ROI), opting for the second method due to being the most adequate to delimit parking spaces and avoid erroneous results when posting. For the graphical user interface (GUI), the Tkinter library was used, creating an interactive window, numerically displaying the detected cars and available spaces. Finally, for the implementation of the real-time system, the YOLO network was chosen, after evaluating certain parameters (mAP, FPS, error rate and false positives) in both CNNs, modifying the detection thresholds and ROIs through a comparison table. Likewise, a mobile phone was installed on the checkpoint of gate 3 of the Ricardo Palma University, using the internet network of the university itself, which was connected to the laptop with the YOLO network, concluding the effectiveness of the selected network, the implementation of the counter and the GUI interface.

Keywords: YOLO v5, Faster RCNN, line of interest, region of interest, graphical user interface, detection and counting of vehicles.

INTRODUCCIÓN

El motivo principal de la investigación es conocer cómo las redes neuronales convolucionales repercuten en un sistema de detección y conteo de vehículos en función a las necesidades que la presente investigación intenta resolver. Así mismo los resultados obtenidos de la investigación sirven de base para incentivar en mejorar la implementación de un sistema de detección y conteo de vehículos utilizando redes neuronales convolucionales en la Universidad Ricardo Palma.

Actualmente, los estudiantes, personal administrativo y docentes esperan mucho tiempo para ingresar con su vehículo por la puerta de entrada N° 3 de la Universidad Ricardo Palma; además, no se tiene un buen control de los espacios disponibles del estacionamiento ocasionando en algunas oportunidades que el vigilante proceda a buscar espacios libres y abandone la garita, lo cual podría causar un retraso de tiempo aproximado de 10 minutos en el momento del ingreso.

Por lo tanto, para el desarrollo de la presente investigación se consideraron como base investigaciones a nivel de tesis y artículos: Detección y conteo de personas en espacios cerrados utilizando estrategias basadas en visión artificial, así mismo Comparison of Yolo, SSD, Faster RCNN for Real Time Tennis Ball Tracking for Action Decision Networks, así como la de Reconocimiento automático de placas de rodaje utilizando una red neuronal convolucional para el ingreso de vehículos en la Universidad Ricardo Palma, entre otras más.

De esta manera, la presente investigación se estructura primero en el planteamiento del problema relacionado a la implementación de una red neuronal para la detección y conteo de vehículos, los objetivos que se pretende lograr para solucionar el problema, así como su justificación y limitaciones. Así mismo, en el segundo capítulo titulado marco teórico, se abarcan todas las investigaciones que se utilizan para desarrollar el proyecto. Luego, el desarrollo del proyecto y todos los pasos que se realizaron para implementarlo, ubicado en el capítulo tres; finalizando con las pruebas y resultados obtenidos que se desarrollaron en el capítulo cuatro.

Para finalizar la investigación se redactan las conclusiones, recomendaciones y se describe la referencia bibliográfica utilizada.

CAPÍTULO I: PLANTEAMIENTO DEL PROBLEMA

1.1. Descripción y formulación del problema general y específicos

Actualmente, los estudiantes, personal administrativo y docentes esperan mucho tiempo para ingresar con su vehículo por la puerta de entrada N° 3 de la Universidad Ricardo Palma; además, no se tiene un buen control de los espacios disponibles del estacionamiento ocasionando en algunas oportunidades que el vigilante proceda a buscar espacios libres y abandone la garita, lo cual podría causar un retraso de tiempo aproximado de 10 minutos en el momento del ingreso.

1.1.1. Problema General

¿Cómo implementar una red neuronal artificial convolucional para la detección y conteo de vehículos en una sección del estacionamiento de la Universidad Ricardo Palma?

1.1.2. Problemas Específicos

a) ¿Cómo aplicar transfer learning a dos arquitecturas de redes neuronales artificiales convolucionales pre-entrenadas, y seleccionar aquella que obtenga el mejor rendimiento para la detección y conteo de vehículos en una sección del estacionamiento de la Universidad Ricardo Palma?
b) ¿Qué estrategia de conteo de vehículos utilizar en una sección del estacionamiento de la Universidad Ricardo Palma?
c) ¿Cómo implementar una interfaz gráfica de usuario (GUI), para la visualización del conteo de vehículos en una sección del estacionamiento de la Universidad Ricardo Palma?

1.2. Objetivo general y específicos

1.2.1. Objetivo General

Implementar una red neuronal artificial convolucional para la detección y conteo de vehículos en una sección del estacionamiento de la Universidad Ricardo Palma.

1.2.2. Objetivos Específicos

a) Aplicar transfer learning a dos redes neuronales artificiales convolucionales pre-entrenadas, y seleccionar la red que obtenga el mejor rendimiento para la detección y conteo de vehículos en una sección del estacionamiento de la Universidad Ricardo Palma.
b) Aplicar la mejor estrategia de conteo de vehículos en una sección del estacionamiento de la Universidad Ricardo Palma.
c) Implementar una interfaz gráfica de usuario (GUI) para visualizar la detección y conteo de vehículos en una sección del estacionamiento de la Universidad Ricardo Palma.

1.3. Delimitación de la investigación

1.3.1. Temporal

Este trabajo se limita a las grabaciones realizadas con la cámara de vídeo en el periodo de tiempo comprendido entre mayo del 2022 y noviembre del 2022.

1.3.2. Espacial

Este proyecto se limita a una sección del estacionamiento que será captada por el campo de visión de la cámara, que se instalará sobre la garita de la puerta N° 3 de la Universidad Ricardo Palma, otra limitante será el uso exclusivo de la cámara en el día y sin lluvia. Una limitante adicional son las 2 estrategias de conteo (línea de interés o región de interés) que limitaron el uso de la red neuronal artificial dentro del área de acción de la cámara de video.

1.3.3. Temática

Este estudio tiene como delimitación teórica la bibliografía relacionada con las redes neuronales artificiales convolucionales.

1.4. Justificación e importancia

1.4.1. Justificación

Este proyecto se justifica por la necesidad que existe de controlar los espacios disponibles del estacionamiento de la Universidad Ricardo Palma para evitar la congestión vehicular.

1.4.2. Importancia

Este proyecto es importante porque se brindó información, en tiempo real, de los espacios libres del estacionamiento al vigilante de la puerta N° 3 de la Universidad Ricardo Palma mediante una aplicación puesto dentro de su computadora, disminuyendo el tiempo que tardan tanto los estudiantes, como los docentes y el personal administrativo en recibir la notificación del número de estacionamientos libres; a la vez, se logró reducir el tiempo de espera para ingresar al estacionamiento, disminuyendo y/o evitando la congestión vehicular.

CAPÍTULO II: MARCO TEÓRICO

2.1. Antecedentes del estudio de investigación

2.1.1. Antecedentes nacionales

Chirinos, X. y Calero, P. (2021) en su tesis titulada Detección del uso correcto de mascarillas utilizando una red neuronal convolucional para el ingreso de personas a un laboratorio de una universidad tiene como objetivo detectar el uso correcto de la mascarilla utilizando una red neuronal convolucional al ingreso de un laboratorio el cual se concluyó que se logró comparar y validar el funcionamiento de las redes neuronales y se consiguió clasificar el uso correcto de la mascarilla.

Este proyecto es importante para este estudio debido a que se utilizó el reescalado de imágenes para evitar el uso exhaustivo del rendimiento del hardware para el entrenamiento de la red neuronal.

Ramírez, B. y Tito, M. (2020) en su tesis titulada Reconocimiento automático de placas de rodaje utilizando una red neuronal convolucional para el ingreso de vehículos en la Universidad Ricardo Palma para el título de Ingeniero Electrónico en la Universidad Ricardo Palma en Lima Perú, concluyeron que se logró implementar el sistema de detección de placas de rodaje utilizando los Toolbox del software Matlab implementaron tres modelos de redes neuronales convolucionales y una cámara web logró un resultado del 95% de asertividad de reconocimiento en todas las placas de rodaje tomadas.

El aporte que tiene esta investigación con el presente proyecto es el uso aplicativo de una interfaz gráfica para la visualización de sus resultados obtenidos por la red neuronal artificial escogida.

Cayllahua, N y Suárez, J. (2019) en su tesis titulada Redes neuronales de aprendizaje profundo para el reconocimiento facial y control de acceso de estudiantes a un laboratorio para optar el título de ingeniero electrónico en la Universidad Ricardo Palma, desarrollaron un sistema de control de ingreso de alumnos al laboratorio de control, mediante el entrenamiento de una red neuronal convolucional con fotografías de los estudiantes, logrando obtener un óptimo reconocimiento, pero solo a los estudiantes cuyas fotografías

captadas tengan el mismo fondo que las imágenes con las que se entrenó la red neuronal.

De esta tesis se rescata la importancia de, no solo una abundante cantidad de imágenes para el entrenamiento de la red, sino de la variedad de las mismas (distintos colores, tamaños, fondo, etc.) para evitar el "sobreajuste" (overfitting) de la red.

Narciso, W. y Manzano, E. (2021) en su artículo titulado "Sistema de visión artificial basado en redes neuronales convolucionales para la selección de arándanos según estándares de exportación", tiene como objetivo la automatización de la selección de arándanos que presenten signos de deterioro o de putrefacción, utilizando la red neuronal convolucional Faster RCNN.

De esta investigación se rescata el uso de una porción de la base de datos de prueba (test), y que no se usó durante el entrenamiento, para comprobar la precisión de la red neuronal y su capacidad de generalizar objetos.

Rios, P. (2022) en su tesis titulada Diseño y entrenamiento de una red neuronal empleando procesamiento de imágenes para diferenciar granos de trigo respecto a malezas, que tuvo objetivo el desarrollo de una red neuronal artificial convolucional para la detección e identificación de granos de trigo con una base de datos de 720 imágenes de 4 tipos de granos distintos, logró obtener una eficacia del 97.56% y una tasa de falsos positivos del 0.83% a través de una distribución de la base de datos de imágenes con un grupo para el entrenamiento, otro para la validación y un tercero para la prueba de la red neuronal.

De esta investigación se rescata la importancia de la distribución de toda la base de datos en subgrupos de entrenamiento, validación y prueba.

2.1.2. Antecedentes internacionales

Jockela, J. (2018), en su tesis titulada *"Person counter using real time object detection and a small neuronal network"* para obtener el grado de bachiller en la carrera de tecnologías de la información y comunicaciones en la Universidad de ciencias aplicadas de Turku, Finlandia; tiene como objetivo

la búsqueda e implementación de una red neuronal artificial convolucional para el conteo de las personas y si sus requisitos acuerdan con los de un hardware asequible. El cual concluye que implementó con éxito dicha CNN, sin embargo, el hardware utilizado no tuvo una eficacia óptima y sugiere utilizar una GPU NVIDIA para mejorar el rendimiento.
El aporte de la investigación hacia este proyecto es que se debe tener una GPU, de preferencia de la marca NVIDIA, para obtener un mejor rendimiento, caso contrario se podría obtener falsos negativos debido a una baja tasa de FPS. Además, de esta tesis se deduce que se debe aplicar una GPU dedicada para mejorar el rendimiento de entrenamiento y velocidad de la CNN en aplicación.

Sarda, Dix & Bhan (2021) en su investigación titulada "*Object Detection for Autonomous Driving using YOLO [You Only Look Once] algorithm*", utiliza la red neuronal YOLO V4 para reconocer objetos en tiempo real para la conducción autónoma aplicando una cámara, ellos concluyeron que su sistema detecta los objetos registrados, pero estos tienen un margen de error mínimo el cual dicen que para el caso de la detección autónoma puede ser fatal.
De esta investigación se rescata el uso del reentrenamiento de la red neuronal en base a la personalización del entrenamiento.

Wu, T., Wang, T. y Liu,Y. (2021) en su artículo titulado "*Real-Time Vehicle and Distance Detection Based on Improved Yolo v5 Network*" el cual tiene como objetivo proponer un sistema de deteccion de vehiculos y distancia basado en una red neuronal Yolo v5 que logre detectar los vehículos en un ambiente simulado, concluye que por el uso del GPU su sistema mejora la tasa de cuadros a 47 FPS.
Esta investigación es importante debido a que resalta las capas de aprendizaje en una red neuronal convolucional y el uso de métricas para verificar el rendimiento de la red.

Deepa, R., Tamiselvan, E., Abrar, E. & Shrinivas, S. (2016) en su proyecto titulado *"Comparison of Yolo, SSD, Faster RCNN for Real Time Tennis Ball*

Tracking for Action Decision Networks" el cual se comparan tres redes neuronales artificiales convolucionales con el objetivo de obtener el mejor rendimiento para el seguimiento de pelotas de tenis en movimiento en base a la métrica de pérdida del objeto en el tiempo, tuvo como resultado que la mejor red para sus requerimientos es SSD debido a que su función de Loss decrece más rápida que las demás teniendo como resultado 20000 pasos.
Este artículo brinda a la presente tesis información de resultados de YOLO y Faster RCNN en la aplicación de seguimiento de objetos, y de comparación de resultados en base a la comparación de las funciones loss.

Tan, L., Huangfu, T., Wu, L. y Chen, W. (2021) en su artículo titulado *"Comparison of YOLO v3, Faster R-CNN, and SSD for Real-Time Pill Identification"* tienen como objetivo la comparación de estas tres redes neuronales para la detección de pastillas en el área médica farmacéutica. Se realizan las comparaciones con 5131 imágenes de las cuales se dividen en 216 tipos de medicamentos entre cápsulas y pastillas. En sus resultados tuvieron como métricas de comparación la media de precisión promedio (MAP) y la tasa de cuadros por segundo (FPS) el cual tienen como resultado que Faster RCNN tiene la mejor tasa de MAP, pero tiene la menor tasa de FPS, YOLO tiene una menor tasa de MAP, pero tiene la mayor tasa de FPS y SSD tiene un balance entre MAP y FPS. Concluyeron que la red más comprometida para la aplicación es YOLO.
Esta investigación brinda una información general de las detecciones en el área médica de las redes neuronales propuestas y aporta a la tesis información de comparaciones entre redes neuronales para la detección de objetos en tiempo real.

2.2. Bases teóricas vinculadas a las variables de estudio

2.2.1. Red neuronal artificial convolucional

Para Gelvez (2019) las redes neuronales convolucionales se definen como:

> Las redes neuronales convolucionales (CNNs, por sus siglas en inglés) son un tipo de redes neuronales artificiales que son utilizadas, entre otras cosas, para el análisis y reconocimiento de imágenes. (…) Lo que

resulta fundamental al querer tratar con el aprendizaje de patrones como los de una imagen. Dentro de este tipo de NNs, las capas convolucionales son la parte más importante de toda la red. Estás capas como lo menciona Khan et al. (como se citó en Gelvez, 2019) son un grupo de filtros (…) que son convolucionados con una cierta entrada (o *input*) para generar un mapa de características en una salida (u *output*). Las CNNs toman un dato, por ejemplo, una imagen, y aplican los filtros, no a toda la imagen, sino a una parte de esta, luego hacen lo mismo en otro sector, y así sucesivamente hasta obtener algo más fácil de procesar. Las primeras capas de estas NNs se encargan de identificar los patrones más básicos, como puede ser una línea recta vertical, o cierta curva, y a medida que cada capa pasa su output a la siguiente, se van identificando patrones cada vez más complejos a partir de los anteriores y así hasta llegar a identificar cosas más abstractas, como puede ser un gato en una imagen. (p. 2)

2.2.2. Arquitectura de la red neuronal artificial convolucional

La arquitectura de una red neuronal se define como la estructura de cómo se conforma una red neuronal convolucional. MathWorks (2022) define la estructura de la siguiente manera:

> (…) está compuesta por una capa de entrada, una capa de salida y muchas capas intermedias ocultas. Estas capas realizan operaciones que alteran los datos con el objetivo de aprender características específicas de dichos datos. Las 3 capas más frecuentes son: convolución, activación o *ReLU*, y *pooling*.
>
> - Convolución: Somete las imágenes de entrada a un conjunto de filtros convolucionales, cada uno de los cuales activa ciertas características de las imágenes.
> - Unidad lineal rectificada (ReLU): Permite un entrenamiento más rápido y eficaz al asignar los valores negativos a cero y mantener los valores positivos. También se lo denomina activación, dado que solo las características activadas pasan a la siguiente capa.

- Pooling: Simplifica la salida al disminuir la tasa de muestreo no lineal, reduciendo así el número de parámetros que la red necesita aprender.

Estas operaciones se repiten en decenas o cientos de capas, de modo que cada capa aprende a identificar diferentes características. (www.la.mathworks.com)

2.2.3. Detección y conteo

MathWorks (2022) define la detección en redes neuronales artificiales convolucionales como:

> La detección de objetos y el reconocimiento de objetos son técnicas similares para identificar objetos, pero varían en cuanto a su ejecución. La detección de objetos es el proceso de localizar objetos presentes en imágenes. En el caso de *deep learning*, la detección de objetos forma parte del reconocimiento de objetos, que no solo identifica el objeto, sino que lo localiza en una imagen. Esto permite identificar y localizar varios objetos en la misma imagen. (www.la.mathworks.com)

2.2.4. Estrategias de conteo

Luna, A. y Rodríguez, N. (2017) definen las estrategias de conteo de la siguiente manera:

> (…) actualmente las estrategias de conteo más populares se encuentran divididas en dos grupos:
>
> - LOI (Línea de interés): Estrategia de conteo basada en el número de personas en movimiento que atraviesan cierta línea de la escena definida por el usuario. El procedimiento común para contar las personas requiere que se detecten las personas y que se haga un seguimiento de cada una de ellas hasta el momento que cruzan la línea trazada por el usuario.
> - ROI (Región de interés): Estrategia de conteo basada en el número de personas estáticas y en movimiento que se encuentran en determinada zona de una escena. A diferencia del método anterior, esta estrategia requiere la detección de las personas, más no su seguimiento. (p. 2)

2.3. Definición de términos básicos

- Deep Learning: En español aprendizaje automático. “Es un conjunto de algoritmos de aprendizaje automático que intenta modelar abstracciones de alto nivel en datos usando arquitecturas computacionales que admiten transformaciones no lineales múltiples e iterativas de datos expresados en forma matricial o tensorial” (Bengio, Coruville & Vincent, 2014, p. 1).
- Convolución: En las redes neuronales convolucionales, consiste en la operación matemática de un grupo de píxeles de la imagen de entrada con una matriz más pequeña.
- Transfer learning: En español transferencia del aprendizaje. Es un método empleado en el campo de Deep Learning, en la que se utiliza una red neuronal capaz de resolver una gran cantidad de problemas gracias a una vasta base de datos, para que solucione un problema específico utilizando el re-entrenamiento.
- YOLOv5: Sistema conformado por una red neuronal artificial convolucional para la detección de objetos en tiempo real. La descripción completa de YOLOv5 se ubica en el apartado 3.3.1 de la presente tesis.
- Faster RCNN: Red neuronal artificial convolucional profunda aplicada a la detección de objetos, el cual posee como característica una precisión acertada. La descripción completa de Faster RCNN se ubica en el apartado 3.4.1 de la presente tesis.
- mAP: En español media de precisión promedio. Es la métrica más común que se utiliza para determinar qué tan bien una red neuronal puede localizar o clasificar objetos y se define como la media de todas las precisiones promedio para cada clase (en el caso del presente proyecto solo es una clase) que indica qué tan buena es la red neuronal para detectar verdaderos positivos frente a un conjunto de verdaderos y falsos positivos.
- FPS: En español cuadros o imágenes por segundos. Indica la cantidad de imágenes sucesivas que se muestran en pantalla en un segundo. En el presente proyecto, se utilizó para determinar la velocidad con la que el algoritmo (red neuronal + método de conteo) es capaz de realizar el procesamiento de los cuadros, así como las detecciones y mostrarlas en la interfaz gráfica de usuario.

CAPÍTULO III: DESARROLLO DEL PROYECTO

En este capítulo se redactó el proceso de aplicación de las redes neuronales artificiales convolucionales, el cual abarcó el software y hardware utilizado, la construcción de la base de datos, el reentrenamiento de las redes neuronales propuestas, implementación del sistema de conteo y de la interfaz gráfica. Dicho proceso se puede apreciar en la figura 1.

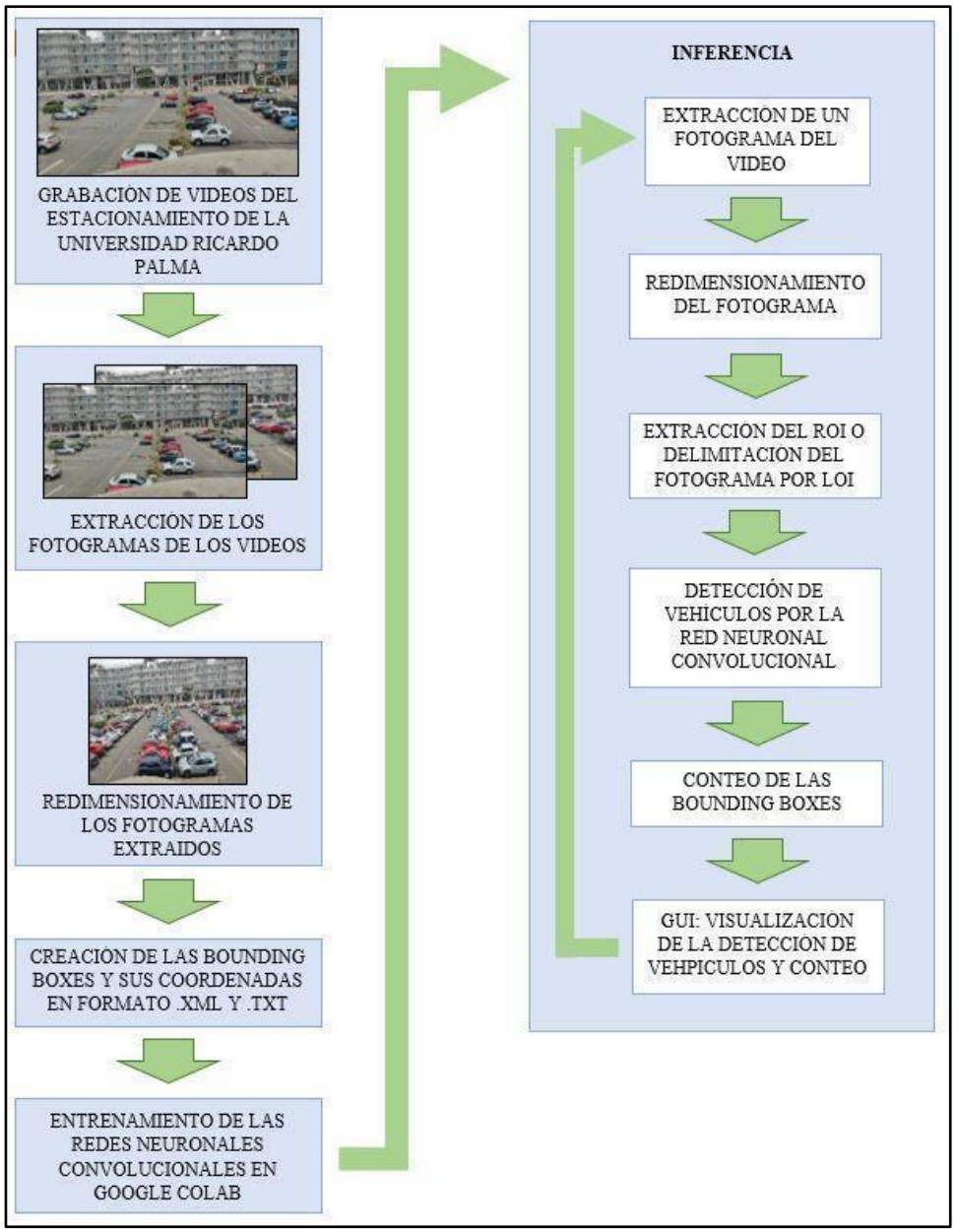

Figura N 1: Diagrama de bloques del proyecto
Fuente: Elaboración propia

3.1. Hardware y software utilizado

Para la obtención de datos y pruebas se utilizaron los siguientes dispositivos de hardware:

Xiaomi Mi 11 Lite: Dispositivo de entrada utilizado para la recolección de imágenes por medio de la cámara de video y enlace entre la computadora por IP. Sus características se registran en la Tabla 1.

Lenovo Ideapad Gaming 3: Dispositivo encargado del esfuerzo computacional generado por las redes neuronales y visualización de los sistemas propuestos. Sus características más importantes se registran en la Tabla 2.

Para el entrenamiento y la inferencia se utilizaron los siguientes programas de software:

Anaconda: Software de distribución libre utilizado para la administración y despliegue de entornos virtuales y librerías de software en lenguaje Python.

Spyder: Software libre encargado de proveer un entorno de desarrollo integrado para la programación del algoritmo de conteo de vehículos utilizando redes neuronales convolucionales, en lenguaje Python.

Google Colab: Entorno interactivo de ejecución online gratuito de código de Python, usado para el entrenamiento de las redes neuronales convolucionales propuestas. Sus especificaciones técnicas se encuentran en la Tabla 3.

Make Sense: Entorno Web encargado de la creación de Labels de Bounding Boxes para data artificial y data real.

Tabla Nº 1

Xiaomi Mi 11 Lite - Especificaciones técnicas

Características	Detalle de características
Dimensiones y peso	160.53×75.73×6.81mm158 gramos
Pantalla	6,55" FHD+ AMOLED 90 Hz, HDR10 Gorilla Glass 5
Procesador	Qualcomm Snapdragon 778G
RAM	6GB / 8GB LPDDR4X
Capacidad	128GB / 256 GB UFS 2.2
Cámaras traseras	Principal: 64 MP (1/1,97", f/1.79) Ultra gran angular: 8 MP (f/2.2, 119°) Macro: 5 MP (f/2.4)
Cámara frontal	20 MP
Conectividad	WiFi 6, Bluetooth 5.2, NFC, GPS, Dual SIM 5G
Batería	4.250 mAh con carga rápida de 33W

Fuente: https://www.xatakandroid.com/analisis/xiaomi-11-lite-5g-ne-analisis-caracteristicas-precio-especificaciones

Tabla Nº 2

Lenovo Ideapad Gaming 3 - Especificaciones técnicas

Características	Detalle de características
Dimensiones y peso	359.6 x 251.9 x 24.2 mm 2.25 Kg
Pantalla	15,6" FHD (1920 x 1080), IPS, antirreflectante, sin capacidad táctil, 45 % NTSC, 250 nits, 120 Hz, retroiluminación LED, marco biselado fino
Procesador	Procesador Intel Core i5™ 5 10300H (2,50 GHz hasta 4.20 GHz)
Tarjeta gráfica	GPU para equipo portátil NVIDIA® GeForce® GTX™ 1650 4 GB GDDR5
RAM	8 GB DDR4-3200MHz (SODIMM)
Capacidad	512 GB SSD M.2 2242 PCIe TLC
Adaptador de corriente	170 W
Batería	Polímero de Litio, 3 celdas, 45 Wh
Conectividad	Wi-Fi 6 2x2 AX & Bluetooth® 5.0

Fuente: https://www.lenovo.com/pe/es/laptops/ideapad/serie-300/Gaming-3-Gen-6-15-AMD/p/WMD00000479?orgRef=https%253A%252F%252Fwww.google.com.pe%252

F&gclid=Cj0KCQjwxveXBhDDARIsAI0Q0x3qPOSbrVaPldDO-XMENOF7X69HrMQKVYPEJJy3N9QHvACWJ-Z-8pYaAkcyEALw_wcB&cid=pe:sem|se|google|lenovo_dsa_totalsitio|total|||9142907751|92029595029|dsa-867340452201|dsa|mixed|all&cid=co:sem|Lenovo_PageFeed_DSA_Tablets|google&s_kwcid=AL!4305!3!435004405840!!!g

Tabla Nº 3

Google Colab - Especificaciones técnicas

Características	Detalle de características
CPU	x2 Intel(R) Xeon(R) CPU @ 2.20GHz
Tarjeta gráfica	GPU NVIDIA® Tesla T4 16GB GDDR6 300 GB/sec
RAM	12.48 GB
Capacidad de disco	108 GB

Fuente: https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/tesla-t4/t4-tensor-core-datasheet-951643.pdf

3.2. Construcción del Dataset

3.2.1. Obtención de la data real

Para la obtención de la data real se realizó la captura de videos de una determinada sección del estacionamiento de la Universidad Ricardo Palma, dicho sector se visualiza en la figura 2, considerando los espacios del E1 al E56, por el cual el dispositivo de recolección fue el Xiaomi Mi 11 Lite el cual se configuró a una resolución de 1920x1080 píxeles para la toma de datos ubicado encima de la garita de seguridad ubicada a 7 metros del suelo, en la puerta número 3 del estacionamiento el cual puede ser visualizado en las figuras 3 y 5. Esta se encargó de la recolección de información en video de los automóviles estacionados.

El dispositivo grabó, por cada visita a la Universidad, un promedio de 40 minutos de video, del cual se extrajeron de 7 a 10 muestras por cada video mostrado y se redimensionó a una resolución de 640 x 640, teniendo un total de 100 imágenes y se guardaron en una carpeta como dataset para la creación de bounding boxes, tal y como se muestra en la figura 6.

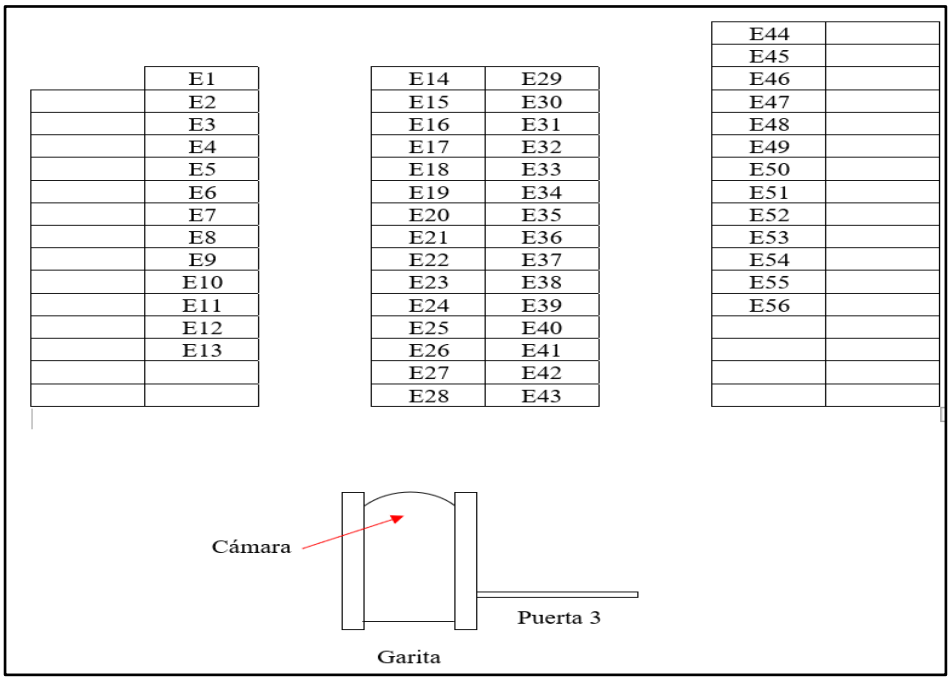

Figura N 2: Sección elegida del estacionamiento de la Universidad Ricardo Palma
Fuente: Elaboración propia

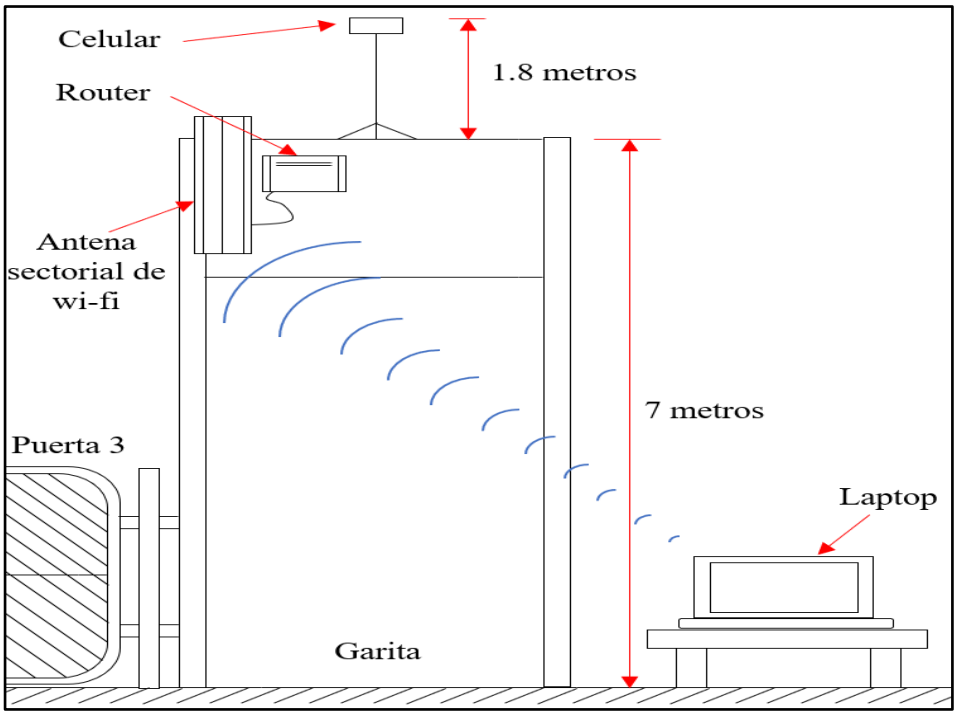

Figura N 3: Diagrama de instalación de cámara en la garita de la puerta 3 del estacionamiento de la Universidad Ricardo Palma
Fuente: Elaboración propia

Figura N 4: Garita de la puerta 3 del estacionamiento de la Universidad Ricardo Palma
Fuente: Elaboración propia

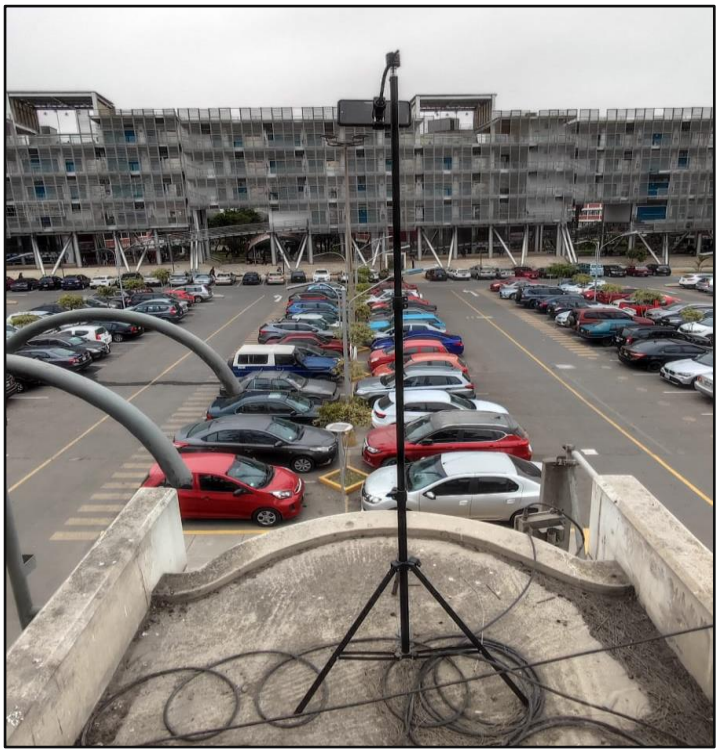

Figura N 5: Instalación de cámara en la garita de la puerta 3 del estacionamiento de la Universidad Ricardo Palma
Fuente: Elaboración propia

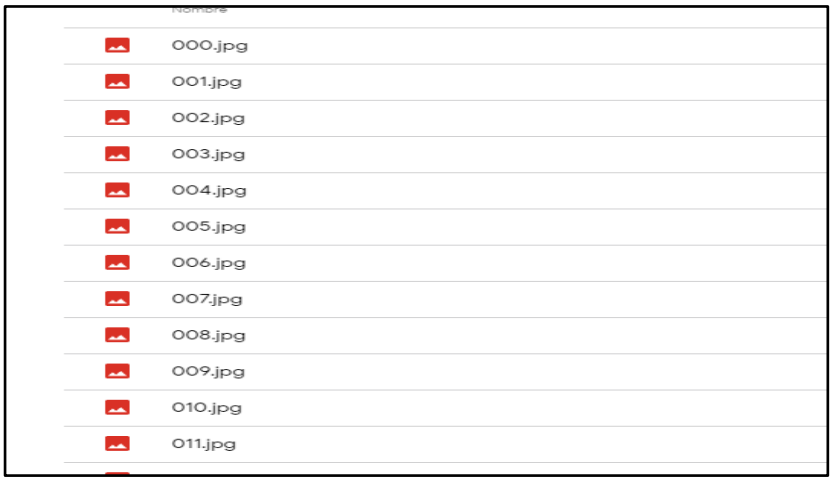

Figura N 6: Fotogramas recolectados de los videos de la sección propuesta del estacionamiento de la Universidad Ricardo Palma
Fuente: Elaboración propia

3.2.2. Obtención de la data artificial

Para el caso de la obtención de imágenes de autos del tipo artificial, se recolectó imágenes de la plataforma de búsqueda Google en internet por medio de la extensión Image Downloader de Chrome, recolectando un total de 360 imágenes el cuales se filtrarán las imágenes que no pertenezcan a automóviles, estas se visualizan en la figura 7.

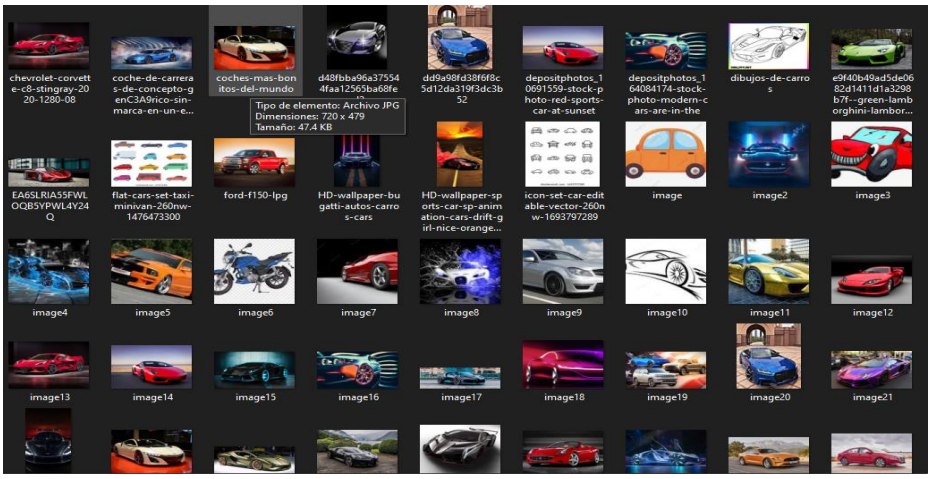

Figura N 7: Data artificial recolectada para entrenamiento de las redes neuronales artificiales convolucionales
Fuente: Elaboración propia

3.2.3. Creación de Labels a partir de Bounding boxes de la data real y artificial

La creación de Labels se realizó utilizando la plataforma web de Make Sense en el cual se suben las imágenes de autos obtenidas el cual se realizó bounding

boxes para cada imagen, teniendo como único Label el tipo carro como se puede visualizar en la figura 8.

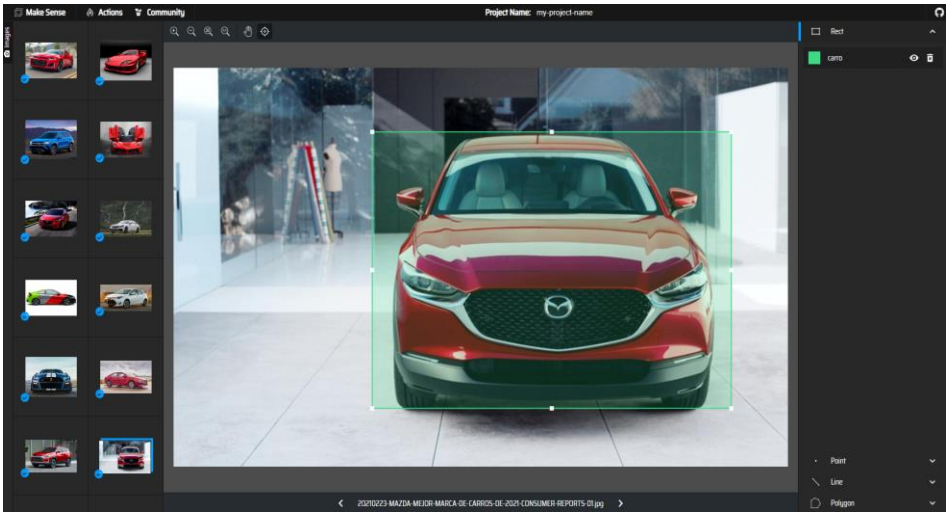

Figura N 8: Data artificial y real utilizada en la plataforma Make Sense para la creación de bounding boxes
Fuente: Elaboración propia

Después se descargaron la data de las bounding boxes tanto en formato .xml y .txt para Faster R-CNN y YOLO respectivamente como se aprecia en la figura 9, el cual será aplicado al entrenamiento de ambas redes neuronales artificiales convolucionales.

Finalmente se organizaron las imágenes y sus respectivos archivos en dos carpetas separadas: train, para el entrenamiento de la red neuronal convolucional; y val, para la validación del entrenamiento.

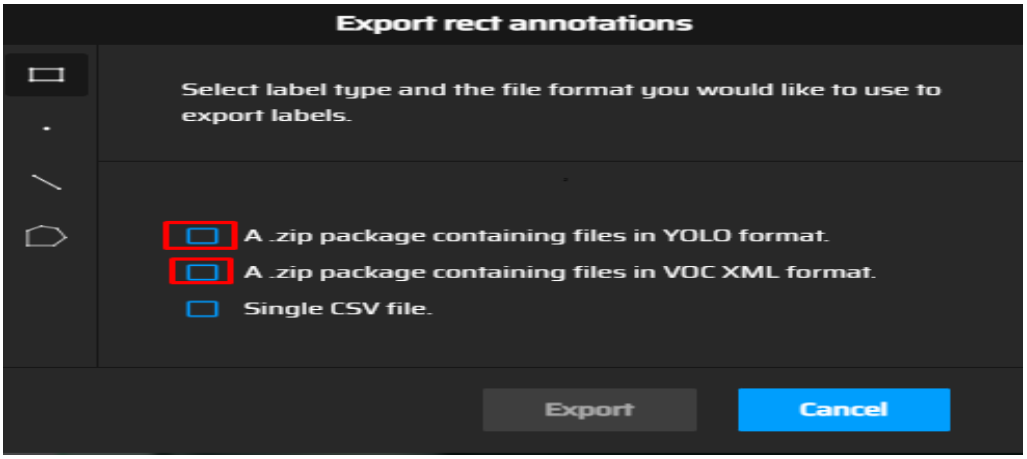

Figura N 9: Exportación de bounding boxes de la data artificial y real para el entrenamiento.
Fuente: Elaboración propia

3.3. Implementación de YOLO v5

3.3.1. Elección de variante de YOLO v5 en base a sus características

Cada variante de Yolo v5 se diferencia en la profundidad de neuronas de cada una desde el .n al .x. Se eligió la variante de YOLO v5 en base a las características brindadas por defecto el cual cada variante de YOLO v5 fue entrenada con un dataset de 1500 imágenes y 300 épocas de entrenamiento el cual obtiene las siguientes métricas de rendimiento y porcentaje de predicción. Los resultados de rendimiento de cada variante realizada por el desarrollador se visualizan en la tabla 4 y figura 10.

Tabla Nº 4

YOLO v5 variantes - Especificaciones técnicas

Model	size (pixels)	mAPval 0.5:0.95	mAPval 0.5	Speed CPU b1 (ms)	Speed V100 b1 (ms)	Speed V100 b32 (ms)	params (M)	FLOPs @640 (B)
YOLOv5n	640	28.0	45.7	45	6.3	0.6	1.9	4.5
YOLOv5s	640	37.4	56.8	98	6.4	0.9	7.2	16.5
YOLOv5m	640	45.4	64.1	224	8.2	1.7	21.2	49.0
YOLOv5l	640	49.0	67.3	430	10.1	2.7	46.5	109.1
YOLOv5x	640	50.7	68.9	766	12.1	4.8	86.7	205.7

Fuente: https://github.com/ultralytics/yolov5

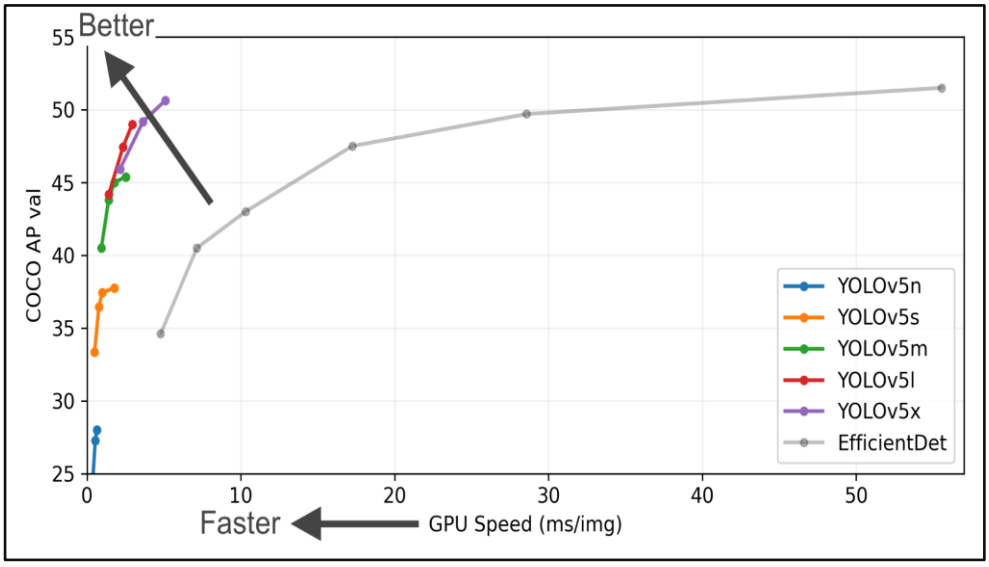

Figura N 10: Comparativa de precisión promedio y velocidad de variantes de YOLO v5
Fuente: https://github.com/ultralytics/yolov5

Se escogió la variante YOLO v5x debido a que posee el mejor porcentaje de predicción debido a la profundidad de cada capa, teniendo como media promedio de precisión en 0.5 (mAp) de 68.9% y la diferencia de velocidad es de milisegundos el cual no afecta al proyecto.

3.3.2. Transfer Learning a la red neuronal aplicando congelamiento de capas

Yolo v5 tiene una arquitectura general conformada por una troncal (Backbone), un cuello (Neck) y cabeza (Head) teniendo partes de CSPDarknet, PANet y YOLO respectivamente como se puede observar en la Figura 11, en la troncal se tienen las capas de cuello de botella CSP y de Pooling espacial piramidal, en el cuello se tienen las capas de convolución 1x1 y 3x3 y funciones de concatenación, por último, la cabeza posee las capas de salida de convolución 1x1.

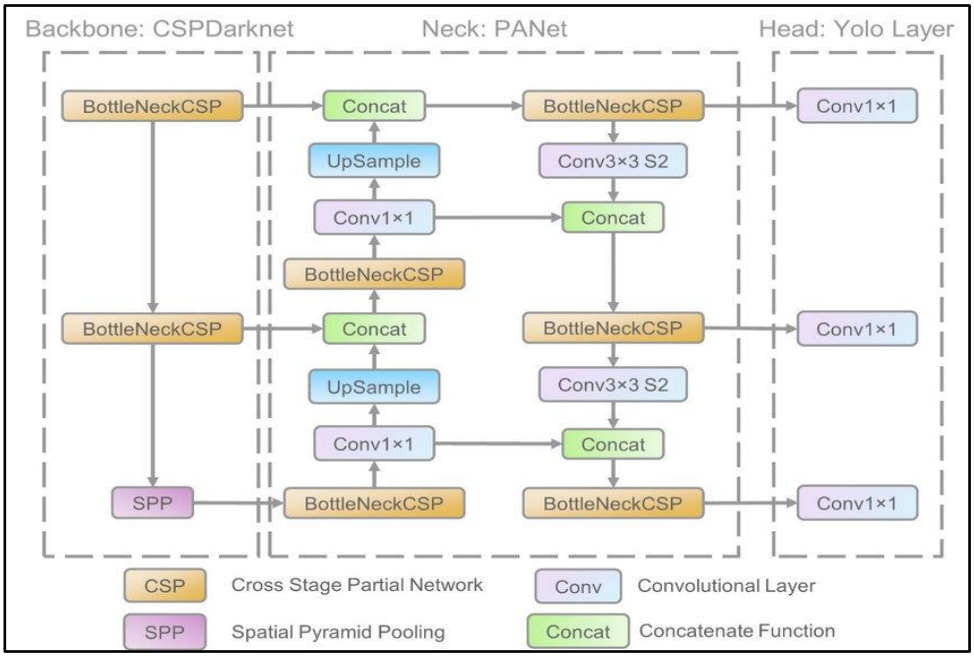

Figura N 11: Arquitectura de red neuronal artificial convolucional YOLO v5
Fuente: https://www.researchgate.net/figure/The-network-architecture-of-Yolov5-It-consists-of-three-parts-1-Backbone-CSPDarknet_fig1_349299852

Para el Transfer Learning, se aplicó el congelamiento de las primeras 20 capas dejando las últimas capas de convolución de la cabeza a reentrenar, teniendo como configuración de etiqueta la categoría de carro. Este re-entrenamiento se realiza en la plataforma de Google Colab debido a que esta plataforma en la nube posee un mejor rendimiento al entrenar redes neuronales.

Finalmente, se realiza el reentrenamiento de las últimas 4 capas utilizando las imágenes recolectadas y las bounding boxes obtenidas en formato .txt utilizando la configuración de 150 épocas y de formato de 640 pixeles con un número de muestras procesadas (batch size) de 16, aplicando los pesos pre entrenados de YOLO v5x. El proceso y resultados del entrenamiento se visualizan en las figuras 12 y 13.

```
     Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
   146/149     7.62G    0.04537    0.05197          0        146       640: 100% 29/29 [00:13<00:00,  2.18it/s]
                Class     Images  Instances          P          R     mAP@.5 mAP@.5:.95: 100% 2/2 [00:01<00:00,  1.71it/s]
                  all         40        317      0.838      0.733      0.817      0.429

     Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
   147/149     7.62G    0.04358    0.05046          0        138       640: 100% 29/29 [00:13<00:00,  2.19it/s]
                Class     Images  Instances          P          R     mAP@.5 mAP@.5:.95: 100% 2/2 [00:01<00:00,  1.74it/s]
                  all         40        317      0.844      0.735      0.817      0.428

     Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
   148/149     7.62G    0.04523    0.05114          0        192       640: 100% 29/29 [00:13<00:00,  2.21it/s]
                Class     Images  Instances          P          R     mAP@.5 mAP@.5:.95: 100% 2/2 [00:01<00:00,  1.71it/s]
                  all         40        317      0.866      0.736       0.82      0.427

     Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
   149/149     7.62G       0.044    0.04959          0        214       640: 100% 29/29 [00:13<00:00,  2.22it/s]
                Class     Images  Instances          P          R     mAP@.5 mAP@.5:.95: 100% 2/2 [00:01<00:00,  1.73it/s]
                  all         40        317      0.862      0.732      0.819      0.428

150 epochs completed in 0.689 hours.
Optimizer stripped from runs/train/exp/weights/last.pt, 173.2MB
Optimizer stripped from runs/train/exp/weights/best.pt, 173.2MB

Validating runs/train/exp/weights/best.pt...
Fusing layers...
Model summary: 444 layers, 86173414 parameters, 0 gradients, 203.8 GFLOPs
                Class     Images  Instances          P          R     mAP@.5 mAP@.5:.95: 100% 2/2 [00:01<00:00,  1.17it/s]
                  all         40        317      0.841      0.717      0.814      0.439
Results saved to runs/train/exp
```

Figura N 12: Entrenamiento por congelamiento de 20 capas en YOLO v5
Fuente: Elaboración propia

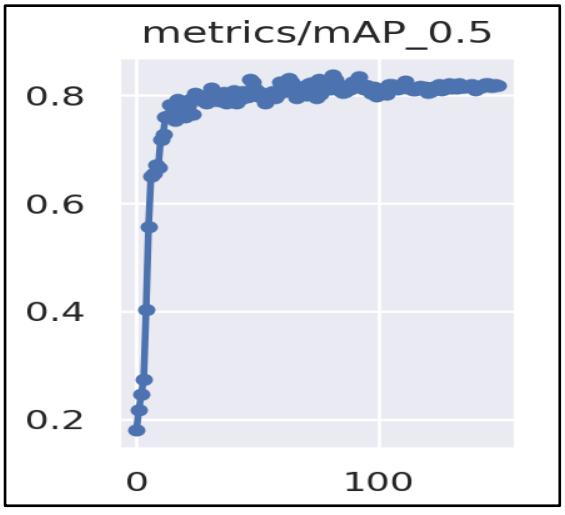

Figura N 13: Métrica mAP vs época en YOLO v5
Fuente: Elaboración propia

3.3.3. Implementación del sistema de conteo por ROI y LOI

Para el sistema de conteo de vehículos, se utilizaron dos métodos: líneas de interés (LOI) y regiones de interés (ROI). Estos permiten delimitar el área de acción de la red neuronal convolucional para detectar solo los vehículos contenidos entre líneas de interés o dentro de regiones de interés.

Se utilizó la biblioteca de visión computacional y procesamiento de imágenes OpenCV, para Python, para el redimensionamiento de los fotogramas utilizando el método de interpolación, para dibujar las líneas delimitadoras LOI y los polígonos que conforman las ROI.

Para las LOI se utilizó las función de *line()* cuyos argumentos utilizados fueron los siguientes:

cv2.line (ImgInOut, Pt1, Pt2, color, thickness)

Donde:

ImgInOut: Es la imagen donde se dibuja la línea.

Pt1: Coordenada del punto de origen.

Pt2: Coordenada del punto final.

Color: Color de la línea dibujada.

Thickness: Grosor de la línea dibujada.

Esta función dibuja una línea comprendida entre los puntos Pt1 y Pt2 basándose en el sistema de coordenadas de imágenes mostrado en la figura 14, donde la coordenada (0,0) le corresponde a la esquina superior izquierda de la imagen.

Se dibujaron un total de 6 líneas para delimitar los espacios de estacionamiento, teniendo como referencia la figura N° 2, en fotogramas de resolución de 640x640, tal y como se muestra en la figura N° 13, con la finalidad de que la red cuente solo los vehículos que traspasen las líneas determinadas.

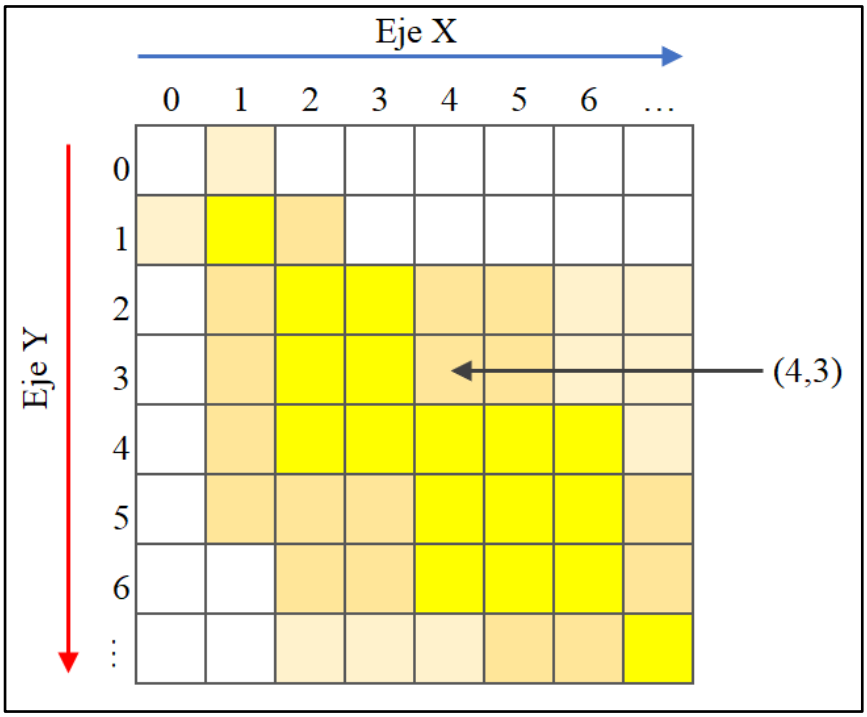

Figura N 14: Sistema de coordenadas de imágenes del OpenCV
Fuente: Elaboración propia

Figura N 15: Delimitación de fotograma por LOI
Fuente: Elaboración propia

Para la delimitación por ROI se utilizó las función de *fillPoly()* cuyos argumentos utilizados fueron los siguientes:

cv2.fillPoly (Img, endPoints, Color)

Donde:

Img: Es la imagen donde se dibuja el polígono

EndPoints: Matriz que contiene las coordenadas de los vértices del polígono

Color: Color del polígono dibujado

Dicha función dibuja un polígono relleno, que tiene como vértices las coordenadas encontradas en la matriz EndPoints, basándose en el sistema de coordenadas de imágenes mostrado en la figura N° 12, donde la coordenada (0,0) le corresponde a la esquina superior izquierda de la imagen. Se establecieron las coordenadas de los vértices que conforman los 3 polígonos que delimitan los espacios de estacionamiento. Después, se cambiaron los valores de todos los píxeles del fotograma a (0,0,0), formato BGR que maneja el OpenCV. Seguidamente, se llenó de blanco todos los píxeles del fotograma, con valores (255,255,255), que están dentro de los polígonos delimitados anteriormente.

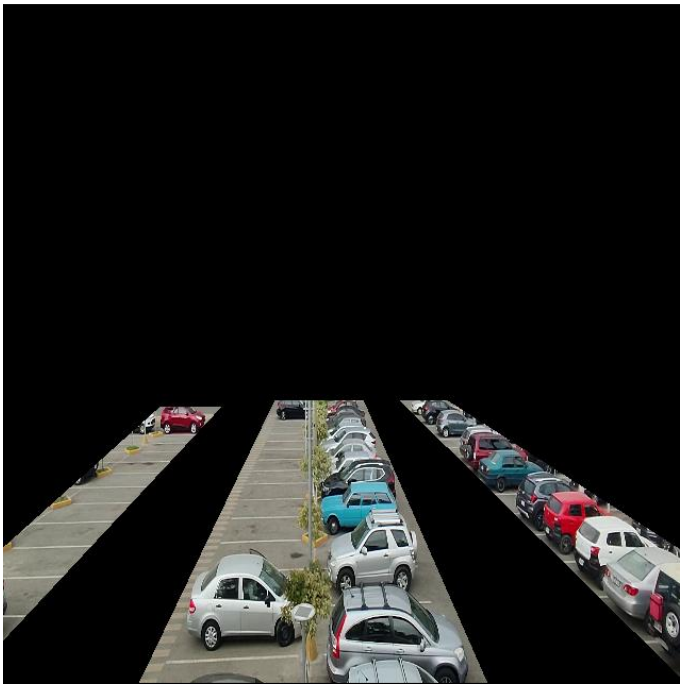

Figura N 16: Zonas del estacionamiento delimitadas por ROI
Fuente: Elaboración propia.

Luego, se realizó una operación AND entre los valores de los píxeles de la nueva máscara y los píxeles del fotograma original, obteniendo como resultado la figura 16, donde se aprecia solo los espacios de estacionamiento delimitados por las ROI. Por último, la imagen creada se pasó a la red neuronal convolucional para la detección de los vehículos, la creación de sus respectivas bounding boxes y del conteo que se extrae de manera automática de la salida de la red neuronal convolucional. En la figura 17 se muestra un diagrama de bloques que resume los pasos dados para la obtención de las ROI, desde la extracción de los fotogramas hasta la imagen final que se le entrega a la red para la detección.

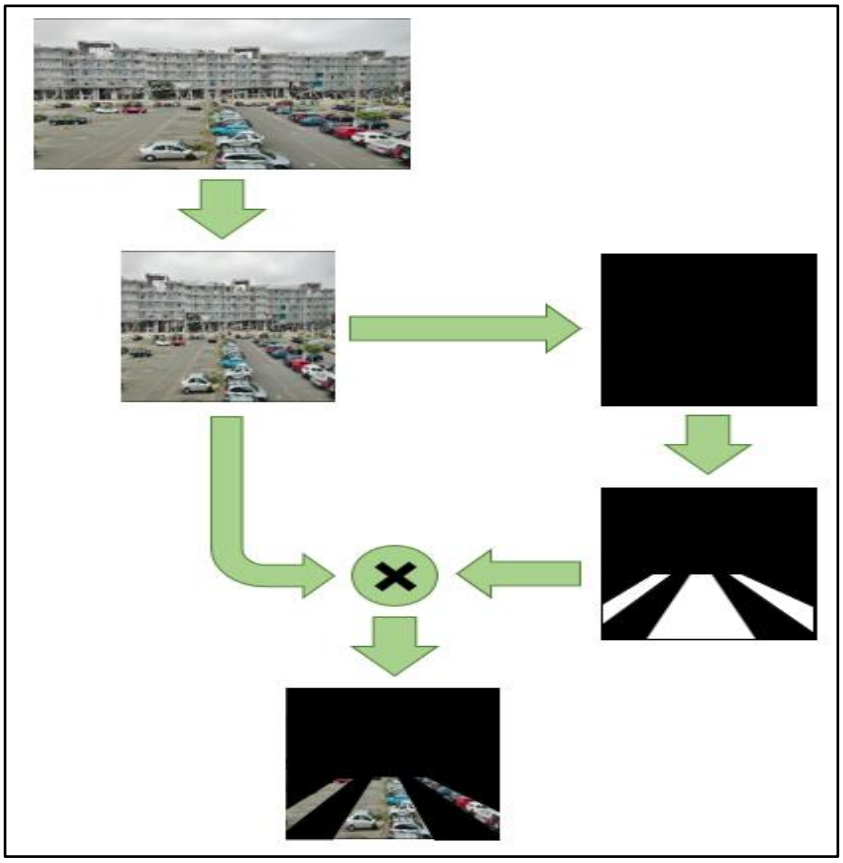

Figura N 17: Diagrama de bloques del método de conteo por ROI
Fuente: Elaboración propia

3.3.4. Implementación del aplicativo GUI para YOLO V5

Para la implementación del interfaz gráfico de usuario se utilizó la librería Tkinter de python. Por el consumo alto de CPU de de la memoria de video, se le agregó un botón que permite iniciar y detener las detecciones de los

vehículos, además de casillas para visualizar la cantidad de vehículos estacionados y los espacios de estacionamiento que quedan.

Figura N 18: Interfaz gráfica de usuario utilizando Tkinter
Fuente: Elaboración propia

3.4. Implementación de Faster RCNN

3.4.1. Transfer Learning a la red neuronal utilizando Fine Tuning

Se utilizó el método de Fine Tuning para el re-entrenamiento de la red Faster RCNN como un método de Transfer Learning. Esto se logró manteniendo los pesos iniciales de la red pre entrenada y haciendo un re-entrenamiento a toda la red, modificando solo la capa de salida encargada de la clasificación para dos clases: background y carros. En la Figura 12 se observa la arquitectura de Faster RCNN y como se estaría solo modificando la capa de softmax regression, encargada de calcular la probabilidad de predicción de cada clase.

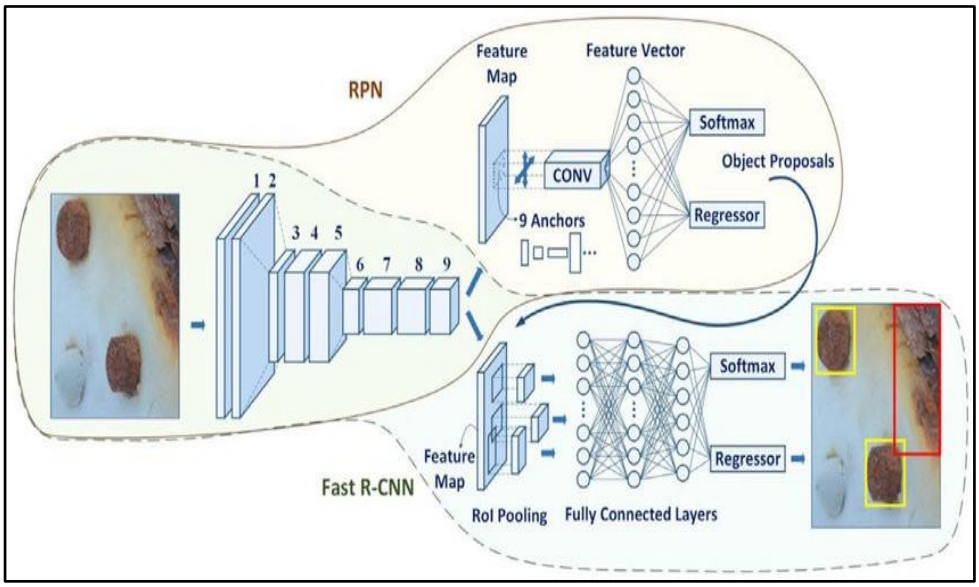

Figura N 19: Arquitectura de la red neuronal artificial convolucional Faster RCNN
Fuente: https://towardsdatascience.com/faster-rcnn-object-detection-f865e5ed7fc4#:~:text=Faster%20RCNN%20is%20an%20object,SSD%20(%20Single%20Shot%20Detector).

3.4.2. Implementación del sistema de conteo por ROI y LOI

Tanto la implementación del sistema de conteo por ROI y LOI para la red neuronal convolucional Faster RCNN, fue la misma descrita en el apartado 3.3.4 de la presente tesis.

3.4.3. Implementación del aplicativo GUI para Faster RCNN

La implementación del aplicativo GUI para la red neuronal convolucional Faster R-CNN, es la misma descrita en el apartado 3.3.5 de la presente tesis.

CAPÍTULO IV: PRUEBAS Y RESULTADOS OBTENIDOS

4.1. Comparación de redes neuronales YOLO v5 y Faster RCNN utilizando la métrica de media de precisión promedio

Para la comparación de las redes neuronales convolucionales, se utilizaron las métricas de media de precisión promedio (mean average precision o mAP) tomando solo las detecciones que tienen una intersección sobre unión (IoU) mayor a 0.5, que es lo convencional para considerar una detección como válida.

En la figura 20 se observa la gráfica de mAP versus el número de época durante el proceso de entrenamiento de la red neuronal convolucional YOLO v5. Como se observa, durante un entrenamiento de 100 épocas la red YOLO v5 obtuvo una mAP de 81.615%. De igual manera, como se muestra en la figura 21, se entrenó la red neuronal convolucional Faster RCNN durante 100 épocas, logrando un mAP de 81.595%, ligeramente menor a la red YOLO v5.

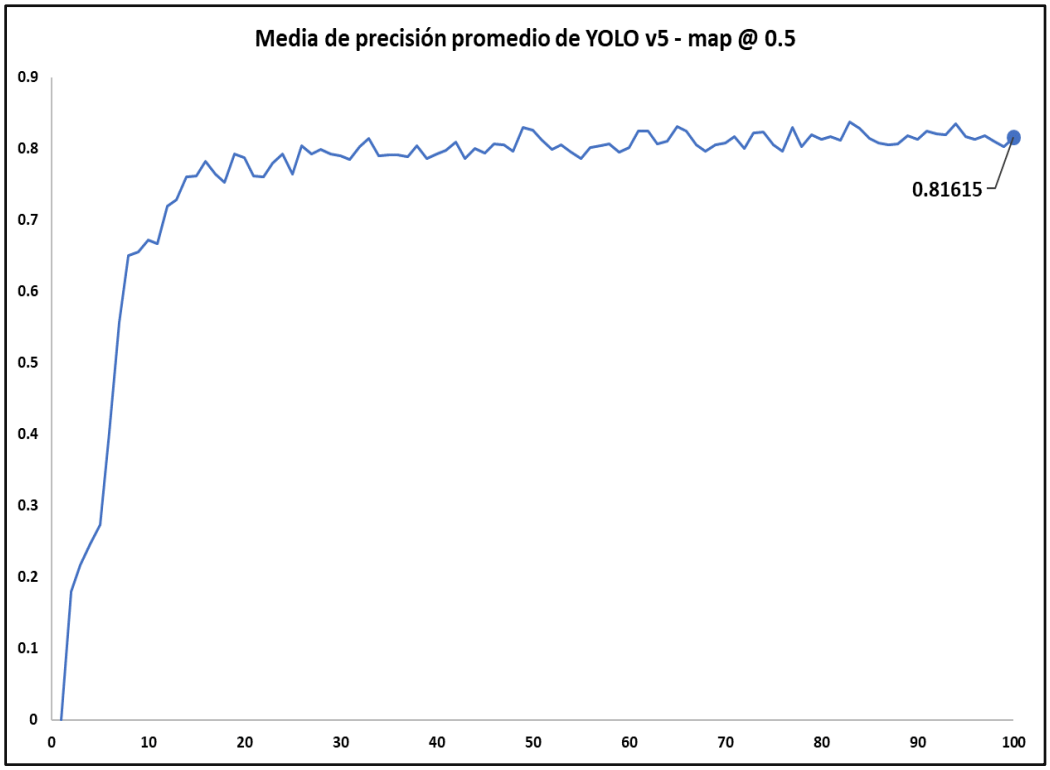

Figura N 20: Gráfico de media de precisión promedio de YOLO v5
Fuente: Elaboración propia

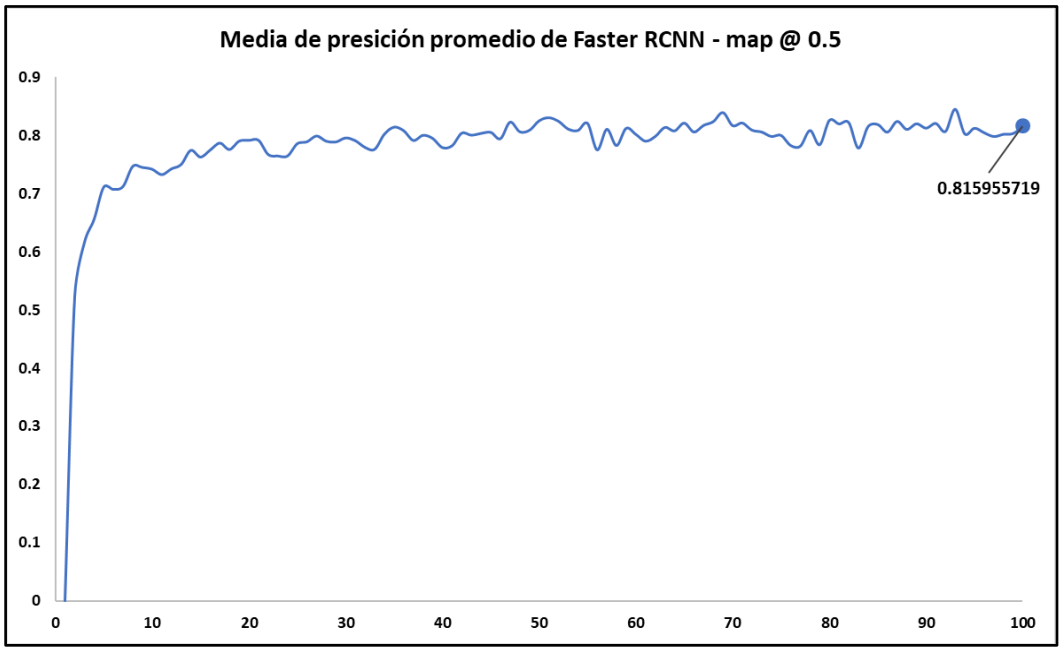

Figura N 21: Gráfico de media de precisión promedio de Faster RCNN
Fuente: Elaboración propia

4.2. Comparación de resultados de los métodos de conteo de vehículos

Para la comparación de los 2 métodos de conteo de vehículos, se utilizó la red neuronal YOLO v5, sin ningún motivo en particular con la red Faster RCNN, puesto que no se tenía como objetivo observar el rendimiento de las redes sino la eficacia de los métodos de conteo.

4.2.1. Método de conteo usando líneas de interés

Para el método de conteo usando líneas de interés se dibujaron líneas que delimitaban las secciones de estacionamiento donde se quería contar los vehículos. Para ello se utilizó el procedimiento descrito en la sección 3.3.3. de la presente tesis, para la implementación de las líneas delimitadoras. Primero se inicializó la red neuronal convolucional para la detección de vehículos que se visualiza en la forma de puntos rojos, como se observa en la figura 22, ubicadas en el centro de las detecciones. A la vez, el algoritmo contó los puntos rojos ubicados solo entre las líneas delimitadoras color azul, observándose la cantidad contada en la parte superior izquierda de la ventana. Se tomó captura de pantalla a los 3 primeros cuadros de video tras el inicio de la red neuronal para observar la eficacia del método de conteo. Como se observan en las figuras 22, 23 y 24, el contador de vehículos aumenta

progresivamente, llegando a contar 86 vehículos a pesar de que solo habían 28 carros estacionados entre las líneas delimitadoras.

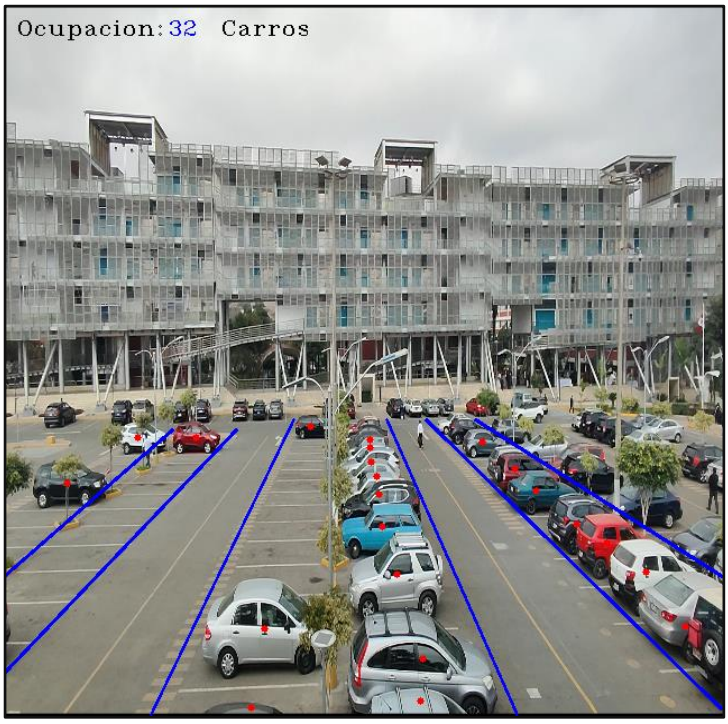

Figura N 22: Implementación de método de conteo por LOI - Primer cuadro de video
Fuente: Elaboración propia

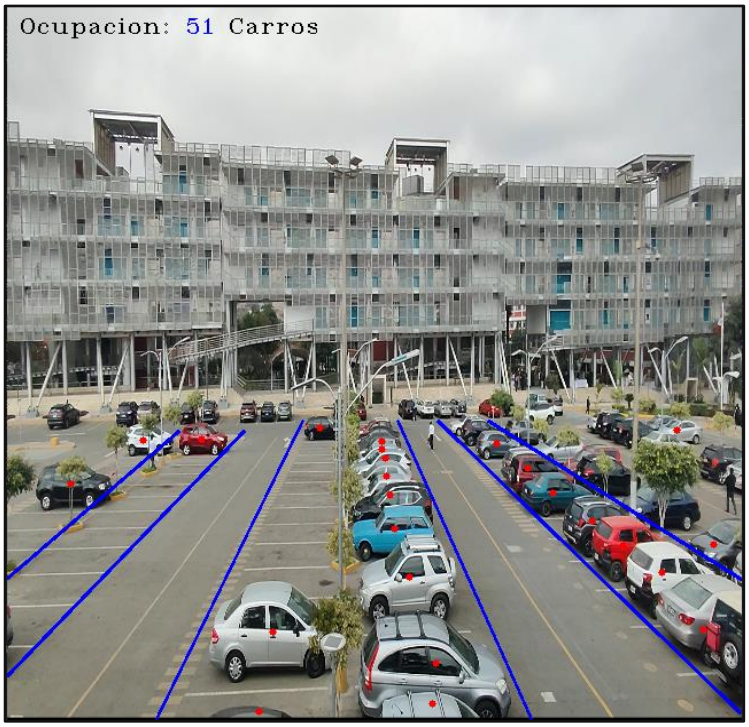

Figura N 23: Implementación de método de conteo por LOI - Segundo cuadro de video
Fuente: Elaboración propia

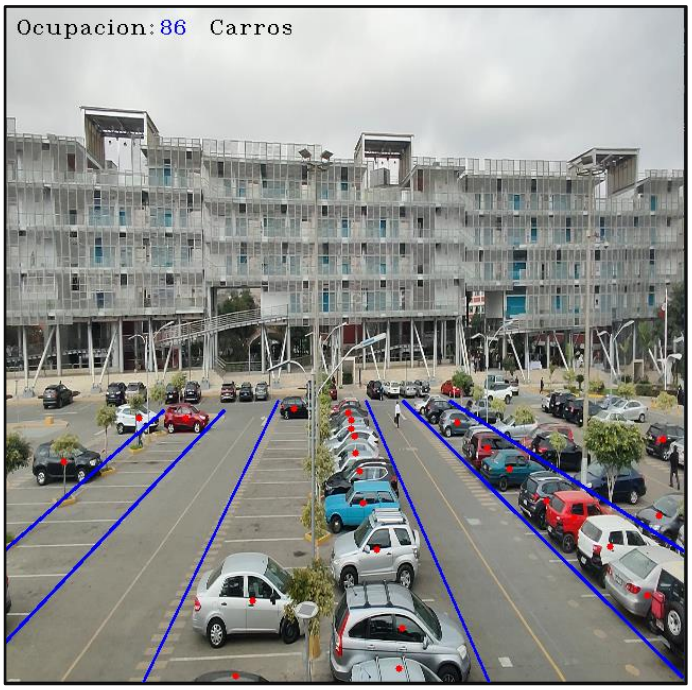

Figura N 24: Implementación de método de conteo por LOI - Tercer cuadro de video
Fuente: Elaboración propia

4.2.2. Método de conteo usando regiones de interés

Para el método de conteo usando regiones de interés se extrajeron secciones de estacionamiento en donde se querían contar los vehículos. Para ello se utilizó el procedimiento descrito en la sección 3.3.3. de la presente tesis. Primero se inicializó la red neuronal convolucional para la detección de vehículos que se visualiza en la forma de rectángulos color verde como se aprecia en la figura 25. A su vez, el algoritmo solo contó los rectángulos verdes ubicados dentro del área delimitadora, observándose la cantidad contada en la parte superior izquierda de la ventana. Se tomó captura de pantalla a los 3 primeros cuadros de video tras el inicio de la red neuronal para observar la eficacia del método de conteo. Como se observa en las figuras 25, 26 y 27, el contador de vehículos no incrementa indefinidamente como sucedió con el método de conteo usando LOI, sino que se mantiene relativamente estable, oscilando entre valores de 22 a 25 vehículos contados. Las siguientes pruebas se realizaron utilizando solo el método de conteo por ROI, ya que el método usando LOI probó ser totalmente ineficaz e inestable y no hubiese proveído ningún aporte a las siguientes pruebas que se muestran en las siguientes secciones.

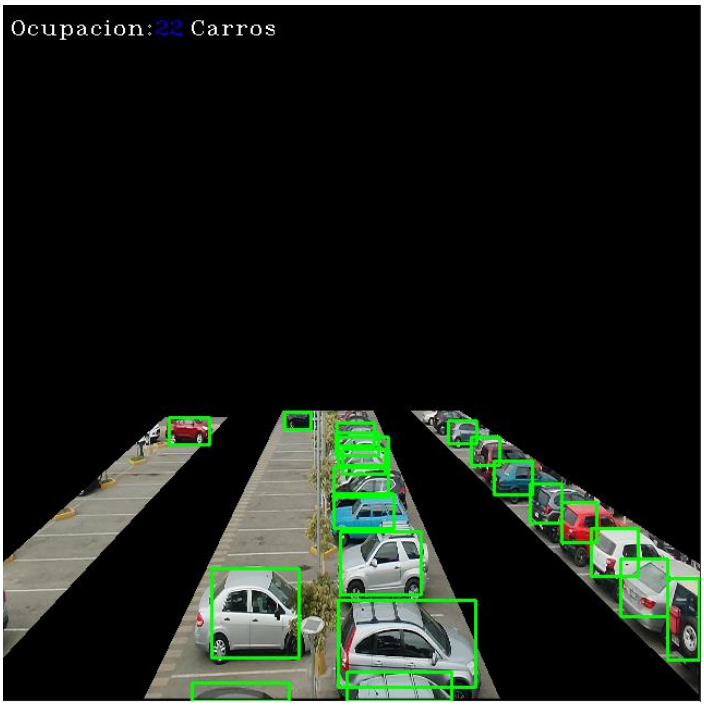

Figura N 25: Implementación de método de conteo por ROI - Primer cuadro de video
Fuente: Elaboración propia

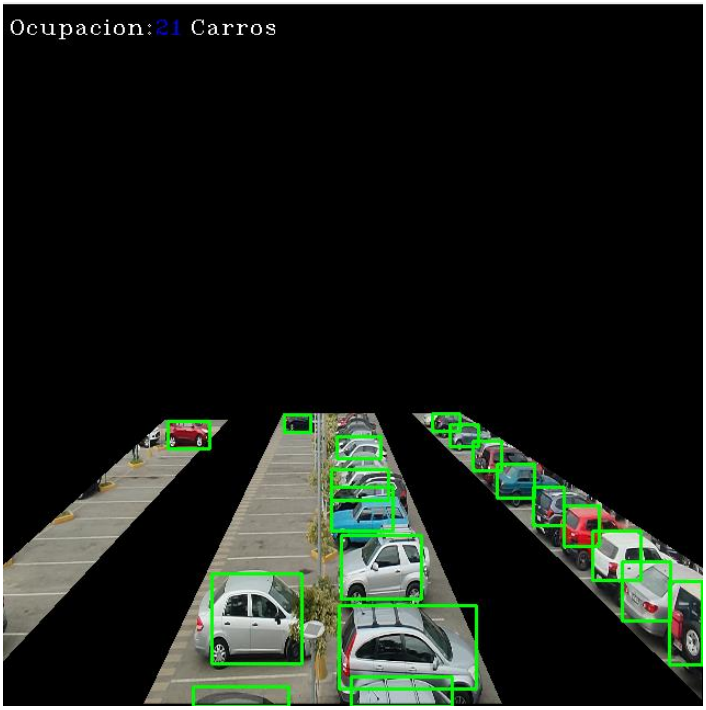

Figura N 26: Implementación de método de conteo por ROI - Segundo cuadro de video
Fuente: Elaboración propia

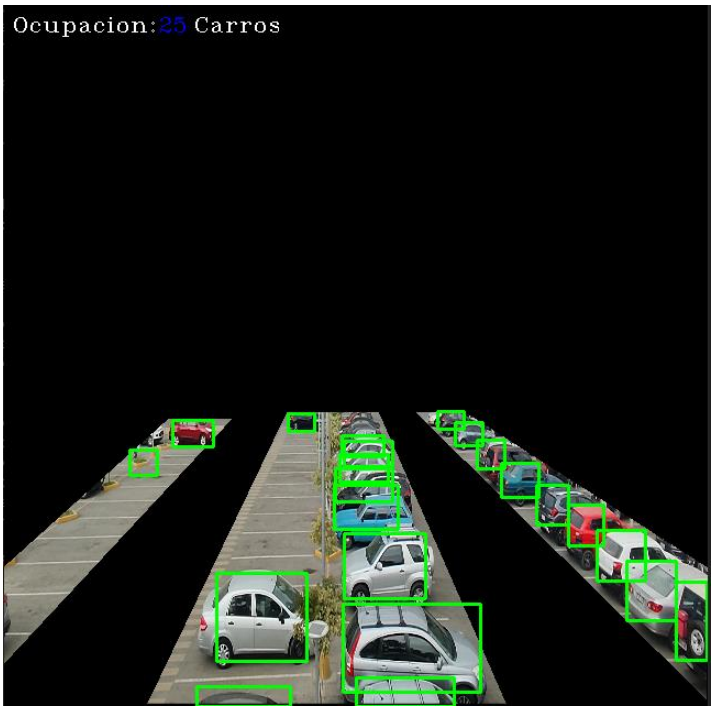

Figura N 27: Implementación de método de conteo por ROI - Tercer cuadro de video
Fuente: Elaboración propia

4.3. Resultados obtenidos de las redes neuronales YOLO v5 y Faster RCNN con 56 espacios de estacionamiento con umbrales de 0.35 y 0.55

Para esta prueba, se abarcó los 56 espacios de estacionamiento, con el fin de observar el rendimiento de las redes neuronales cuando tienen muchos vehículos presentados. Adicionalmente, se utilizaron dos valores de umbral (threshold) de 0.35 y 0.55 para estudiar la estabilidad del algoritmo al limitar la cantidad de detecciones que las redes neuronales convolucionales entregan al contador. Además, se agregó un contador de cuadros por segundo (FPS o frames per second) para observar la velocidad de la red a la hora de detectar un vehículo. Tanto el video en vivo, como las bounding boxes, el contador de vehículos y de FPS se observaron a través de una GUI programada con dos botones para iniciar y finalizar las detecciones. Finalmente, cabe mencionar que se utilizó la red wi-fi de estudiantes de la Universidad Ricardo Palma y que la laptop, utilizada para la recepción de los datos de video que envíaba el celular y el funcionamiento del resto del algoritmo, se posicionó al lado derecho de la garita de la puerta 3, para aprovechar al máximo la señal de wi-fi; sin embargo, aun tomando en cuenta todas estas consideraciones, se realizó un ping desde la laptop al celular conectado a la red y se mostró que tiene un delay de casi medio segundo como se observa en la figura 28.

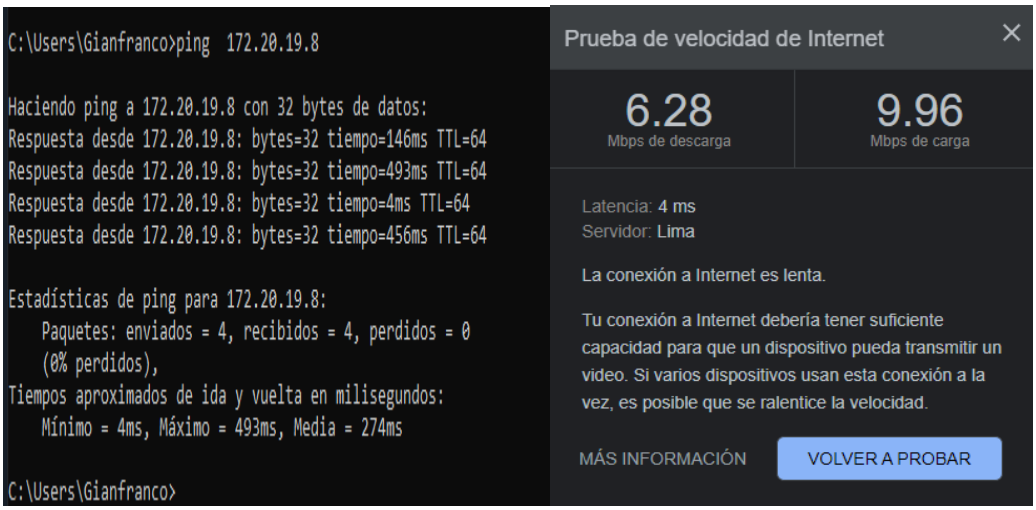

Figura N 28: Prueba de latencia y velocidad de internet realizado desde la laptop al celular conectado a la red de Universidad Ricardo Palma
Fuente: Elaboración propia

4.3.1. Resultados de los FPS obtenidos con 56 espacios de estacionamiento

Se extrajo la cantidad de FPS observados en el GUI en una grabación con una duración de un minuto y medio, para analizar si es que existía una diferencia en velocidad entre las 2 redes neuronales y al cambiarles su valor de umbral. Cabe destacar que, durante las pruebas realizadas conectadas a la red de la Universidad Ricardo Palma, se experimentó un delay, adicional al de la figura 28, de aproximadamente 5 segundos, que afectó seriamente la velocidad del algoritmo en general.

En la figura 29 se observa los FPS extraídos en cada segundo durante 90 segundos de video, se colocaron en una tabla en excel para obtener el gráfico mostrado, así como su promedio de FPS mostrado a través de la línea horizontal color naranja. El mismo procedimiento se realizó para los tres casos restantes que son el de la red neuronal YOLO v5 con umbral de 0.55, Faster RCNN con umbral de 0.35 y Faster RCNN con un umbral de 0.55, cómo se pueden observar en las figuras 30, 31 y 32 respectivamente.

Como se aprecia en las figuras mostradas, se obtuvieron unas cantidades bajas de FPS; sin embargo, la red neuronal YOLO v5 muestra una ligera tasa mayor de FPS que a la red neuronal Faster RCNN. También se observó que el cambio de umbral no afecta significativamente la velocidad, puesto que en las figuras 31 y 32 la cantidad de FPS disminuye ligeramente para la red Faster RCNN, mientras que en las figuras 29 y 30 aumentan para la red YOLO v5.

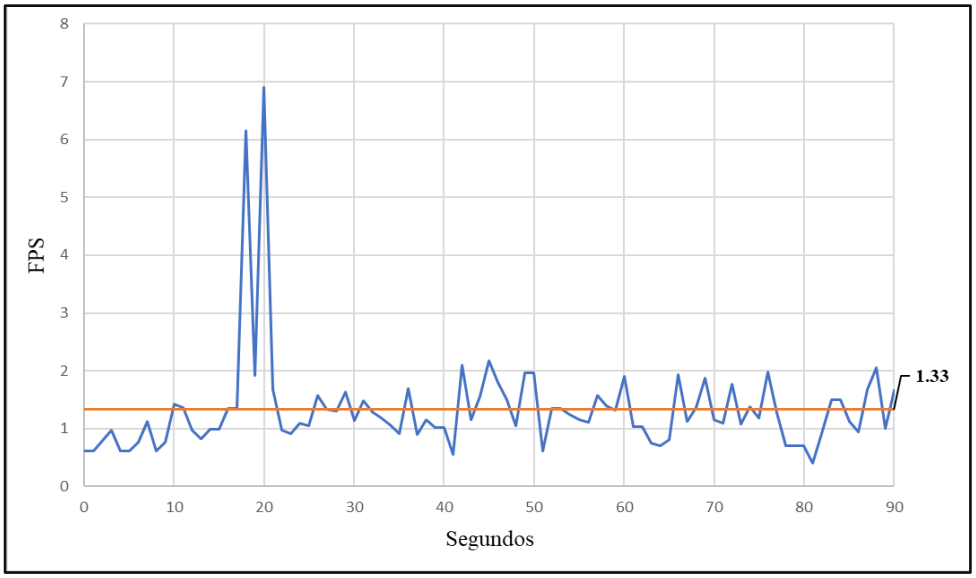

Figura N 29: FPS obtenidos con YOLO v5 y con umbral de 0.35 - 56 espacios
Fuente: Elaboración propia

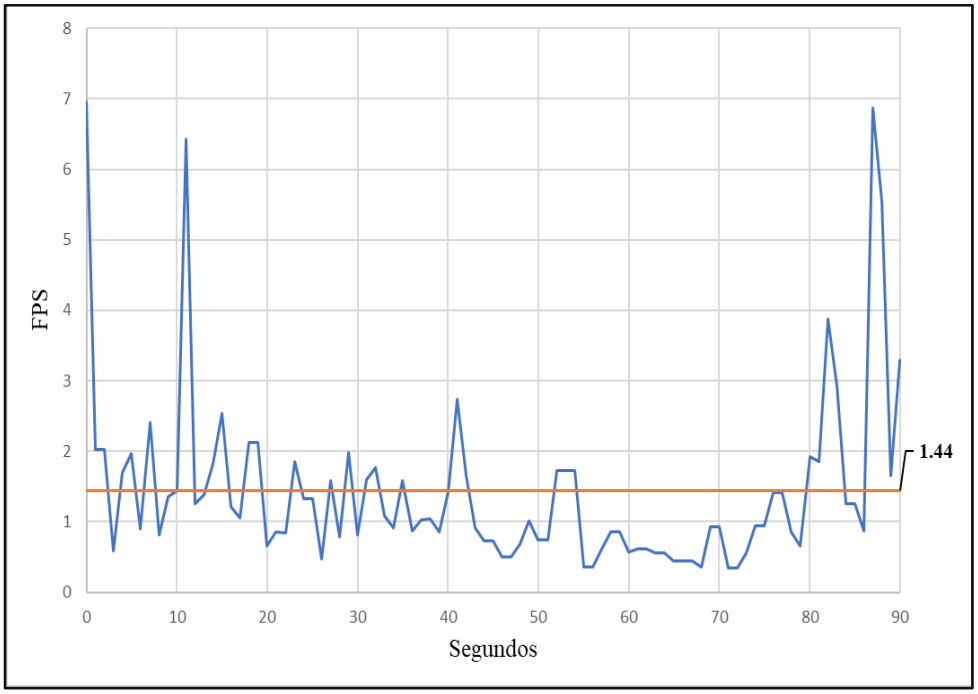

Figura N 30: FPS obtenidos con YOLO v5 y con umbral de 0.55 - 56 espacios
Fuente: Elaboración propia

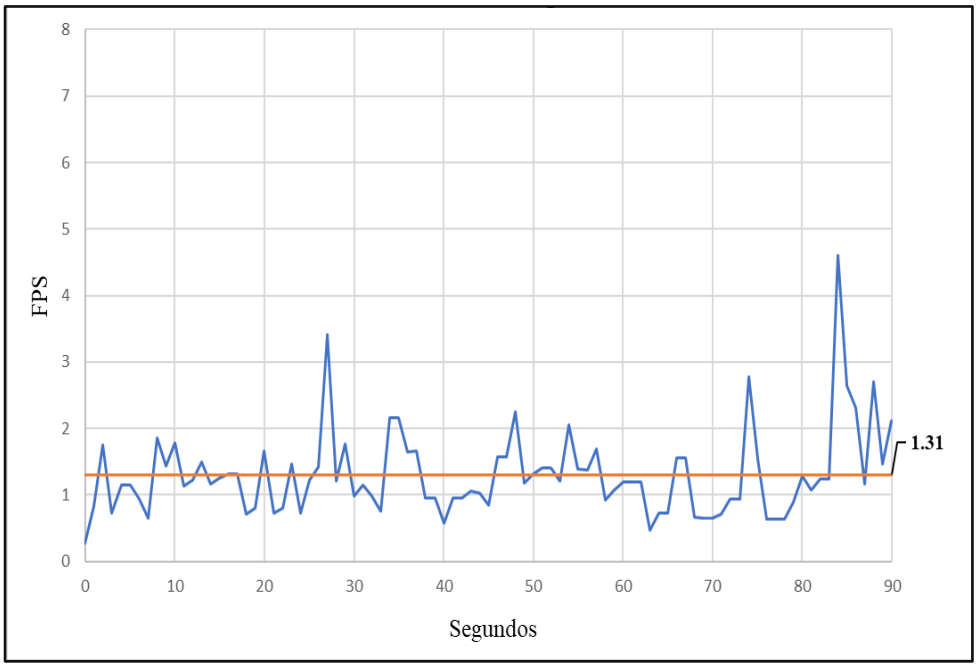

Figura N 31: FPS obtenidos con Faster RCNN y con umbral de 0.35 - 56 espacios
Fuente: Elaboración propia

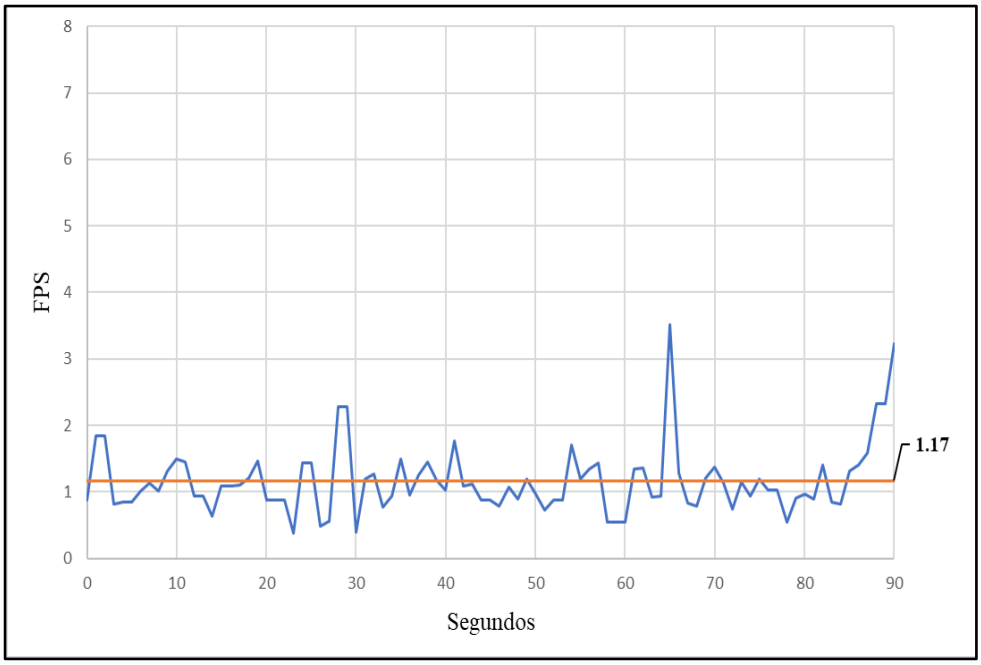

Figura N 32: FPS obtenidos con Faster RCNN y con umbral de 0.55 - 56 espacios
Fuente: Elaboración propia

4.3.2. Resultados de la prueba de precisión y estabilidad del contador y de las redes neuronales convolucionales con 56 espacios de estacionamiento

Para esta prueba se extrajeron el número de espacios disponibles mostrados en la GUI, en cada segundo durante una grabación de 90 segundos en total, con la intención de observar la variación del contador en el tiempo con respecto al número de espacios disponibles reales y determinar su estabilidad, así como ver la cercanía de los valores del contador al número real de espacios disponibles para determinar su precisión.

En la figura 33 se aprecia la cantidad de espacios disponibles para la red neuronal YOLO v5 con umbrales de 0.35 (línea azul) y 0.55 (línea verde), además del número de espacios reales (línea naranja), cuya cantidad era 2 en el momento de la implementación. Así mismo, en la figura 34, se muestra la gráfica de la cantidad de espacios disponibles para la red Faster RCNN con umbrales de 0.35 (línea azul) y 0.55 (línea verde), y la cantidad de espacios reales disponibles cuya cantidad era 1 en el momento de la prueba.

Las redes son más precisas cuanto más cerca estén los valores de la línea naranja, y más estable cuantos menos picos presentan las líneas azules y verdes.

Comparando las figuras 33 y 34, resultó que red neuronal Faster RCNN era más precisa, puesto que obtuvo valores más cercanos a la línea naranja en comparación a la red neuronal YOLO v5; sin embargo, ambas redes presentaron menor precisión cuando se aumentaba el umbral de 0.35 a 0.55, pero a la vez mostraban mayor estabilidad, más precisamente durante los segundos 45 a 65.

En la figura 34, la red Faster RCNN obtuvo valores negativos debido a falsos positivos, es decir, detectaba vehículos donde no los había o detectaba un vehículo 2 veces, como se muestra en la figura 35, en el rectángulo rojo grande del lado derecho que encerró 2 vehículos, afectando la precisión del contador. Adicionalmente, se calculó el porcentaje de error en cada segundo durante una duración de 90 segundos, tal y como se muestra en la figura 36 y luego se calculó su promedio. Este procedimiento se realizó para calcular el porcentaje de error de la red neuronal YOLO v5 con umbrales de 0.35 y 0.55, y para la red neuronal Faster RCNN con umbrales de 0.35 y 0.55 como se muestra en las figuras 36, 37, 38 y 39 respectivamente. También, en las

figuras 40, 41, 42 y 43 se observan la mínima y máxima cantidad de números de autos contados por parte de la red neuronal YOLO v5 con umbrales de 0.35 y 0.55, y las figuras 44, 45, 46 y 47 por parte de la red neuronal Faster RCNN con umbrales de 0.35 y 0.55 respectivamente.

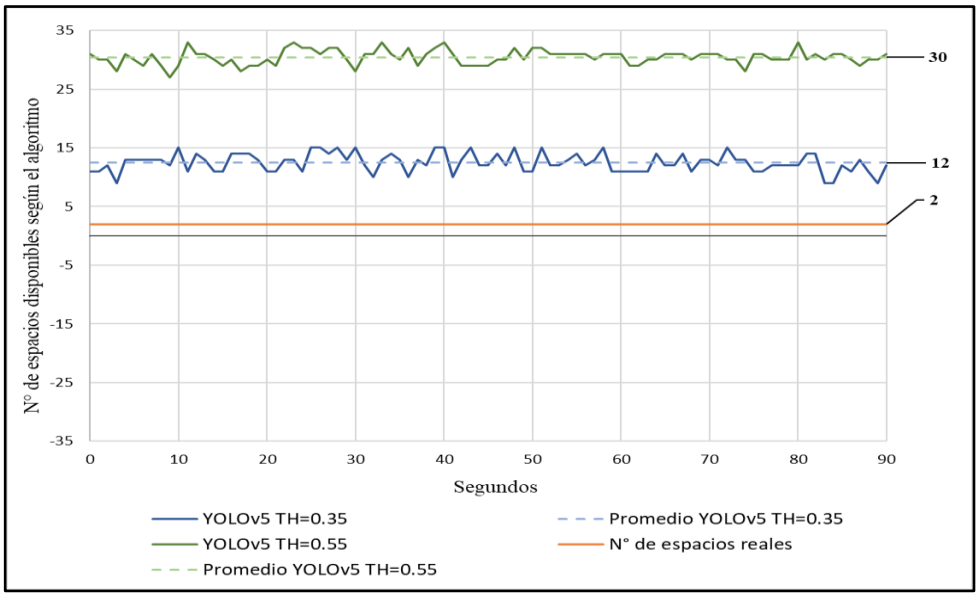

Figura N 33: Número de espacios disponibles con YOLO v5 con 56 espacios
Fuente: Elaboración propia

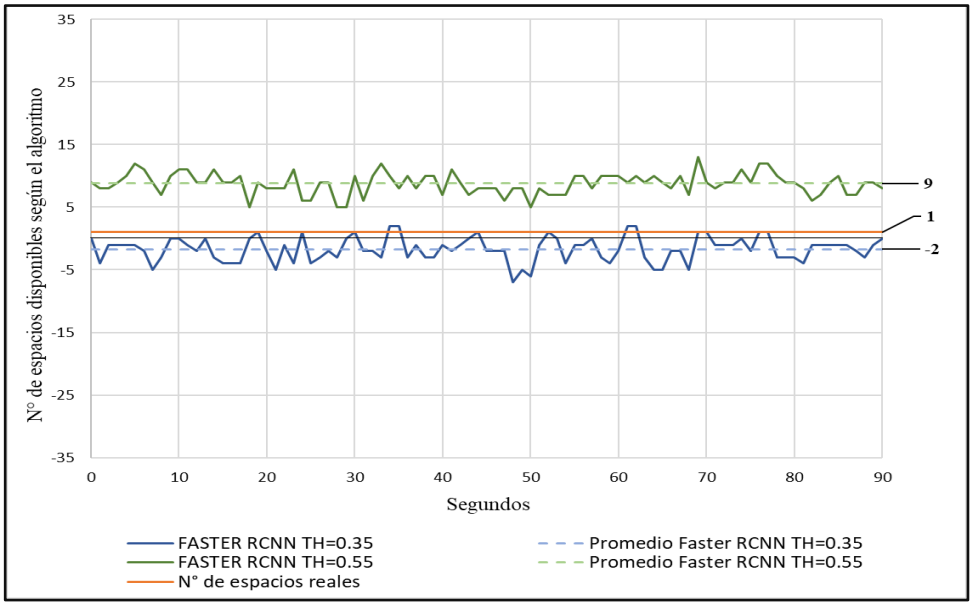

Figura N 34: Número de espacios disponibles con Faster RCNN con 56 espacios
Fuente: Elaboración propia

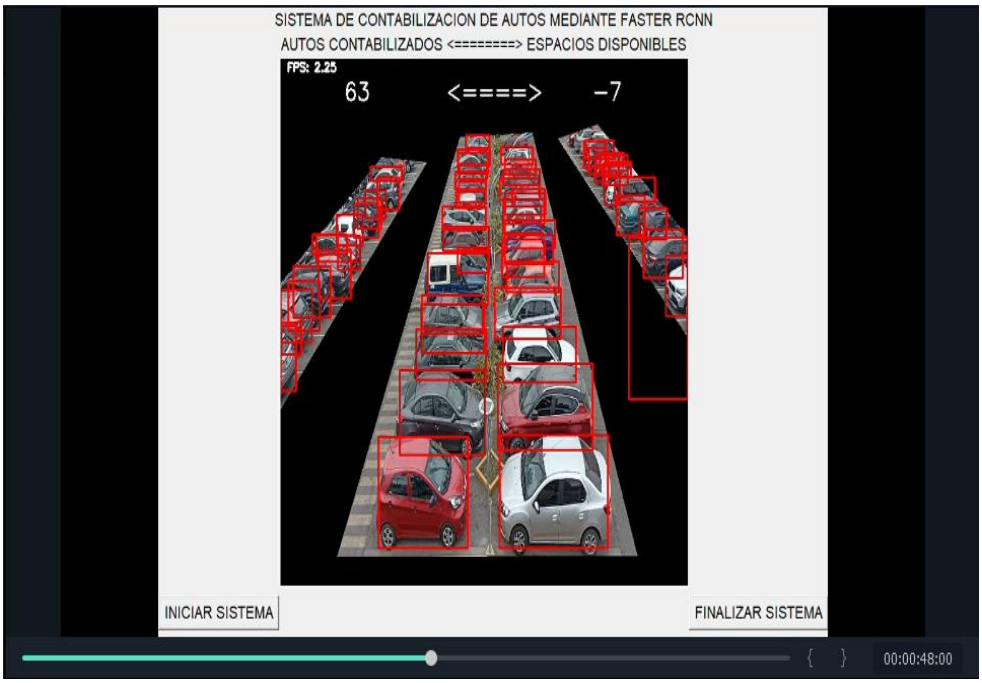

Figura N 35: Caso de falso positivo en la red neuronal Faster RCNN con umbral de 0.35 y 56 espacios
Fuente: elaboración propia

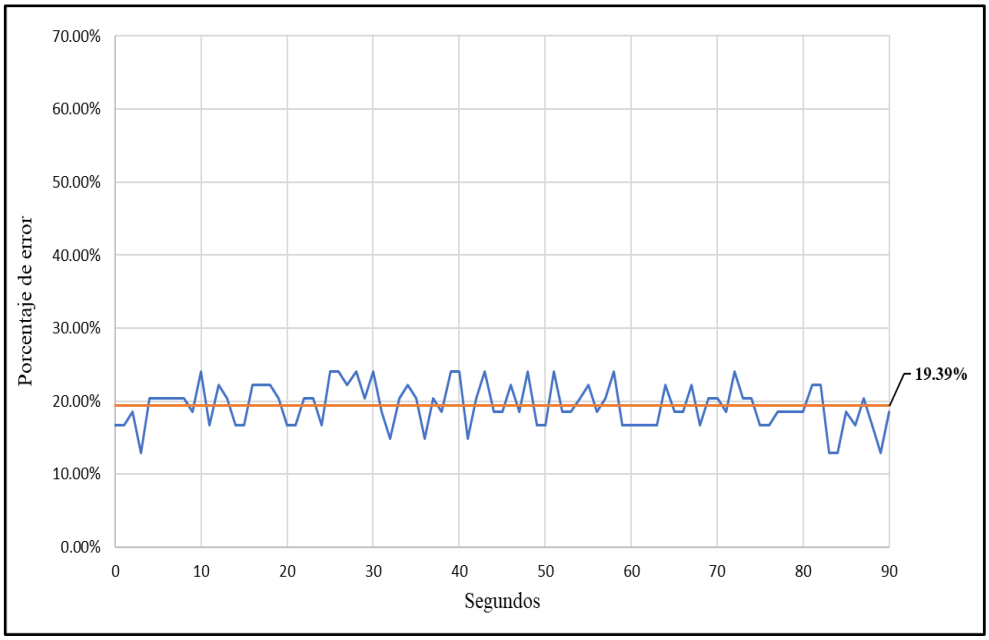

Figura N 36: Porcentaje de error YOLO v5 con umbral de 0.35 y con 56 espacios
Fuente: Elaboración propia

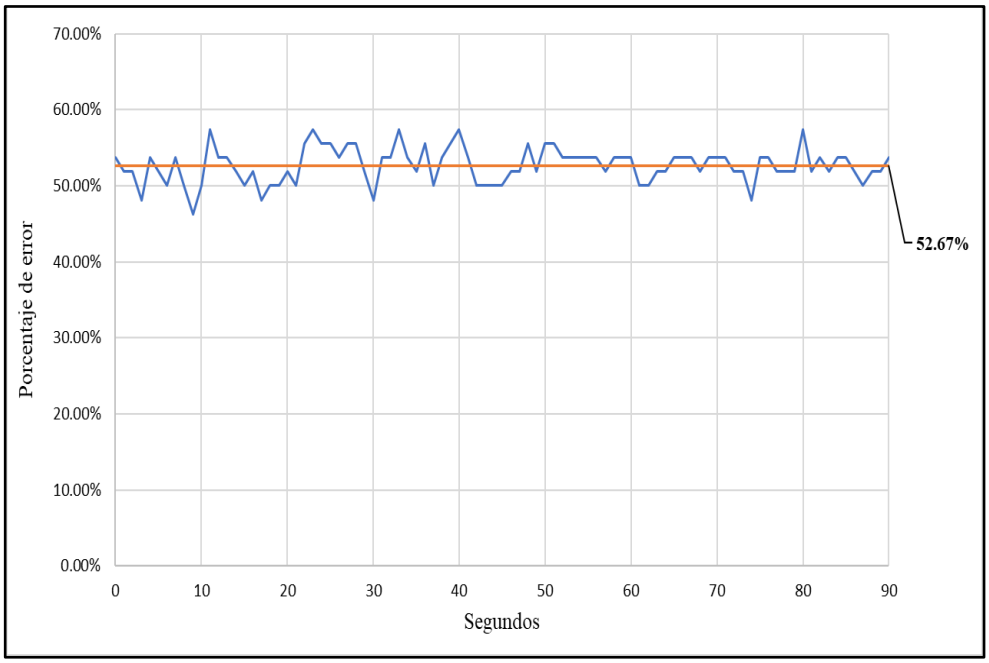

Figura N 37: Porcentaje de error YOLO v5 con umbral de 0.55 y con 56 espacios
Fuente: Elaboración propia

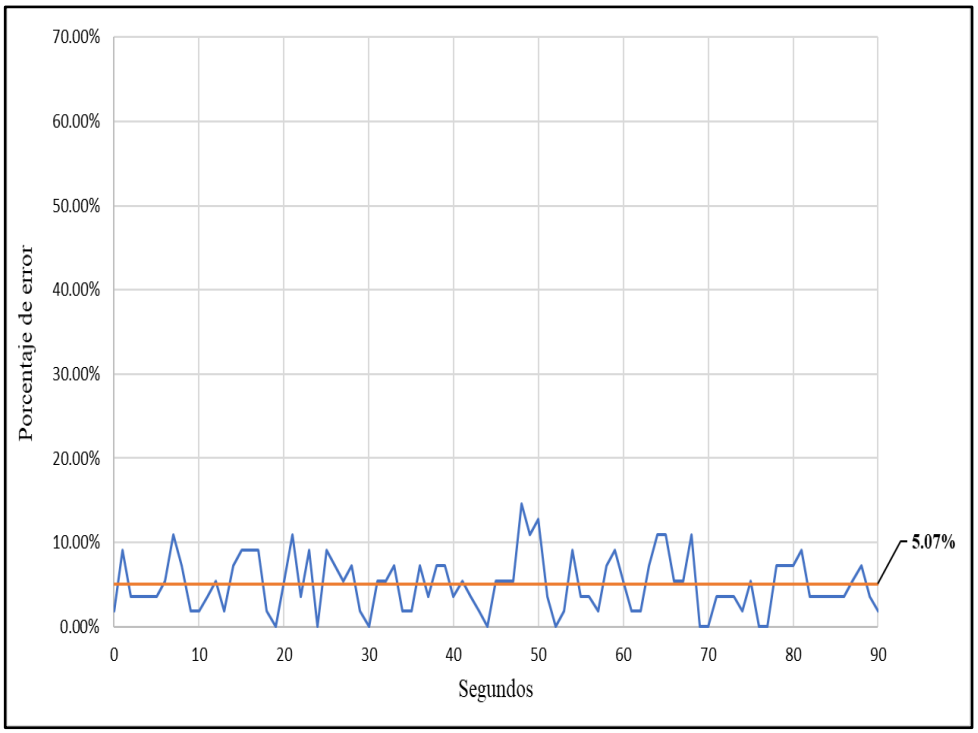

Figura N 38: Porcentaje de error Faster RCNN con umbral de 0.35 y con 56 espacios
Fuente: Elaboración propia

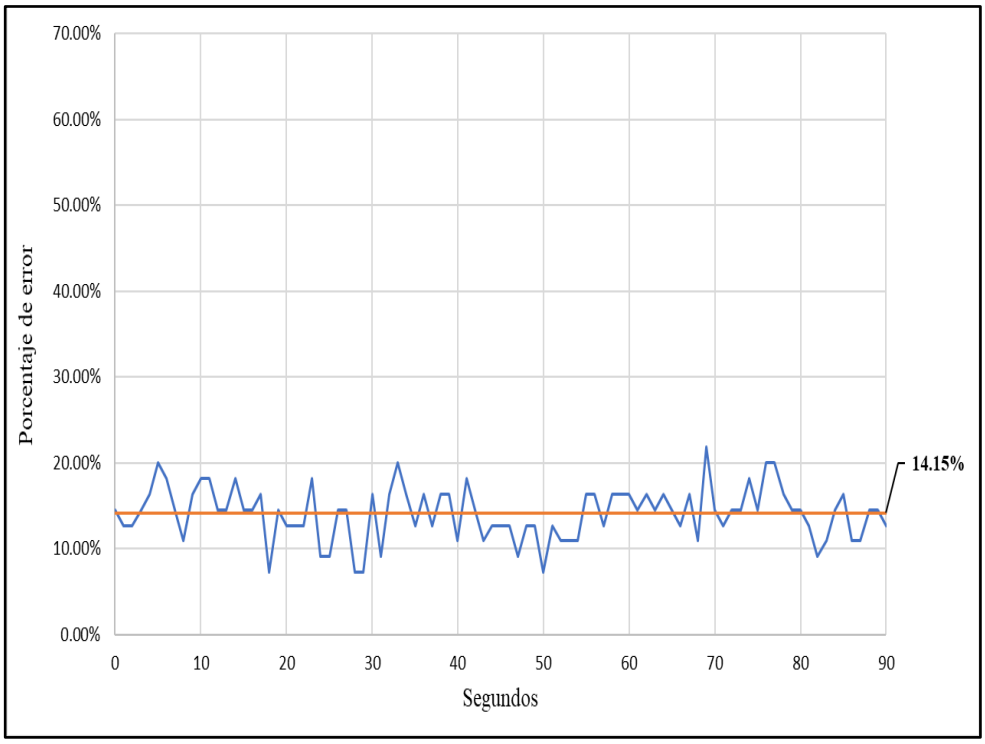

Figura N 39: Porcentaje de error Faster RCNN con umbral de 0.55 y con 56 espacios
Fuente: Elaboración propia

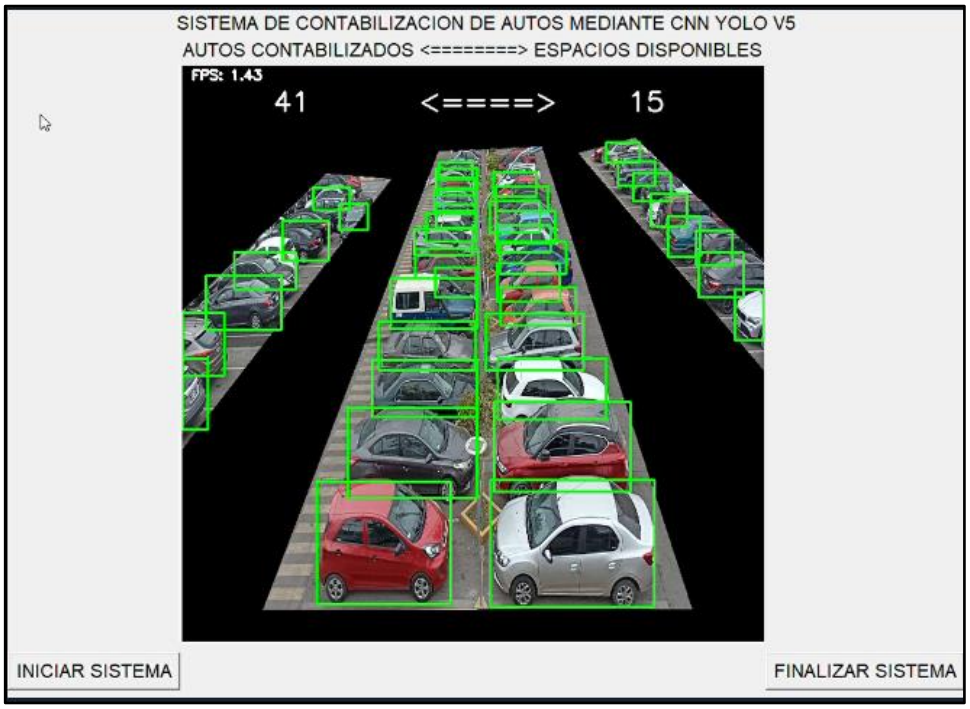

Figura N 40*: Cantidad mínima de autos contados por YOLO v5 con umbral de 0.35 y con 56 espacios*
Fuente: Elaboración propia

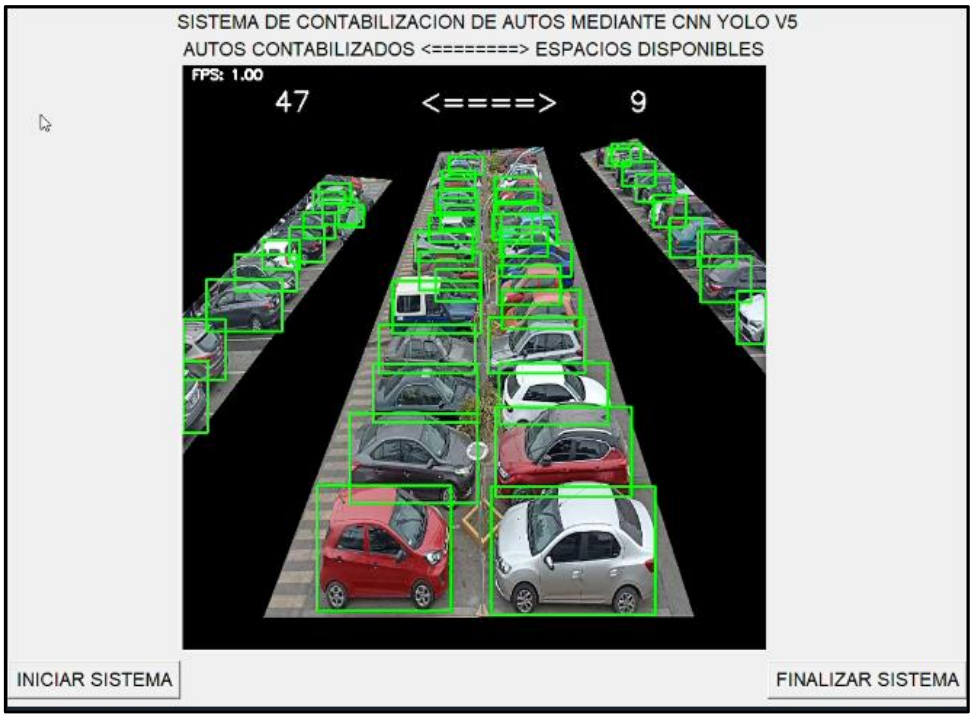

Figura N 41: Cantidad máxima de autos contados por YOLO v5 con umbral de 0.35 y con 56 espacios
Fuente: Elaboración propia

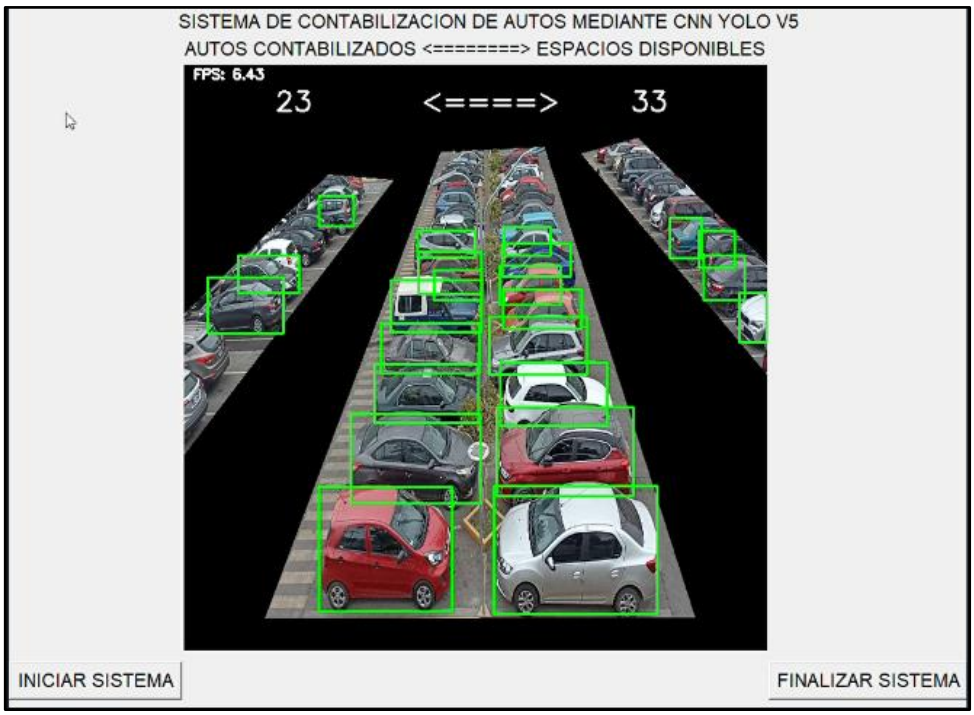

Figura N 42: Cantidad mínima de autos contados por YOLO v5 con umbral de 0.55 y con 56 espacios
Fuente: Elaboración propia

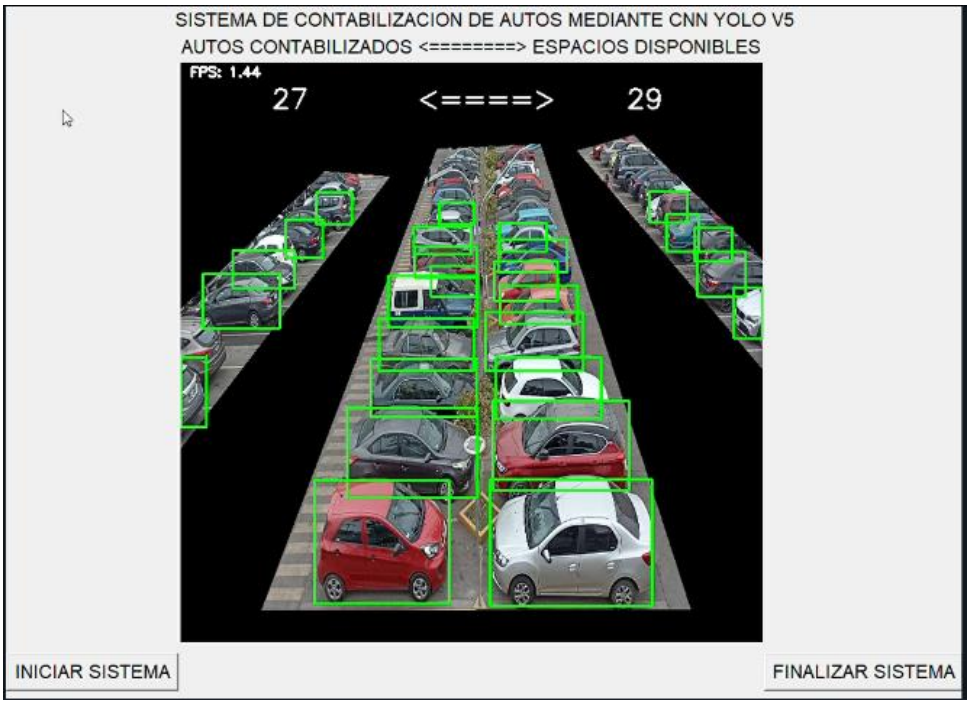

Figura N 43: Cantidad máxima de autos contados por YOLO v5 con umbral de 0.55 y con 56 espacios
Fuente: Elaboración propia

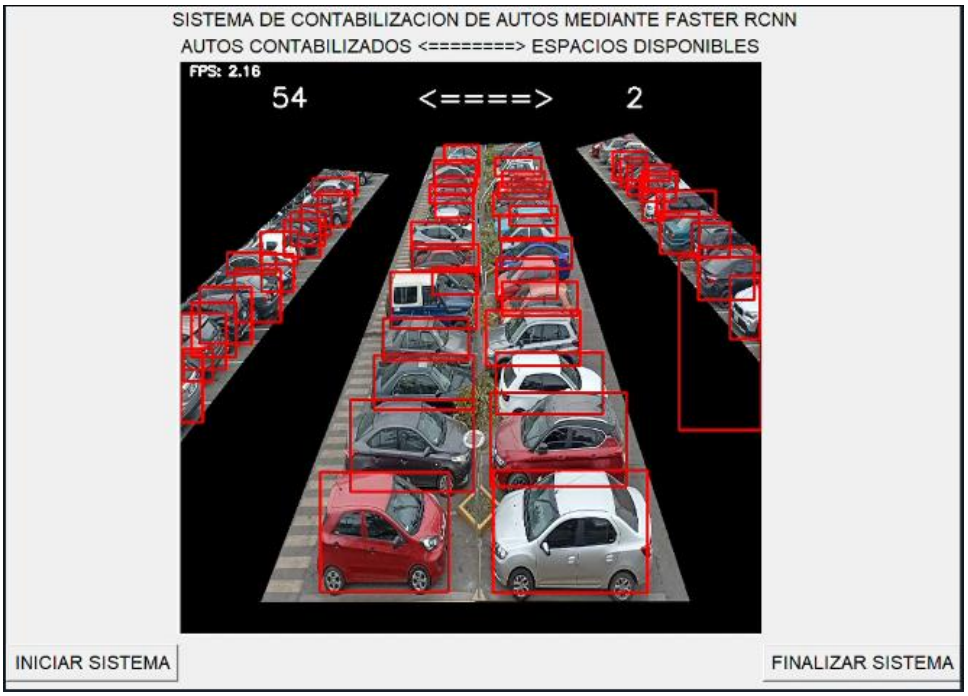

Figura N 44: Cantidad mínima de autos contados por Faster RCNN con umbral de 0.35 y con 56 espacios
Fuente: Elaboración propia

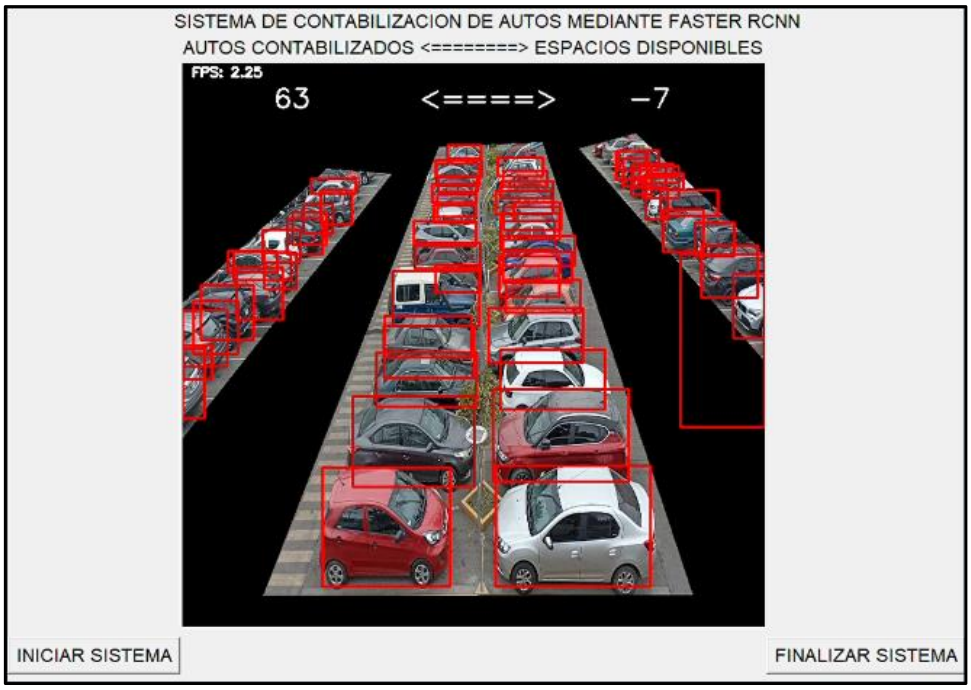

*Figura N 45: C*antidad máxima de autos contados por Faster RCNN con umbral de 0.35 y con 56 espacios.
Fuente: Elaboración propia

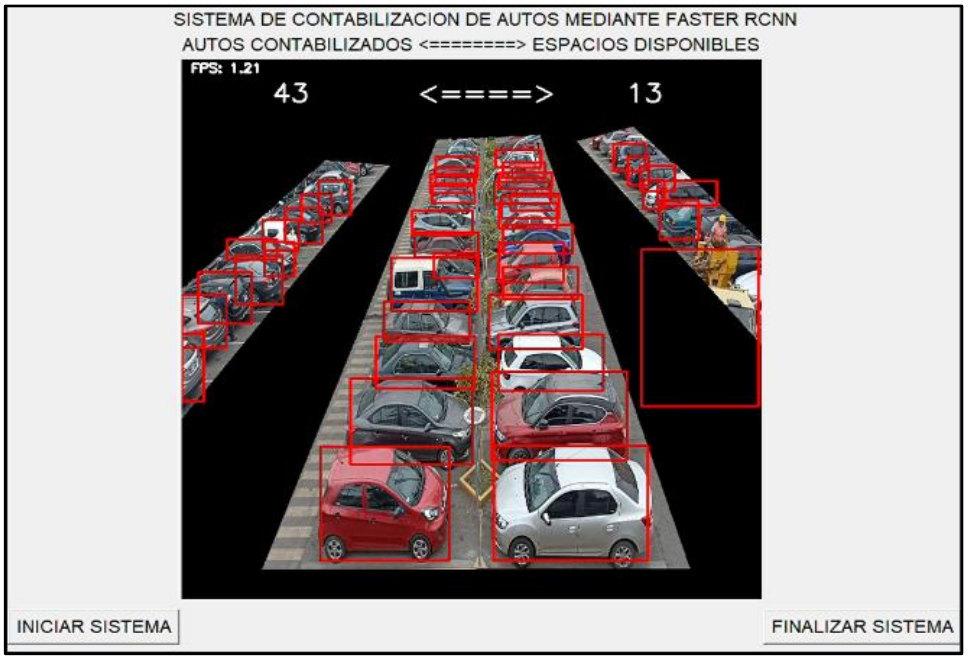

Figura N 46: Cantidad mínima de autos contados por Faster RCNN con umbral de 0.55 y con 56 espacios
Fuente: Elaboración propia

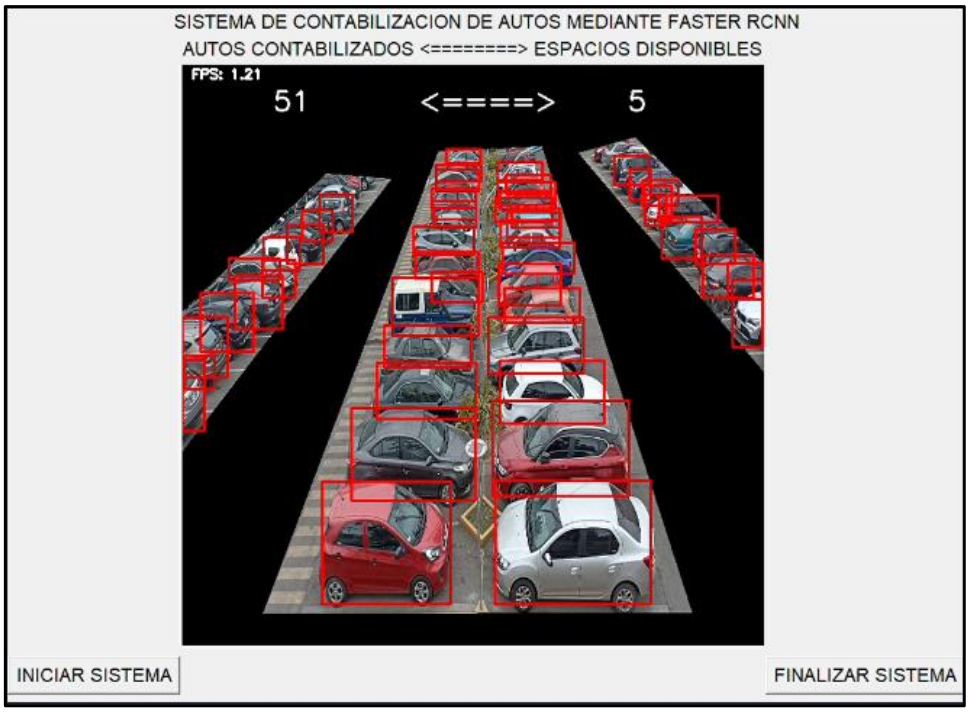

Figura N 47: Cantidad máxima de autos contados por Faster RCNN con umbral de 0.55 y con 56 espacios
Fuente: Elaboración propia

4.4. Resultados obtenidos de las redes neuronales YOLO v5 y Faster RCNN con 27 espacios de estacionamiento con umbrales de 0.35 y 0.55

Para esta prueba, se redujo a 27 espacios de estacionamiento, con el fin de observar el rendimiento de las redes neuronales cuando se les presenta a una menor cantidad de vehículos. Adicionalmente, se utilizaron dos valores de umbral (threshold) de 0.35 y 0.55 para estudiar la estabilidad del algoritmo al limitar la cantidad de detecciones que las redes neuronales convolucionales entregan al contador. Además, se agregó un contador de cuadros por segundo (FPS o frames per second) para observar la velocidad de la red a la hora de detectar un vehículo. Tanto el video en vivo, como las bounding boxes, el contador de vehículos y de FPS se observaron a través de una GUI programada con dos botones para iniciar y finalizar las detecciones. Finalmente, cabe mencionar que se utilizó la red wi-fi de estudiantes de la Universidad Ricardo Palma y que la laptop, utilizada para la recepción de los datos de video que enviaba el celular y el funcionamiento del resto del algoritmo, se posicionó al lado derecho de la garita de la puerta 3, para aprovechar al máximo la señal de wi-fi; sin embargo, aun tomando en cuenta todas estas consideraciones, se realizó un ping desde la laptop al celular conectado a la red y se mostró que tiene un delay de casi medio segundo como se observa en la figura 28.

4.4.1. Resultados de los FPS obtenidos con 27 espacios de estacionamiento

El procedimiento de extracción de los FPS es el mismo realizado en el apartado 4.3.1. de la presente tesis. También se experimentó un delay de aproximadamente 5 segundos adicional al retraso mostrado en la figura 28. En la figura 48 se observa los FPS extraídos en cada segundo durante 90 segundos de video, se colocaron en una tabla en excel para obtener el gráfico mostrado, así como su promedio de FPS mostrado a través de la línea horizontal color naranja. El mismo procedimiento se realizó para los tres casos restantes que son el de la red neuronal YOLO v5 con umbral de 0.55, Faster RCNN con umbral de 0.35 y Faster RCNN con un umbral de 0.55, cómo se pueden observar en las figuras 49, 50 y 51 respectivamente. Como se aprecia en las figuras mostradas, se obtuvieron unas cantidades bajas de FPS; sin embargo, son ligeramente superiores a las pruebas que se realizaron con 56 espacios de estacionamiento. La red neuronal YOLO v5 vuelve a tener una ligera cantidad mayor de FPS a la red neuronal Faster RCNN cuando tienen un umbral de 0.35, como se observa en las figuras 48 y 50; pero es inferior cuando tienen un umbral de 0.55, que se aprecia en las figuras 49 y 51.

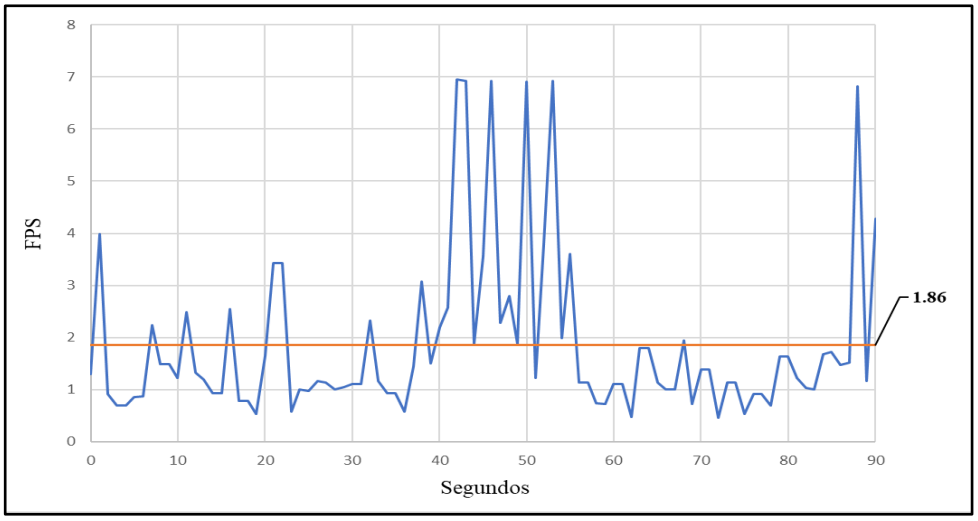

Figura N 48: FPS obtenidos con YOLO v5 y con umbral de 0.35 - 27 espacios
Fuente: Elaboración propia

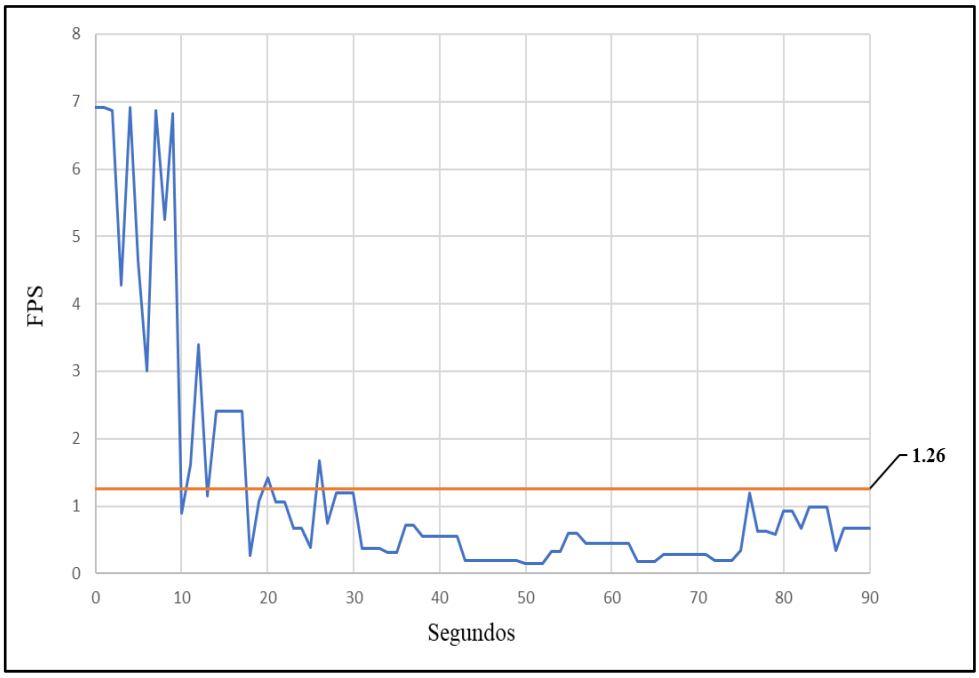

Figura N 49: FPS obtenidos con YOLO v5 y con umbral de 0.55 - 27 espacios
Fuente: Elaboración propia

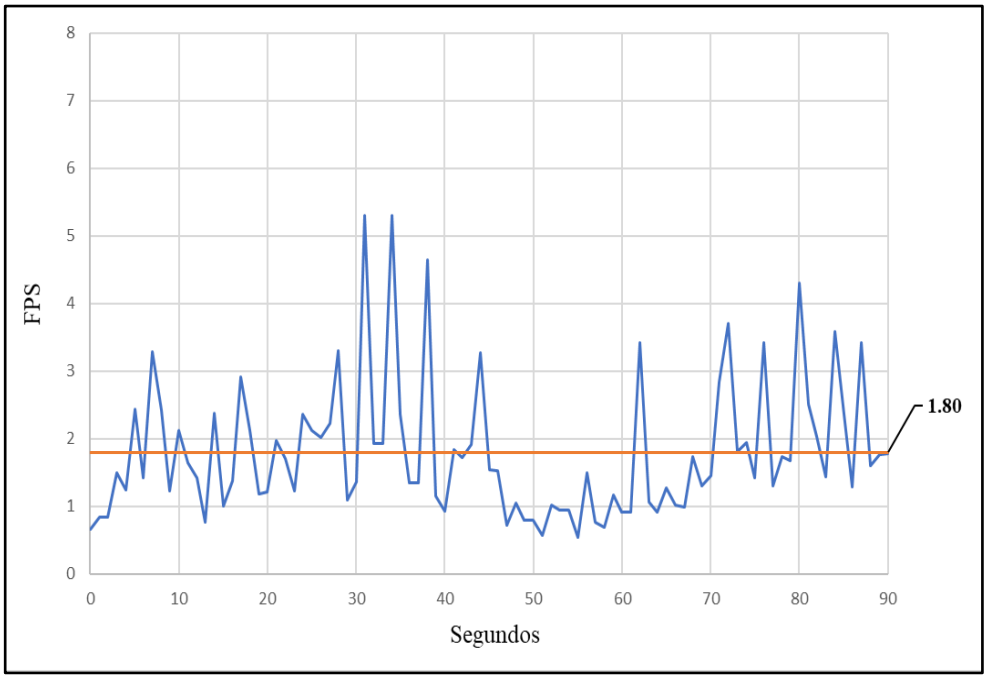

Figura N 50: FPS obtenidos con Faster RCNN y con umbral de 0.35 - 27 espacios
Fuente: Elaboración propia

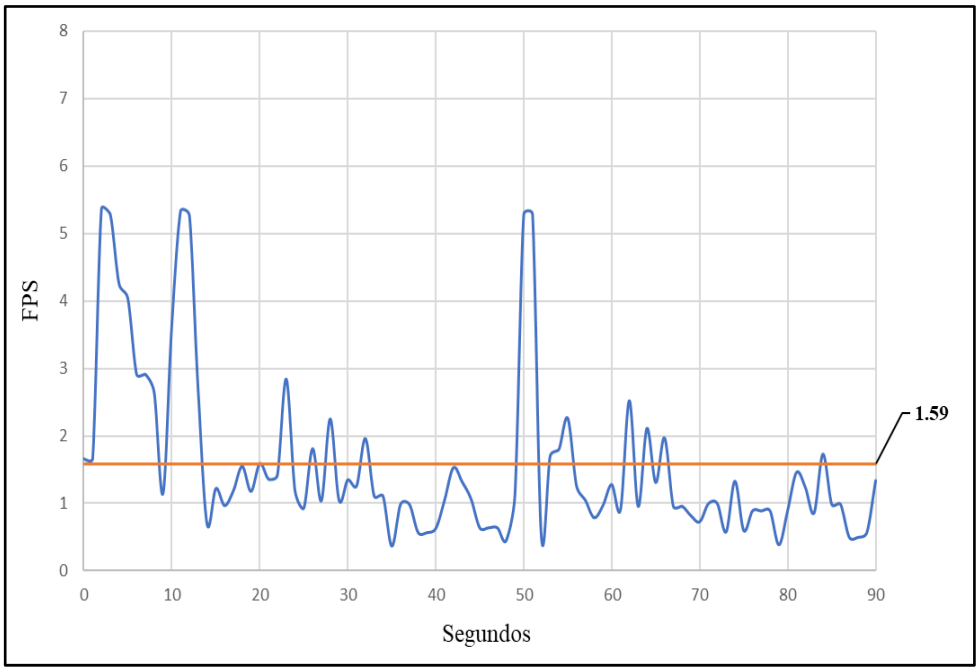

Figura N 51: FPS obtenidos con Faster RCNN y con umbral de 0.55 - 27 espacios
Fuente: Elaboración propia

4.4.2. Resultados de la prueba de precisión y estabilidad del contador y de las redes neuronales convolucionales con 27 espacios de estacionamiento

Para esta prueba se extrajeron el número de espacios disponibles mostrados en la GUI, en cada segundo durante una grabación de 90 segundos en total, con la intención de observar la variación del contador en el tiempo con respecto al número de espacios disponibles reales y determinar su estabilidad, así como ver la cercanía de los valores del contador al número real de espacios disponibles para determinar su precisión.

En la figura 52 se aprecia la cantidad de espacios disponibles para la red neuronal YOLO v5 con umbrales de 0.35 (línea azul) y 0.55 (línea verde), además del número de espacios reales (línea naranja), cuya cantidad era 2 en el momento de la implementación. Así mismo, en la figura 53, se muestra la gráfica de la cantidad de espacios disponibles para la red Faster R-CNN con umbrales de 0.35 (línea azul) y 0.55 (línea verde), y la cantidad de espacios reales disponibles cuya cantidad era 1 en el momento de la prueba.

Las redes son más precisas cuanto más cerca estén los valores de la línea naranja, y más estable cuantos menos picos presentan las líneas azules y verdes. Comparando las figuras 52 y 53, resultó que red neuronal Faster R-CNN era más precisa, puesto que obtuvo valores más cercanos a la línea

naranja en comparación a la red neuronal YOLO v5; sin embargo, ambas redes presentaron menor precisión cuando se aumentaba el umbral de 0.35 a 0.55, pero a la vez mostraban mayor estabilidad, más precisamente durante los segundos 43 a 65.

De nuevo, la red Faster RCNN obtuvo valores negativos debido a falsos positivos, es decir, detectaba vehículos donde no los había o detectaba un vehículo 2 veces, como se muestra en la figura 54, en el rectángulo rojo grande del lado derecho que encerró 2 vehículos, afectando la precisión del contador. Adicionalmente, se calculó el porcentaje de error en cada segundo durante una duración de 90 segundos, tal y como se muestra en la figura 55 y luego se calculó su promedio. Este procedimiento se realizó para calcular el porcentaje de error de la red neuronal YOLO v5 con umbrales de 0.35 y 0.55, y para la red neuronal Faster RCNN con umbrales de 0.35 y 0.55 como se muestra en las figuras 55, 56, 57 y 58 respectivamente. También, en las figuras 59, 60, 61 y 62 se observan la mínima y máxima cantidad de números de autos contados por parte de la red neuronal YOLO v5 con umbrales de 0.35 y 0.55, y las figuras 63, 64, 65 y 66 por parte de la red neuronal Faster RCNN con umbrales de 0.35 y 0.55 respectivamente.

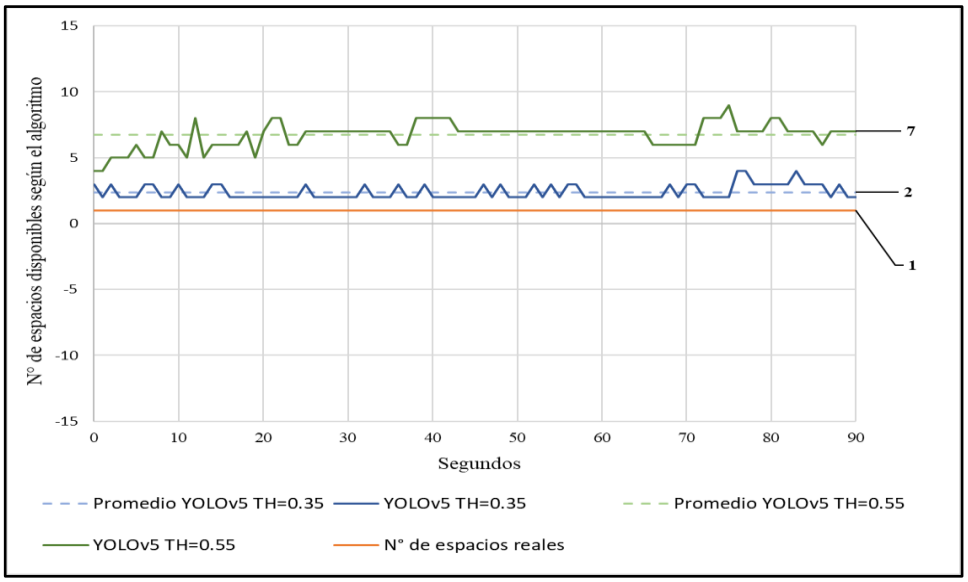

Figura N 52: Número de espacios disponibles con YOLO v5 con 27 espacios
Fuente: Elaboración propia

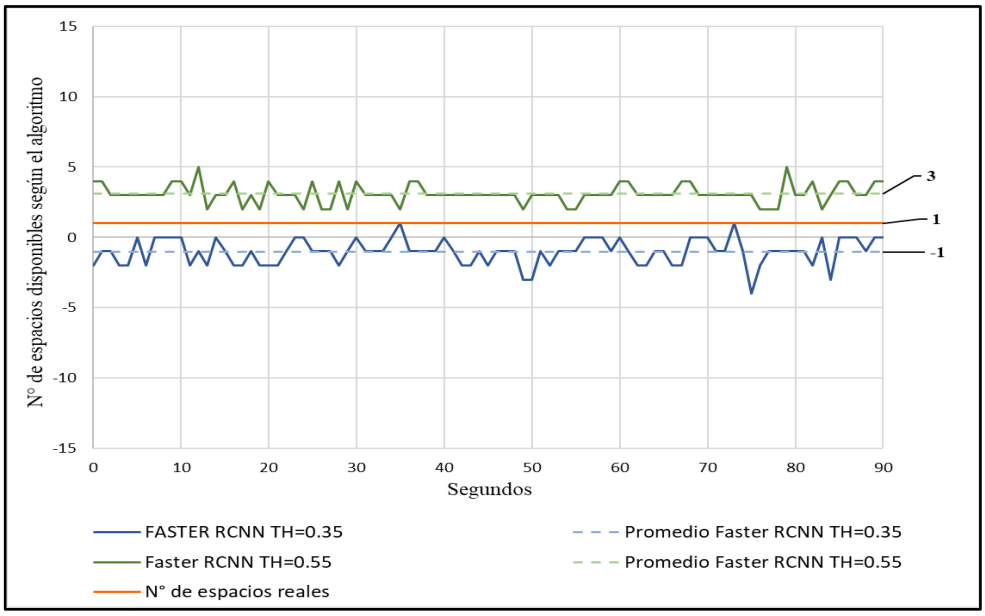

Figura N 53: Número de espacios disponibles con Faster RCNN con 27 espacios
Fuente: Elaboración propia

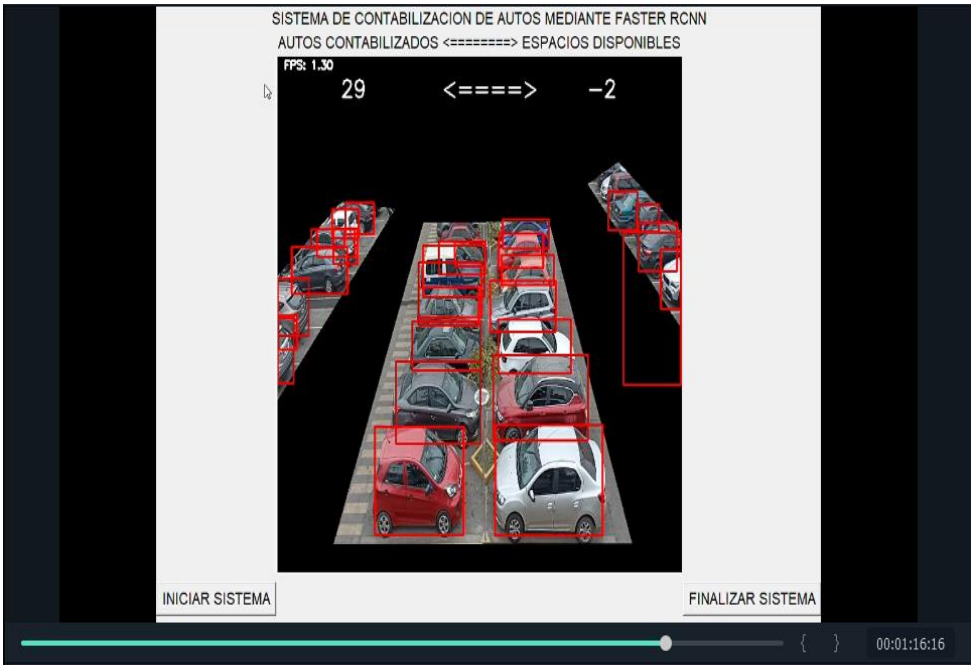

Figura N 54: Caso de falso positivo en la red neuronal Faster RCNN con umbral de 0.35 y 27 espacios
Fuente: Elaboración propia

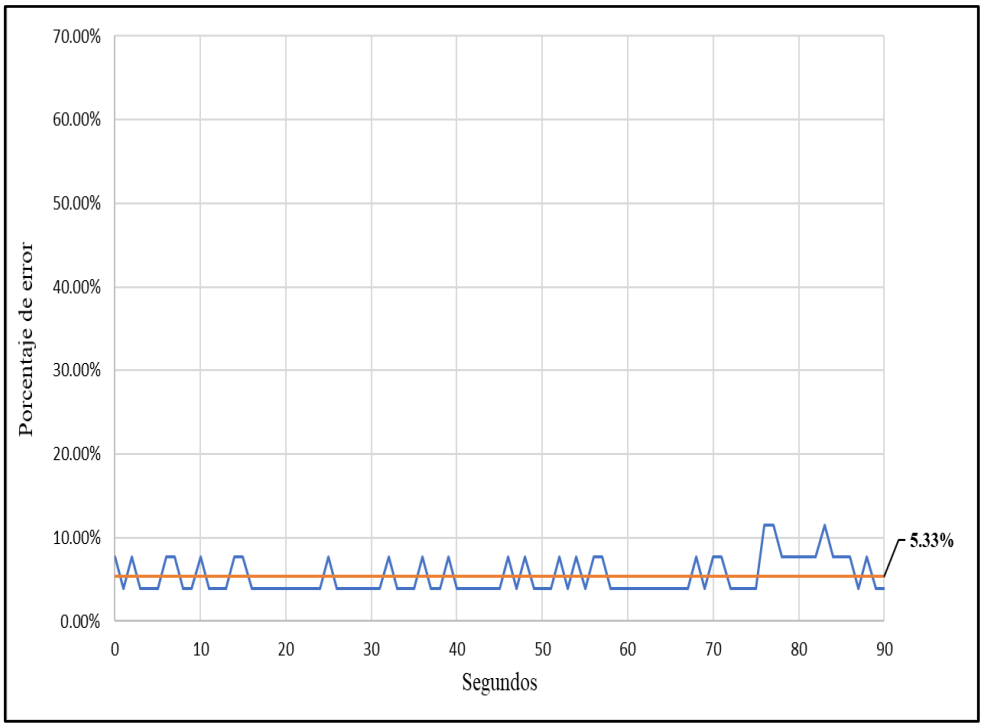

Figura N 55: Porcentaje de error YOLO v5 con umbral de 0.35 y con 27 espacios
Fuente: Elaboración propia

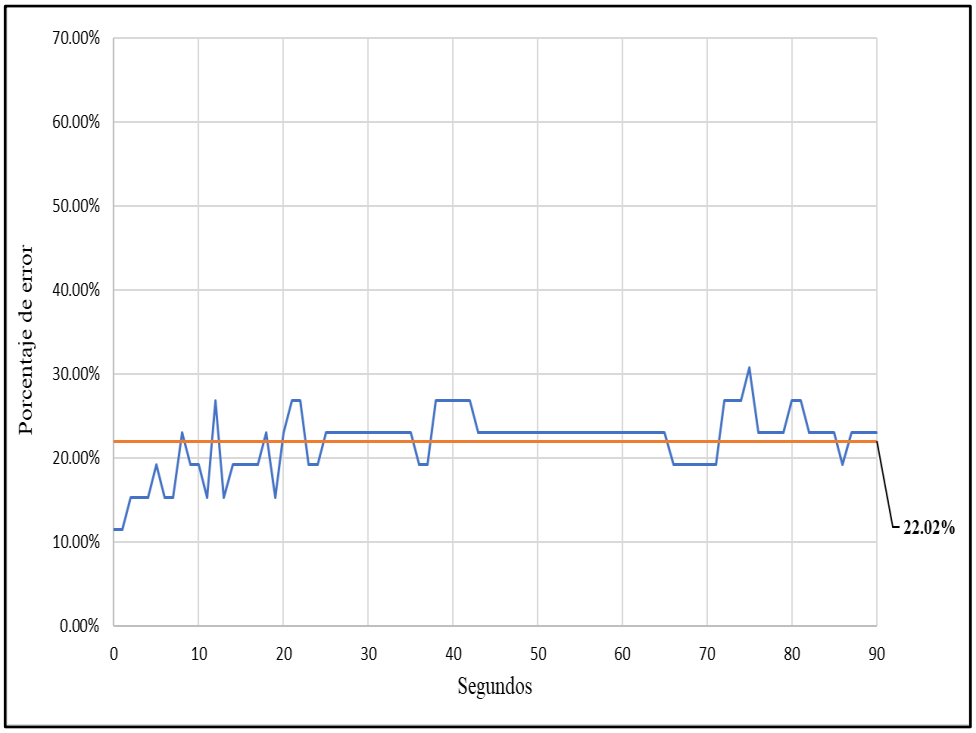

Figura N 56: Porcentaje de error YOLO v5 con umbral de 0.55 y con 27 espacios
Fuente: Elaboración propia

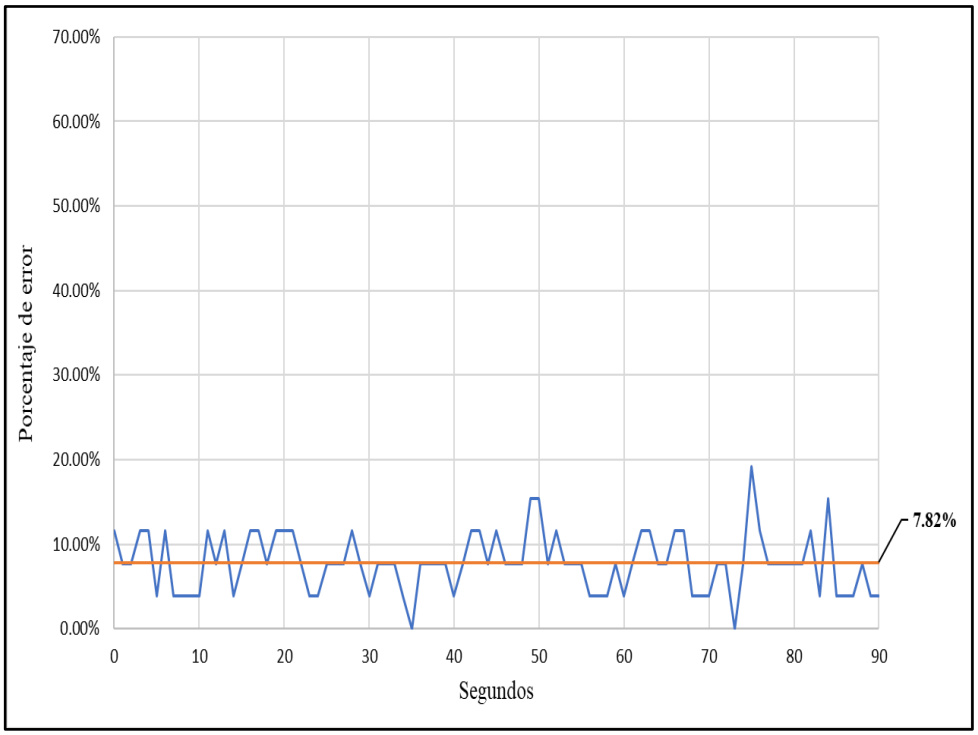

Figura N 57: Porcentaje de error Faster RCNN con umbral de 0.35 y con 27 espacios
Fuente: Elaboración propia

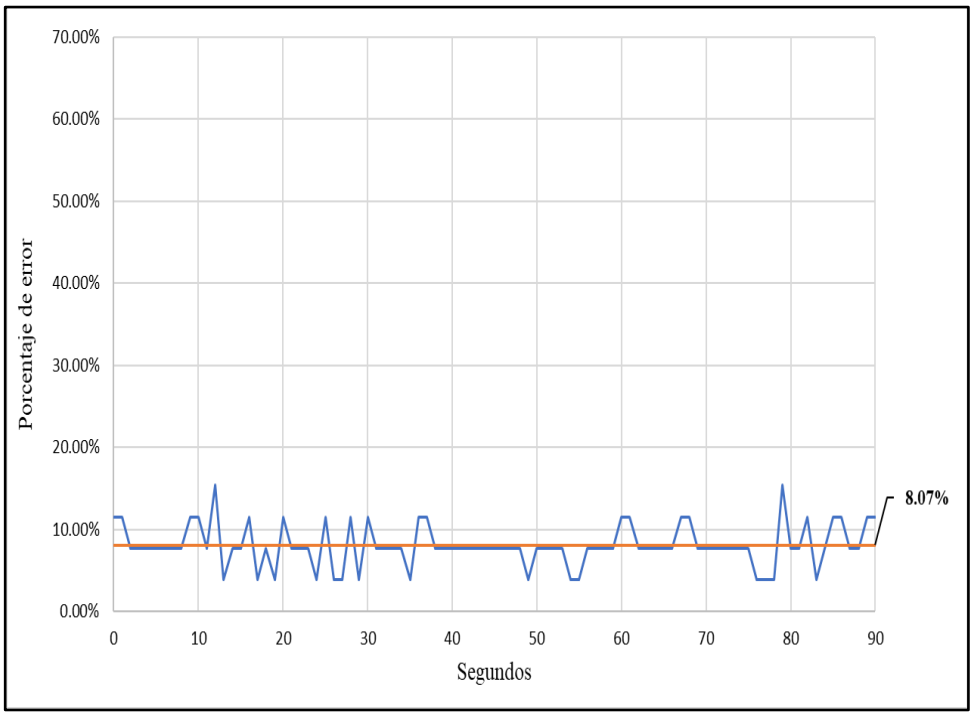

Figura N 58: Porcentaje de error Faster RCNN con umbral de 0.55 y con 27 espacios
Fuente: Elaboración propia

Figura N 59: Cantidad mínima de autos contados por YOLO v5 con umbral de 0.35 y con 27 espacios
Fuente: Elaboración propia

Figura N 60: Cantidad máxima de autos contados por YOLO v5 con umbral de 0.35 y con 27 espacios
Fuente: Elaboración propia

Figura N 61: Cantidad mínima de autos contados por YOLO v5 con umbral de 0.55 y con 27 espacios
Fuente: Elaboración propia

Figura N 62: Cantidad máxima de autos contados por YOLO v5 con umbral de 0.55 y con 27 espacios
Fuente: Elaboración propia

Figura N 63: Cantidad mínima de autos contados por Faster RCNN con umbral de 0.35 y con 27 espacios
Fuente: Elaboración propia

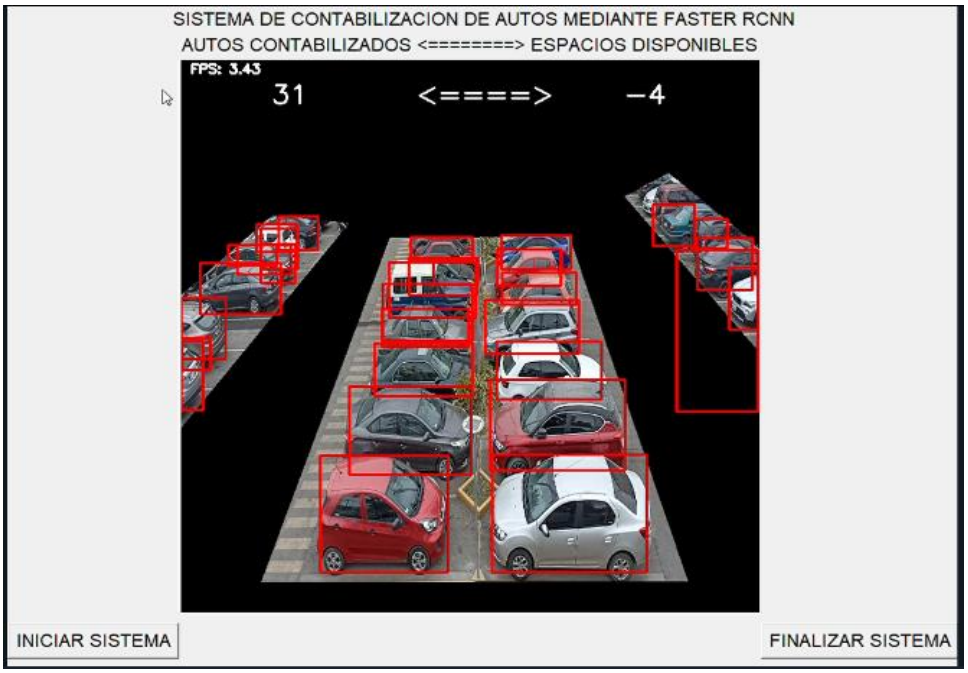

Figura N 64: Cantidad máxima de autos contados por Faster RCNN con umbral de 0.35 y con 27 espacios
Fuente: Elaboración propia

Figura N 65: Cantidad mínima de autos contados por Faster RCNN con umbral de 0.55 y con 27 espacios
Fuente: Elaboración propia

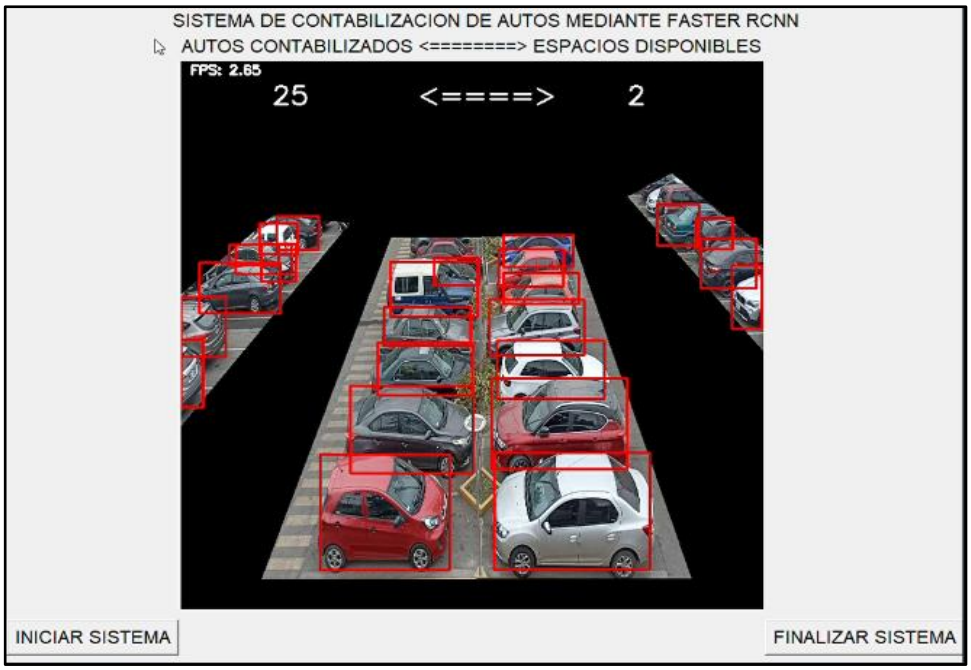

Figura N 66: Cantidad máxima de autos contados por Faster RCNN con umbral de 0.55 y con 27 espacios
Fuente: Elaboración propia

4.5. Comparación de las redes neuronales convolucionales YOLO v5 y Faster RCNN y elección de red para el sistema

En la tabla 5, se presenta toda la información que ha sido recopilada de las redes neuronales con la finalidad de tener todas las métricas resumidas y establecer una conclusión en cuanto que red neuronal presentó mejor rendimiento para el problema

que se intenta resolver en el proyecto. En la tabla 6, se muestra una comparativa en cuanto que red neuronal presenta el más alto valor, indicado en la tabla como un check, en las métricas de mAP, FPS, porcentaje de error promedio y menor cantidad de falsos positivos, teniendo en cuenta toda la información expuesta en el presente capítulo el cual se elige a YOLO v5 como red escogida.

Tabla Nº 5

Tabla resumen de las redes neuronales YOLO v5 y Faster RCNN

		56 espacios		27 espacios	
		Umbral a 0.35	Umbral a 0.55	Umbral a 0.35	Umbral a 0.55
YOLO v5 mAP@0.5 = 0.81615	FPS (promedio)	1.33	1.44	1.86	1.26
	Contador (promedio)	12	30	2	7
	Contador (máx.)	15	33	4	9
	Contador (mín.)	9	27	2	4
	% Error Promedio	19.39%	52.67%	5.33%	22.02%
Faster RCNN mAP@0.5 = 0.81595	FPS (promedio)	1.31	1.17	1.80	1.59
	Contador (promedio)	-2	9	-1	3
	Contador (máx.)	2	13	1	5
	Contador (mín.)	-7	5	-4	2
	% Error Promedio	5.07%	14.15%	7.82%	8.07%

Fuente: Elaboración propia

Tabla Nº 6

Tabla comparativa de las redes neuronales YOLO v5 y Faster RCNN

	YOLO v5	*Faster RCNN*
mAP@0.5	✔	
Cantidad de FPS	✔	
% Error promedio		✔
Menor cantidad de falsos positivos	✔	

Fuente: Elaboración propia

CONCLUSIONES

1. Se implementaron las redes neuronales artificiales convolucionales YOLO v5 y Faster RCNN y se realizó el transfer learning a ambas redes, por los métodos de congelamiento de capas y fine tuning respectivamente, aplicando un dataset de un total de 460 imágenes visualizadas en las figuras 16, 17, 18 y 19, para el cual en base a sus resultados en precisión, cantidad de FPS, porcentaje de error promedio y menor cantidad de falsos positivos tanto en umbrales de detección del 35% y 55% en 27 espacios y 56 espacios visualizados en la tabla 6, se eligió a YOLO v5 como red principal para la detección de automóviles teniendo como promedio de porcentaje de error en detecciones del 5.33%.

2. Se aplicaron los sistemas de limitación para conteo de vehículos por líneas de interés y región de interés; además se probaron en tiempo real por el cual debido a la morfología de los estacionamientos escogidos y tasa de error menor en las detecciones de los espacios comparados en las figuras 24 y 27 respectivamente, por lo cual se escogió el método de selección por región de interés delimitando las secciones a contabilizar mediante procesamiento de imágenes.

3. Se logró implementar una interfaz gráfica de usuario para ambas redes neuronales artificiales convolucionales, el cual tiene la funcionalidad de inicializar la cámara y la red neuronal visualizando los resultados de la contabilización de espacios de estacionamiento; asimismo contó con un botón para finalizar la ejecución del sistema de detección y conteo de vehículos, tal como se puede apreciar en la figura 18.

RECOMENDACIONES

1. Optar por una cámara de 180 grados para exteriores para ampliar el número de detecciones en el estacionamiento, así como condiciones de uso como en el caso de lluvias.

2. Optar por una red de internet dedicada para la aplicación del sistema de detección con un mínimo de 10 Mbps de velocidad simétrica con el fin de evitar retardos en el envío de información entre el teléfono y la laptop, o en todo caso contar con una conexión de internet por cable

3. Tener la posibilidad de disponer de un amplio dataset conformado por automóviles que hacen uso del estacionamiento de la universidad, también aplicar técnicas de balanceo de datos y entrenar completamente la red neuronal desde cero, para mejorar el porcentaje de detecciones.

4. Para trabajos futuros es recomendable utilizar la versión GPU de OpenCV para elevar la tasa de FPS y poder aplicar filtros para la visualización en espacios con poca iluminación o situaciones nocturnas y poder utilizar múltiples cámaras a la vez.

REFERENCIAS BIBLIOGRÁFICAS

Bengio, Y., Courville, A.,& Vincent, P. (2014). Representation Learning: A Review and New Perspectives. *IEEE Trans. PAMI, special issue Learning Deep Architectures*, vol (35), 1798-1828. doi: 10.1109/TPAMI.2013.50.

Cayllahua, N y Suárez, J. (2019). *Redes neuronales de aprendizaje profundo para el reconocimiento facial y control de acceso de estudiantes a un laboratorio* (Tesis de Titulación). Universidad Ricardo Palma, Lima, Perú.

Chirinos, X. y Calero, P. (2021). *Detección del uso correcto de mascarillas utilizando una red neuronal convolucional para el ingreso de personas a un laboratorio de una universidad* (Tesis de titulación). Universidad Ricardo Palma, Lima, Perú.

Deepa, R., Tamiselvan, E., Abrar, E. & Shrinivas, S. (2016). Comparison of Yolo, SSD, Faster RCNN for Real Time Tennis Ball Tracking for Action Decision Networks. *2019 International Conference on Advances in Computing and Communication Engineering (ICACCE), 2019*, 1-4. doi: 10.1109/ICACCE46606.2019.9079965.

Gelvez, J. (2019). Redes neuronales convolucionales y redes neuronales recurrentes en la transcripción automática. *ResearchGate*, 2. doi: 10.13140/RG.2.2.10855.39843

Jokela, J. (2018). *Person counter using real-time object detection and a small neural network* (Tesis de pregrado). Turku University of Applied Sciences, Finlandia.

Luna, A. y Rodríguez, N. (2017). *Detección y conteo de personas en espacios cerrados utilizando estrategias basadas en visión artificial* (Tesis). Pontificia Universidad Javeriana, Bogotá D.C., Colombia.

MathWorks (2022). *Procesamiento de imágenes y visión artificial*. https://la.mathworks.com/solutions/image-video-processing/object-recognition.html

MathWorks (2022). *Redes neuronales convolucionales*. https://la.mathworks.com/discovery/convolutional-neural-network-matlab.html

Narciso, W. y Manzano, E. (2021). Sistema de visión artificial basado en redes neuronales convolucionales para la selección de arándanos según estándares de exportación. *Universidad San Martín de Porres*, 1-12.

Ramírez, B. y Tito, M. (2020). *Reconocimiento automático de placas de rodaje utilizando una red neuronal convolucional para el ingreso de vehículos en la Universidad Ricardo Palma* (Tesis de titulación). Universidad Ricardo Palma, Lima, Perú.

Rios, P. (2022). *Diseño y entrenamiento de una red neuronal empleando procesamiento de imágenes para diferenciar granos de trigo respecto a malezas* (Tesis de Titulación). Universidad Tecnológica del Perú, Lima, Perú.

Sarda, A., Dixit S., & Bhan A. (2021). Object Detection for Autonomous Driving using YOLO [You Only Look Once] algorithm. *Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)*, 1370-1374. doi: 10.1109/ICICV50876.2021.9388577

Tan, L., Huangfu, T., Wu, L. & Chen, W. (2021). Comparison of YOLO v3, Faster R-CNN, and SSD for Real-Time Pill Identification. *BMC Medical Informatics and Decision Making, 2021, 21* (324). doi: 10.1186/s12911-021-01691-8.

Wu, T, Wang, T., & Liu, Y. (2021). Real-Time Vehicle and Distance Detection Based on Improved Yolo v5 Network. *2021 3rd World Symposium on Artificial Intelligence (WSAI), 24-28.* doi: 10.1109/WSAI51899.2021.9486316

ANEXO 1

Coordenadas de las líneas de interés en Python utilizando la librería OpenCV y la función cv2.line()

```
#Líneas de Interés
frame = cv2.line(frame, (149,382), (0,513), (255, 0, 0), 2)
frame = cv2.line(frame, (206,382), (0,600), (255, 0, 0), 2)
frame = cv2.line(frame, (257,373), (130,638), (255, 0, 0), 2)
frame = cv2.line(frame, (340,373), (456,638), (255, 0, 0), 2)
frame = cv2.line(frame, (375,373), (638,633), (255, 0, 0), 2)
frame = cv2.line(frame, (419,373), (639,514), (255, 0, 0), 2)
```

Matriz que contiene las coordenadas de los vértices que definen las regiones de interés en Python utilizando la librería OpenCV y la función cv2.polyline()

```
#Matriz de vértices de las ROI
contours3 = np.array([[149,379], [206,379], [0,600], [0,513]], np.int32)
contours2 = np.array([[257,373], [340,373], [456,638], [130,638]], np.int32)
contours1 = np.array([[375,373], [419,373], [639,514], [638,633]], np.int32)

#Extracción de las ROI
mask = np.zeros(frame.shape, dtype=np.uint8)
cv2.fillPoly(mask, pts=[contours1], color=(255, 255, 255))
cv2.fillPoly(mask, pts=[contours2], color=(255, 255, 255))
cv2.fillPoly(mask, pts=[contours3], color=(255, 255, 255))
frame = cv2.bitwise_and(frame, mask)
```