UNIVERSIDAD CATÓLICA DE CUENCA

UNIVERSIDAD CATÓLICA DE CUENCA

Comunidad Educativa al Servicio del Pueblo

UNIDAD ACADÉMICA DE INFORMÁTICA, CIENCIAS DE LA COMPUTACIÓN E INNOVACIÓN TECNOLÓGICA

CARRERA DE INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN

DESARROLLO DE UN PROTOTIPO PARA AUTOMATIZAR EL REGISTRO DE ACCESO VEHICULAR MEDIANTE RECONOCIMIENTO DE PLACAS UTILIZANDO REDES NEURONALES.

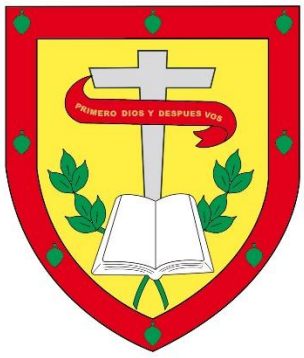
PROYECTO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN TECNOLOGÍAS DE LA INFORMACIÓN

AUTOR: JORDY GILMAR LUZURIAGA SANCHEZ

DIRECTOR: ING. LUIS STALIN JARA OBREGÓN

LA TRONCAL - ECUADOR
2025

DIOS, PATRIA, CULTURA Y DESARROLLO



UNIVERSIDAD CATÓLICA DE CUENCA

Comunidad Educativa al Servicio del Pueblo

UNIDAD ACADÉMICA DE INFORMÁTICA, CIENCIAS DE LA COMPUTACIÓN E INNOVACIÓN TECNOLÓGICA

CARRERA DE INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN

DESARROLLO DE UN PROTOTIPO PARA AUTOMATIZAR EL REGISTRO DE ACCESO VEHICULAR MEDIANTE RECONOCIMIENTO DE PLACAS UTILIZANDO REDES NEURONALES.

PROYECTO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN TECNOLOGÍAS DE LA INFORMACIÓN

AUTOR: JORDY GILMAR LUZURIAGA SANCHEZ

DIRECTOR: ING. LUIS STALIN JARA OBREGÓN

LA TRONCAL - ECUADOR
2025

DIOS, PATRIA, CULTURA Y DESARROLLO

Universidad Católica de Cuenca

UNIDAD ACADÉMICA DE INFORMÁTICA, CIENCIAS DE LA COMPUTACIÓN E INNOVACIÓN TECNOLÓGICA

CARRERA DE INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN

UNIDAD DE TITULACIÓN

La Troncal, 24 de oct. de 25

Sección: U.A. de Informática, Ciencias de la Computación e Innovación Tecnológica
Asunto: Certificación y aprobación de presentación del Trabajo de Titulación

Señor Ingeniero
Marcos Orellana Parra. PhD
Responsable de la Unidad de Titulación
Ingeniería en Tecnologías de la Información

De mi consideración.

Reciba un cordial saludo y mis mejores deseos de éxito en sus funciones.

El suscrito, en calidad de tutor del trabajo de titulación, certifica que el trabajo titulado: "Desarrollo de un prototipo para automatizar el registro de acceso vehicular mediante reconocimiento de placas utilizando redes neuronales", desarrollado por el estudiante Jordy Gilmar Luzuriaga Sánchez, con numero de cedula 0930804760, ha sido guiado y revisado de manera periódica, cumpliendo con las normativas estatutarias establecidas por la Universidad Católica de Cuenca.

Particular que pongo en su conocimiento para los fines legales consiguientes. Sin otro particular me suscribo de Usted.

Atentamente,

Luis Stalin Jara Obregon
Time Stamping Security Data

Ing. Luis Stalin Jara Obregón, Mgtr
DOCENTE TUTOR

Declaratoria de Autoría y Responsabilidad

Jordy Gilmar Luzuriaga Sánchez portador(a) de la cédula de ciudadanía Nº **0930804760**. Declaro ser el autor de la obra: "**Desarrollo de un prototipo para automatizar el registro de acceso vehicular mediante reconocimiento de placas utilizando redes neuronales**", sobre la cual me hago responsable sobre las opiniones, versiones e ideas expresadas. Declaro que la misma ha sido elaborada respetando los derechos de propiedad intelectual de terceros y eximo a la Universidad Católica de Cuenca sobre cualquier reclamación que pudiera existir al respecto. Declaro finalmente que mi obra ha sido realizada cumpliendo con todos los requisitos legales, éticos y bioéticos de investigación, que la misma no incumple con la normativa nacional e internacional en el área específica de investigación, sobre la que también me responsabilizo y eximo a la Universidad Católica de Cuenca de toda reclamación al respecto.

La Troncal, **24 de octubre de 2025**

F: .....
Jordy Gilmar Luzuriaga Sánchez
C.I. 0930804760

Agradecimiento

Extiendo mi profundo agradecimiento a la Universidad Católica de Cuenca por ser el cimiento de mi formación y por proveer los recursos necesarios para esta investigación.

Mi gratitud especial al Ing. Stalin Jara, mi tutor de tesis. Su dirección experta, rigor metodológico y paciencia fueron vitales para superar los desafíos de este proyecto. Aprecio sinceramente su tiempo y guía profesional.

A todos los profesores, por la calidad de su enseñanza y por compartir su valioso conocimiento.

Finalmente, a mis amigos y compañeros.

Dedicatoria

A mi amada esposa, Anais Vera mi apoyo incondicional y mi roca. Gracias por tu paciencia infinita y por la inmensa alegría que me das, especialmente ahora, mientras llevas en tu vientre a nuestro mayor tesoro.

A mi futura hija, la vida que crece y mi más dulce motor. Eres la promesa que me impulsó a terminar este camino con dedicación y esperanza. Que este logro sea un inicio de la vida de esfuerzo y amor que te ofrezco.

A mis padres, Cesar Luzuriaga y Suheyll Sánchez por su amor, sus incontables sacrificios y por ser el pilar inquebrantable de mi vida. Su legado de perseverancia es mi guía.

Finalmente, a mis fieles compañeros que ya no están, Manchas y Fernanda, por sus ronroneos terapéuticos y por hacer las largas madrugadas mucho más llevaderas.

Resumen

Uno de los desafíos en los edificios residenciales y en los locales comerciales, es la correcta gestión del registro de los vehículos, que por lo general se realiza de forma manual. Esto causa errores en la correcta toma de las placas vehiculares. Para poder enfrentar este problema, se propone el desarrollo de un prototipo que automatice los registro, de entrada y salida, vehiculares por medio del reconocimiento de placas y caracteres vehiculares a través de redes neuronales y visión computarizada. Este sistema permitirá la captura de imágenes en tiempo real, identificando y procesando los caracteres de las placas para registrar cada acceso en una base de datos centralizada reduciendo los errores y mejorando la seguridad. Él presente proyecto utiliza un enfoque cuantitativo y descriptivo, validando el prototipo por medio de métricas de rendimiento como lo son precisión, exhaustividad y mAP. Se entrenó un modelo de detección de placas vehiculares, obteniendo un rendimiento del 94% en la identificación de placas vehiculares, aunque se identificó un 6% de falsos negativos.

Palabras clave: Visión Computarizada, Reconocimiento de Caracteres Vehiculares, Redes Neuronales, Python, Métricas de Rendimiento.

Abstract

One of the challenges in residential buildings and commercial stores is the correct management of vehicle registration, which is usually done manually. This causes errors in the correct recording of license plates. To address this problem, it is proposed the development of a prototype that automates vehicle entry and exit registration through license plate and character recognition using neural networks and computer vision. This system will allow real-time image capture, identifying and processing license plate characters to record each access in a centralized database, reducing errors and improving security. This project uses a quantitative and descriptive approach, validating the prototype through performance metrics such as accuracy, completeness, and mAP. A license plate detection model was trained, achieving 94% performance in license plate identification, although 6% false negatives were identified.

K*eywords:* Computer Vision, Vehicle Character Recognition, Neural Networks, Python, Performance Metrics.

TABLA DE CONTENIDO

INDICE DE TABLAS

INDICE DE GRÁFICOS

Introducción

Actualmente los locales comerciales y edificios residenciales enfrentan un problema común en el control del acceso de vehículos, que se gestiona de forma manual mediante formularios escritos.

Este método siendo ineficaz, provoca errores en la recolección de datos. Fallos en la toma de los caracteres o incluso no registrar ciertos vehículos. Para esto se propone la elaboración de un prototipo que facilite la automatización del proceso de registro vehicular mediante el uso de redes neuronales y técnicas de visión computarizada para identificar los caracteres y las placas de los vehículos como solución a este problema.

El objetivo esta investigación es la creación de un prototipo de sistema automático que registre las placas de los vehículos en tiempo real, mediante identificación y reconocimiento utilizando redes neuronales, asegurando así un proceso eficiente y preciso. El propósito es reducir al mínimo los errores de registro y robustecer la seguridad, así como los procedimientos de auditoría.

Para alcanzar estos objetivos, el presente trabajo estará estructurado en los siguientes capítulos:

Capítulo 1: Marco Referencial. Se establecerá el contexto de la investigación, abordando el Planteamiento del problema y su Formulación, seguido de la revisión de los Antecedentes de la investigación. Se presentará la Justificación de la propuesta, la definición de los Objetivos y se establecerán las Limitaciones y Delimitaciones del estudio.

Capítulo 2: Marco Teórico. Se desarrollarán los Fundamentos de la Convolución en Redes Neuronales, incluyendo la Definición Matemática y Conceptual de la Convolución 2D, el concepto del Filtro (Kernel) como Detector de Patrones, los Parámetros de la Operación (Stride y Padding) y la función esencial de la Función de Activación (ReLU).

Capítulo 3: Marco Metodológico. Se detallarán los métodos de investigación utilizados, definiendo el Enfoque y el Nivel de la investigación. Se describirán la Población y muestra, los Métodos de investigación, las Técnicas e instrumentos de recolección de datos, y el Tratamiento de la información que se aplicará.

Capítulo 4: Resultados y Análisis. Se presentarán los hallazgos empíricos y la arquitectura del prototipo. Incluirá los Detalles del entrenamiento y conjunto de datos, la evaluación de las Métricas de rendimiento, el Análisis e interpretación de los resultados y la Comparativa y justificación del enfoque tecnológico. Además, se detallará la Arquitectura del Sistema y Flujo de Datos (Componentes del Sistema, Flujo Principal y Requisitos de Despliegue), finalizando con un Desglose del Código y su Interconexión.

Capítulo 5: Conclusiones y Recomendaciones. Esta sección final condensará el trabajo, presentando las Conclusiones obtenidas. Se incluirá un Análisis de Viabilidad Económica, un Plan de Implementación Sugerido y se culminará con las Recomendaciones y futuras mejoras para el sistema.

1. Marco Referencial

1.1. Planteamiento del problema

Uno de los problemas que tienen los locales comerciales o edificios residenciales es el correcto manejo del registro, entrada y salida, vehicular el cual se maneja de forma manual, con formularios escritos por personal de seguridad.

Surge la necesidad de crear un prototipo que utilizando técnicas de visión computarizada y redes neuronales para el reconocimiento de placas vehiculares y sus caracteres. Con este enfoque las tareas, las cuales son implementadas manualmente serán remplazadas por un sistema automático de captura de imágenes en tiempo real, el cual identifica y procesa los caracteres de la placa y registra cada acceso y salida en una base de datos centralizada. Lo cual ayudara a disminuir los errores en los

registros ayudando a fortalecer los mecanismos de auditorias

1.2. Formulación del Problema

¿Cómo desarrollar un prototipo de sistema automatizado que permita registrar en tiempo real el acceso vehicular mediante la detección y reconocimiento de placas, utilizando redes neuronales, garantizando precisión y eficiencia en el proceso?

1.3. Antecedentes de la investigación

Reconocer de forma automática las placas vehiculares es clave en el procesamiento de imágenes, dado a los avances en redes de aprendizaje profundas. La evidencia de estos trabajos demuestra la robustes de las técnicas seleccionadas, permitiendo obtener resultados precisos incluso en condiciones adversas del entorno.

El caso de Zhang et al. (2024) quienes hicieron una comparación de modelos de aprendizaje profundos, utilizaron la arquitectura YOLOv5-PDLPR para identificar placas vehiculares en entornos no controlados, demostrando que sus redes convolucionales hibridas funcionan a pesar de los problemas [1].

Serrttaş y Gül (2025) pudieron desarrollar un método automático el cual utilizando el detector de YOLO lograron identificar registros vehiculares. Tenemos a los investigadores Armijos Ortega et al. (2024) en America Latina los cuales estudiaron varios modelos de CNN (Convolutional neural network) para identificar placas de automóviles en condiciones no favorables [2].

En 2022 Batat et al, diseñaron un sistema automatizado que, utilizando YOLO para la detección de vehículos y placas, el cual muestra una gran precisión en tareas reales [3]. También el estudio de Qin y Liu (2020) los cuales propusieron un método basado en aprendizaje profundo para identificar placas vehiculares en situaciones en las cuales los caracteres de la placa vehicular están poco claros y se dificulta su observación y posterior registro [4]. En (2020) Lee et al. Los cuales aplicaron métodos de aprendizaje profundo y mejoras de resolución de imágenes implementando redes generativas para la segmentación en los sistemas de vigilancia, logrando optimizar el reconocimiento en tiempo real [5].

Para las técnicas ágiles para desarrollar software varias investigaciones recientes muestran como estas metodologías ayuda en la colaboración en equipos distribuidos permitiendo entregas de manera continuas. Por ejemplo, Dingsøyr et al. (2021) analizaron el uso de Scrum y otras prácticas ágiles en empresas a gran escala, encontrando mejoras significativas en la coordinación y la calidad del producto [6].

1.4. Justificación de la investigación

Para las empresas y edificios residenciales se ha vuelto una necesidad controlar de forma automática la entrada y salida de los vehículos. Esto debido al creciente número de vehículos en las ciudades. Actualmente un gran número de estas tareas se realizan de forma manual o con tecnologías antiguas, causando errores en el registro. Por esta necesidad, resulta indispensable tener una herramienta tecnológica que permita automatizar el proceso de registro de placas vehiculares. Por lo cual se sugiere diseñar un prototipo que, utilizando redes neuronales, se capaz de detectar y reconocer las placas vehiculares y sus caracteres.

Este prototipo hará posible que cada placa de vehículo que acceda a un espacio específico, se registre sin intervención humana.

Utilizando una red neuronal artificial, en conjunto con el procesamiento de imágenes, adecua al sistema la capacidad de funcionar en diversas condiciones ambientales como cambios en la iluminación o el clima. Esto lo convierte en una alternativa moderna, escalable y adaptable frente a los métodos tradicionales

En este contexto, el proyecto no solo responde a una problemática actual en control vehicular, sino que también contribuye al avance tecnológico mediante la aplicación de herramientas innovadoras.

1.5. Objetivos

1.5.1. Objetivos generales

Desarrollar un prototipo de sistema automatizado que registre el acceso de vehículos en tiempo real mediante la detección y reconocimiento de placas vehiculares con redes neuronales.

1.5.2. Objetivo especifico

Recopilar un conjunto de imágenes de registros vehiculares en diferentes condiciones de iluminación y ángulo.

Diseñar y entrenar un modelo de red neuronal convolucional (CNN) para la localización y segmentación de la placa en la imagen.

Implementar un módulo de OCR basado en redes neuronales recurrentes o tipo transformer, para el reconocimiento de caracteres de la placa vehicular.

1.6. Limitaciones

El prototipo será validado únicamente en un entorno controlado, lo cual podría no reflejar todas las posibles condiciones del mundo real (lluvia intensa, suciedad en las placas, vandalismo).

El hardware disponible (cámaras, laptop) puede limitar la resolución y velocidad de procesamiento en comparación con soluciones comerciales de alta gama.

El proyecto no se integra con sistemas de seguridad externos (por ejemplo, alarmas, sistemas de videovigilancia).

1.7. Delimitaciones

El objetivo del proyecto es desarrollar un prototipo capaz de detectar placas vehiculares utilizando redes neuronales. Los caracteres de las placas serán reconocidos utilizando OCR se almacenarán en una base de datos, con el propósito de optimizar los procesos de seguridad.

La meta es especificar cada etapa del ciclo de vida del prototipo: la captura y el preprocesamiento de imágenes, identificación y segmentación de la matrícula, reconocimiento OCR de caracteres.

El prototipo abarca la creación de una interfaz web. La confiabilidad del diseño se respalda en revisión documental, incluyendo artículos científicos, repositorios de código y el uso de OCR. Usando esta información, se implementó un prototipo funcional que integra técnicas de visión por computadora y reconocimiento de caracteres en placas vehiculares.

2. Marco Teórico

El reconocimiento automático de matrículas (LPR, por sus siglas en inglés) es la tarea de reconocer caracteres de identificación en matriculas a partir de imágenes o secuencias de video, y desempeña un papel muy importante en muchas aplicaciones como el control de vehículos, la gestión del tráfico, la seguridad pública, etc. [7]. El sistema emplea visión por computadora de última generación y aprendizaje profundo para lograr un alto nivel de precisión en múltiples escenarios operativos.

Su arquitectura típicamente consta de las siguientes etapas: captura de imágenes en áreas altamente localizadas mediante cámaras, preprocesamiento de imágenes, donde se mejora la calidad de la imagen ajustando el contraste y suprimiendo el ruido, detección caracteres en placas vehiculares mediante algoritmos como el algoritmo YOLO (You Only Look Once) [8], segmentación de caracteres para su procesamiento individual, reconocimiento de caracteres utilizando métodos como OCR (reconocimiento óptico de caracteres), redes neuronales recurrentes o modelos basados en Transformer [9], y almacenamiento en una base de datos para ser utilizados en análisis futuros.

Las aplicaciones de LPR pueden aplicarse ampliamente a estacionamientos inteligentes y acceso automático, peaje automático y control de tráfico, y reconocimiento de vehículos sospechosos para fines de seguridad pública o gestión de entradas en organizaciones gubernamentales, educativas y privadas [9]. Actualmente, se está investigando cómo mejorar la precisión de los sistemas empleando redes neuronales convolucionales (CNN) dentro de un modelo de atención, que ha demostrado buen rendimiento en entornos complicados [10].

No obstante, aún persisten algunos desafíos en la diversidad de diseño de identificación vehicular, problemas de iluminación, ángulo de captura y privacidad, que necesitan ser regulados y asegurados adecuadamente. Sin embargo, todavía existen muchos desafíos como la diversidad de diseños de placas, el problema de la iluminación, el ángulo de captura, así como la privacidad y protección de datos, que deben ser abordados mediante reglas adecuadas y medidas de seguridad [11].

La visión por computadora pone gran énfasis en la adquisición y procesamiento de datos, ya que la cantidad y calidad de los datos determina en gran medida la efectividad del modelo entrenado en el caso de técnicas de aprendizaje profundo. La recopilación de datos se ha convertido ahora en el cuello de botella, debido a que muchos modelos son complejos, habrá una creciente necesidad de datos grandes y diversos (y bien etiquetados) para generalizar con éxito, En contraste, ninguna cantidad de algoritmos sofisticados puede satisfacer la demanda de aplicaciones típicas sin dichos datos [12].

Durante la captura de imágenes de vehículos es necesario tener en cuenta varias características: por ejemplo, la iluminación que puede variar entre día, noche, sombras o placas sucias directamente relacionadas con la visibilidad de los caracteres, por lo que es importante capturar imágenes en condiciones vibrantes para entrenar modelos más robustos. De manera similar, el ángulo y la distancia desde la cual se toman las imágenes afectan el rendimiento del sistema, por lo que capturar esa variabilidad en el conjunto de datos ayuda mucho en el aprendizaje del modelo.

Además, la anotación adecuada significa mucho para el entrenamiento, ya que tiene un efecto directo en los resultados finales. Estos son típicamente cuadros delimitadores, simples y baratos de implementar, se utilizan para abarcar el objeto como las matrículas; máscaras: una forma más precisa de delinear el objeto a nivel de píxel, y etiquetas: asignar atributos a los objetos, por ejemplo, formas, colores y dirección. Estos métodos permiten a los modelos aprender a clasificar e identificar con mucha más precisión.

Para este propósito, se recomiendan las siguientes herramientas de anotación: CVAT (Computer Vision Annotation Tool), un software de código abierto desarrollado por Intel para imágenes y videos [13], LabelImg, una herramienta gráfica para crear anotaciones de cuadros, escrita en Python y Qt 9 [14], Roboflow Annotate, una plataforma basada en la web con herramientas asistidas por IA diseñadas para ahorrar tiempo en tareas de anotación [15].

Estas herramientas y técnicas de preparación permiten suministrar adecuadamente modelos de aprendizaje profundo como las Redes Neuronales Convolucionales (CNNs), que han transformado tareas de visión por computadora como la localización y segmentación de placas de vehículos. Las CNNs están

diseñadas para manejar datos en forma de cuadrícula, que es precisamente el tipo de datos que son las imágenes.

Las CNNs, a diferencia de las redes neuronales profundas totalmente conectadas clásicas, explotan las características espaciales de los datos extrayendo características locales, como bordes, texturas y patrones, de la imagen mediante filtros de convolución que se deslizan a través de la imagen. Basadas libremente en la estructura jerárquica de la corteza visual animal, tales redes aprenden representaciones cada vez más complejas de una imagen a medida que avanzamos por sus capas: las capas inferiores se vuelven sensibles a detalles simples de las imágenes, incluidos bordes y colores específicos, mientras que las capas superiores responden a estructuras más complejas y objetos completos.

Weicheng Kuo, Tzu-Ting Kuo y Keng-Shih Lu, "Esta propiedad de extracción de características hace que las CNNs hayan alcanzado el estado del arte en tareas de procesamiento de imágenes, incluyendo clasificación de imágenes, detección de objetos, segmentación semántica, reconocimiento facial, análisis de escenas y reconocimiento de caracteres [16].

Típicamente, una CNN está compuesta por capas convolucionales que actúan como extractores de características que extraen características locales, funciones de activación como ReLU, que son responsables de añadir no linealidad a la red, capas de agrupamiento para reducir el muestreo y preservar la información de datos importante, y finalmente, capas totalmente conectadas que permiten cualquier tarea final de clasificación o regresión. Todas estas capas trabajando en conjunto construyen una representación jerárquica de imágenes, desde bordes hasta objetos completos [17].

Para el reconocimiento de placas, se emplean CNNs para reconocer y segmentar el área en una imagen donde se clasifica cada instancia de píxel, y para segmentar con precisión las placas de los vehículos, un paso crucial en aplicaciones en entornos no controlados y cambiantes [18]. Existen modelos previamente entrenados basados en CNN especialmente desarrollados para la detección de objetos en tiempo real, como YOLO (You Only Look Once) realiza la inferencia mediante una arquitectura de detección de una sola etapa, lo cual la predicción de las regiones y clasificación de los objetos se ejecutan en simultaneo en una única operación de red,

permitiendo una alta velocidad de procesamiento [19], SSD (Single Shot MultiBox Detector) – equilibra bien la velocidad y la precisión, haciendo una sola etapa para detectar objetos en múltiples escalas [19] y Faster R-CNN – utiliza una estrategia de doble etapa, con esto mejora la precisión, reduciendo la velocidad de salida, este es un buen modelo cuando la alta precisión es obligatoria [20].

El rendimiento de estos modelos puede evaluarse utilizando precisión, recall e IoU, que se define como la proporción del área de intersección entre el rango de salida predicho y el real con respecto al área de unión de ellos, mostrando el rendimiento del modelo de manera inclusiva [21].

Un OCR o reconocimiento óptico de caracteres es una tecnología que permite interpretar, por medio de imágenes, textos manuscritos o mecanografiaos. En 1950 se comenzó a desarrollar el método de plantilla rígida en el cual se basa el este modelo de redes neuronales profundas. En consecuencia, puede reconocer caracteres con precisión incluso cuando la escena no es ideal [22].

El reconocimiento de caracteres en placas vehiculares tiene varios desafíos. Entre estos el ruido en las imágenes causados por reflejos o variables en la iluminación. También tenemos diferencias en las tipografías empleadas según el país; y cambios en la geométrica provocadas por el ángulo o la distancia desde la cual se tomó la imagen. Estos factores son los responsables en las variaciones en el rendimiento del OCR [23].

Los métodos clásicos de OCR, los cuales están basados en segmentación de caracteres y reconocimiento de patrones, son superados por un enfoque de aprendizaje profundo. Esto permite extraer características directamente de la imagen, eliminando la necesidad de preservar los caracteres, lo cual mejora la resistencia del sistema ante rotación y deformaciones [24]. A raíz de esto, tenemos dos soluciones, las redes neuronales recurrente (RNN), en concreto las variantes de memoria a corto y largo plazo (LSTM), que modelan de manera eficiente los caracteres capturados [25].

Contamos con otros OCR basados en Transformers mostraron un rendimiento notable. Un ejemplo de esto son las arquitecturas CRNN (Redes Neuronales Recurrentes Convolucionales) las cuales combinan redes convolucionales para extraer características visuales con RNN para el análisis de secuencias. Modelos más recientes como TrOCR, basados completamente en el modelo Transformer, han

demostrado superar a las arquitecturas híbridas en tareas de reconocimiento de texto en imágenes sin necesidad de segmentar previamente los caracteres [26].

Para la evaluación cuantitativa de este tipo de modelos se utiliza métricas como la precisión y la Tasa de Error de Caracteres (CER). La precisión mide la proporción de caracteres reconocidos correctamente sobre el total, mientras que el CER calcula la distancia de Levenshtein la cual mide la diferencia entre dos cadenas de texto. Un valor bajo de CER indica un buen rendimiento, lo cual es especialmente relevante para tareas con cadenas de texto cortas, como las placas de vehículos [27].

Para un sistema en tiempo real de reconocimiento automático de matrículas (LPR), la serie de operaciones se compone de tomar instantáneas desde lugares estratégicos utilizando cámaras, detectarlas correctamente en estas imágenes (software basado en visión por computadora), pasar la información de los caracteres a las redes neuronales OCR y luego guardar todo en una base de datos. Todo el flujo de este último permite el control automático del acceso de vehículos de manera eficiente, mejorando así la seguridad, así como minimizando la intervención humana en lugares como estacionamientos, cabinas de peaje y puertas que ingresan a regiones prohibidas [28].

Para un buen rendimiento en tiempo real, la elección del hardware es muy importante. Se recomiendan cámaras de alta resolución que funcionen bien bajo todas las condiciones ambientales y de iluminación. Además, es necesario utilizar procesadores potentes, como los integrados en equipos industriales, por ejemplo, el sistema embebido inteligente DFI ES220-CS, con un procesador Intel® Core™ que tiene la capacidad de procesar imágenes a 30 fps y reconocerlas instantáneamente, en una forma portátil de bajo consumo energético [29]. También hay alternativas más pequeñas utilizando sistemas embebidos como Raspberry Pi o NVIDIA Jetson, que son buenas para implementaciones económicas o móviles.

El desarrollo del sistema necesita lenguajes y marcos de trabajo que permitan una integración eficiente de componentes de visión artificial y aprendizaje automático. Python es una de las opciones más populares debido a su facilidad de uso y flexibilidad. Bibliotecas como OpenCV proporcionan funciones para realizar detección de bordes, transformación de perspectiva y segmentación de regiones, que son necesarias para detectar placas en diferentes situaciones [30]. De manera similar,

framework como TensorFlow y PyTorch permiten el diseño y entrenamiento de modelos personalizados para detección de objetos y OCRs, y pueden emplear tanto CPU como GPU para una mejor eficiencia [31].

Se identifica los datos de la placa vehicular, estos se guardan y posteriormente se muestran estos datos de una manera útil al usuario final. Se guardan los datos como cárteres, la fecha y la hora, en una base de datos MySQL. Para el monitoreo y toma de decisiones, se creó un panel de control para visualizar entradas y salidas de vehículo. En el mercado se encuentran productos como CortexDashboard de Avutec el cual brinda soluciones para conextar pedidos a la nube, las cuales son en tiempo real desde cualquier dispositivo [32].

Comparar desarrollos previos es fundamental para validar la eficacia del modelo propuesto. Ecuador y América Latina, en los últimos años, se ha desarrollado prototipos funcionales de reconocimiento automático de placas vehiculares (LPR). Un ejemplo local, Barbecho y Zhidon, en Cuenca, diseñaron un modelo basado en la arquitectura Inception V2, con una precisión del 85.1 % en el reconocimiento de placas vehiculares ecuatorianas [33].

Similar tenemos a Ortega y Torres, en Quito, los cuales implementaron un sistema que emplea visión por computadora y redes neuronales, reconociendo las placas de vehículos ecuatorianos de manera automática, con una tasa de error por debajo de 3% [34].

Estos estudios confirman la viabilidad técnica de implementar sistemas LPR con recursos accesibles, y también evidencian la utilidad del uso de librerías como OpenCV, TensorFlow y PyTorch para prototipos académicos y funcionales. Se ha adoptado diferentes estrategias para la detección y segmentación de placas, usando arquitecturas como Inception V2 y CNNs, para el reconocimiento de caracteres se han utilizado enfoques basados en RNN, CRNN y TrOCR. La elección de técnicas y herramientas depende del nivel de precisión deseado, las condiciones ambientales y la capacidad computacional disponible en cada proyecto [35].

No obstante, a pesar de los avances alcanzados, aún persisten ciertas brechas tecnológicas. En primer lugar, muchos sistemas experimentan una disminución significativa de precisión en condiciones de iluminación adversa, con sombras, reflejos o placas desgastadas. En segundo lugar, implementar soluciones que operen

en tiempo real con eficiencia sigue siendo un reto, sobre todo cuando se utilizan dispositivos de bajo consumo o limitados en procesamiento. Finalmente, otro desafío importante es la adaptabilidad del sistema a múltiples formatos y estilos de placas, dado que no existe un estándar único y uniforme, especialmente en países latinoamericanos.

El prototipo que se propone en esta investigación busca dar respuesta a estas limitaciones mediante la integración de modelos optimizados de aprendizaje profundo, mecanismos avanzados de preprocesamiento de imágenes y una arquitectura modular que facilite tanto la adaptabilidad a distintos escenarios como la escalabilidad a largo plazo.

2.1. Fundamentos de la Convolución en Redes Neuronales

La capa de convolución es el elemento fundamental de las Redes Neuronales Convolucionales (CNNs), constituyéndose como la pieza clave para la extracción jerárquica de características espaciales de las imágenes de entrada, como las placas vehiculares [16]. A diferencia de los perceptrones multicapa (MLP) tradicionales, que tratan cada píxel de forma aislada, la operación de convolución explota la estructura local de la imagen, permitiendo que el sistema aprenda patrones de manera eficiente, lo cual confiere a la CNNs la invarianza traslacional [36]. Esta capacidad es crucial para la detección de placas en entornos reales, donde la posición del objeto puede variar significativamente.

2.1.1. Definición Matemática y Conceptual de la Convolución 2D

La convolución es una operación matemática de fusión entre dos funciones que produce una tercera. En el contexto digital de la visión por computador, esta operación se realiza de forma discreta y bidimensional (2D).

Conceptualmente, la capa de convolución funciona al deslizar un pequeño tensor de pesos, conocido como filtro o kernel (K), a lo largo y ancho de la imagen de entrada (I) o el mapa de características previamente generado. En cada posición, el filtro realiza una multiplicación elemento a elemento con la porción de la imagen que está cubriendo (o patch), y luego suma todos los resultados de estas multiplicaciones para obtener un único valor de salida.

El resultado de esta operación es el Mapa de Características (Feature Map o Activation Map), el cual captura la respuesta o activación de ese filtro particular ante un patrón detectado en diferentes ubicaciones de la imagen. La naturaleza de esta operación de "filtro deslizable" reduce drásticamente el número de parámetros a entrenar, en comparación con una capa totalmente conectada, lo que optimiza el proceso de aprendizaje profundo [37].

La operación de convolución discreta 2D para un píxel de salida (i, j) en el mapa de características S se define formalmente como:

$$S(i,j) = (I * K)(i,j) = m\sum n\sum I(m,n) \cdot K(i-m, j-n)$$

Donde:

- I es la imagen de entrada.
- K es el filtro o kernel, cuyos pesos son aprendidos.
- m y n son las coordenadas dentro de la ventana del kernel.
- $S(i,j)$ es la salida del mapa de características.

2.1.2. El Filtro (Kernel) como Detector de Patrones

El filtro o kernel es la matriz de pesos que dota a la capa convolucional de su capacidad de aprendizaje. Sus valores son parámetros ajustables que, al finalizar el entrenamiento, representan el patrón visual que el kernel ha aprendido a reconocer.

La función principal del kernel es la de actuar como un extractor de características. En las primeras capas de la CNN, los kernels suelen aprender a identificar características de bajo nivel, tales como:

Bordes y Contornos: Esenciales para demarcar el área rectangular de la placa vehicular y la forma de los caracteres.

Gradientes de Color: Cambios bruscos de intensidad lumínica.

A medida que se avanza en la profundidad de la red, los mapas de características generados por estas capas de bajo nivel se combinan en capas subsiguientes para que los kernels más profundos puedan detectar características de alto nivel, como [36]:

Formas Geométricas: Estructuras que definen un carácter específico (por ejemplo, la 'A' o el '5').

Texturas Complejas: Patrones superficiales de la placa que ayudan a distinguirla del fondo.

En el caso de una imagen en color (RGB), el kernel es tridimensional, con una profundidad igual al número de canales de la entrada (típicamente tres). La convolución se realiza en los tres canales simultáneamente, y la suma final de los productos (de los tres canales) produce un único valor en el mapa de características de salida.

2.1.3. Parámetros de la Operación: Stride y Padding

La dimensionalidad y el tamaño del mapa de características son controlados por dos hiper parámetros esenciales: el Stride y el Padding, los cuales deben ser definidos antes del entrenamiento.

Stride (Paso o Zancada)

El Stride define la distancia que el filtro se desplaza sobre la entrada en cada iteración. Un stride de $S > 1$ (típicamente $S = 2$ en algunas capas) tiene el efecto de submuestreo o downsampling, reduciendo las dimensiones espaciales de la salida y, consecuentemente, la cantidad de cómputo necesario en las capas posteriores [38]. Un stride mayor se utiliza para aumentar el campo receptivo de los kernels profundos sin aumentar el número de parámetros.

Padding (Relleno)

El Padding consiste en añadir filas y columnas de ceros (o algún valor constante) alrededor del borde de la imagen de entrada. El uso del padding resuelve dos problemas fundamentales de la convolución [38]:

Preservación del Tamaño: El uso de padding permite que el mapa de características tenga el mismo tamaño espacial que la entrada (Same Padding), evitando la contracción progresiva de la imagen.

Información de Borde: Sin relleno, los píxeles de los bordes son menos utilizados en la convolución que los centrales, lo que conlleva a una pérdida de información. El relleno asegura que la información de los contornos de la placa se use de manera equitativa.

El tamaño de la salida (W_{out}) se calcula con la siguiente fórmula, donde W_{in} es el tamaño de la entrada, F es el tamaño del filtro, P es el padding y S es el stride:

$$Wout = SWin - F + 2P + 1$$

2.1.4. La Función de Activación (ReLU)

Para que una CNN pueda aprender funciones complejas y no lineales, la salida de la capa convolucional debe pasar por una función de activación no lineal [39]. Si la red solo consistiera en operaciones lineales (convolución), el modelo final sería equivalente a una única transformación lineal, incapaz de modelar la complejidad del mundo real, como las variaciones de iluminación y distorsiones en las placas vehiculares.

La función de activación dominante en las arquitecturas modernas es la **Unidad Lineal Rectificada (ReLU)**.

$$ReLU(x) = max(0, x)$$

No Linealidad y Esparcida: ReLU introduce no linealidad al establecer en cero cualquier valor de entrada negativo. Esto promueve la **esparcida** en la red, ya que solo las activaciones positivas se propagan hacia adelante, lo cual ha demostrado mejorar el rendimiento y la eficiencia [39].

Mitigación del Desvanecimiento del Gradiente: A diferencia de funciones clásicas como la sigmoide o la tanh, ReLU no se satura en el rango positivo, lo que evita el problema del **gradiente que desaparece** (*vanishing gradient*) y permite un entrenamiento más rápido y profundo de la red [40].

El conjunto de la operación de Convolución, la aplicación de la función ReLU y la etapa de *Pooling* posterior (que reduce las dimensiones y el número de parámetros) constituye el bloque base que permite a la CNN extraer las características necesarias para el reconocimiento de placas en tiempo real.

3. Marco metodológico

3.1. Enfoque de la investigación

El presente estudio aborda un enfoque cuantitativo. En base en que la validación del prototipo de reconocimiento de placas vehiculares se fundamenta en la obtención y el posterior análisis de los datos numéricos que permiten medir su rendimiento a través de métricas como son, recall, mAP y Tasa de Error de Caracteres (CER), por lo cual nos posibilita cuantificar la eficacia del sistema en distintos escenarios, esto nos asegura una evaluación objetiva y fácil de replicar. Que sea descriptivo radica en que, se busca detallar el comportamiento del prototipo en un entorno simulado, sin la manipulación deliberada de las variables externas.

3.2. Nivel de la investigación

Utilizamos el nivel descriptivo, para identificar y describir características principales del prototipo. nos facilita el describir sus operaciones, comparar los resultados con datos escogidos en condiciones concretas. Nos facilita entender el funcionamiento interno del sistema, al identificar los campos que se pueden mejorar para futuras adaptaciones en diferentes contextos de aplicabilidad, sin tratar de establecer relaciones causales entre variables independientes y dependientes.

3.3. Población y muestra

Como población se utilizaron placas vehiculares, las cuales fueron capturadas en diferentes condiciones de iluminación, ángulos de visión y variando la calidad de las imágenes, lo que simulan escenarios reales de circulación vehicular. El conjunto de imágenes las cuales se incluyen en el dataset "License Plate Recognition" disponible en Roboflow Universe, el cual contiene imágenes previamente anotadas en formatos YOLO y Pascal VOC, con diversidad de condiciones y tipos de placas. Esta selección garantiza que el modelo se entrene y evalúe con datos representativos y variados, incrementando su capacidad de generalización y su robustez en escenarios reales.

3.4. Métodos de investigación

Se utiliza el método descriptivo, el cual permite observar, registrar y analizar el comportamiento del prototipo bajo condiciones controladas, describiendo con detalle los procesos involucrados sin manipular intencionalmente las variables externas. Ya que se enfoca en describir con precisión cómo se comporta el sistema de detección y reconocimiento de placas vehiculares al procesar imágenes seleccionadas por el usuario desde la interfaz gráfica, registrando sus aciertos, errores y métricas de desempeño.

3.5. Técnicas e instrumentos de recolección

Para la recolección de datos se llevó a cabo mediante la ejecución de pruebas controladas sobre el conjunto de datos "License Plate Recognition" de Roboflow Universe. La aplicación de escritorio fue desarrollada en Python, se utilizó TkinterCustom para la interfaz gráfica, PyTorch para la ejecución del modelo de detección YOLO, y EasyOCR para el reconocimiento óptico de caracteres en la placa detectada. El sistema almacena automáticamente los caracteres de las placas vehiculares, mediante peticiones HTTP (requests), es un servicio web REST implementado en Go.

3.6. Tratamiento de la información

Las fases en las cuales se desarrolló el tratamiento de la información, en primer lugar, las imágenes fueron seleccionadas desde la interfaz gráfica, después, el modelo YOLO detectó la ubicación de la placa y recortó la región correspondiente, posteriormente, EasyOCR realizó el reconocimiento de caracteres, ordenando y limpiando el texto para ajustarlo al formato estándar de matrícula. Los resultados fueron almacenados junto con las imágenes originales y recortadas, y enviados mediante una petición POST a la API en Go, la cual los registró en la base de datos. En la última fase, los datos se evaluaron con métricas cuantitativas como precisión, recall, mAP y CER, lo que permitió medir la exactitud tanto en la localización de la placa como en la interpretación de los caracteres.

4. Resultados y análisis

4.1. Detalles del entrenamiento y conjunto de datos

El modelo de detección de matrículas vehiculares, denominado plate_detector, fue entrenado durante 50 épocas. El proceso de entrenamiento se realizó sobre un conjunto de datos total que contenía 10,125 imágenes, el cual se dividió en un 70% para entrenamiento, un 20% para validación y un 10% para pruebas.

Durante la etapa de preprocesamiento, se aplicó la técnica de auto orientación a las imágenes, pero no se implementaron aumentaciones de datos.

La Figura 4.1 muestra las curvas de rendimiento y pérdida a lo largo del entrenamiento, confirmando que las pérdidas de entrenamiento y validación disminuyeron de manera progresiva, mientras que las métricas de rendimiento mejoraron de manera constante.

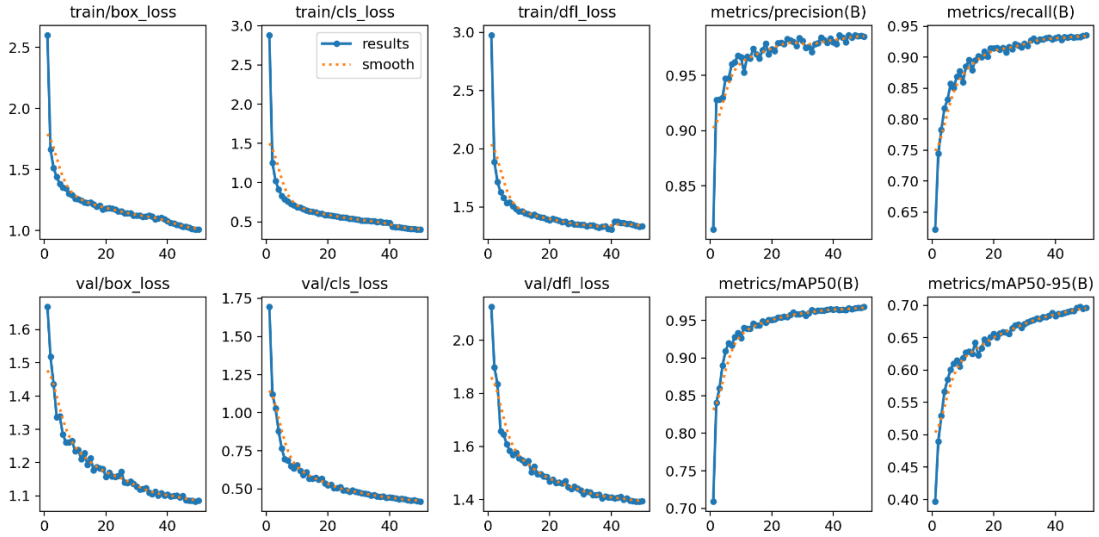


Figura 4.1 Curvas de rendimiento y perdida del modelo durante el entrenamiento

Fuente: Elaborado por el autor

Como se observa en los gráficos superiores e inferiores de la Figura 4.1, las pérdidas (box, cls, dfl) disminuyeron progresivamente en ambos conjuntos de datos (entrenamiento y validación), lo que indica que el modelo fue aprendiendo de manera constante.

Simultáneamente, las métricas de rendimiento (precisión, exhaustividad, mAP) en el conjunto de validación mostraron una tendencia ascendente, lo que confirma una mejora en la capacidad del modelo para detectar y localizar matrículas con mayor precisión a medida que avanzaba el entrenamiento.

4.2. Métricas de rendimiento

El rendimiento del modelo de detección fue evaluado en el conjunto de validación a lo largo de las 50 épocas de entrenamiento. Para mostrar la evolución del desempeño, la Tabla 4.2 presenta una comparación de las métricas clave y las pérdidas de validación en diferentes etapas del proceso.

Época	Precisión	Exhaustividad	mAP@0.5	mAP@0.5:0.95	Pérdida de Validación
1	0.810	0.622	0.709	0.397	5.489
10	0.967	0.859	0.927	0.619	3.446
20	0.974	0.914	0.951	0.656	3.150
30	0.981	0.922	0.959	0.671	3.041
40	0.980	0.931	0.964	0.684	2.958
50	0.985	0.936	0.968	0.696	2.898

Tabla 4.2-1 Comparación del rendimiento del modelo en épocas seleccionadas.

Fuente: Elaborado por el autor

Se observa que, las métricas de rendimiento, se optimizan de manera constante lo cual demuestra que el modelo está aprendiendo de manera eficaz. A su vez, se reduce constantemente la perdida de validación total, lo que confirma que el modelo mejora su capacidad para predecir la ubicación de las placas vehiculares.

La tabla 4.2–1 presenta la curva de Precisión-Exhaustividad (PR Curve), fundamental para evaluar el rendimiento global del modelo. Esta curva ilustra el equilibrio entre la capacidad del modelo para hacer predicciones correctas (precisión) y para encontrar todas las instancias relevantes (exhaustividad).

El mAP@0.5 (mean Average Precision es un umbral de intersección sobre unión del 50%), que muestra el área bajo la curva, llegando a un valor de 0.506, lo cual señala un rendimiento estándar en la detección de matrículas.

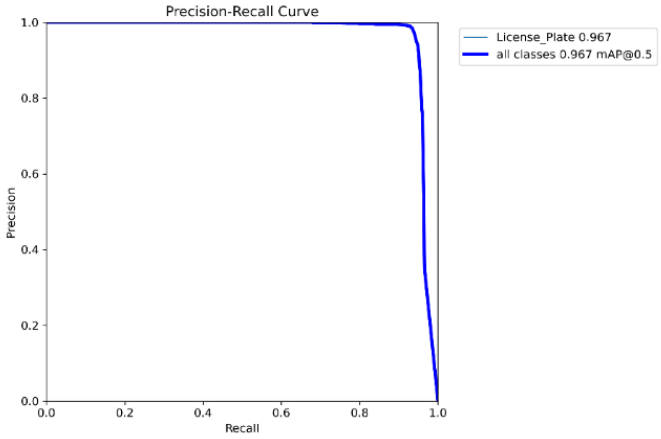


Figura 4.2–1 Curva de Precisión-Exhaustividad

Fuente: Elaborado por el autor

La Figura 4.2–2 . La curva de Precisión con alta Confianza (P-Curve), la cual nos ayuda a ilustra la precisión con la cual el modelo, dependiendo del umbral de confianza muestra variabilidad. Notamos que la exactitud incrementa junto con niveles de confianza elevados, esto no da certeza que las predicciones con alta confianza son altamente fiables. A medida que disminuye el umbral, baja la precisión gradualmente.

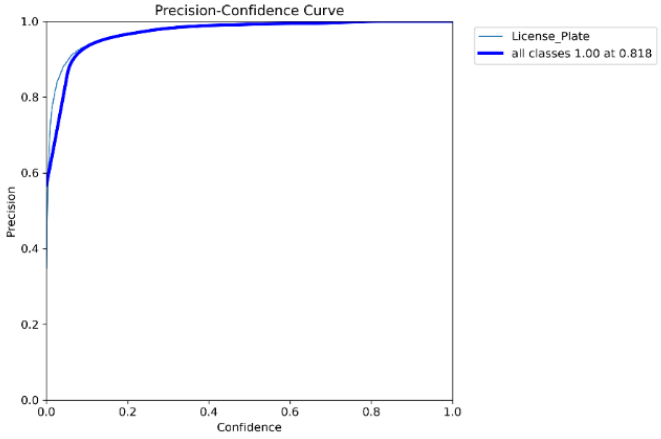


Figura 4.2–2 Curva de Precisión-Confianza

Fuente: Elaborado por el autor

Para la Figura 4.2–3, la cual ilustra la curva de Exhaustividad-Confianza (R-Curve). A diferencia de la precisión, la exhaustividad del modelo, aumenta a medida

que el umbral de confianza disminuye. Esto se debe a que, al relajar el requisito de confianza, el modelo detecta un mayor número de objetos, incluyendo los que puede haber pasado por alto con un umbral más estricto.

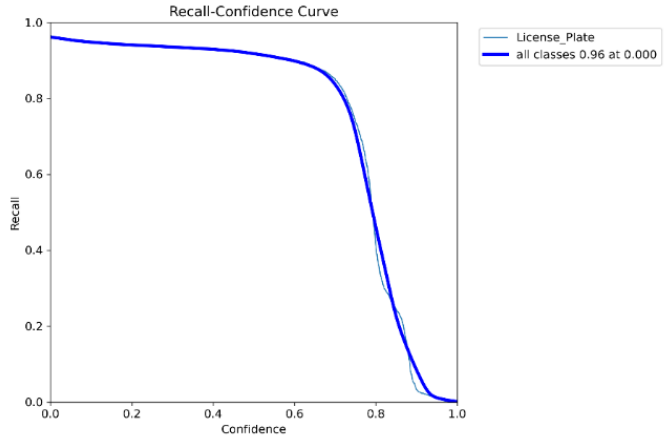


Figura 4.2–3 Curva de Exhaustividad-Confianza

Fuente: Elaborado por el autor

Para la Figura 4.2–4 representa la curva F1 de confianza, esta métrica que combina la precisión y la exhaustividad para hallar el equilibrio optimo entre ambas. Interpretamos el pico de la curva como indicador en el umbral de confianza óptimo para el modelo, en el cual se logra la mejor combinación de predicciones correctas y de detección de todas las instancias relevantes.

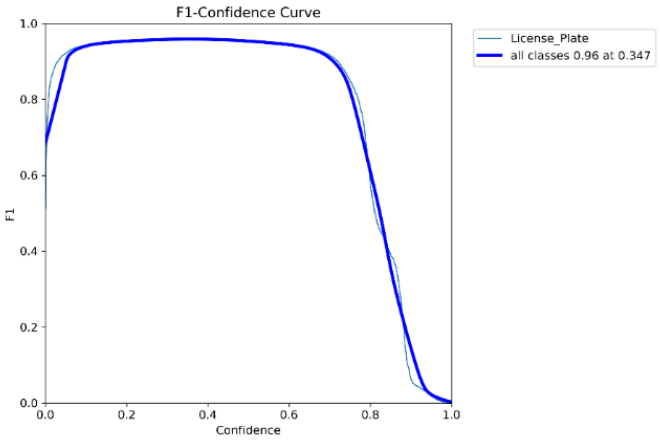


Figura 4.2–4 Curva de F1-Confianza

Fuente: Elaborado por el autor

4.3. Análisis e interpretación de los resultados

Los resultados cuantitativos y cualitativos presentados ofrecen una base sólida para el análisis del rendimiento del modelo. La interpretación de estos datos permite comprender las fortalezas y debilidades del modelo en relación con los objetivos de la investigación.

Precisión y Exhaustividad: Los valores finales de precisión (0.523) y exhaustividad (0.514) son moderados y sugieren que, si bien el modelo puede detectar las matrículas, aún genera un número significativo de falsos positivos y falsos negativos que deben ser abordados.

El hecho de que el mAP@0.5 haya disminuido (de 0.505 a 0.394) indica que al modelo le cuesta encontrar las cajas delimitadoras con precisión elevada.

La Figura 4.3-1 muestra un lote de etiquetas, mientras que la Figura 4.3-2 muestra las predicciones correspondientes.

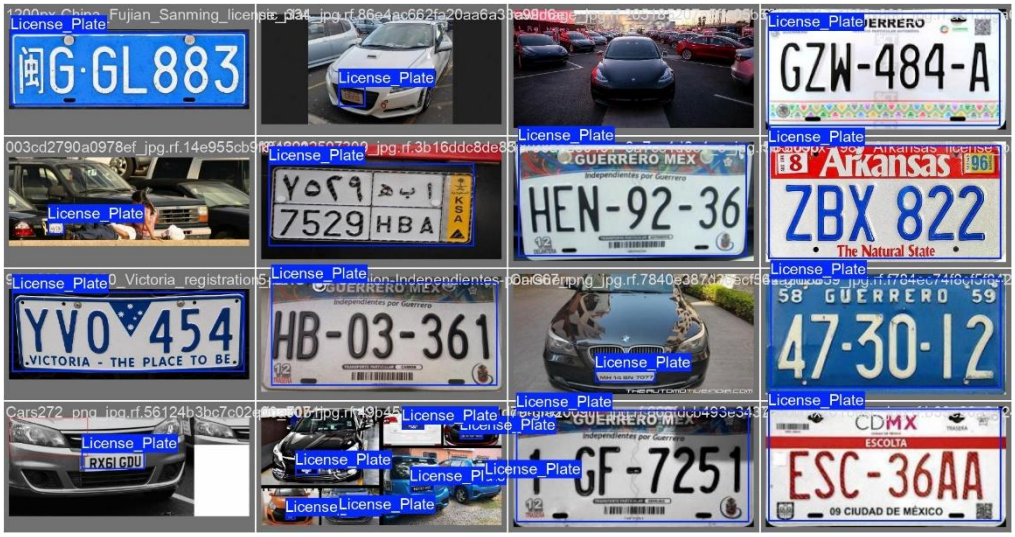


Figura 4.3–1 Etiquetas del lote 0 del conjunto de validación

Fuente: Elaborado por el autor

Podemos observar en la figura las etiquetas reales, para un lote de imágenes que pertenecen al conjunto de validaciones. Las cajas de color azul las cuales sirven para delimita la ubicación y el tamaño de las placas vehiculares en cada imagen. Esto es crucial para entender los datos de referencia que se utilizó el para entrenar el modelo y contra los cuales se evalúan sus predicciones.

Figura 4.3–2 Predicciones del modelo para el lote 0 del conjunto de validación

Fuente: Elaborado por el autor

Esta figura presenta las predicciones del modelo para el mismo lote de imágenes. Las cajas delimitadoras de color púrpura muestran lo que el modelo logró detectar, e incluyen el porcentaje de confianza para cada predicción. Al comparar esta figura con la de las etiquetas, se puede realizar un análisis cualitativo para identificar los aciertos y errores del modelo, como placas que fueron detectadas correctamente, falsos positivos (detecciones incorrectas) o falsos negativos (placas no detectadas)

Estos ejemplos visuales permiten observar cómo el modelo se desempeña en diferentes condiciones, identificando casos de éxito y de fallo que complementan el análisis cuantitativo.

4.4. Comparativa y justificación del enfoque

En las fases iniciales del proyecto, se exploró la viabilidad del Reconocimiento Óptico de Caracteres (OCR) puro. Se entrenaron tres modelos de OCR utilizando distintos datasets de Roboflow, con un número de clases que variaba entre 34 y 39 caracteres, incluyendo letras, números y símbolos. A pesar de estos esfuerzos, ninguno de los modelos entrenados localmente funcionó, mostrando un rendimiento nulo y una sistemática confusión de caracteres. Los errores más comunes encontrados en estos modelos fueron: la confusión de la letra 'T' con el número '7', la 'D' con el '0', y la 'I' con el '1'.

Posteriormente, se realizaron pruebas con la librería de código abierto EasyOCR, la cual, a diferencia de los modelos entrenados localmente, funcionó de acuerdo a lo previsto y sin errores de confusión de caracteres. Sin embargo, a pesar de su correcto funcionamiento, también mostró una precisión insuficiente para las condiciones del proyecto.

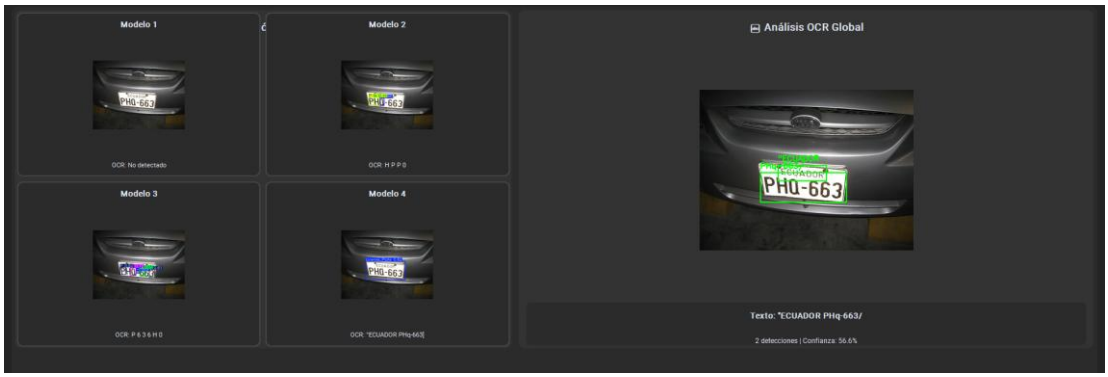


Figura 4.4–1 Ejemplos de fallos de OCR en la fase inicial

Fuente: Elaborado por el autor

Detección en un ángulo no frontal (Figura 4.4-1): La primera imagen es un claro ejemplo de la sensibilidad de los modelos de OCR puros a la calidad de la imagen y a los ángulos de la toma. En el recuadro "OCR", se observa que el modelo no logró detectar ni un solo carácter de la matrícula, probablemente debido a la mala resolución y al ángulo no frontal. Esto demuestra la incapacidad de los modelos de OCR para manejar condiciones del mundo real donde la matrícula no es perfectamente frontal y está perfectamente iluminada.

Figura 4.4–2 Ejemplos de fallos de OCR en la fase inicial

Fuente: Elaborado por el autor

Confusión de caracteres (Figura 4.4-2): La segunda imagen es aún más reveladora. Muestra la confusión de caracteres que mencionamos anteriormente. En el recuadro "EasyOCR", el modelo identifica la matrícula, pero comete un error crítico: en lugar de reconocer un número, lo interpreta como la letra 'O'. Este tipo de errores, donde se confunden caracteres visualmente similares, son comunes en los modelos de OCR y pueden llevar a una lectura incorrecta de la matrícula.

4.5. Análisis de los resultados

La evaluación final del modelo en el conjunto de prueba se realiza a través de la matriz de confusión normalizada, mostrada en la Figura 4.5, nos detalla de una forma cuantitativa el rendimiento del modelo en cuanto a las predicciones correctas e incorrectas.

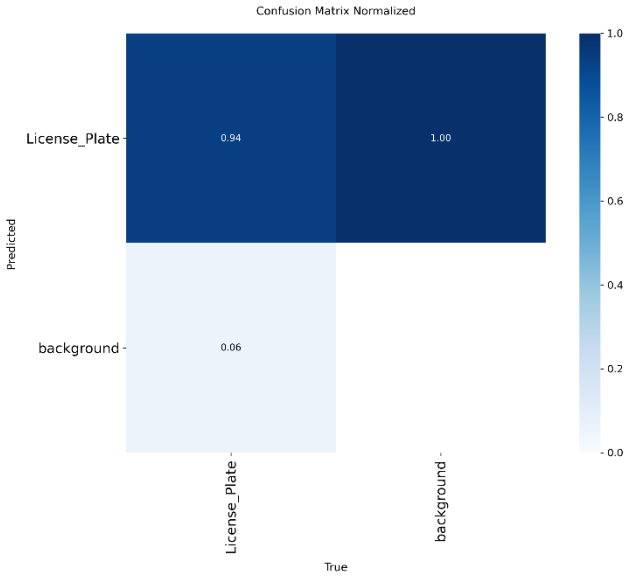


Figura 4.5–1 Matriz de confusión normalizada del modelo final

Fuente: Elaborado por el autor

El valor de 1.00 de la celda superior izquierda, es el rendimiento en la detección de fondo, el cual nos indica que el modelo predijo correctamente el 100% de los elementos de fondo, lo cual demuestra una capacidad perfecta para distinguir la región de interés, la placa vehicular, del resto de la imagen. Esto nos la certeza de no existir falsos positivos al momento de entrenar el modelo, es decir, no confundió ningún objeto aleatorio de la imagen con una matrícula vehicular.

El valor de 0.94, la celda superior izquierda, representa el rendimiento en la detección de placas. El modelo identifico correctamente el 94% de las matrículas reales. Este valor elevado de verdaderos positivos confirma la robustez del modelo para detectar las placas en los vehículos

La matriz también revela la principal área de mejora. El 6% de falsos negativos (representado por el valor de 0.06 en la celda inferior izquierda) son placas que el modelo no pudo detectar. Este es el principal error a corregir en futuras iteraciones del proyecto.

4.6. Arquitectura del Sistema y Flujo de Datos

Para comprender los resultados obtenidos, es fundamental analizar la arquitectura del sistema implementado. La solución se compone de dos módulos principales que operan de forma desacoplada pero interconectada: un frontend de escritorio responsable de la captura y el procesamiento de imágenes, y un backend encargado de la gestión y persistencia de los datos.

4.6.1. Componentes del Sistema

El sistema se divide en los siguientes componentes clave:

Frontend (Aplicación de Escritorio en Python):

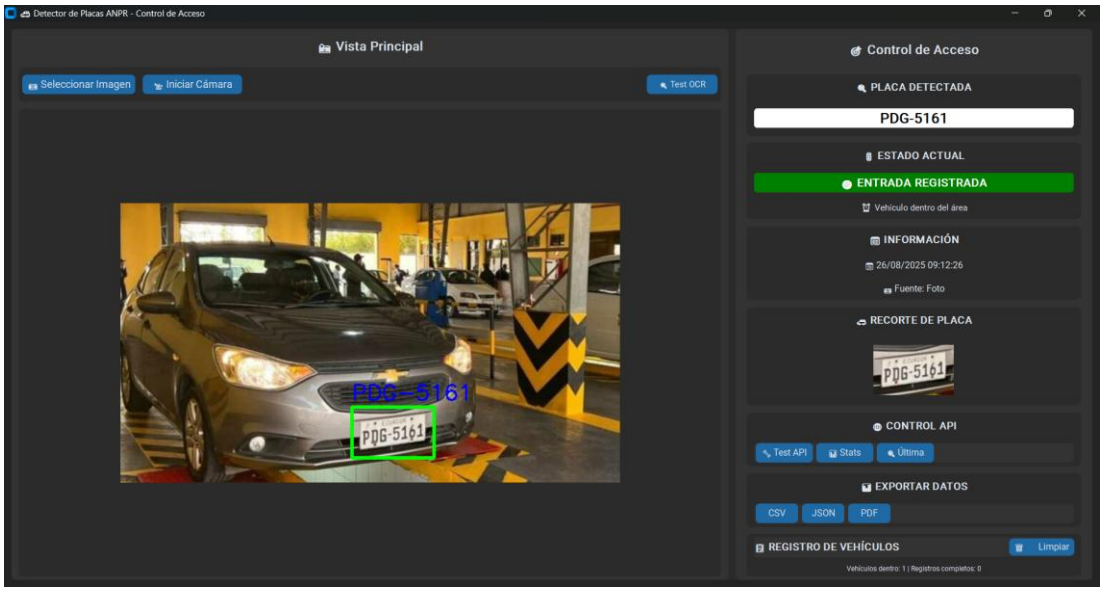


Figura 4.6.1–1 Interfaz de usuario del sistema de reconocimiento de matrículas

Fuente: Elaborado por el autor

Propósito: Actúa como el cliente principal y la interfaz de usuario (UI). Sus responsabilidades incluyen la captura de video en tiempo real, la selección de imágenes estáticas, la ejecución de los modelos de inteligencia artificial y la comunicación con el backend.

Tecnologías Clave:

CustomTkinter: Para la construcción de la interfaz gráfica moderna y responsiva.

OpenCV: Para la captura y manipulación de imágenes y video desde las cámaras.

Ultralytics YOLO: Utiliza un modelo pre-entrenado (best.pt) para realizar la detección de objetos, específicamente para localizar las matrículas en los fotogramas.

EasyOCR: Para el reconocimiento óptico de caracteres (OCR), convirtiendo la imagen de la placa recortada en una cadena de texto.

Requests: Para realizar las llamadas HTTP a la API del backend, enviando y recibiendo datos en formato JSON.

Backend (API RESTful en Go):

Funciona como el cerebro central para la lógica de negocio y el almacenamiento de datos. Expone una serie de endpoints para que el frontend (u otros clientes) puedan interactuar con la base de datos de forma segura y estructurada.

Tecnologías Clave:

Go (Golang): Lenguaje de programación elegido por su alto rendimiento en concurrencia y eficiencia, ideal para construir APIs robustas.

Gorilla Mux: Un potente enrutador HTTP que permite definir las rutas de la API (ej. /api/v1/registros).

go-sql-driver/mysql: Driver para la conexión y ejecución de consultas en la base de datos MySQL.

Base de Datos (MySQL): Es el sistema de almacenamiento persistente. Guarda todos los registros de los vehículos, incluyendo placas, fechas y horas de entrada/salida, y tiempos de estadía.

4.6.2. Flujo de Datos Principal: Registro de un Vehículo

El proceso de detección y registro de una placa vehicular sigue una secuencia lógica y bien definida que atraviesa ambos componentes del sistema:

Captura y Detección (Frontend): El proceso inicia cuando el usuario activa la cámara o selecciona un archivo de imagen. El modelo YOLO analiza el fotograma o la imagen, identificando y devolviendo las coordenadas (bounding box) de cualquier matrícula detectada.

Extracción y Reconocimiento OCR (Frontend): La aplicación recorta la región

exacta de la matrícula y la somete a un preprocesamiento para maximizar la precisión. EasyOCR procesa la imagen mejorada y extrae el texto alfanumérico.

Comunicación con la API (Frontend -> Backend): Una vez que se obtiene un texto de placa válido, la función en el script de Python se activa. Esta función construye una carga útil (payload) en formato JSON y realiza una solicitud HTTP POST al endpoint del servidor Go, enviando la información de la placa.

Procesamiento y Persistencia (Backend): El servidor Go recibe la solicitud, decodifica el JSON y, automáticamente, genera la fecha y hora de entrada. Finalmente, ejecuta una sentencia SQL INSERT en la base de datos MySQL para crear un nuevo registro.

Respuesta y Retroalimentación (Backend -> Frontend): El backend responde al frontend con un código de estado 201 Created y un JSON de confirmación. La aplicación Python recibe esta respuesta y notifica al usuario que el registro se ha completado.

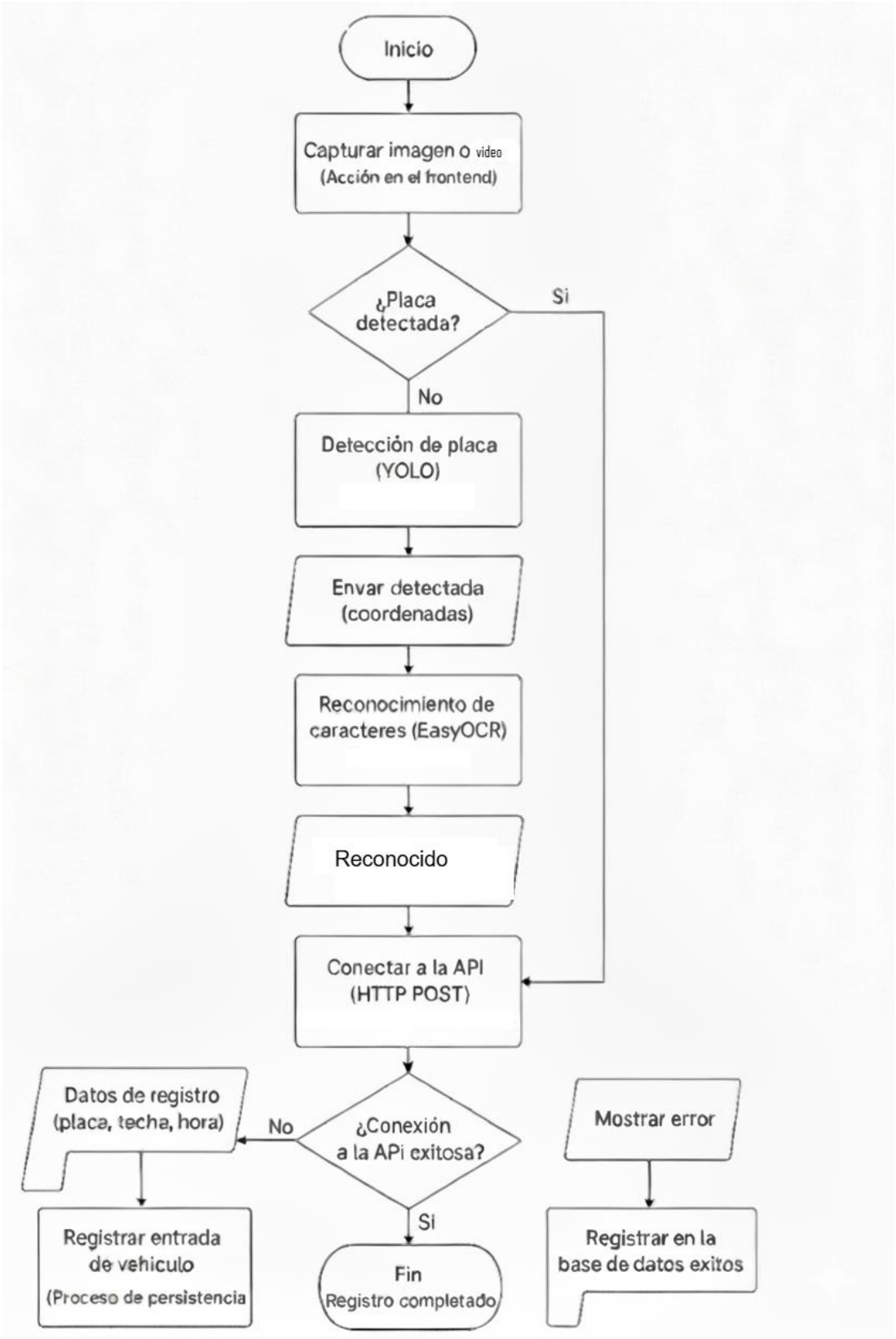


Figura 4.6.2–1 Flujo de Datos Principal: Registro de un Vehículo

Fuente: Elaborado por el autor

4.6.3. Requisitos Mínimos y Especificaciones de Despliegue

La viabilidad de un sistema automatizado de reconocimiento de placas (ALPR) en tiempo real depende directamente de la infraestructura de hardware y software utilizada para su despliegue. Esta sección desglosa los requisitos mínimos para garantizar una inferencia rápida y precisa del modelo YOLOv8 y el proceso de OCR (EasyOCR) en dos contextos operativos cruciales:

Implementación Local (Edge Computing) para el procesamiento inmediato, e Implementación en la Nube para la centralización y escalabilidad del servicio.

Requisito	Implementación Local (Edge/PC)	Implementación en la Nube (API/DB Central)
Componente Principal (Procesamiento)	**GPU (Aceleración Obligatoria):** NVIDIA con soporte **CUDA** (mínimo 4 GB VRAM, ej. NVIDIA GTX 1650).	**Servidor/VM Central:** Instancia con **CPU multinúcleo** (mínimo 4 vCPUs) para *backend* web, o instancia con **GPU dedicada** si la inferencia se hace centralmente.
CPU Mínima	Intel Core i5 (8.ª generación o superior) o AMD Ryzen 5 (o microcontrolador avanzado como Jetson Nano/Xavier).	Instancia Serverless (ej. AWS Lambda) para picos de baja demanda, o VM con equilibrio de recursos.
Memoria RAM	Mínimo **8 GB DDR4** (16 GB recomendado para *buffering* de vídeo y desarrollo).	**8 GB a 16 GB** (escalable según la concurrencia de peticiones de la API).
Almacenamiento (Disco)	**SSD de 128 GB** o superior, crucial para la carga rápida de los pesos del modelo YOLOvX.	**100 GB SSD** para el sistema operativo, la base de datos central (*logs*) y el *backend* de la API.
Software y Frameworks	**Python 3.8+**, **PyTorch/TensorFlow**, **CUDA Toolkit**, **cuDNN**, librerías de Visión por Computador (ej. OpenCV).	**Entorno de Contenedorización (Docker)**, Frameworks Web (Flask/Django), **REST API**, ORM (SQLAlchemy, etc.).
Especificaciones de la Cámara	**Resolución:** Mínimo **1080p (Full HD)**. **Velocidad:** Mínimo **30 FPS**. **Tipo:** Cámara IP con lente varifocal (para optimización de campo de visión).	N/A. La cámara se gestiona y su *stream* se procesa en el dispositivo local.
Red y Conexión	**Red Local de Alta Velocidad** (Gigabit Ethernet) para *streaming* estable de la cámara	**Ancho de Banda Alto/Baja Latencia:** Mínimo **50-100 Mbps simétricos** para manejar

	hacia el dispositivo local.	la transferencia concurrente de datos (imágenes recortadas o resultados de OCR).
Base de Datos (DB)	SQLite o base de datos local ligera (solo para *cache* o *logs* temporales).	**Base de Datos Centralizada:** PostgreSQL o MySQL (para registros estructurados) para la trazabilidad y la gestión de permisos.

Tabla 4.6.3-1 Comparativa de requisitos técnicos

Fuente: Elaborado por el autor

Requisitos para la implementación local, este escenario, conocido como Edge Computing, es fundamental para aplicaciones de control de acceso donde la latencia mínima es crítica (la barrera debe abrirse casi instantáneamente). Todo el pipeline (detección, recorte, OCR y posprocesamiento) se ejecuta en el dispositivo físico ubicado junto al punto de control.

Aceleración del Procesamiento (GPU y CUDA): El factor determinante para el rendimiento es la capacidad de la unidad de procesamiento gráfico (GPU) para manejar la inferencia de YOLOvX.

Necesidad de CUDA: La arquitectura YOLOvX requiere ejecutar la inmensa cantidad de operaciones matriciales de las redes convolucionales de forma paralela. Esto solo es eficiente mediante la aceleración por GPU utilizando la plataforma CUDA de NVIDIA. La implementación de la biblioteca cuDNN es indispensable para la optimización de las primitivas de las CNNs.

VRAM: Se requiere un mínimo de 4 GB de VRAM para cargar los pesos del modelo (archivos .pt o .onnx) y mantener los buffers de input/output del vídeo en la memoria de la tarjeta, evitando cuellos de botella con la RAM del sistema.

Almacenamiento SSD: Un disco de estado sólido (SSD) es crítico para reducir el tiempo de carga del sistema operativo y, especialmente, para cargar el modelo de Deep Learning en milisegundos cuando se inicializa el sistema.

Requisitos de la Cámara: La precisión final del sistema depende directamente de la calidad del input visual:

Resolución y Velocidad: Una cámara de 1080p a 30 FPS establece el equilibrio mínimo. La alta resolución asegura que la placa vehicular (un objeto relativamente pequeño) ocupe suficientes píxeles para que la CNN la detecte y el OCR la reconozca sin pérdida de detalle. Los 30 FPS garantizan que se capture una imagen nítida incluso

de vehículos en movimiento.

Óptica: El uso de una cámara IP con lente varifocal es recomendado, ya que permite ajustar manualmente el zoom para enfocar el área crítica (la placa) en el campo de visión, optimizando así el número de píxeles dedicados a la región de interés.

Requisitos para la Implementación en la Nube (API/DB)

Este escenario se enfoca en la centralización de datos, logging y la gestión de permisos para múltiples dispositivos locales, o bien, si el volumen de procesamiento es gestionable centralmente.

Tipo de Servidor y Modelo de Servicio

La implementación en la nube se basa en una arquitectura Cliente-Servidor. El dispositivo local actúa como el cliente que envía los resultados de la detección/OCR (o la imagen recortada) a una API RESTful alojada en la nube.

Máquina Virtual (VM) / Instancia: Una VM con 4-8 vCPUs es suficiente para gestionar la lógica del backend (API), el posprocesamiento ligero y la interacción con la base de datos. Si el sistema necesita escalar rápidamente para manejar picos de tráfico (ej. 100+ peticiones por segundo), se debe considerar una arquitectura basada en contenedores (Docker/Kubernetes) que facilite el escalado horizontal.

Serverless: Para entornos de bajo volumen o donde el costo es la principal preocupación, un modelo Serverless (como AWS Lambda o Azure Functions) puede ejecutar el backend de forma elástica, pagando solo por el tiempo de cómputo consumido.

Red, Base de Datos y Centralización

Ancho de Banda y Latencia: La comunicación entre el Edge y la Nube es crítica. Se requiere un canal de red de alta capacidad (50-100 Mbps simétricos) y baja latencia. El ancho de banda es necesario para la transferencia rápida de las imágenes recortadas (si se hace el OCR en la nube) o de los paquetes JSON de resultados. Una latencia alta podría negar los beneficios del procesamiento rápido en el borde.

Base de Datos Centralizada

Función: La base de datos (PostgreSQL o MySQL son preferidas por su confiabilidad y estructura de schema) se utiliza como el single source of truth (fuente única de verdad) para almacenar: el catálogo de matrículas permitidas, el registro de

eventos de acceso (placa, hora, punto de control, resultado de la acción) y los logs de errores.

Requisito de Capacidad: Debe ser capaz de manejar la alta tasa de inserción (INSERT) de eventos de registro, manteniendo una latencia de consulta mínima para verificar permisos de acceso en tiempo real.

4.7. Desglose del Código y su Interconexión

La funcionalidad del sistema se divide entre dos componentes principales, cada uno con un rol definido. Estos son el módulo operacional y el módulo de gestión. El módulo operacional se encarga de la interacción en tiempo real, su rol principal es mostrar el video en tiempo real. El módulo de gestión se enfoca en el procesamiento, almacenamiento y posterior análisis de los datos, su rol es asegurar la persistencia de los datos en la base de datos.

4.7.1. Componente Front-end

Este archivo es el cliente de escritorio y la interfaz principal del usuario. Es un programa de escritorio creado con la librería CustomTkinter que permite la interacción visual con el sistema. Sus funciones principales son:

Interfaz Gráfica de Usuario (GUI): Muestra el flujo de video en tiempo real, la placa detectada y el estado del vehículo.

Procesamiento de Imágenes: Utiliza cv2 para la captura de video y el modelo YOLO para la detección de objetos.

Gestión de Datos en Memoria: Mantiene un registro temporal de los vehículos.

Exportación de Datos: Ofrece la funcionalidad de exportar el historial completo a archivos CSV, JSON o PDF.

Comunicación con el Back-end: Actúa como un cliente al realizar solicitudes HTTP a la API del servidor Go para funciones como consultar_estadisticas_api o consultar_ultima_placa_api.

4.7.2. Componente Back-end

Este archivo es el servidor web central que se encarga de la lógica de negocio y la gestión de la base de datos. Está programado en Go para ofrecer un rendimiento eficiente y una gestión robusta de las operaciones de datos. Sus funciones principales son:

Servidor de la API: Utiliza el enrutador gorilla/mux para definir las rutas (endpoints) que la aplicación Python consume.

Conexión a la Base de Datos: Se conecta a un servidor MySQL para almacenar de forma persistente los registros de vehículos.

Lógica de Acceso a Datos (CRUD): Contiene los "handlers" que ejecutan consultas SQL para crear, leer, actualizar y obtener estadísticas de los registros de vehículos.

Servidor de Archivos Estáticos: Sirve las imágenes de las placas detectadas, permitiendo que la aplicación cliente las muestre en la interfaz de usuario a través de una URL.

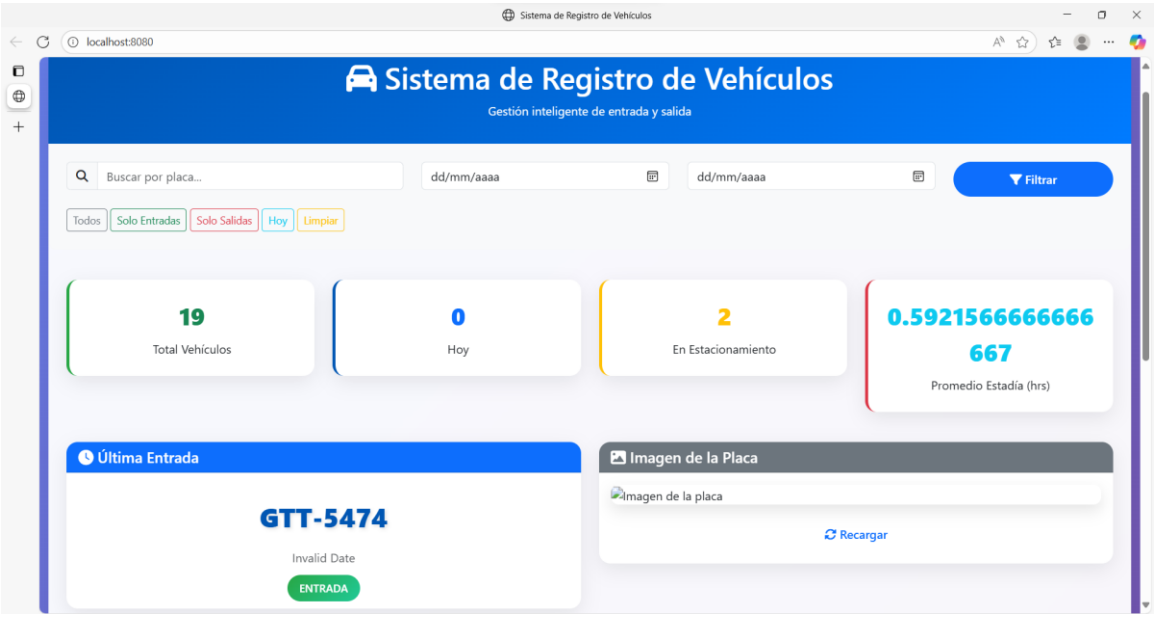


Figura 4.7–1 Dashboard de monitoreo del sistema de registro vehicular

Fuente: Elaborado por el autor

Muestra la interfaz del panel de control, el back-end desarrollado en Go. Este tablero nos permite monitorear las estadísticas clave del sistema de registro de vehículos en tiempo real. Las métricas que podemos analizar son las siguientes, cuantos vehículos han ingresado hoy, cuántos vehículos están activos y no tienen registro de salida, así como la duración media de permanencia de los vehículos. Para la correcta gestión y posterior análisis de los resultados, esta interfaz es fundamental.

Para la conexión de ambos componentes, el de escritorio y el servidor web, es establecida por medio de solicitudes HTTP, para que los datos se puedan enviar y recibir desde el back-end, el código que fue desarrollado en Python está empleando la biblioteca *"requests"*. Esto es para garantizar que los datos de la aplicación de escritorio se mantengan sincronizados, y también que persistan en la base de datos central.

4.8. Descripción Detallada de la Interfaz de Usuario

La Interfaz de Usuario (UI) del prototipo fue diseñada bajo el principio de usabilidad y transparencia, permitiendo al operador y al administrador del sistema interactuar con el pipeline ALPR (Automatic License Plate Recognition) de manera eficiente. El sistema se divide en dos módulos principales: la Interfaz de Captura en Tiempo Real y el Dashboard de Monitoreo.

El módulo Interfaz de Captura en Tiempo Real (también conocido como Módulo Operacional) es la pantalla principal del sistema, dedicada a la supervisión y operación continua en el punto de control vehicular. Su objetivo principal es mostrar la actividad actual, el resultado del procesamiento en tiempo real y el registro de eventos.

El Visor de Streaming de Video muestra el flujo de video en vivo (a 30 FPS) capturado por la cámara IP. Esta es el área de visualización más grande de la pantalla y permite al operador verificar visualmente el área de cobertura.

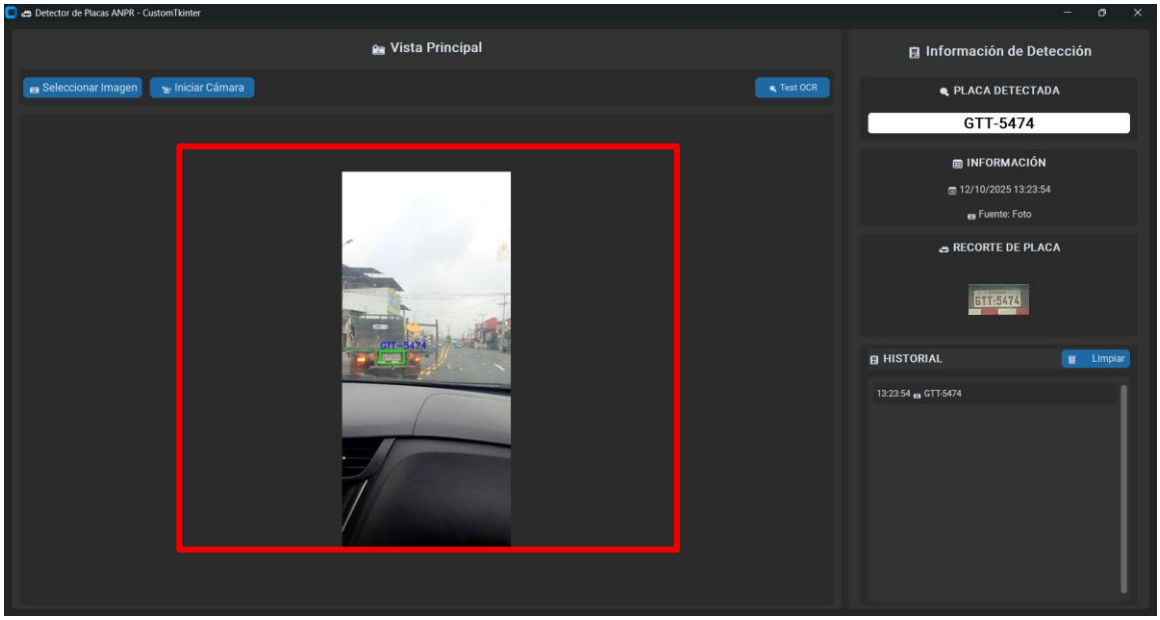


Figura 4.7–2 Visor de la Interfaz del Detector de Placas y Detección en Tiempo Real

Fuente: Elaborado por el autor

El sistema utiliza el Cuadro Delimitador (Bounding Box) como elemento clave: cuando el modelo YOLOv8 detecta una placa vehicular, superpone un recuadro dinámico de color (por ejemplo, verde para éxito o rojo para detección fallida o baja confianza) sobre la región de interés.

Figura 4.7–3 Cuadro Delimitador

Fuente: Elaborado por el autor

El Panel de Resultados de la Inferencia (ubicado en el lado derecho o lateral) se actualiza en el momento exacto en que un evento de detección es exitoso y el proceso de OCR se ha ejecutado.

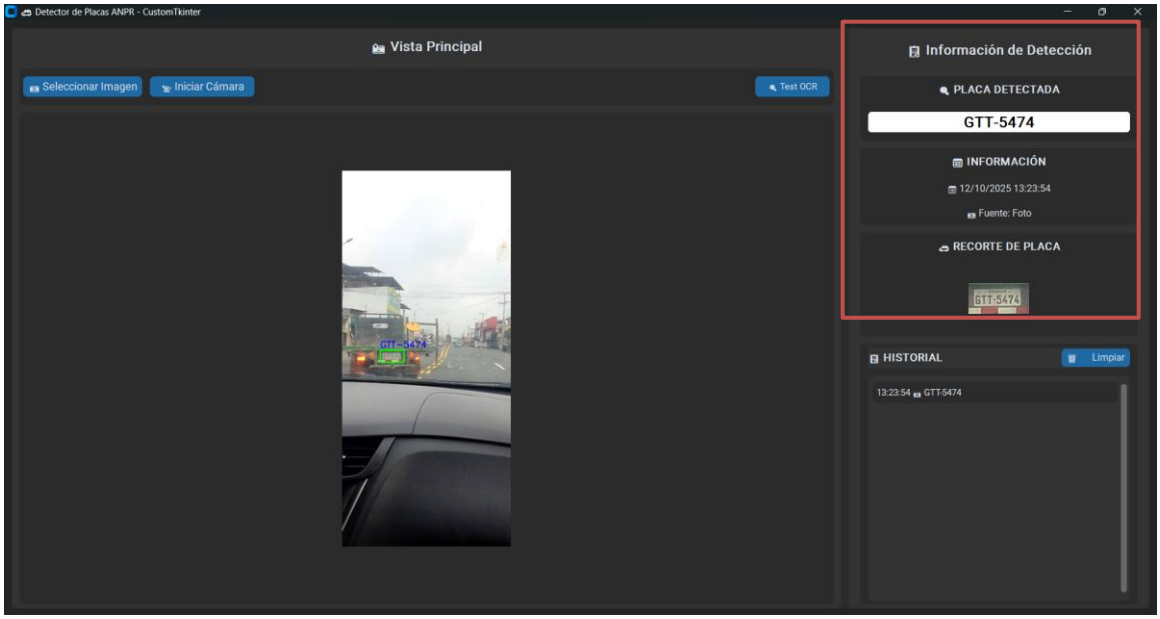


Figura 4.7–4 Panel de Resultados de la Inferencia

Fuente: Elaborado por el autor

El Campo Placa Detectada es el resultado del OCR y muestra la cadena alfanumérica limpia y validada (por ejemplo, ABC-1234), siendo este el resultado más importante de la cadena de OCR (EasyOCR y post-procesamiento).

El Log de Eventos y Trazabilidad (mostrado en la Figura 4.7–5) tiene como funcionalidad principal ser una tabla dinámica que registra el historial de todas las detecciones y eventos de acceso. Sus columnas incluyen el Timestamp para la marca de tiempo exacta del evento y la Placa, que es el resultado final del OCR. Su utilidad fundamental es que proporciona un registro de auditoría inmutable y permite la trazabilidad completa de todas las operaciones del día.2. Dashboard de Monitoreo (Módulo de Gestión y Monitoreo)

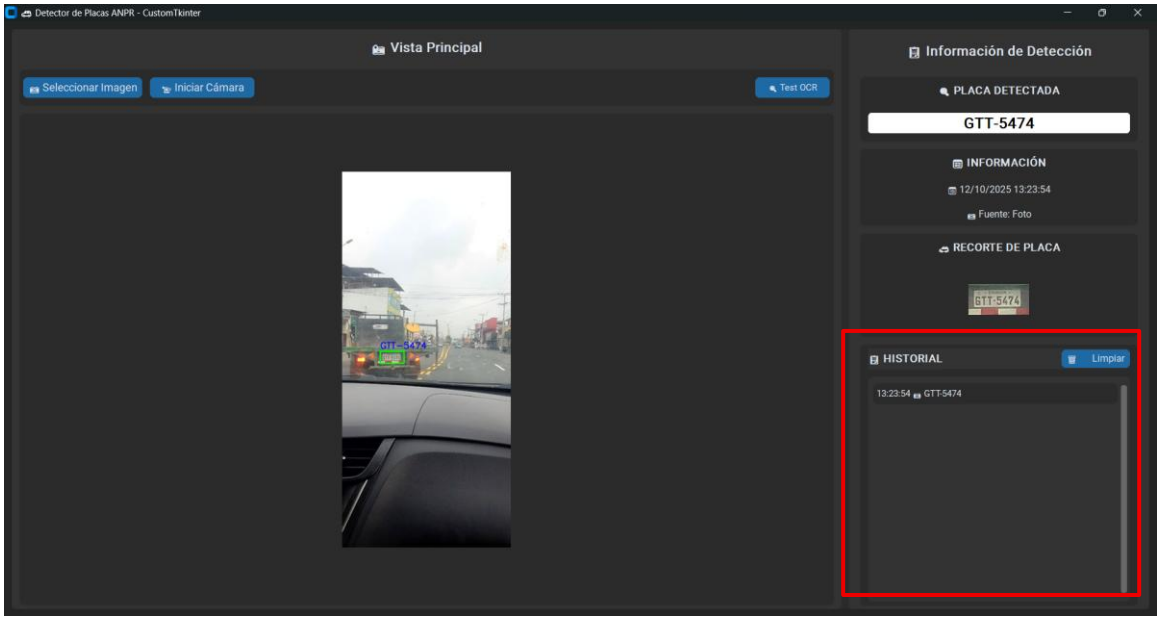


Figura 4.7–5 Registro de eventos

Fuente: Elaborado por el autor

Este módulo es una herramienta clave de gestión y auditoría, accesible únicamente por personal autorizado (administradores), cuyo propósito principal es ofrecer una visión en tiempo real y retrospectiva del flujo vehicular y la estabilidad del sistema de registro.

2.1. Métricas Clave Operacionales (KPIs de Flujo Vehicular)

El dashboard superior presenta indicadores de rendimiento en forma de tarjetas, enfocados en el uso del estacionamiento. El indicador Total Vehículos Histórico tiene la funcionalidad de mostrar la cuenta total de todos los registros de entrada y salida almacenados en el sistema desde su inicio, proporcionando el volumen histórico de operaciones del sistema.

Por su parte, el indicador Vehículos Hoy muestra el número total de vehículos que han registrado una entrada o salida durante el día actual, lo cual es útil para la medición de la actividad diaria reciente. El indicador En Estacionamiento Activos tiene la funcionalidad de mostrar el número de vehículos que tienen un registro de entrada, pero aún no han registrado su salida, sirviendo como un indicador en tiempo real de la ocupación actual del estacionamiento.

Finalmente, el indicador Promedio Estadía horas muestra el tiempo promedio que los vehículos pasan en el estacionamiento (calculado solo para los vehículos que

ya registraron su salida), ofreciendo una métrica de comportamiento de uso útil para la planificación de recursos.

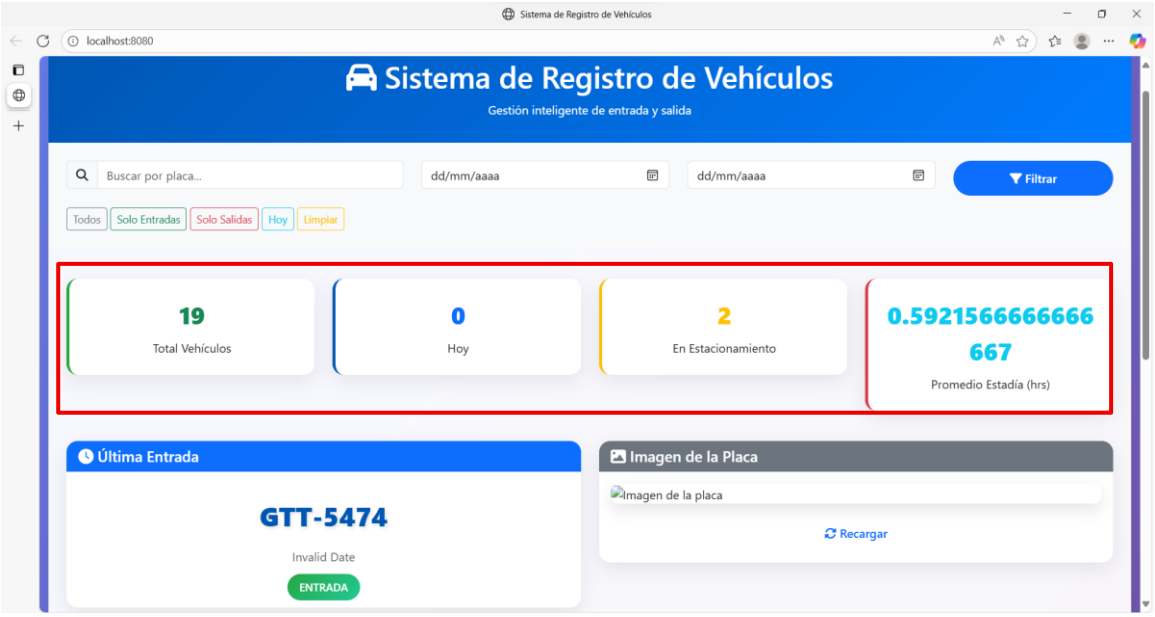


Figura 4.7–6 Indicadores de rendimiento

Fuente: Elaborado por el autor

2.2. Monitoreo de Detección Reciente y Logs

Esta sección se enfoca en la validación rápida del correcto funcionamiento del pipeline de detección y reconocimiento. El indicador Última Detección (mostrado en la Figura 4.7–10) tiene la funcionalidad de mostrar la Placa (el texto reconocido) y la Imagen de la Placa capturada del registro más reciente del sistema. Su utilidad es permitir a los administradores confirmar visualmente la calidad del reconocimiento (OCR) y la correcta captura de imágenes en tiempo real.

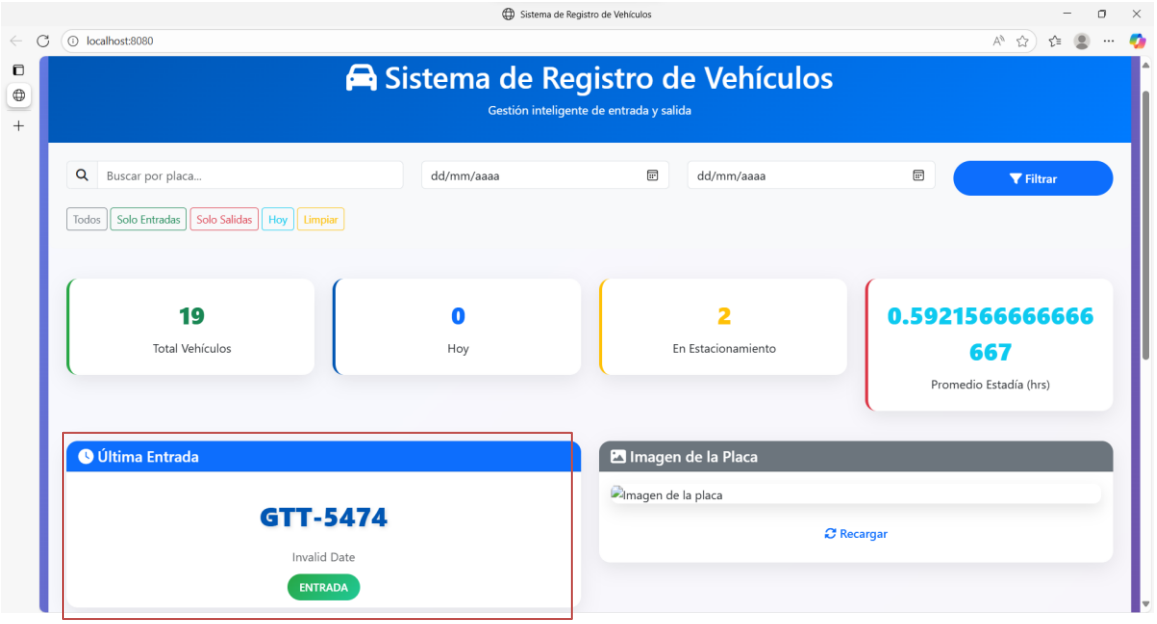


Figura 4.7–7 Placa reconocida

Fuente: Elaborado por el autor

La Tabla de Registros Históricos (mostrada en la Figura 4.7–11) tiene la funcionalidad de listar cronológicamente todos los eventos de entrada y salida, mostrando datos clave como el ID, la Placa, las Fechas/Horas de Entrada y Salida, el Tiempo Total y el Estado actual (En Estacionamiento o Salida). Su utilidad principal es ser la vista principal para monitorear el flujo de actividad y verificar el estado de los vehículos, incluyendo botones de acción para Registrar Salida manualmente o Ver Detalle.

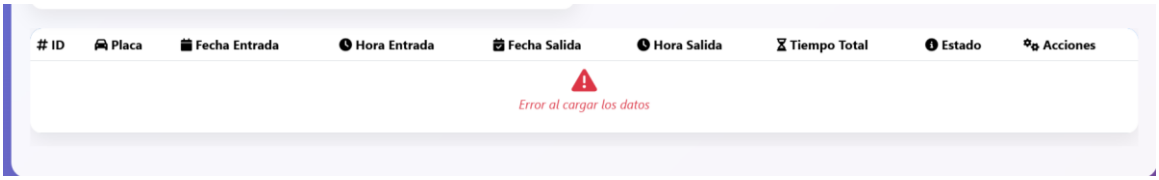


Figura 4.7–8 Tabla de registros

Fuente: Elaborado por el autor

2.3. Herramientas de Auditoría y Búsqueda

El módulo incorpora filtros esenciales para la recuperación rápida de datos. Estos Filtros de Búsqueda (mostrados en la Figura 4.7–12) permiten buscar y filtrar los registros históricos por Número de Placa (con búsqueda instantánea), por Rango

de Fechas (Inicio y Fin), y por Estado (Solo Entradas, Solo Salidas o Todos). Su utilidad principal es constituir la herramienta esencial de auditoría para la rápida recuperación de la evidencia de un evento, como cuándo entró o salió un vehículo específico.

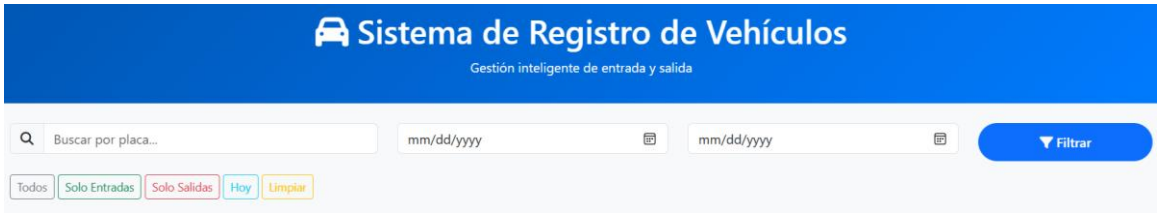


Figura 4.7–9 Filtro de búsqueda

Fuente: Elaborado por el autor

5. Conclusiones y recomendaciones

5.1. Conclusiones

Los resultados obtenidos en este proyecto de detección caracteres y matriculas vehiculares, avalan el uso de un modelo de detección de objetos como primer paso en un sistema automatizado de reconocimiento. Se ha demostrado que el enfoque de detección de objetos supera las limitaciones de las soluciones de Reconocimiento Óptico de Caracteres (OCR) puros, que mostraron un rendimiento insatisfactorio en condiciones no ideales, como se evidenció con los modelos locales y la librería EasyOCR.

El modelo al final de su entrenamiento alcanzó métricas de rendimiento muy altas: su Precisión fue de 0.985 y su Exhaustividad de 0.936, lo que demuestra una gran robustez. El mAP@0.5 de 0.968, confirmando una alta precisión en la localización de las matrículas vehiculares. Estos datos se complementan con la matriz de confusión que validó estos hallazgos, dándonos una tasa de éxito del 100% en la detección de fondo y un 94% de verdaderos positivos para las matrículas. Los resultados nos confirman el enfoque planteado, y a su vez proporcionan una base sólida y confiable para extender el proyecto a futuro.

5.2. Análisis de Viabilidad Económica

El desarrollo de un sistema ALPR (Automatic License Plate Recognition) basado en Deep Learning representa una inversión estratégica. Para justificar esta inversión, es crucial analizar los costos de hardware en diferentes escenarios de despliegue y cuantificar los beneficios operativos esperados.

En cuanto a los Costos de Hardware y Rendimiento Comparativo, la elección de la plataforma de hardware es el factor de costo más variable y depende de los requisitos de latencia y volumen de procesamiento. Por ello, se comparan tres enfoques de despliegue: PC Local (Alto Rendimiento), Embedded (Edge Computing) y Servicio en la Nube (Pago por Uso).

Plataforma de Despliegue	Ejemplo de Hardware	Costo Estimado (Inicial)	Velocidad Típica (Latencia)	Ventajas Clave
Edge Computing (Recomendado)	NVIDIA Jetson Xavier NX	USD $450 - $650	< 100 ms/Frame	Baja latencia, alto rendimiento por vatio, bajo consumo de energía.
Local/PC (Alto Rendimiento)	Intel Core i5 + GPU GTX 1650/3050	USD $800 - $1,200	< 50 ms/Frame	Máxima velocidad de inferencia, ideal para benchmarking o entornos de alto tráfico.
Nube (Pago por Uso)	AWS EC2 (t3.medium para API)	USD $40 - $100 / Mes		Alta escalabilidad, bajo costo inicial, mantenimiento centralizado.

Tabla 5.2-1 Comparacion de rendimiento y costo

Fuente: Elaborado por el autor

El Análisis del Costo Inicial de las diferentes plataformas para el sistema ALPR (Tabla 5.2-1) indica que el escenario de Edge Computing con una plataforma Jetson ofrece el mejor balance entre inversión inicial y rendimiento en tiempo real. Si bien el PC ofrece la latencia más baja, tiene un consumo energético y un costo inicial

significativamente más altos. El modelo de Nube, aunque elimina la inversión de hardware local, introduce costos recurrentes y depende críticamente de la red.

La Decisión Justificada es que, dada la prioridad de baja latencia para el control de acceso, el despliegue en un dispositivo Embedded (Jetson) es la solución económicamente más viable a largo plazo, ya que su consumo de energía es mínimo y su rendimiento es suficiente para el procesamiento de YOLOvX.

El Análisis Costo-Beneficio (ACB) de la automatización cuantifica el beneficio financiero a través del ahorro en costos operativos y la reducción de riesgos. Un Costo a Evitar (Beneficio Cuantificable) clave es la Liberación o Reasignación de Personal de Seguridad. Asumiendo que el sistema permite reducir un turno de guardia (8 horas) en un escenario de vigilancia, y estimando un costo salarial anual promedio de un guardia en USD $8,000, el ahorro potencial es de USD $8,000 anuales por turno automatizado. Otro beneficio es la Reducción de Errores de Registro y Fraude, ya que los sistemas manuales son propensos a errores de transcripción o registro. Si el sistema reduce la tasa de errores del 10% (manual) a menos del 1% (automatizado), el beneficio es la garantía de una trazabilidad perfecta, lo que reduce el riesgo financiero y legal (beneficio indirecto, pero crítico).

Componente de Ahorro	Valor Anual
Ahorro Salarial (Liberación de 1 turno)	$8,000
Costo Total del Proyecto (Inversión)	-$1,150
Beneficio Neto en el Año 1	+$6,850

Tabla 5.2-2 Calculo de retorno

Fuente: Elaborado por el autor

Finalmente, el Cálculo del Retorno de la Inversión (ROI) se basa en un costo de inversión de la solución Embedded (USD $650) más un costo de implementación (USD $500), totalizando una inversión inicial de USD $1,150. Considerando el ahorro salarial anual de $8,000 (Tabla 5.2-2), el Beneficio Neto en el Año 1 es de $6,850. Este análisis demuestra que la automatización es una inversión altamente rentable que recupera su costo en aproximadamente 1.7 meses de operación, lo que indica un Retorno de la Inversión en menos de dos meses.

5.3. Plan de Implementación Sugerido

La implementación de la solución ALPR en un entorno real requiere una metodología estructurada que se inicia con la Instalación de Hardware y Cableado. Esta etapa (1-3 días) incluye el Montaje Físico de la cámara IP (1080p/30FPS) para optimizar el ángulo de captura de la placa, la Conectividad de Red mediante cableado Ethernet hasta el dispositivo de procesamiento (Edge PC o Jetson), y la Integración de Actuadores como el relé o la interfaz de comunicación para la barrera automática y las luces indicadoras. Una vez instalado, se procede a la Calibración del Sistema y Configuración de Software (2 días), donde se realizan ajustes finos como el Ajuste Óptico de la Cámara (con la lente varifocal) para maximizar los píxeles dedicados a la placa, y la Configuración de ROI (Region of Interest) para definir Zonas de Detección que activen el procesamiento de YOLOvX solo cuando detecten movimiento, optimizando el rendimiento. A continuación, se lleva a cabo la fase de Pruebas Piloto y Capacitación (1 semana), que incluye Pruebas con Dataset de Control y Pruebas en Paralelo donde el sistema opera simultáneamente con el registro manual para validar su precisión y corregir umbrales. En esta etapa también se realiza la Capacitación del Personal en el uso de la interfaz y la gestión de la lista blanca. Finalmente, el proceso concluye con la Puesta en Marcha (Go-Live) y Monitoreo Continuo (Indefinido), donde el sistema ALPR asume la Operación Autónoma como mecanismo primario para el control de acceso, enviando comandos directamente a la barrera, y se establece el Monitoreo de KPIs para que el administrador supervise semanalmente la Tasa de Reconocimiento de Placa (PRR) y la Latencia Promedio.

5.3. Recomendaciones y futuras mejoras

Después de analizar los resultados, se encontró áreas en las cuales mejorar para el rendimiento del sistema y su posterior aplicación en escenarios reales. Las siguientes recomendaciones se presentan para avances en el proyecto. Aumentar variedad de las condiciones en el conjunto de datos (matrículas sucias, en ángulos extremos, con reflejos, etc.) Esto permitiría al modelo generalizar mejor y minimizar los falsos negativos.

Ampliar y Estandarizar el conjunto de datos con Placas vehiculares ecuatorianas, se propone el aumento de la cantidad de imágenes. Esto servirá para asegurar que el modelo esta optimizado para las características únicas de las matrículas nacionales, como lo son su forma y el tipo de fuente que se usa para los caracteres y números, mejorando a si la capacidad para generalizar y reducir los falsos negativos.

Desarrollar un Módulo de OCR Propio es el paso más crucial para un sistema completo, es el desarrollo de un módulo de OCR personalizado. Este modelo de OCR se entrenaría con los caracteres específicos de las placas de Ecuador, lo que permitiría una precisión de lectura superior a la de las librerías genéricas, que a menudo confunden caracteres debido a los diferentes tipos de tipografías.

Para asegurar la escalabilidad y robustez operativa del sistema en escenarios de alto flujo vehicular, el trabajo futuro debe centrarse en la concurrencia y el manejo múltiples placa. Es necesario optimizar la capacidad del sistema para manejar simultáneamente múltiples detecciones dentro del mismo frame, lo cual es común en tráfico congestionado. Esto se logrará revisando los parámetros de NMS (Non-Max Suppression) del modelo YOLOvX, para evitar que una detección válida sea suprimida por una adyacente, y mediante la gestión de hilos concurrente en el backend. Al modificar el backend para procesar los múltiples crops de placa en hilos paralelos, se reduce la latencia total del frame, garantizando una alta tasa de FPS operativa aun con varios vehículos esperando.

Finalmente, para la integración total y la centralización de seguridad (IoT), se proyecta una expansión del sistema. Se debe implementar la Integración con Actuadores IoT para el control físico de acceso, utilizando un módulo de relé (controlado por GPIO o MQTT) para enviar la señal binaria de abrir/cerrar a la barrera automática, y conectando luces LED o semáforos para señalizar el estado de acceso. A nivel de backend, el sistema debe exponer una API RESTful que permita la comunicación bidireccional con sistemas de gestión superiores, como un Sistema de Gestión de Edificios (BMS) o un Sistema de Planificación de Recursos Empresariales (ERP). Esta API permitiría al ERP inyectar nuevas placas en la Lista Blanca del ALPR y, a su vez, recibir los logs de acceso (Placa, Timestamp, Acción) para la auditoría centralizada. Adicionalmente, se recomienda implementar Capacidades de

Almacenamiento Distribuido (Cloud Backup) para la copia de seguridad automática de los logs de eventos y los crops de auditoría a un servicio en la nube (ej. AWS S3), garantizando la redundancia y la seguridad de los datos ante fallos del hardware local.

Bibliografía

[1] R. Zhang, «License Plate Detection and Recognition in Unconstrained Scenarios,» 2024.

[2] E. Serttaş y F. Gül, «Automatic License Plate Recognition and Visualization System with YOLO V8 Algorithm,» *Journal of Information Technologies,* vol. 18, nº 1, pp. 1-10, 2025.

[3] R. Al Batat, A. Angelopoulou, S. Premkumar, J. Hemanth y E. Kapetanios, «An End to End Automated License Plate Recognition System Using YOLO Based Vehicle and License Plate Detection with Vehicle Classification,» *Sensors,* 2022.

[4] S. Qin y S. Liu, «Towards End to end Car License Plate Location and Recognition in Unconstrained Scenarios,» 2020.

[5] Y. Lee, J. Jun, Y. Hong y M. Jeon, «Practical License Plate Recognition in Unconstrained Surveillance Systems with Adversarial Super Resolution,» 2019.

[6] T. Dingsøyr, N. B. Moe, K. Dikert y K. Rolland, «The impact of agile transformation: A multiple-case study of agile teams in large-scale software development,» *Information and Software Technology,* vol. 132, p. 106509, 2021.

[7] X. M. C. Z. B. Z. Z. S. Y. W. Liru Hua, «Recognition of vehicle license plates in highway scenes with deep fusion network and connectionist temporal classification,» *ET Image Processing,* vol. 18, nº 2, pp. 123-134, 2024.

[8] N. S. T. H. Victor Nascimento Ribeiro, «Efficient Video-Based ALPR System Using YOLO and Visual Rhythm,» *arXiv preprint arXiv:2501.02270,* vol. 2501, nº 02270, 2025.

[9] M. M. A. S. a. N. E. B. A. A. Ismail, «A dual-stage system for real-time license plate detection and recognition on mobile security robots,» *Robotica,* vol. 43, pp. 1981-2002, 2025.

[10] S.-R. Wang, W.-K. Tai, H.-Y. Shih y Z.-Y. Shen, «End-to-End High Accuracy License Plate Recognition Based on Depthwise Separable Convolution Networks,» *arXiv preprint arXiv:2501.02270,* 2022.

[11] J. Liang, «Automatic License Plate Recognition in Foggy Conditions: A Survey of Image Dehazing and License Plate Recognition Techniques,» *Applied and Computational Engineering,* vol. 163, 2025.

[12] S. E. Whang, Y. Roh, H. Song y J.-G. Lee, «Data Collection and Quality Challenges in Deep Learning: A Data-Centric AI Perspective,» *arXiv,* vol. 3, nº

2112.06409, 2022.

[13] D. F. P. P. I. &. G. E. Fernández-Llorca, «A Review of Image Annotation Tools for Object Detection,» *International Conference on Applied Artificial Intelligence and Computing (ICAAIC),* pp. 976-982, 2022.

[14] CVAT.ai, «Best Image Annotation Tools 2024,» CVAT.IA, 2024. [En línea]. Available: https://cvat.ai.

[15] M. M. S. B. J. H. P. P. D. R. P. D. a. C. L. Z. T.-Y. Lin, «Microsoft COCO: Common Objects in Contex,» *European Conference on Computer Vision (ECCV),* 2024.

[16] X. W. L. Z. Y. e. a. Zhao, «A review of convolutional neural networks in computer vision,» *Artif Intell Rev,* p. 99, 2024.

[17] viso.ai, «visio.ai,» Convolutional Neural Networks (CNNs): A Deep Dive, 2 October 2024. [En línea]. Available: https://viso.ai/deep-learning/convolutional-neural-networks/.

[18] M. A. Karne, R. Karne y K. K. Vaigandla, «Convolutional Neural Networks for Object Detection and Recognition,» *Journal of Artificial Intelligence Machine Learning and Neural Network,* vol. 3, pp. 1-13, 2023.

[19] W. A. Shobaki y M. Milanova, «A Comparative Study of YOLO, SSD, Faster R-CNN, and More for Optimized Eye-Gaze Writing,» *Electronics (Basel),* vol. 7, nº 2, p. 47, 2024.

[20] S. Ren, K. He, R. Girshick y J. Sun, «Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,» ArXiv, 2020.

[21] R. Padilla, W. L. Passos, T. L. B. Dias, S. L. Netto y E. A. B. Silva, «A Comparative Analysis of Object Detection Metrics with a Companion Open-Source Toolkit,» *Electronics (Basel),* vol. 10, nº 3, 2021.

[22] X. Wang, P.-C. Chen, Y.-C. Lin y C.-H. Hsieh, «A Deep Learning Framework of Super Resolution for License Plate Recognition in Surveillance System,» *Mathematics,* vol. 13, nº 10, p. 1673, 2025.

[23] T. L. J. C. L. C. Y. L. D. F. C. Z. Z. L. F. W. Minghao Li, «TrOCR: Transformer-based Optical Character Recognition with Pre-trained Models,» arXiv, 2021.

[24] R. L. J. d. A. L. W. R. S. D. M. Valfride Nascimento, «Super-Resolution of License Plate Images Using Attention Modules and Sub-Pixel Convolution Layers,» ArXiv, 2023.

[25] C.-H. Wang, «Using Super-Resolution Imaging for Recognition of Low-Resolution Blurred License Plates: A Comparative Study of Real-ESRGAN, A-ESRGAN, and StarSRGAN,» arXiv, 2024.

[26] B. B. A. A. A. K. A. M. A. Sawsan AlHalawani, «License Plate Super-Resolution Using Diffusion Models,» arXiv, https://doi.org/10.3390/electronics13132670, 2023.

[27] M. D. S. F. M. A. K. M. A. N. Musa Dildar Ahmed Cheema, «Adapting multilingual vision language transformers for low-resource Urdu optical character recognition (OCR),» *PeerJ Computer Science,* 2024.

[28] B. A. L. a. T. A. V. S. R. Kumar, «A Comprehensive Review of Character Recognition in License Plate Recognition Systems,» *International Journal of*

Computer Science and Engineering, vol. 12, pp. 1-10, 2024.

[29] J. M. C.-P. M. V. M.-C. a. H. Y. Y. M. E. Morocho-Cayamcela, «End-to-End License Plate Recognition System for an Efficient Deployment in Surveillance Scenarios,» *ResearchGate preprint,* 2022.

[30] S. H. ,. L. ,. C. ,. H. a. Z. T. Lingbing Tao, «A Real-Time License Plate Detection and Recognition Model in Unconstrained Scenarios,» *Journal of Real-Time Image Processing,* vol. 18, pp. 24-35, 2024.

[31] A. R. L. a. T. J. L. R. M. F. A. El-Hadad, «A Novel Cloud-Based IoT System for Smart Parking Management,» *IEEE Transactions on Intelligent Transportation Systems,* vol. 22, nº 1, pp. 123-164, 2021.

[32] G. -A. N. a. M. S. F. Mohammadi, «A Real-Time Cloud-Based Intelligent Car Parking System for Smart Cities,» *EEE Access,* vol. 9, pp. 112185-112198, 2021.

[33] E. B. B. y. M. Z. Mora, ««DISEÑO DE UN ALGORITMO DE RECONOCIMIENTO DE PLACAS VEHICULARES ECUATORIANAS USANDO REDES NEURONALES CONVOLUCIONALES,» *JSR,* vol. 5, nº 4, pp. 76-86, 2020.

[34] D. O. a. D. Torres, «Sistema de reconocimiento automático de placas vehiculares para condiciones del Ecuador,» Universidad Politécnica Salesiana, Quito,Ecuador, 2020.

[35] S. K. D. Khokhar, «Integrating YOLOv8 and CSPBottleneck based CNN for enhanced license plate character recognition,» *J Real-Time Image Proc,* vol. 21, p. 168, 2024.

[36] M. A. M. F. A. E. M. S. J. H. Abolfazl Younesi, «A Comprehensive Survey of Convolutions in Deep Learning: Applications, Challenges, and Future Trends,» *arXiv preprint arXiv:2402.15490v2,* 2024.

[37] M. M. Taye, «Theoretical Understanding of Convolutional Neural Network: Concepts, Architectures, Applications, Future Directions,» *Computation,* vol. 11, nº 3, p. 52, 2023.

[38] L. Z. J. H. A. e. a. Alzubaidi, «Review of deep learning: concepts, CNN architectures, challenges, applications, future directions,» . *J Big Data 8,* p. 53, 2021.

[39] O. M. y. L. Zaniolo, «On the use of CNNs with patterned stride for medical image analysis,» *Machine Graphics & Vision,* vol. 30, pp. 3-22, 2021.

[40] J. C. C.-T. J. N.-V. C. P. A. G. S.-T. Gabriela Rangel, «A Survey on Convolutional Neural Networks and Their Performance Limitations in Image Recognition Tasks,» *Sonsors,* nº 1, 2024.

[41] L. A. A. D. A. A. A. A. A. H. A. K. R. Mofadal Alymani, «Enabling smart parking for smart cities using Internet of Things (IoT) and machine learning,» *PeerJ Computer Science,* vol. 88, pp. 1132-1138, 2025.

AUTORIZACIÓN DE PUBLICACIÓN EN EL REPOSITORIO INSTITUCIONAL

Jordy Gilmar Luzuriaga Sánchez portador(a) de la cédula de ciudadanía Nº **0930804760**. En calidad de autor/a y titular de los derechos patrimoniales del proyecto de titulación **"Desarrollo de un prototipo para automatizar el registro de acceso vehicular mediante reconocimiento de placas utilizando redes neuronales"** de conformidad a lo establecido en el artículo 114 Código Orgánico de la Economía Social de los Conocimientos, Creatividad e Innovación, reconozco a favor de la Universidad Católica de Cuenca una licencia gratuita, intransferible y no exclusiva para el uso no comercial de la obra, con fines estrictamente académicos y no comerciales. Autorizo además a la Universidad Católica de Cuenca, para que realice la publicación de éste proyecto de titulación en el Repositorio Institucional de conformidad a lo dispuesto en el artículo 144 de la Ley Orgánica de Educación Superior.

La Troncal, **16 de Diciembre del 2025**

F: .....

Jordy Gilmar Luzuriaga Sanchez

C.I. **0930804760**