UNIVERSIDAD CATÓLICA SANTO TORIBIO DE MOGROVEJO

FACULTAD DE INGENIERÍA

ESCUELA DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

Desarrollo e implementación de una solución basada en inteligencia artificial para la categorización de vehículos en el Proyecto Especial de Infraestructura de Transporte Nacional – Provías Nacional

TESIS PARA OPTAR EL TÍTULO DE

INGENIERO DE SISTEMAS Y COMPUTACIÓN

AUTOR

Pedro Ricardo Pejerrey Gomez

ASESOR

Miguel Orlando Diaz Vidarte

https://orcid.org/0000-0002-7403-0304

Chiclayo, 2026

Desarrollo e implementación de una solución basada en inteligencia artificial para la categorización de vehículos en el Proyecto Especial de Infraestructura de Transporte Nacional – Provías Nacional

PRESENTADA POR

Pedro Ricardo Pejerrey Gomez

A la Facultad de Ingeniería de la
Universidad Católica Santo Toribio de Mogrovejo
para optar el título de

INGENIERO DE SISTEMAS Y COMPUTACIÓN

APROBADA POR

Héctor Miguel Zelada Valdivieso
PRESIDENTE

Huilder Juanito Mera Montenegro
SECRETARIO

Miguel Orlando Díaz Vidarte
VOCAL

Desarrollo e implementación de una solución basada en inteligencia artificial para la categorización de vehículos en el Proyecto Especial de Infraestructura de Transporte Nacional – Provías Nacional

INFORME DE ORIGINALIDAD

10%	10%	2%	4%
INDICE DE SIMILITUD	FUENTES DE INTERNET	PUBLICACIONES	TRABAJOS DEL ESTUDIANTE

FUENTES PRIMARIAS

1	tesis.usat.edu.pe Fuente de Internet	1%
2	repositorio.uchile.cl Fuente de Internet	1%
3	hdl.handle.net Fuente de Internet	<1%
4	Submitted to Consorcio CIXUG Trabajo del estudiante	<1%
5	cict.umcc.cu Fuente de Internet	<1%
6	github.com Fuente de Internet	<1%
7	docplayer.es Fuente de Internet	<1%
8	www.slideshare.net Fuente de Internet	<1%

Índice

Resumen

La clasificación de vehículos en las estaciones de peaje de PROVIAS NACIONAL se realiza mediante sensores instalados de manera invasiva en la superficie de la losa de concreto. Aunque este método resulta funcional, presenta inconvenientes, ya que el desgaste generado por el flujo vehicular reduce progresivamente la efectividad de los sensores. En el mercado existen alternativas menos invasivas, como los sensores infrarrojos; sin embargo, para PROVIAS NACIONAL su implementación supondría un incremento considerable en los costos, al requerir modificaciones tanto en la infraestructura como en el software. Este estudio surgió de la necesidad de incorporar métodos innovadores que aprovecharan las tecnologías y técnicas actuales, con el propósito de reemplazar los enfoques tradicionales en la identificación y categorización de vehículos dentro de las estaciones de peaje. En consecuencia, el objetivo principal de la investigación fue desarrollar e implementar una solución basada en inteligencia artificial para llevar a cabo dicha clasificación. Para el desarrollo del modelo se adoptó la metodología CRISP-DM, mientras que la implementación de la solución se ejecutó siguiendo los principios de la metodología ágil XP. Los resultados obtenidos demostraron que la solución propuesta alcanzó una precisión del 95% en la clasificación de vehículos, superando ampliamente a los métodos tradicionales. Asimismo, se evidenció una reducción en los costos de mantenimiento. Estos hallazgos confirmaron la viabilidad técnica y económica de implementar soluciones basadas en inteligencia artificial en el contexto de PROVIAS NACIONAL.

Palabras clave: Inteligencia artificial, redes neuronales artificiales, procesamiento de imágenes, visión por computadora, aprendizaje automático, aprendizaje profundo.

Abstract

Vehicle classification at PROVIAS NACIONAL toll plazas is performed using sensors installed invasively on the surface of the concrete slab. Although this method is functional, it has drawbacks, as wear and tear caused by traffic flow progressively reduces the sensors' effectiveness. Less invasive alternatives, such as infrared sensors, are available on the market; however, for PROVIAS NACIONAL, their implementation would entail a considerable increase in costs, requiring modifications to both the infrastructure and the software. This study arose from the need to incorporate innovative methods that leverage current technologies and techniques, with the aim of replacing traditional approaches to vehicle identification and categorization within toll plazas. Consequently, the main objective of the research was to develop and implement an artificial intelligence-based solution for this classification. The CRISP-DM methodology was adopted for the development of the model, while the solution was implemented following the principles of the Agile XP methodology. The results obtained demonstrated that the proposed solution achieved 95% accuracy in vehicle classification, significantly outperforming traditional methods. A reduction in maintenance costs was also evident. These findings confirmed the technical and economic feasibility of implementing artificial intelligence-based solutions in the context of PROVIAS NATIONAL.

Keywords: Artificial intelligence, artificial neural networks, image processing, computer vision, machine learning, deep learning.

Introducción

PROVIAS NACIONAL, entidad pública vinculada al Ministerio de Transportes y Comunicaciones (MTC), está encargada de gestionar y mantener la red vial nacional no concesionada, es decir, aquellas vías que no han sido transferidas a empresas privadas y cuya conservación recae en el Estado [1]. Estas vías son clave para el transporte nacional, ya que conectan regiones y permiten el flujo eficiente de personas y mercancías. Sin embargo, su mantenimiento depende exclusivamente de recursos públicos [2]. Uno de los principales mecanismos para financiar estas labores es el cobro en estaciones de peaje, cuyos ingresos se destinan a la conservación y mejora de dicha red vial [3]. PROVIAS NACIONAL aplica un sistema de tarifas basado en la clasificación de vehículos, conforme al Reglamento Nacional de Vehículos aprobado por el D.S. Nº 058-2003-MTC [4].

Actualmente, la entidad utiliza un software propietario para el proceso de cobro, en el que un operador ingresa la placa de un vehículo para identificar su categoría y tarifa correspondiente. Si la información obtenida presenta inconsistencias, el operador tiene la facultad de realizar las correcciones pertinentes antes de emitir el ticket, asegurando así la precisión del cobro. Una vez emitido el ticket, se implementa un mecanismo de control que valida la categoría asignada al vehículo. Este mecanismo utiliza sensores de contacto instalados en una losa de concreto [5], que son activados por la presión que ejerce el paso de las llantas de los vehículos. Al ser activados, los sensores generan señales eléctricas que permiten contar el número de ejes. Cabe señalar que estos sensores tienen una vida útil estimada de 1 000 000 de ciclos.

Debido a que los sensores están expuestos y en contacto constante con las llantas de los vehículos, sufren un desgaste progresivo. Además, factores climáticos como la humedad, el calor y las precipitaciones contribuyen a una reducción gradual en la capacidad de detección de los ejes a lo largo del tiempo. Otro factor que impacta en el índice de detección es el estado de la losa de concreto. Cuando circulan vehículos de alto tonelaje, la losa experimenta daños y deformaciones alrededor de los sensores, lo que disminuye la presión y el contacto necesarios para activarlos correctamente. Según las estadísticas registradas en enero de 2025, el peaje con mayor afluencia vehicular generó un promedio de 276 000 activaciones por sensor. Proyectando este ritmo de uso y considerando que los sensores fueron instalados el 1 de enero, se estima que será necesario reemplazarlos aproximadamente cada cuatro meses para garantizar un funcionamiento óptimo.

En varios países vecinos, entre ellos Chile, Brasil, Ecuador y Bolivia, además de emplear sensores de contacto, se utilizan cortinas de sensores ópticos para clasificar vehículos [6] [7].

Esta tecnología también es conocida en Perú, pero no se aplica en peajes gestionados por PROVIAS NACIONAL.

La presente investigación se justifica en distintos ámbitos. En el aspecto social, el proceso de clasificación será mucho más rápido y eficiente, beneficiando a los ciudadanos. En el aspecto de investigación, permitirá implementar un nuevo método para el control e identificación de vehículos basado en inteligencia artificial. En el ámbito económico, se requerirá una menor inversión del tesoro público para el control en la clasificación de los vehículos. Finalmente, en el aspecto tecnológico, se integrarán distintas herramientas, tales como visión artificial con hardware de cámaras IP y lenguajes de programación con inteligencia artificial.

¿Cómo el desarrollo e implementación de una solución basada en inteligencia artificial puede mejorar la categorización de vehículos en el Proyecto Especial de Infraestructura de Transporte Nacional - PROVIAS NACIONAL?

En ese marco, el objetivo general es desarrollar e implementar una solución basada en inteligencia artificial para la categorización de vehículos en el Proyecto Especial de Infraestructura de Transporte Nacional - PROVIAS NACIONAL. Para ello, se plantean cuatro objetivos específicos: entrenar un modelo de inteligencia artificial que mejore la precisión y velocidad del proceso de clasificación en tiempo real; desarrollar una aplicación de escritorio que integre dicho modelo y ofrezca una interfaz intuitiva y funcional; determinar la infraestructura de TI necesaria para la operación de la solución, incluyendo servidores, red local y cámaras; y evaluar el rendimiento del modelo propuesto en comparación con los métodos tradicionales, midiendo eficiencia, precisión y capacidad de adaptación en diversas condiciones operativas.

Revisión de literatura

Antecedentes:

La investigación realizada por *Pérez* [8] se toma como primer antecedente, para el autor el monitoreo de vehículos en vías urbanas e interurbanas representa un problema que se ve reflejado en altos costos operativos y logísticos, esto debido a que se deben instalar grandes redes de sensores en las vías. Como propuesta de solución se hace uso de modelos basados en inteligencia artificial para la detección y clasificación de vehículos en video. Producto de la investigación el autor concluye que el uso de redes neuronales convolucionales resulto ser satisfactorio, puesto se logró realizar un sistema independiente, robusto y escalable haciendo uso del modelo YOLOv3.

Como segundo antecedente, se toma en cuenta el estudio presentado por *García* [9], el cual aborda la detección en tiempo real de tres categorías de vehículos que circulan

por las vías: motocicletas, vehículos particulares y camiones. La información obtenida a través de este proceso tiene aplicaciones diversas, entre las que destacan la gestión del tráfico y el desarrollo de tecnologías para la conducción autónoma. A partir de los resultados obtenidos, el autor concluye que, mediante el uso de inteligencia artificial y la combinación de datos provenientes de un sensor LIDAR 3D con una cámara, fue posible diseñar un algoritmo innovador, eficaz tanto en precisión como en velocidad. Entre sus principales atributos, se resalta su capacidad para operar en tiempo real, su amplio alcance, la precisión en la detección y la capacidad de reunir una amplia cantidad de datos útiles sobre los vehículos para distintos propósitos.

Como tercer antecedente, se considera el estudio desarrollado por *Coanqui et al* [10], el cual resalta la importancia del conteo y la clasificación vehicular para una adecuada planificación de obras viales, tales como el diseño y mantenimiento de pavimentos, así como proyectos de señalización, iluminación, entre otros. Como parte de esta investigación, se plantea aplicar inteligencia artificial como una alternativa al enfoque tradicional basado en hardware. Para abordar esta problemática, el autor diseñó dos métodos: el primero emplea un algoritmo de inteligencia artificial para la detección y el seguimiento de vehículos en tiempo real; el segundo, utiliza técnicas de visión artificial para clasificar los vehículos de acuerdo con sus áreas promedio. Tras examinar los resultados obtenidos, se determina que el enfoque basado en visión artificial ofrece un desempeño superior en términos de precisión.

Según lo expuesto por *Ocampo* [11], a pesar de los avances conseguidos en la mejora de infraestructura vial en Colombia, el alto volumen de tráfico en los accesos a las ciudades continúa siendo un problema significativo, especialmente en las estaciones de peaje, donde se generan frecuentes congestiones. Ante esta situación, el autor propone un sistema que permita la clasificación de vehículos y el reconocimiento de matrículas, ambos apoyados en técnicas de inteligencia artificial. Para la tarea de clasificación, se emplearon dos modelos de redes neuronales: ResNet y MobileNet, considerando variaciones en características visuales como la intensidad del color, acercamientos y rotaciones. Los resultados mostraron un mejor desempeño por parte del modelo ResNet. En cuanto al reconocimiento de matrículas, se aplicó una metodología basada en reconocimiento óptico de caracteres (OCR), alcanzando niveles de precisión aceptables.

Como antecedente final, se considera el estudio desarrollado por *Córdova et al* [12], en el cual el autor plantea que los semáforos son una de las principales fuentes de la

congestión vehicular, debido a que operan con programaciones fijas que no responden a variaciones imprevistas en el flujo de vehículos. El autor destaca la necesidad de implementar un sistema de semaforización adaptativo, capaz de ajustar los tiempos de señalización mediante el uso de inteligencia artificial. Para ello, se integró una cámara que permite realizar el conteo de vehículos en tiempo real, y, a través de la comunicación con una plataforma de desarrollo basada en PyCharm, se logró optimizar la programación de los semáforos de acuerdo con la demanda vehicular.

Bases teóricas científicas:

Inteligencia artificial:

Según *Ponce et al* [13] el apelativo inteligencia artificial se remonta a la conferencia realizada en Darmouth College en la que múltiples científicos e investigadores en el área de las ciencias computacionales se reunieron para discutir la posibilidad de construir maquinas inteligentes, sin embargo en la actualidad definir el termino inteligencia representa todo un reto debido a que no se ha establecido una definición clara y exclusiva de lo que significa serlo, *Benítez et al* [14] añade que el cerebro humano destaca como principal exponente de inteligencia, ello se debe en gran medida a su capacidad para identificar y reconocer patrones complejos de forma eficiente, del mismo modo *López et al* [15] expresa que es necesario investigar la naturaleza de la mente humana para que a través de conceptos y técnicas se logre dotar a una máquina de un nuevo tipo de inteligencia.

Los tres autores sostienen una postura común al afirmar que la inteligencia artificial busca crear sistemas que puedan pensar, aprender y enfrentar problemas como lo haría un ser humano.

Aprendizaje automático:

De acuerdo con *Soria et al* [16], el aprendizaje automático es el campo de estudio que, mediante modelos, da la posibilidad a las máquinas de asimilar a partir del procesamiento de datos, sin necesidad de programar reglas específicas, a diferencia de los enfoques tradicionales, en los que cada acción debe ser explícitamente codificada. *Sánchez et al* [17] considera que el desarrollo de esta disciplina se debe en gran medida al incremento de la capacidad de cálculo de los equipos informáticos. *Bobadilla* [18] destaca que se pueden clasificar 4 tipos de aprendizaje.

1. El aprendizaje supervisado se basa en la utilización de conjuntos de datos que incluyen tanto las entradas como las salidas esperadas, permitiendo a los modelos aprender la relación que existe entre ambas.

2. El aprendizaje no supervisado se enfoca en analizar los datos sin etiquetas predefinidas, con el fin de identificar patrones, agrupamientos o estructuras subyacentes en el conjunto.
3. El aprendizaje por refuerzo se inspira en mecanismos de recompensa, donde un agente lleva a cabo decisiones en un entorno determinado y obtiene retroalimentación a través de penalizaciones o recompensas, optimizando así su comportamiento condicionado por las consecuencias derivadas de sus acciones.
4. El aprendizaje semi-supervisado, el cual combina elementos de los enfoques supervisado y no supervisado, ya que emplea una cantidad limitada de datos etiquetados junto con una gran cantidad de datos sin etiquetar, permitiendo una mejora significativa en el rendimiento con un menor costo de anotación de datos.

Para el desarrollo del aprendizaje automático *Sánchez et al* [17] expone 4 técnicas principales, entre ellas menciona.

1. El aprendizaje evolutivo inspirado en principios biológicos como la mutación y la selección natural para resolver problemas de optimización.
2. El razonamiento basado en casos aplica soluciones previas a situaciones similares.
3. El aprendizaje inductivo permite generalizar a partir de ejemplos concretos.
4. Las redes neuronales artificiales, por su parte, simulan el comportamiento de las neuronas humanas, facilitando la modelación de relaciones no lineales y complejas.

Aprendizaje profundo:

En opinión de *Bosch et al* [19], el aprendizaje profundo nace como un área dentro del aprendizaje automático en donde los modelos no son capaces de modelar ideas abstractas y complejas. *Torres* [20] refuerza esta idea al mencionar que en el aprendizaje profundo los modelos procesan la información en múltiples niveles o capas, aprendiendo desde rasgos simples (como formas o bordes) hasta conceptos abstractos como escenas y objetos completos.

Para *Bobadilla* [18], el aprendizaje profundo está fuertemente asociado al concepto de redes neuronales, en donde se destacan dos arquitecturas principalmente. Por un lado las redes neuronales convolucionales, eficaces en la detección de patrones espaciales, muy utilizadas en reconocimiento de imágenes y diagnóstico médico; y las redes

neuronales recurrentes, que procesan secuencias temporales, fundamentales en tareas como traducción automática, análisis de sentimiento o procesamiento de lenguaje natural (ver anexo 1).

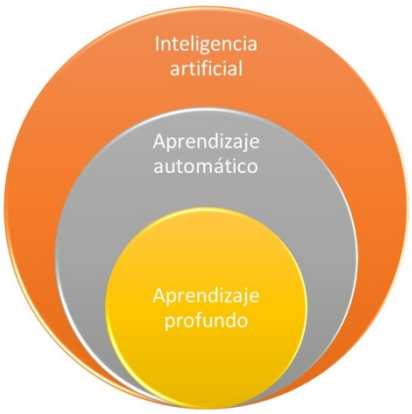


Fig. 1. Relación de pertenencia de aprendizaje profundo según Yepes [21].

Redes neuronales artificiales:

Ruiz et al [22] señala que las redes neuronales artificiales constituyen un modelo computacional basado en el funcionamiento del cerebro humano. Se componen de unidades llamadas neuronas artificiales, interconectadas entre sí, que permiten identificar patrones, tomar decisiones y aprender a partir de los datos. *Lara* [23] detalla que los modelos de neuronas artificiales comparten cuatro componentes básicos esenciales.

1. El receptor, que recibe señales de entrada, ajustadas mediante un peso que modula su importancia.
2. El sumador, que combina estas señales de forma ponderada.
3. La función de activación, que determina si la neurona debe activarse, y que varía según la red o tarea.
4. El elemento de salida transmite la señal activada a otras neuronas o al entorno externo.

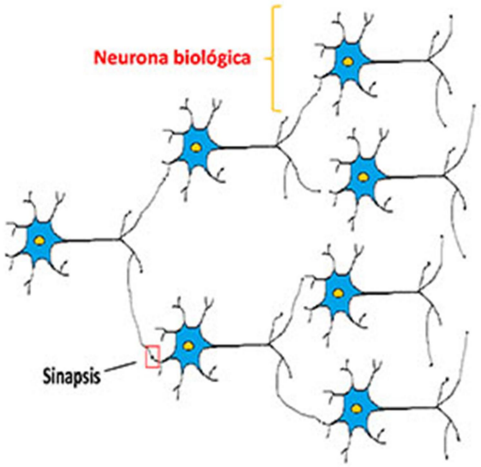


Fig. 2. Red neuronal biológica según Franco el at. [24].

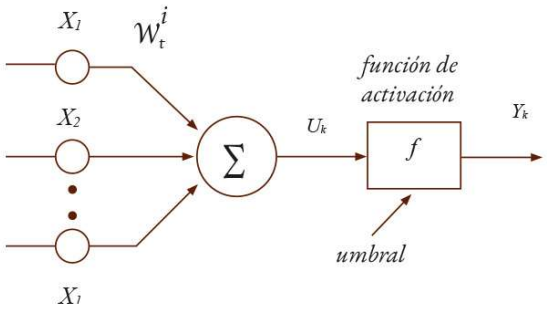


Fig. 3. Modelo de neurona artificial según Incio et al. [25].

En palabras de *Hilera et al* [26], las redes neuronales artificiales están estructuradas en capas, compuestas por múltiples neuronas conectadas. La cantidad de neuronas y capas se determina según la complejidad del problema a resolver. Generalmente, se distinguen tres tipos.

1. La capa de entrada obtiene datos del entorno y los envía al sistema para su procesamiento.
2. Las capas ocultas, situadas entre la entrada y la salida, constituyen el núcleo operativo, donde se realizan los cálculos que permiten identificar patrones o relaciones internas en los datos. Estas capas pueden ser una o múltiples, dependiendo del diseño del modelo.
3. La capa de salida emite los resultados del procesamiento hacia el exterior, adaptándolos a la tarea correspondiente.

La manera en que estas capas se conectan e interactúan define el rendimiento del modelo, influyendo directamente en su capacidad para aprender, generalizar y resolver tareas del aprendizaje automático.

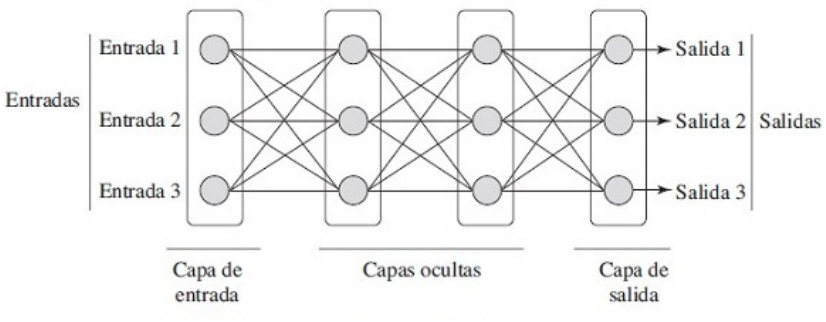


Fig. 4. Estructura de red neuronal artificial Sánchez [27].

Aprendizaje en redes neuronales artificiales:

Basogain [28] sostiene que las redes neuronales artificiales muestran un notable potencial para adquirir conocimientos, desarrollado a través de un proceso llamado entrenamiento. Este proceso implica modificar los pesos de las conexiones neuronales para que, ante ciertas entradas, la red genere salidas coherentes o deseadas. El entrenamiento se clasifica en supervisado y no supervisado.

En el aprendizaje supervisado, un tutor externo proporciona las salidas correctas para cada entrada. La red compara estas salidas con las propias, generando un error que guía la modificación de los pesos para minimizar dicha discrepancia. Entre sus principales métodos se encuentran:

- Corrección de error, donde los valores de los pesos se ajustan basándose en la diferencia entre la salida esperada y la obtenida.
- Refuerzo, que emplea señales indicadoras del acierto o error sin necesidad de conocer la salida exacta.
- Aprendizaje estocástico, que introduce ajustes aleatorios en los pesos para evaluar su impacto en la salida.

El aprendizaje no supervisado no necesita respuestas esperadas ni tutor. La red analiza directamente los datos y los organiza internamente en categorías. Entre sus principales métodos se encuentran:

- Hebbiano, que modifica los pesos según la correlación entre activaciones de neuronas conectadas.
- Competitivo y cooperativo, donde las neuronas compiten por representar patrones específicos y colaboran para una mejor representación conjunta.

Redes neuronales convolucionales:

Torres [20] compara las redes neuronales convolucionales con la manera en que el cerebro humano procesa e interpreta estímulos visuales, al inferir patrones complejos como rostros a partir de rasgos individuales. *Bosch et al* [19] señala que estas redes están optimizadas con el propósito de analizar datos estructurados a manera de cuadrícula, como las imágenes, gracias a su capacidad para ajustarse a variaciones espaciales (como posición, tamaño y orientación).

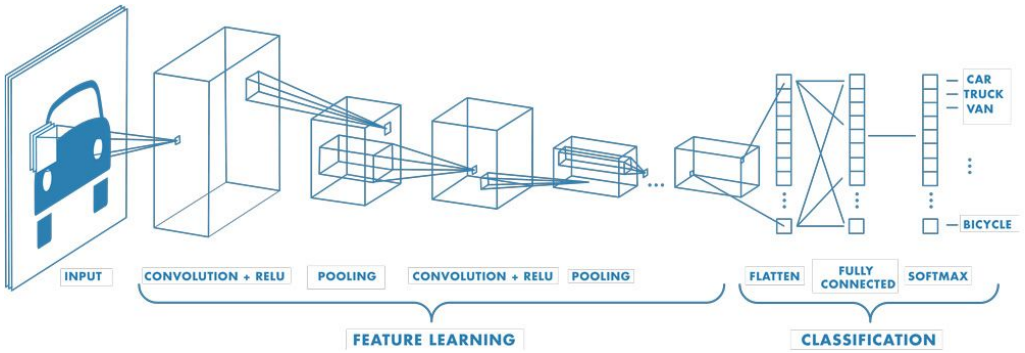


Fig. 5. Red neuronal convolucional según Mathworks [29].

Esto se logra mediante la operación de convolución, que extrae rasgos locales aplicando filtros, simulando parcialmente el funcionamiento del sistema visual humano.

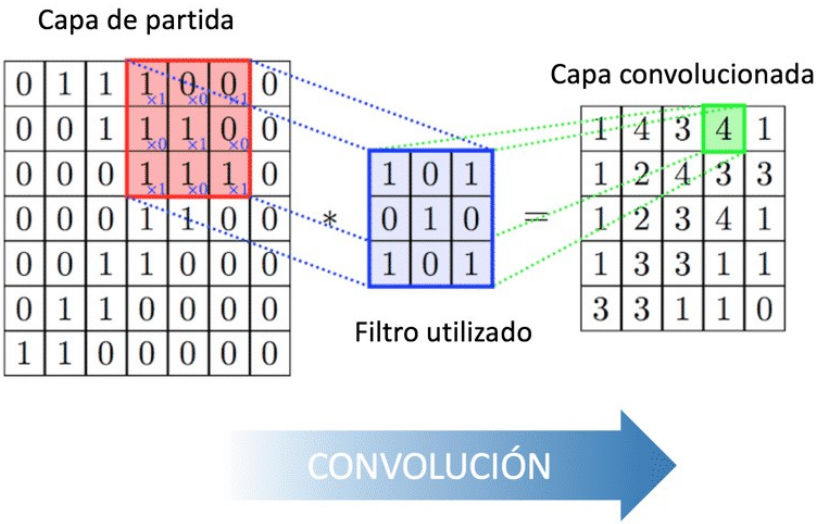


Fig. 6. Proceso de convolución Calvo [30].

Capas en redes neuronales convolucionales:

De acuerdo con *IBM* [31], las redes neuronales convolucionales se componen de tres tipos esenciales de capas, que actúan en conjunto para transformar progresivamente una imagen de entrada en una representación abstracta útil para la clasificación. Cada una cumple una función distinta y trabaja de manera secuencial para extraer, simplificar y utilizar las características detectadas.

- Capa convolucional: Es el primer componente funcional de la red y está diseñado para identificar características locales presentes en los datos, como líneas, texturas o colores. Esta capa ejecuta la operación central del modelo, que consiste en aplicar filtros sobre la imagen para generar representaciones intermedias.
- Capa de agrupación (Pooling): También conocida como capa de submuestreo, tiene como objetivo reducir el tamaño de las representaciones generadas en la etapa anterior. Esto disminuye el número de parámetros y el costo computacional sin perder información esencial.
- Capa totalmente conectada: Representa la fase final del procesamiento. Aquí, cada unidad está enlazada con todas las unidades de la capa anterior, lo que permite integrar toda la información extraída para producir una decisión final. Esta capa es responsable de la clasificación.

Proceso de reconocimiento de imágenes:

De acuerdo con *Barrios* [32], para que una red neuronal pueda analizar una imagen, esta debe convertirse primero en una matriz numérica. En dicho formato, cada celda representa un píxel y su valor indica la intensidad luminosa o cromática correspondiente. Por ejemplo, una imagen de 28 por 28 píxeles genera una matriz de igual tamaño, en la que cada valor codifica la información visual.

El volumen de datos que debe ser procesado depende del tipo de imagen. En caso de que la imagen sea en escala de grises, solo se requiere un canal de información, lo que implica que el total de valores será equivalente al número de píxeles. En cambio, para imágenes a color, se utilizan tres canales correspondientes a los componentes rojo, verde y azul (modelo RGB), lo que multiplica por tres la cantidad de datos requeridos.

En la práctica, las imágenes con baja resolución no son comunes en aplicaciones avanzadas de visión por computadora, ya que se necesita un nivel de detalle adecuado para lograr una identificación precisa de objetos. Esto implica que, frecuentemente, se trabaja con imágenes de alta resolución, lo que incrementa notablemente la complejidad computacional.

De la misma forma *Bosch et al* [19] advierte que el tamaño de estas imágenes puede suponer un desafío importante. Por ejemplo, una imagen de 1000 por 1000 píxeles en formato RGB requiere procesar tres millones de valores. Este volumen de datos exige

tanto una alta capacidad de memoria como una infraestructura adecuada para su análisis en tiempo real.

Ante esta situación, los mismos autores señalan que aplicar métodos para reducir la cantidad de datos sin comprometer su valor informativo resulta fundamental. Estrategias como el uso de capas de agrupación o la adopción de arquitecturas optimizadas permiten gestionar imágenes complejas de forma eficiente, manteniendo la precisión del modelo y reduciendo la demanda de recursos computacionales.

Fig. 7. Canales RGB de imagen según Reader [33].

Metodología CRISP-DM:

Para IBM [34], CRISP-DM es una metodología ampliamente utilizada en proyectos de minería de datos, esta metodología no es lineal, sino iterativa, lo que permite volver a fases anteriores para refinar el análisis. Su fortaleza radica en su enfoque sistemático, que facilita la transformación de grandes volúmenes de datos en conocimiento útil para la toma de decisiones. CRISP-DM promueve la comprensión profunda del contexto del negocio como punto de partida esencial, asegurando que los modelos generados estén alineados con los objetivos estratégicos de la organización.

Metodología ágil:

Según Fernández [35], XP es una metodología ágil orientada al desarrollo de software que promueve una mejora continua, entre sus principales fortalezas menciona que cuenta con una alta adaptabilidad a los cambios de requerimientos y una estrecha colaboración con el cliente. XP se caracteriza por ciclos de desarrollo muy cortos, entregas frecuentes y retroalimentación constante, todo con el objetivo de aumentar la calidad del software.

Materiales y métodos

Tipo de investigación:

La presente investigación será del tipo aplicado y experimental. Es aplicado porque en base a teorías generales existentes, se destinarán esfuerzos usando nuevos métodos o formas para la resolución de un problema existente específico en PROVIAS

NACIONAL. Asimismo, es experimental porque se manipularán variables, que afecten el comportamiento dentro de la estructura del modelo de inteligencia artificial, para evaluar su impacto sobre el desempeño [36].

Métodos de investigación:

TABLA I
MÉTODOS DE INVESTIGACIÓN

Método	**Sustento**
Deductivo	Este método se utilizará para plantear una solución a la problemática identificada.
Experimental	Es método se utilizará para entrenar, validar y comparar el rendimiento respecto de los métodos tradicionales.

Técnicas e instrumentos de recolección de datos:

TABLA II
TÉCNICAS DE INVESTIGACIÓN

Técnicas	**Instrumentos**
Entrevista	Guía de entrevista
Observación	Guía de observación
Revisión de literatura	Literatura científica
Revisión de documentación	Reportes de software propietario

Metodología de desarrollo:

Metodología CRISP-DM para el desarrollo del modelo basado en inteligencia artificial, a continuación, se detallan las fases:

- Comprensión del negocio: En esta fase se realizará un análisis de los objetivos estratégicos de PROVIAS NACIONAL, identificando la necesidad de automatizar la categorización de vehículos en las estaciones de peaje. Se evaluarán las limitaciones del sistema actual y se definirán los beneficios esperados de una solución basada en inteligencia artificial, orientada a mejorar la eficiencia operativa, reducir el mantenimiento y optimizar la toma de decisiones.
- Comprensión de los datos: Se llevará a cabo una exploración detallada del conjunto de datos disponible, enfocado principalmente en imágenes de

vehículos captadas en peajes. Se analizarán aspectos como calidad, resolución, distribución de clases (tipos de vehículos), posibles inconsistencias y la relevancia de la información para el cumplimiento del objetivo del modelo.

- Preparación de los datos: Esta fase se incluirá el preprocesamiento de las imágenes para adecuarlas al modelo de aprendizaje automático. Se realizarán tareas como redimensionamiento, normalización, etiquetado de clases y, en algunos casos, aumentación de datos para mejorar la generalización del modelo. El objetivo fue construir un conjunto de datos limpio, balanceado y representativo.
- Modelado: Se diseñará y entrenará un modelo de inteligencia artificial utilizando técnicas de visión por computadora, como redes neuronales convolucionales. Se probarán diferentes arquitecturas y parámetros para encontrar la configuración más adecuada según los requerimientos del negocio.
- Evaluación: Una vez entrenado el modelo, se evaluará su rendimiento mediante métricas (precisión). Además, se validará su alineación con los objetivos definidos por PROVIAS NACIONAL, asegurando que los resultados obtenidos sean útiles y confiables para su integración en el sistema de clasificación vehicular.
- Despliegue: Finalmente, el modelo será integrado en una aplicación funcional capaz de operar en entornos reales de peaje. Esta fase incluirá pruebas de funcionamiento, validación en campo y ajustes finales para garantizar su desempeño en condiciones reales de operación.

Para desarrollar la aplicación, se ha adoptado la metodología ágil XP. Las fases se explican a continuación:

- Fase de planificación: Se establecerá comunicación directa y continua con el personal responsable de PROVIAS NACIONAL, con el fin de identificar y definir los requerimientos funcionales de la aplicación. Los requerimientos serán documentados y organizados en forma de historias de usuario, facilitando su análisis, priorización y posterior desarrollo incremental.
- Fase de diseño: Dado que la aplicación se desarrollará como una solución de escritorio, en esta etapa se definirán los prototipos de interfaz de usuario, la arquitectura general del sistema y la estructura de componentes. Se seleccionarán tecnologías proporcionadas por Microsoft, como la plataforma

.NET para el desarrollo de la interfaz y lógica de negocio, y SQL Server como sistema gestor de base de datos. El diseño se mantuvo simple, flexible y modular, para facilitar futuros ajustes.

- Fase de codificación: Durante esta fase se implementará la funcionalidad de la aplicación utilizando el lenguaje de programación C# (C Sharp), en conformidad con las buenas prácticas de programación promovidas por XP, tales como programación simple, desarrollo incremental y trabajo en pequeñas iteraciones. La base de datos será estructurada y administrada mediante SQL Server, garantizando integridad, seguridad y eficiencia en el manejo de la información.
- Fase de pruebas: Se realizarán pruebas unitarias y funcionales sobre cada uno de los componentes desarrollados, verificando que cumplan con los requisitos establecidos en las historias de usuario. Las pruebas continuas permitirán identificar errores tempranos y mantener la calidad del software en cada iteración.
- Fase de lanzamiento: Finalmente, se entregará la aplicación completamente funcional junto con su respectiva documentación técnica y de usuario. Además, se brindará una capacitación básica al personal de PROVIAS NACIONAL sobre el uso general de la aplicación, asegurando una correcta adopción y operación inicial del sistema.

Prototipos de la solución:

Requerimientos funcionales:

- La solución debe tener un apartado para configurar las cámaras que se usaran.
- La solución debe realizar la clasificación de los vehículos según el requerimiento de los interesados.
- La solución debe almacenar los metadatos tales como fecha, hora y nivel de confianza de los resultados obtenidos del proceso de clasificación.
- La solución debe contar con un monitor para hacer el seguimiento de la funcionalidad en tiempo real.

Requerimientos no funcionales:

- La solución debe estar preparada para adaptarse a cualquier situación del entorno (lluvia, bajas condiciones de luz, etcétera).

- La solución debe alcanzar al menos un 90% de confiabilidad en los resultados obtenidos.
- El tiempo para la clasificación no debe superar los 2 segundos.

Arquitectura de la solución propuesta:

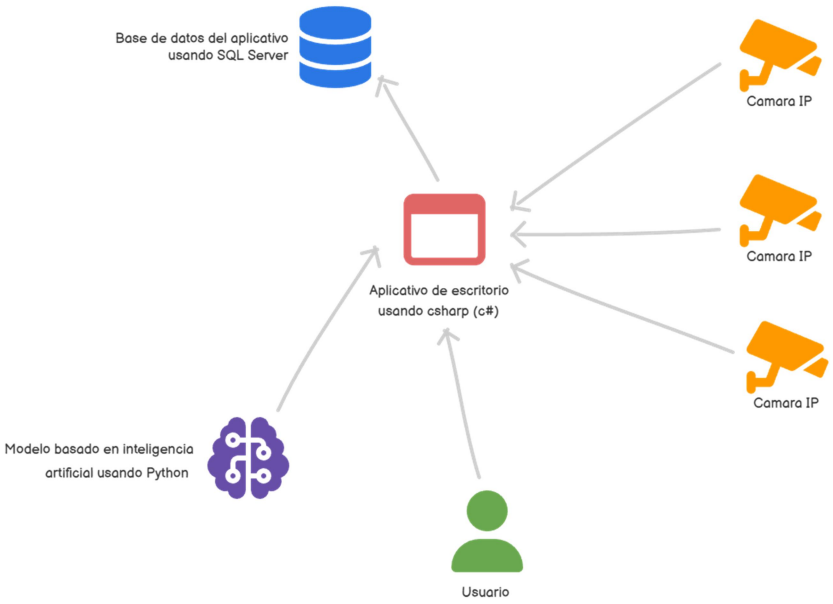


Fig. 8. Arquitectura de solución propuesta.

La infraestructura tecnología estará conformada por:

- Cámara IP: Representa la cámara IP desde donde se capturará la imagen en tiempo real.
- Modelo basado en inteligencia artificial: Modelo encargado de realizar la clasificación de los vehículos.
- Aplicativo: Aplicación con interfaces de usuario.
- Base de datos de aplicativo: Base de datos donde se almacenarán datos del aplicativo.
- Usuario: Usuarios que harán uso del aplicativo.

Colaboradores:

TABLA III
COLABORADORES

Nº	Colaborador	Tipo de apoyo	Especificación del apoyo
1	PROVIAS NACIONAL	Información	Información sobre funcionamiento de software propietario actual.

Resultados y discusión

En base a la metodología usada:

Metodología CRISP-DM:

Fase 1, comprensión del negocio:

En la actualidad el proceso de categorización de vehículos se realiza mediante el uso de una tecnología pasiva invasiva, esta presenta limitaciones, tales como la exposición a condiciones climatológicas adversas y el desgaste derivado del uso constante, lo que conlleva a deficiencias en la detección y categorización de vehículos.

Fig. 9. Tecnología actual, sensores de contacto.

En este contexto, se propone utilizar el modelo YOLO especializado en la detección de objetos en imágenes. Para ello, YOLO será entrenado con un conjunto de datos representativo, considerando diferentes tipos de vehículos, escenarios de tránsito y condiciones ambientales. Se utilizará la "versión 12" de YOLO en su variante "x" debido que se requiere obtener la máxima precisión posible en la categorización de vehículos.

El modelo YOLO será entrenado para reconocer vehículos ligeros, vehículos pesados, ejes y ejes levantados, posteriormente se aplicará un post procesamiento a fin de categorizar los vehículos según su cantidad de ejes. Finalmente, el modelo YOLO deberá integrarse de forma eficiente con los entornos existentes, permitiendo la exportación de resultados en formatos compatibles.

Fase 2, comprensión de los datos:

Para realizar el entrenamiento del modelo YOLOv12s se extraerán imágenes desde el servidor SQL Server de la entidad, dichas imágenes

fueron capturadas desde cámaras IP ubicadas en la periferia de cada unidad de peaje.

El conjunto de imágenes fue sometido a una fase de organización y curaduría, seleccionando aquellas que contenían vehículos claramente visibles, bien definidos y representativos de las clases que se busca identificar: vehículos ligeros, vehículos pesados, ejes y ejes levantados.

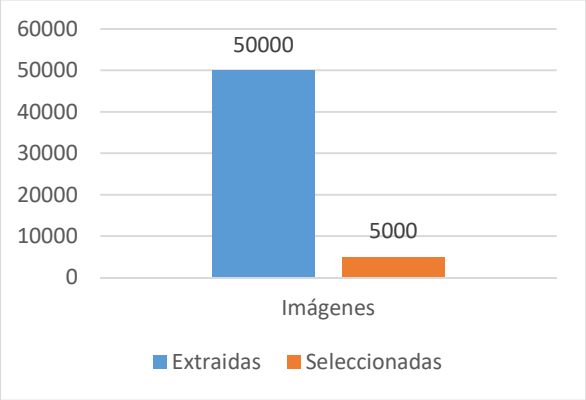


Fig. 10. Imágenes extraídas y seleccionadas.

Las imágenes recolectadas presentan una amplia diversidad de condiciones, se dispone de registros tomados en diferentes horarios, con variaciones notorias en el nivel de luminosidad, presencia de sombras, condiciones climáticas adversas como lluvia o neblina, y situaciones de tráfico denso u oclusión parcial entre vehículos. Esta variabilidad, lejos de ser un inconveniente, contribuye a robustecer el conjunto de datos al exponer al modelo a un espectro realista de escenarios operativos.

A partir de las imágenes seleccionadas, se procedió a su anotación manual, estableciendo un marco referencial para cada clase visible mediante cajas delimitadoras y asignándole su etiqueta correspondiente. Este proceso riguroso garantiza que los datos estén alineados con los requisitos del modelo y facilita su lectura eficiente durante el entrenamiento.

Fig. 11. Anotación mediante cajas delimitadoras.

Durante la exploración inicial del conjunto de datos, se identificaron diversos factores que influyen en la complejidad del problema. Entre ellos, se destaca la alta variabilidad en las dimensiones de los vehículos según la distancia focal, así como la presencia de vehículos parcialmente visibles, lo que podría inducir errores de clasificación.

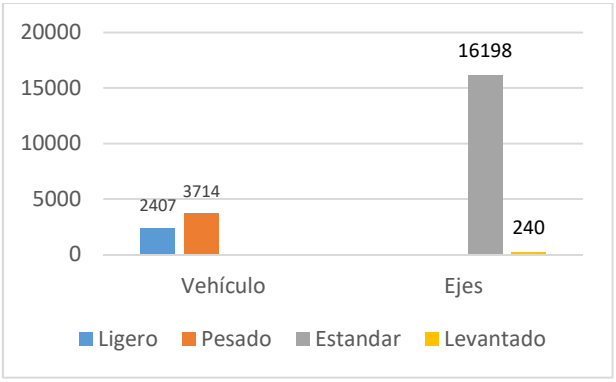


Fig. 12. Detalle de objetos etiquetados.

En suma, la comprensión exhaustiva del conjunto de datos ha permitido establecer una base sólida para las fases posteriores del proyecto. La riqueza visual de las imágenes, la calidad de las anotaciones y la diversidad de escenarios capturados aseguran que el modelo YOLOv12s será expuesto a un entorno de entrenamiento realista y exigente.

Fase 3, preparación de los datos:

La preparación de los datos representó una etapa determinante ya que implicó transformar un conjunto diverso de imágenes crudas en un recurso estructurado, limpio y útil para el proceso de entrenamiento del modelo de detección. A partir de las imágenes recopiladas en distintas ubicaciones, se consolidó suficiente variedad en cuanto a condiciones

ambientales, tipos de vehículos, ángulos de captura y niveles de resolución.

La consolidación del conjunto de datos requirió un proceso riguroso de filtrado y selección, priorizando aquellas imágenes que mostraban escenas representativas del tránsito real. Se descartaron registros borrosos, con baja iluminación o con obstrucciones graves, preservando únicamente aquellas imágenes en las que la presencia y forma de los vehículos fuese claramente distinguible. El objetivo fue asegurar que el modelo no solo aprenda a identificar objetos, sino que también desarrolle la capacidad de diferenciarlos con precisión bajo condiciones operativas realistas.

Las imágenes seleccionadas fueron anotadas manualmente usando Roboflow. Este proceso consistió en identificar cada objeto presente en la imagen mediante cajas delimitadoras y asignarle la clase correspondiente, ya sea como vehículo ligero, vehículo pesado, eje o eje levantado, según criterios visuales consistentes. Aunque el número de ejes no siempre es directamente visible, se incorporaron suficientes casos de referencia para que el modelo reconozca patrones que permitan inferir la categoría general de forma confiable.

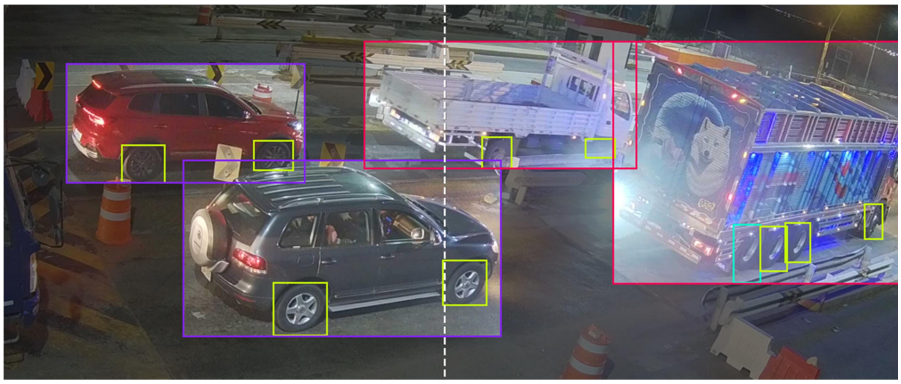


Fig. 13. Etiquetado de imágenes usando Roboflow.

Cada anotación fue almacenada en un formato estructurado, compatible con el modelo YOLOv12s. Las coordenadas de las cajas delimitadoras fueron normalizadas respecto al ancho y alto de la imagen, permitiendo su lectura eficiente durante el proceso de entrenamiento.

```
0 0.73046875 0.521875 0.0390625 0.05859375
0 0.8171875 0.50234375 0.03125 0.05234375
0 0.93046875 0.48203125 0.02265625 0.04140625
0 0.96015625 0.4703125 0.01640625 0.03359375
0 0.44375 0.5734375 0.05078125 0.071875
0 0.259375 0.60546875 0.0578125 0.078125
0 0.23828125 0.38984375 0.02734375 0.0453125
0 0.0265625 0.3921875 0.04765625 0.0609375
2 0.2828125 0.34765625 0.2453125 0.15078125
2 0.31796875 0.51953125 0.33125 0.24765625
2 0.73359375 0.4625 0.215625 0.17734375
2 0.92109375 0.44453125 0.10546875 0.115625
```

Fig. 14. Coordenadas de cajas delimitadoras en imágenes.

Asimismo, se organizó la estructura del directorio de trabajo, estableciendo carpetas diferenciadas para las imágenes y sus respectivas etiquetas, asegurando la integridad y coherencia del dataset.

TABLA IV

ESTRUCTURA DE DIRECTORIOS

Estructura	Carpeta	Observación
Yolov12x/ ├── valid/ │ ├── labels │ └── images ├── train/ │ ├── labels │ └── images └── test/ ├── labels └── images	Yolov12	Carpeta raíz.
	valid	Datos usados durante el entrenamiento para validar el rendimiento del modelo.
	train	Datos que serán utilizados para entrenar el modelo.
	test	Datos para realizar evaluación final del modelo, no influyen en el entrenamiento.

Con el fin de ampliar la capacidad de generalización del modelo y el enriquecimiento de la diversidad del conjunto de imágenes, se aplicaron las siguientes técnicas de aumento de datos:

TABLA V

TÉCNICAS PARA EL AUMENTO DE DATOS UTILIZADAS

Técnica	Valor	Descripción
Rotación	De 0º a 15º	Agrega variabilidad a las rotaciones para ayudar a que el modelo sea más resistente al giro

		de la cámara.
Recorte	De 0% a 30%	Agrega variabilidad al posicionamiento y al tamaño para ayudar a que el modelo sea más resistente a las traslaciones del sujeto y a la posición de la cámara.
Empañamiento	2.5 pixeles	Agrega desenfoque gaussiano aleatorio para ayudar a que el modelo sea más resistente al enfoque de la cámara
Ruido	De 0% a 1.9%	Agrega ruido para ayudar a que el modelo sea más resistente a los artefactos de la cámara.
Brillo	De 0% a 25% de brillo De 0% a 25% de oscuridad	Agrega variabilidad al brillo de la imagen para ayudar a que el modelo sea más resistente a los cambios de iluminación y configuración de la cámara.
Voltear	Horizontal Vertical	Agrega giros horizontales o verticales para ayudar a que el modelo sea insensible a la orientación del sujeto.
Inclinación	De 0º a 15º horizontal De 0º a 15º vertical	Agrega variabilidad a la perspectiva para ayudar a que el modelo sea más resistente a la inclinación de la cámara.
Escala de grises	De 0% a 25%	Aplicar de forma probabilística la escala de grises a un subconjunto del conjunto de entrenamiento.

Estas transformaciones fueron aplicadas de forma aleatoria y controlada, garantizando que los datos modificados mantuvieran su integridad semántica.

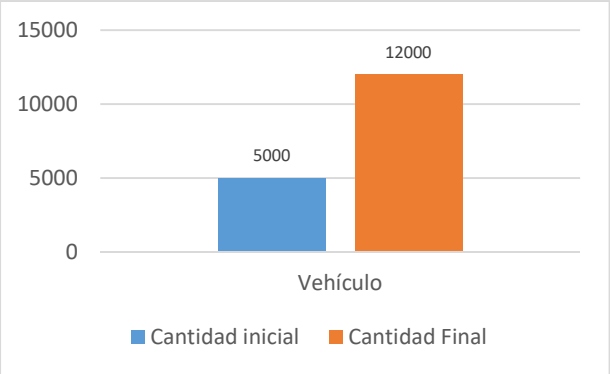


Fig. 15. Conjunto de datos resultante post aumento de datos.

Para maximizar la utilidad del conjunto de datos, se realizó una partición en tres subconjuntos: entrenamiento, validación y prueba. Esta distribución se efectuó de manera porcentual, garantizando que la proporción entre clases se mantenga constante en cada partición.

TABLA VI
VALORES EXPRESADOS EN PORCENTAJES PARA SUBCONJUNTOS

Subconjunto	**Valor en %**	**Valor en cantidades**
Entrenamiento	88	10500
Validación	8	1000
Prueba	4	500

Finalmente, se definieron las clases objetivo y se documentaron las características del conjunto de datos. La calidad visual, la diversidad de condiciones y la precisión de las anotaciones brindaron una base sólida y confiable para el entrenamiento del modelo. Esta preparación cuidadosa del conjunto de datos no solo fortaleció la etapa de modelado, sino que también aseguró que los resultados obtenidos fueran representativos, consistentes y extrapolables a escenarios reales dentro del ámbito de operación de PROVIAS NACIONAL.

Fig. 16. Clases objetivo.

Fase 4, modelado:

Una vez estructurado y preparado el conjunto de datos, se procedió con la implementación del modelo destinado a la detección y categorización automática de vehículos. Para ello, se seleccionó el modelo YOLOv12s, por su reconocida eficiencia en tareas de visión por computadora, particularmente en entornos donde la precisión y la velocidad de inferencia resultan críticas.

Ver anexo 1

A través de múltiples iteraciones de entrenamiento, se ajustaron los parámetros del modelo hasta obtener un equilibrio entre exactitud, sensibilidad y generalización frente a datos no vistos.

Ver anexo 2.

Dónde:

TABLA VII

PARÁMETROS PARA ENTRENAMIENTO

Parámetro	Valor	Descripción
task	detect	Tipo de tarea (detección)
mode	train	Modo de entrenamiento
model	yolo12s.pt	Modelo base a usar
data	data.yaml	Archivo de configuración
epochs	150	Número máximo de épocas de entrenamiento
imgsz	640	Tamaño de imagen (640 * 640 pixeles)
patience	20	Detección temprana si no hay mejora en

		20 epocas
device	0	Uso de la GPU 0
batch	16	Grupo de imágenes que pasan juntas por la red en una sola iteración

Durante las sesiones de entrenamiento, el modelo fue expuesto a imágenes etiquetadas que representaban diversas condiciones operativas, incluyendo variaciones de iluminación, ángulos de cámara y tipos de vehículos. Las etiquetas incluyeron cuatro clases principales: vehículo ligero, vehículo pesado, eje y eje levantado, determinadas sobre la base de características visuales inferidas por el modelo a partir de las anotaciones previamente realizadas.

Fig. 17. Resultado reconocimiento de objetos según su porcentaje de confiabilidad.

Finalizado el entrenamiento del modelo se exporto un archivo en formato .pt, compatible con los entornos de despliegue previstos. Este archivo constituye el núcleo del sistema de inferencia que será integrado en la solución destinada a PROVIAS NACIONAL.

Una vez obtenido el modelo entrenado, se incorporó una etapa de posprocesamiento que permitió complementar la clasificación realizada por YOLOv12s con información adicional sobre la cantidad de ejes por vehículo. Esta etapa analizó cada detección e identificó, a partir de sus características visuales, el número estimado de ejes visibles.

Ver anexo 3.

Como resultado, la salida del sistema no se limitó a clasificar los vehículos como “ligeros” o “pesados”, sino que precisó su configuración, generando etiquetas del tipo “ligero 2”, “pesado 3”, “pesado 5”, entre otras. Este enriquecimiento de la salida permitió alinear la solución con los criterios técnicos utilizados por PROVIAS NACIONAL para la categorización funcional de vehículos en su infraestructura vial.

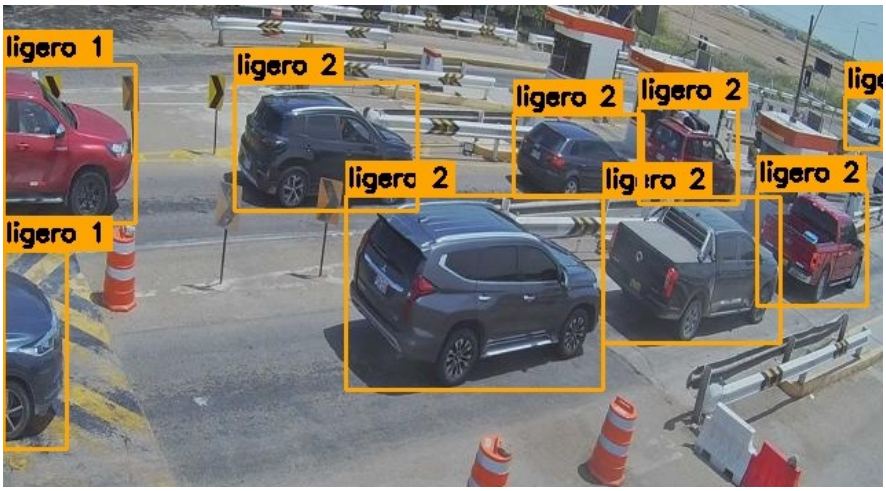


Fig. 18. Resultado post procesamiento.

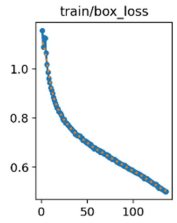
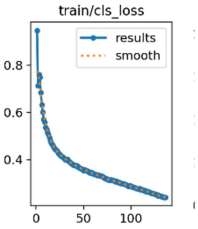
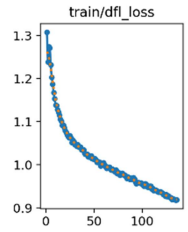
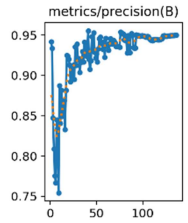
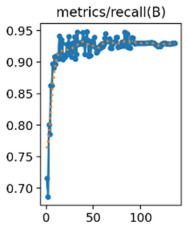
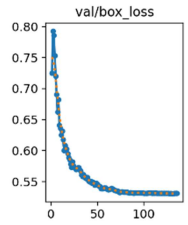
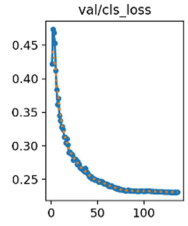
Fase 5, evaluación:

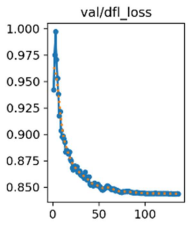
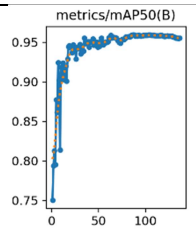
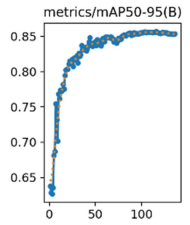
Finalizado el proceso de entrenamiento, se llevó a cabo una evaluación exhaustiva del modelo con el objetivo de verificar su capacidad para cumplir con los requisitos técnicos y funcionales definidos al inicio del proyecto. La precisión en la detección y clasificación de vehículos fue sometida a un análisis cuantitativo sobre un conjunto de imágenes no utilizadas en el entrenamiento, lo cual permitió valorar la capacidad del modelo para generalizar frente a nuevas situaciones. Producto del análisis cuantitativo se obtuvieron los siguientes resultados:

TABLA VIII

ANÁLISIS DE RESULTADOS DE MÉTRICAS

Métrica	**Explicación**

	Perdida de cajas delimitadoras durante el entrenamiento disminuye constantemente, lo que indica que el modelo mejora su capacidad para localizar objetos correctamente.
	Perdida de clasificación de objetos disminuye constantemente, lo que indica que el modelo mejora su capacidad para identificar clases correctamente.
	Precisión más fina en localización de bordes de las cajas delimitadoras disminuye constantemente, lo que indica mejor definición en la predicción de bordes.
	Precisión del modelo aumenta y se estabiliza cerca del 0.95.
	Capacidad de encontrar todos los objetos reales aumenta y se mantiene estable alrededor del 0.93.
	Perdida de caja delimitadora en el conjunto de validación disminuye y se estabiliza, lo que sugiere que no hay sobre ajuste.
	Perdida de clasificación en conjunto de validación disminuye, el modelo generaliza bien.

	Precisión más fina en la localización de los bordes de las cajas delimitadoras del conjunto de validación disminuye.
	Calidad de detección aumenta rápido y se estabiliza en 0.95.
	Precisión más estricta aumenta rápido y se estabiliza en 0.86, el modelo mantiene buena precisión incluso bajo criterios más exigentes.

El desempeño del modelo evidenció resultados altamente consistentes en contextos variados, logrando identificar con eficacia vehículos tanto ligeros como pesados en condiciones visuales complejas. En escenarios con buena iluminación y visibilidad, las predicciones alcanzaron altos niveles de confianza, mientras que, en escenas con oclusiones parciales o iluminación deficiente, el modelo mantuvo un comportamiento estable, aunque con leves reducciones en la precisión de clasificación.

TABLA IX

RESUMEN DE RESULTADOS DEL MODELO ENTRENADO

Métrica	**Valor obtenido**	**Observación**
mAp@50	≈ 0.95	Alta calidad en detección con lo mínimo del 50%.
mAP@50-95	≈ 0.86	Buen rendimiento incluso en condiciones estrictas.
Precisión y exhaustividad	≈ 0.95	El modelo detecta bien y comete pocos falsos positivos.

Recall	≈ 0.93	El modelo tiene un alto porcentaje de detección de los objetos presentes.
Losses	Decrecen	Las perdidas descienden de forma estable, sin señales de sobreajuste.

Las métricas obtenidas reflejaron un rendimiento robusto, con una precisión media promedio (mAP) superior al 80 % y una tasa de verdaderos positivos significativamente alta en las clases evaluadas.

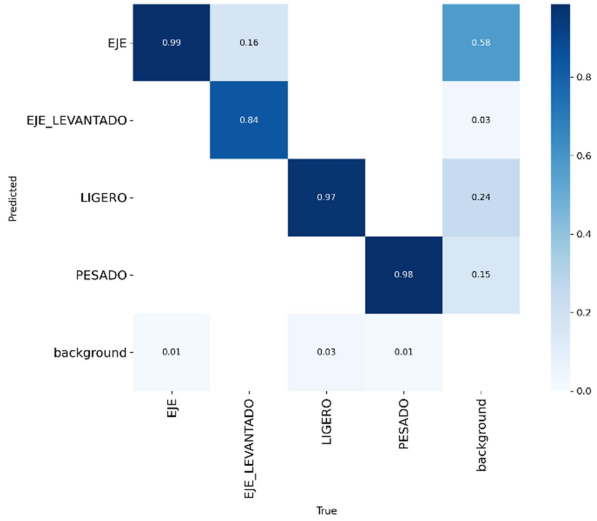


Fig. 19. Matriz de confusión.

De la matriz de confusión se puede concluir que se tiene un alto porcentaje de exactitud para las clases eje, ligero y pesado; la clase eje levantado presenta un bajo rendimiento producto del bajo número de muestras.

Más allá de los resultados numéricos, la evaluación permitió validar la coherencia entre los resultados obtenidos y las necesidades institucionales de PROVIAS NACIONAL.

En conclusión, la evaluación del modelo confirmó su pertinencia como solución tecnológica efectiva para la categorización vehicular automatizada. Si bien se identificaron áreas susceptibles de refinamiento, los resultados alcanzados cumplen con los estándares de calidad esperados y proporcionan una base sólida para el despliegue e integración del sistema en el entorno operativo de PROVIAS NACIONAL.

Fase 6, despliegue:

Para el desarrollo de la fase de despliegue se ejecutó el script encargado de migrar el modelo en formato ONNX, el mismo que será usado en la solución dentro de la metodología XP.

Ver anexo 4.

Metodología XP:

Fase 1, planificación:

TABLA X

HISTORIA DE USUARIO GESTIÓN DE CÁMARAS

Numero: HU-01	Usuario: Provias Nacional
Nombre de la historia: Gestión de cámaras	
Prioridad en negocio: Baja	Riesgo de desarrollo: Bajo
Puntos estimados: 1	Iteración asignada: 1
Programador responsable: Pedro Ricardo Pejerrey Gómez	
Descripción: El usuario podrá gestionar cámaras.	
Criterios de aceptación: - Se deben listar cámaras existentes. - Se debe poder editar cámara existente. - Se debe poder agregar nueva cámara. - Se debe poder eliminar una cámara existente siempre y cuando no existan métricas existentes asociadas. - Cada cámara debe almacenar datos como stream, nombre, usuario, contraseña y dirección. - Se debe incluir un botón para realizar prueba de stream.	
Observación: No se requieren permisos especiales para la gestión de cámaras.	

TABLA XI

HISTORIA DE USUARIO GESTIÓN DE CATEGORÍAS

Numero: HU-02	Usuario: Provias Nacional

Nombre de la historia: Gestión de categorías	
Prioridad en negocio: Bajo	Riesgo de desarrollo: Bajo
Puntos estimados: 1	Iteración asignada: 1
Programador responsable: Pedro Ricardo Pejerrey Gómez	
Descripción: El usuario podrá gestionar categorías.	
Criterios de aceptación: - Se deben listar categorías existentes. - Se debe poder editar categoría existente. - Se debe poder agregar nueva categoría. - Se debe poder eliminar una categoría existente siempre y cuando no existan métricas existentes asociadas. - Cada categoría debe almacenar datos como nombre, abreviatura, si es primaria y color para caja delimitadora.	
Observación: No se requieren permisos especiales para la gestión de categorías.	

TABLA XII

HISTORIA DE USUARIO DETECCIÓN Y CLASIFICACIÓN

Numero: HU-03	Usuario: Provias Nacional
Nombre de la historia: Detección y clasificación de vehículos	
Prioridad en negocio: Medio	Riesgo de desarrollo: Alto
Puntos estimados: 2	Iteración asignada: 2
Programador responsable: Pedro Ricardo Pejerrey Gómez	
Descripción: La solución debe detectar y clasificar automáticamente los vehículos según las categorías definidas adicionando su número de ejes.	

Criterios de aceptación: - La precisión en la clasificación del vehículo debe ser >= 95% - La solución debe contar el número de ejes visibles y mostrarlo adicionándolo a la categoría del vehículo. - El resultado del proceso de clasificación debe generarse en menos de 20 milisegundos por imagen. - El formato de salida debe ser compatible con el sistema actual de Provias Nacional.
Observación: No se requieren permisos especiales para la detección y clasificación de vehículos.

TABLA XIII

HISTORIA DE USUARIO GENERACIÓN DE MÉTRICAS

Numero: HU-04	Usuario: Provias Nacional
Nombre de la historia: Visualización de métricas	
Prioridad en negocio: Media	Riesgo de desarrollo: Alto
Puntos estimados: 2	Iteración asignada: 2
Programador responsable: Pedro Ricardo Pejerrey Gómez	
Descripción: Se debe visualizar métricas de los vehículos clasificados.	
Criterios de aceptación: - La visualización debe incluir datos como fecha, hora, categoría, numero de ejes y nivel de confianza. - Debe poder exportarse a formatos compatibles con el sistema de PROVIAS NACIONAL. - Debe poder generar reporte de manera retroactiva.	
Observación: No se requieren permisos especiales para generación de métricas.	

TABLA XIV

HISTORIA DE USUARIO VISUALIZACIÓN EN TIEMPO REAL

Numero: HU-05	Usuario: Provias Nacional
Nombre de la historia: Visualización en tiempo real	
Prioridad en negocio: Media	Riesgo de desarrollo: Media
Puntos estimados: 2	Iteración asignada: 3
Programador responsable: Pedro Ricardo Pejerrey Gómez	
Descripción: Se debe poder visualizar la clasificación en tiempo real.	
Criterios de aceptación: - La solución debe mostrar el video en tiempo real con las cajas delimitadoras. - El retardo en la visualización no debe superar las 500 milis segundos. - Se debe mostrar el porcentaje de precisión para cada vehículo.	
Observación: No se requieren permisos especiales para la visualización en tiempo real.	

TABLA XV

HISTORIA DE USUARIO REENTRENAMIENTO DEL MODELO

Numero: HU-06	Usuario: Provias Nacional
Nombre de la historia: Reentrenamiento del modelo	
Prioridad en negocio: Alta	Riesgo de desarrollo: Alto
Puntos estimados: 3	Iteración asignada: 4
Programador responsable: Pedro Ricardo Pejerrey Gómez	
Descripción: Se debe poder reentrenar el modelo existente en base a nuevas categorías.	

Criterios de aceptación: - La solución debe permitir reentrenar el modelo existente actual. - Se debe contar con parámetros por defecto, sin embargo, se debe permitir modificarlos. - Se debe versionar los modelos entrenados.
Observación: No se requieren permisos especiales para el entrenamiento del modelo.

TABLA XVI

REQUERIMIENTOS FUNCIONALES

Identificador	**Prioridad**	**Relación con HU**	**Descripción corta**	**Descripción larga**
RF-01	Medio	HU-01, HU-03	Detección de vehículos	La solución debe identificar automáticamente vehículos en videos provenientes de cámaras registradas, diferenciando entre las distintas categorías.
RF-02	Medio	HU-02,	Clasificación	La

		HU-03	por ejes	solución debe estimar y registrar el número de ejes visibles de cada vehículo, generando etiquetas enriquecidas.
RF-03	Medio	HU-05	Visualización en tiempo real	La solución debe mostrar video en vivo con cajas delimitadoras, etiquetas y porcentaje de confianza para cada detección.
RF-04	Medio	HU-04	Visualización de métricas	La solución debe

				mostrar métricas retroactivas con las detecciones realizadas, incluyendo tipo de vehículo, número de ejes y confianza de predicción.
RF-05	Alta	HU-06	Reentrenamiento del modelo	La solución debe permitir reentrenar el modelo, generando una nueva versión optimizada.

TABLA XVII

REQUERIMIENTOS NO FUNCIONALES

Identificador	**Categoría**	**Descripción corta**	**Descripción larga**
RNF-01	Rendimiento	Precisión mínima	El modelo debe lograr una precisión mayor o igual al 95% en la clasificación.
RNF-02	Rendimiento	Tiempo de inferencia	El tiempo de detección y clasificación no debe superar los 200 ms en hardware GPU definido para producción.
RNF-03	Arquitectura	Escalabilidad	La solución debe permitir procesar datos de múltiples cámaras en paralelo sin pérdida significativa de rendimiento.
RNF-04	Calidad	Robustez ante variabilidad	La solución debe mantener precisión ≥ 90 % en condiciones de baja

			iluminación, lluvia o neblina.
RNF-05	Implementación	Portabilidad del modelo	El modelo entrenado debe exportarse en formato .pt y poder desplegarse en diferentes entornos Windows sin modificaciones de código.
RNF-06	Experiencia de Usuario	Usabilidad de la interfaz	La interfaz debe ser intuitiva y accesible para operadores con conocimientos técnicos básicos.

TABLA XVIII

LISTA DE RIESGOS Y MITIGACIONES

Identificador	**Impacto**	**Probabilidad**	**Riesgo**	**Mitigación**
R-01	Alto	Alto	Baja calidad de videos (desenfoque, baja resolución, oclusiones).	Definir parámetros para la configuración de las cámaras.

R-02	Alto	Medio	Tiempo de inferencia superior a lo esperado afectando visualización en tiempo real	Usar hardware GPU adecuado para producción.
R-03	Alto	Alto	Clasificación errónea de ejes por ángulo de cámara o poca visibilidad	Definir parámetros para la ubicación de cámaras.
R-04	Alto	Bajo	Incompatibilidad con sistema de gestión de peajes.	Diseñar formato de exportación validado con TI de PROVIAS antes del desarrollo final.
R-05	Alto	Medio	Pérdida de datos o resultados por fallos del sistema.	Implementar backups automáticos y redundancia en almacenamiento de datos.

R-06	Alto	Alto	Modelo obsoleto ante aparición de nuevos tipos de vehículos.	Planificar reentrenamientos periódicos e incluir proceso ágil de anotación de nuevos datos.
R-07	Medio	Alto	Errores de usuario en la operación (uso incorrecto de interfaz)	Capacitar a operadores y crear un manual de usuario con ejemplos prácticos.

TABLA XIX

ASIGNACIÓN DE ROLES Y RESPONSABILIDADES

Rol	**Responsable**	**Funciones clave**	**Historias de usuario asociadas**
Desarrollador	Pedro Ricardo Pejerrey Gómez	Desarrollo de modelos entidad - relación, interfaces de usuario y codificación de módulos.	Todas

Tester	Pedro Ricardo Pejerrey Gómez	Diseñar y ejecutar pruebas unitarias, funcionales e integradas, documentar incidencias.	Todas

Fase 2, diseño:

Arquitectura cliente – servidor:

- Aplicación de escritorio: Aplicación desarrollada en CSharp para gestionar cámaras, categorías, métricas y visualización en vivo.
- Base de datos: SQL Server para almacenamiento de configuraciones, cámaras, categorías y métricas.

Relación de componentes:

- Módulo de gestión de cámaras (HU-01).
- Módulo de gestión de categorías (HU-02).
- Módulo de métricas (HU-04).
- Módulo de monitor (HU-03, HU-05).
- Módulo de reentrenamiento (HU-06)

Modelo entidad – relación:

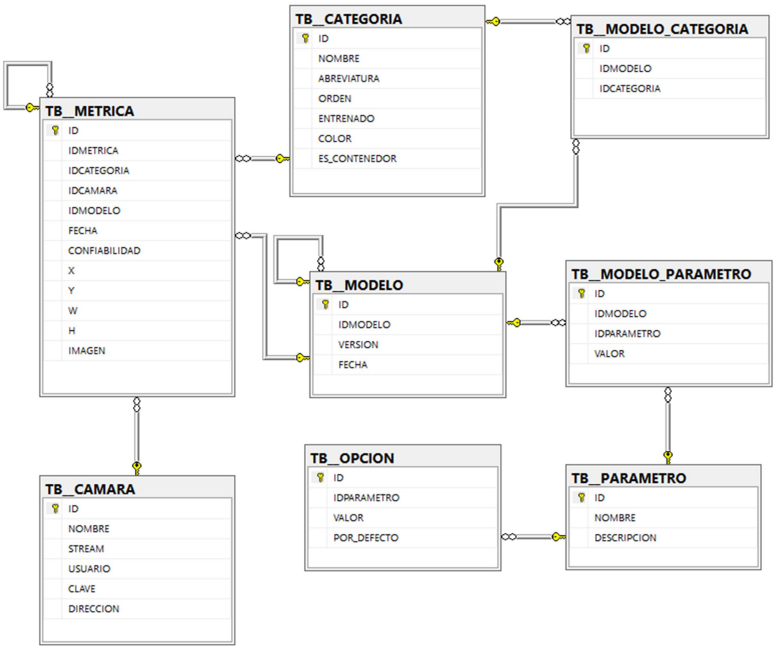


Fig. 20. Diagrama entidad – relación.

Interfaces de usuario:

Ver anexo 5.

Fase 3, codificación:

Herramientas utilizadas:

- Lenguaje de programación: Csharp.
- Base de datos: Microsoft SQL Server 2019.
- Entorno de desarrollo: Visual Studio 2022.
- Librerías externas: OnnxRuntime v1.14.1, openCVSharp v4.

Resultados obtenidos:

- Módulo de gestión de cámaras (HU-01): Se implementó un formulario en csharp que permite realizar las operaciones de alta, baja, edición y consulta de cámaras registradas en la base de datos.

  Ver anexo 6.
- Módulo de gestión de categorías (HU-02): Se implementó un formulario en csharp que permite realizar las operaciones de alta, baja, edición y consulta de categorías registradas en la base de datos.

  Ver anexo 7.

- Módulo de métricas (HU-03): Se implementó un formulario en csharp que permite visualizar las métricas registradas en la base de datos.
  Ver anexo 8.
- Módulo monitor (HU-05): Se implementó un formulario en csharp que ejecuta la inferencia de los vehículos y permite visualizar el monitoreo en tiempo real.
  Ver anexo 9.
- Módulo monitor (HU-06): Se implementó un formulario en csharp permite realizar el reentrenamiento del modelo.
  Ver anexo 10.

Fase 4, pruebas:

Con el propósito de garantizar la calidad de la solución, se llevaron a cabo diversas pruebas que abarcaron desde validaciones unitarias hasta pruebas integradas, orientadas a verificar el cumplimiento de los requisitos funcionales y no funcionales previamente definidos.

Se aplicaron tres niveles:

1. Pruebas unitarias: se enfocaron en validar el correcto funcionamiento de cada módulo de manera independiente (gestión de cámaras, gestión de categorías, métricas, monitor y entrenamiento del modelo).
2. Pruebas funcionales: se verifico que las funcionalidades implementadas respondan a las historias de usuario y a los criterios de aceptación definidos en la fase de planificación.
3. Pruebas de integración: se evaluó la interoperabilidad entre los diferentes módulos y la correcta comunicación con la base de datos SQL Server, asegurando la coherencia en la visualización de métricas, categorización vehicular y monitoreo en tiempo real.

TABLA XX

PRINCIPALES CASOS DE PRUEBA

Caso de prueba	HU relacionada	Descripción	Resultado esperado	Resultado obtenido	Estado

CP-01	HU-01	Registrar nueva cámara	La cámara se guarda en la base de datos y se puede visualizar en lista	Registro correcto	Aprobado
CP-02	HU-01	Editar cámara existente	La actualización de los datos de una cámara se guarda en la base de datos y se puede visualizar en lista	Registro actualizado	Aprobado
CP-03	HU-01	Eliminar cámara existente	Se puede eliminar una cámara existente siempre y cuando no existan métricas relacionadas a ella en la base de datos	Registro eliminado	Aprobado
CP-04	HU-01	Probar stream de cámara	Se puede visualizar la transmisión	Visualización fluida	Aprobado

			de una cámara en tiempo real		
CP-05	HU-02	Registrar nueva categoría	La categoría se guarda en la base de datos y se puede visualizar en lista	Registro correcto	Aprobado
CP-06	HU-02	Editar categoría existente	La actualización de los datos de una categoría se guarda en la base de datos y se puede visualizar en lista	Registro actualizado	Aprobado
CP-07	HU-02	Eliminar categoría existente	Se puede eliminar una categoría existente siempre y cuando no existan métricas relacionadas a ella en la base de	Registro eliminado	Aprobado

			datos		
CP-08	HU-03	Detección y clasificación de vehículos	El sistema identifica y clasifica vehículos con >= 95% de precisión	Precisión promedio de 95%	Aprobado
CP-09	HU-03	Conteo de ejes	Se visualiza el número de ejes junto a la categoría	Correcto en el 93% de casos	Aprobado
CP-10	HU-04	Visualización de métricas	Se visualizan datos de métricas generadas tales como fecha, hora, ejes y confianza de manera retroactiva	Visualización correcta	Aprobado
CP-11	HU-04	Exportación de métricas	Los datos pueden ser exportados a otros formatos para que puedan ser usados por sistemas de terceros	Las métricas se almacenan en base de datos por lo que pueden ser consultados por otras aplicaciones	Aprobado

CP-12	HU-05	Visualización en tiempo real	La transmisión muestra detecciones con < 500 ms de retraso	Retraso promedio de 420 ms	Aprobado
CP-13	HU-06	Reentrenamiento del modelo	Se genera nuevo modelo versionado con parámetros configurables	Modelo entrenado y registrado	Aprobado

Se realizaron pruebas de cargas y rendimiento, evaluando los siguientes parámetros:

- Tiempo de inferencia promedio: 180 ms por imagen (cumple con RNF-02).
- Precisión global del modelo: 95% (cumple con RNF-01).
- Robustez en condiciones adversas: Precisión >= 90% en escenas con baja iluminación (cumple con RNF-04).

Fase 5, lanzamiento:

Se realizó la instalación de 1 cámara IP tubular en exterior de peaje, la cámara se utilizó para el funcionamiento de la solución.

Fig. 21. Cámara IP tubular usada para funcionamiento de solución.

Fig. 22. Ubicacion de camara IP tubular.

En la siguiente imagen se aprecia la aplicación corriendo en un servidor de una estación de peaje.

Fig. 23. Servidor en gabinete de piso en unidad de peaje.

En base a los objetivos del proyecto:

El desarrollo de la presente investigación se estructuró en torno a los objetivos planteados, los cuales guiaron cada una de las fases metodológicas, desde la preparación de los datos hasta el despliegue de la solución final.

Objetivo 1. Entrenar un modelo basado en inteligencia artificial para la categorización de vehículos, para mejorar la precisión y velocidad del proceso de clasificación en tiempo real.

Resultado 1. Se implementó el modelo **YOLOv12s**, el cual fue entrenado con un conjunto de imágenes representativas de escenarios reales. Dicho entrenamiento permitió obtener un modelo robusto, capaz de identificar vehículos ligeros y pesados, así como el número de ejes visibles, alcanzando niveles de precisión superiores al 95 % en pruebas controladas. Este resultado evidencia una mejora sustancial frente a los métodos tradicionales, que se caracterizan por ser más lentos y menos confiables en condiciones adversas.

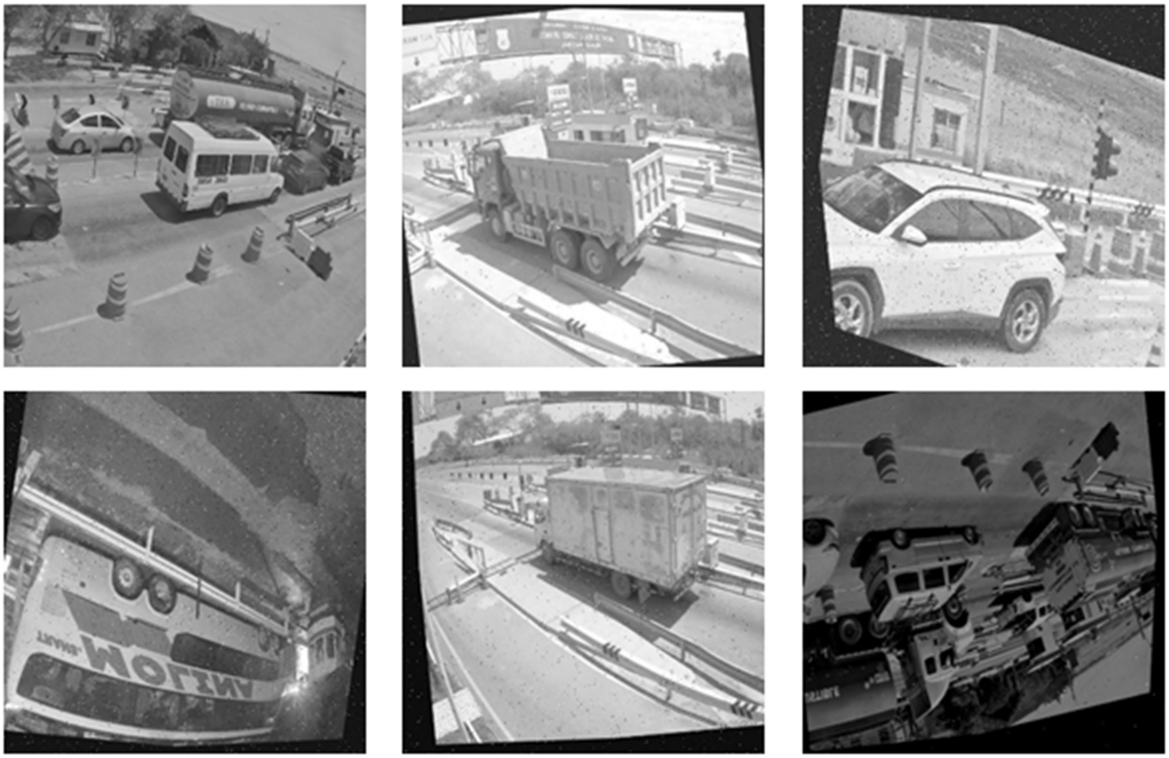


Fig. 24. Conjunto de datos para el entrenamiento.

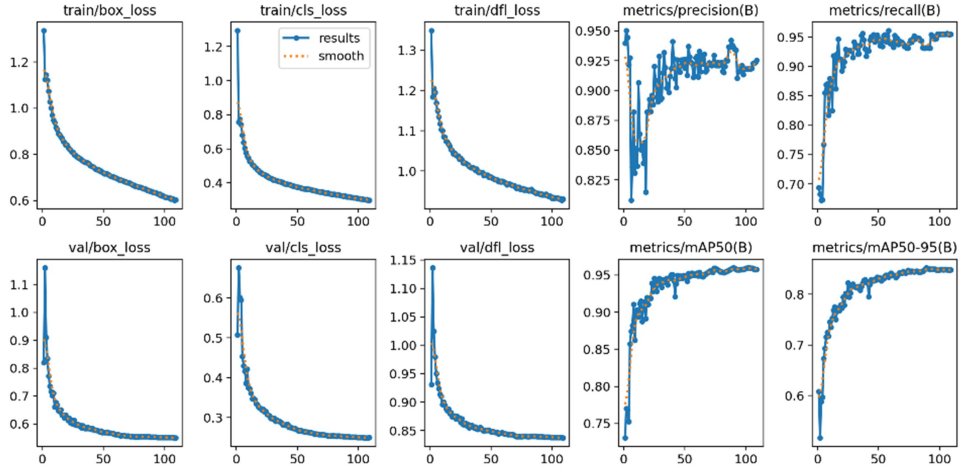


Fig. 25. Resultados obtenidos después del entrenamiento del modelo.

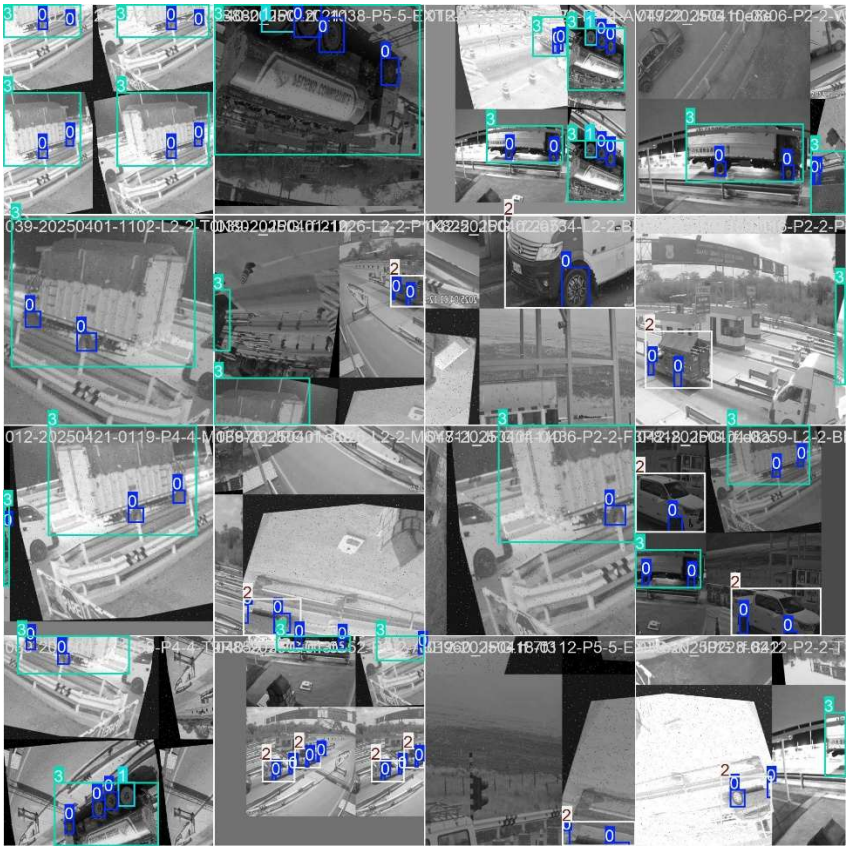


Fig. 26. Detecciones durante el entrenamiento.

Objetivo 2. Desarrollar una aplicación de escritorio que integre el modelo basado en inteligencia artificial, proporcionando una interfaz de usuario intuitiva y funcional.

Resultado 2. Se construyó una solución en **C#** con integración a **SQL Server 2019**, la cual incorpora módulos de gestión de cámaras, gestión de categorías, visualización de métricas, monitoreo en tiempo real y reentrenamiento del modelo. La interfaz fue diseñada con un enfoque en la usabilidad, permitiendo que los operadores interactúen con el sistema de manera sencilla e intuitiva.

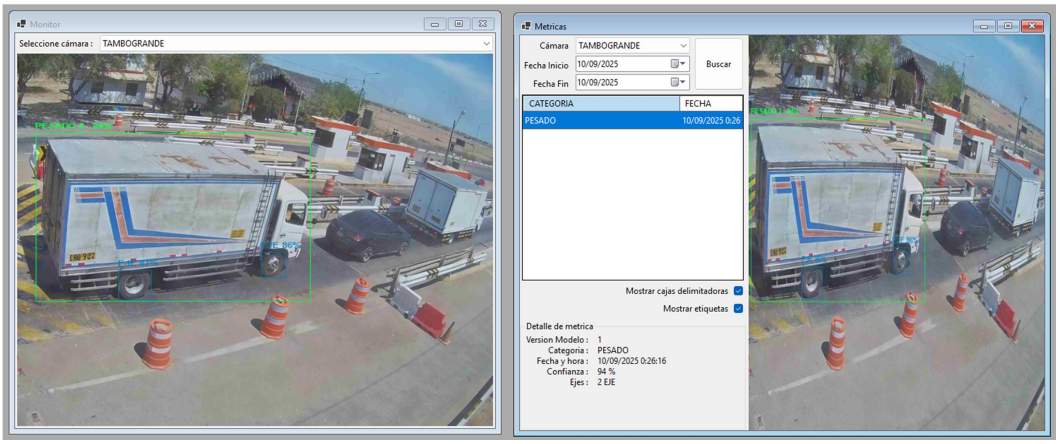


Fig. 27. Funcionamiento de aplicación de escritorio.

Objetivo 3. Determinar la infraestructura de TI que permita la operación de la solución, integrando servidores, red local y cámaras para el procesamiento de datos.

Resultado 3. Se definió una arquitectura cliente-servidor que integra cámaras IP, un servidor de procesamiento con GPU dedicada y una red local de comunicación confiable. Este diseño asegura un flujo de datos eficiente, garantizando que las imágenes capturadas sean procesadas en tiempo real y que los resultados se almacenen de manera segura en la base de datos institucional.

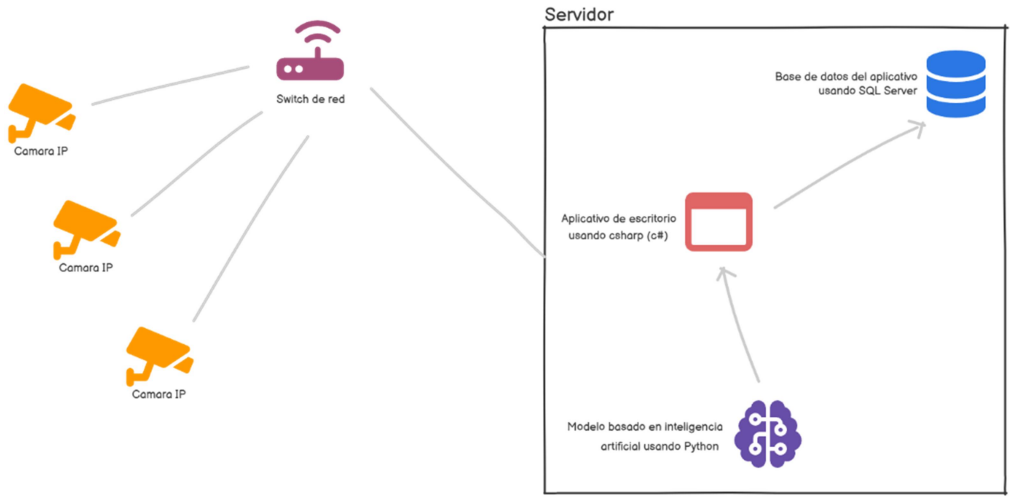


Fig. 28. Infraestructura TI necesaria para operación de solución.

Objetivo 4. Evaluar el rendimiento del modelo propuesto en comparación con los métodos tradicionales, midiendo la eficiencia, la precisión y la capacidad de adaptación a diferentes condiciones operativas.

Resultado 4. Se llevaron a cabo pruebas comparativas que evidenciaron mejoras significativas en precisión, velocidad de clasificación e independencia frente a condiciones adversas como baja iluminación y presencia de oclusiones. Mientras que las técnicas convencionales no consideran dichas situaciones, la solución desarrollada mantuvo un desempeño estable con métricas superiores al 90 % en escenarios de alta variabilidad.

ID	IDMETRICA	IDCATEGORIA	IDCAMARA	IDMODELO	FECHA	CONFIABILIDAD
1	NULL	3	1	1	2025-09-10 00:58:30.520	0,9413229
4	NULL	3	1	1	2025-09-10 00:58:41.260	0,9415041
7	NULL	3	1	1	2025-09-10 00:59:17.383	0,9409508
9	NULL	3	1	1	2025-09-10 01:01:32.580	0,9396706
12	NULL	3	1	1	2025-09-10 01:03:42.603	0,9299257
15	NULL	3	1	1	2025-09-10 01:03:57.820	0,9385152
18	NULL	3	1	1	2025-09-10 01:04:32.833	0,9366494
21	NULL	3	1	1	2025-09-10 01:05:10.307	0,936369
24	NULL	3	1	1	2025-09-10 01:05:23.570	0,9319212

Fig. 29. Porcentaje de confiabilidad de categorización.

Conclusiones

1. El modelo de inteligencia artificial entrenado logró una categorización precisa y eficiente de los vehículos, superando los métodos tradicionales en términos de exactitud y velocidad de procesamiento. El uso de técnicas de aprendizaje profundo permitió optimizar la detección y clasificación en escenarios reales, demostrando la viabilidad de implementar un sistema automatizado que reduzca la intervención humana y los errores en la identificación de categorías vehiculares.
2. Se desarrolló una aplicación de escritorio funcional en lenguaje csharp, haciendo uso de librerías como openCV, onnxruntime y CUDA, que integran el modelo de inteligencia artificial entrenado, ofreciendo una interfaz gráfica amigable y de fácil uso. La integración permitió ejecutar el proceso de clasificación en tiempo real, visualizar los resultados y gestionar la información de manera eficiente. Esta implementación demostró que la herramienta puede ser utilizada por personal no técnico, mejorando la operatividad del sistema y la toma de decisiones.
3. Se definió una infraestructura tecnológica óptima que garantiza el funcionamiento eficiente de la solución propuesta. La integración de servidores locales, cámaras y red permitió el procesamiento simultáneo de datos y la comunicación fluida entre los componentes del sistema. Este diseño de infraestructura aseguró la escalabilidad, estabilidad y seguridad necesarias para una implementación real en entornos operativos.
4. El análisis comparativo evidenció que el modelo propuesto basado en inteligencia artificial superó significativamente a los métodos tradicionales en precisión, velocidad y adaptabilidad. El sistema demostró un desempeño consistente bajo diferentes condiciones de iluminación y ángulos de captura, consolidando su efectividad como alternativa tecnológica avanzada para la clasificación automatizada de vehículos.

Recomendaciones

1. Se recomienda implementar la solución en forma piloto en distintas estaciones de peaje, con el fin de evaluar su comportamiento en diversos entornos geográficos y de tráfico, antes de una adopción a nivel nacional.
2. Es necesario realizar reentrenamientos periódicos del modelo con nuevas imágenes y categorías de vehículos, asegurando que la precisión se mantenga frente a la aparición de nuevos tipos de transporte o modificaciones en la flota vehicular.
3. Se sugiere integrar la solución con módulos de reconocimiento de matrículas y control de evasión de peaje, lo que permitiría ampliar su utilidad en la gestión y fiscalización del tránsito.
4. Se recomienda utilizar servidores con GPU dedicadas y redes de comunicación seguras, que garanticen la estabilidad del procesamiento en tiempo real y el almacenamiento confiable de los registros generados.

Referencias

[1] P. Nacional, «Información Institucional - Proyecto Especial de Infraestructura de Transporte Nacional,» [En línea]. Available: https://goo.su/xbcG3. [Último acceso: 15 Abril 2025].

[2] M. d. T. y. Comunicaciones, «Ministerio de Transportes y Comunicaciones,» [En línea]. Available: https://goo.su/W8jjT. [Último acceso: 17 Abril 2025].

[3] Ferrovial, «Que son los peajes, tipos, beneficios, curiosidades,» [En línea]. Available: https://goo.su/lzOw6. [Último acceso: 17 Abril 2025].

[4] M. d. T. y. Comunicaciones, «DECRETO SUPREMO Nº 058-2003-MTC,» [En línea]. Available: https://goo.su/ueKRzQ. [Último acceso: 18 Abril 2025].

[5] Infraex, «Funcionamiento de pisones de peaje en Chile y Perú,» [En línea]. Available: https://goo.su/XqJTFC. [Último acceso: 18 Abril 2025].

[6] Wikipedia, «Cobro electrónico de peajes,» [En línea]. Available: https://bit.ly/3GymgyW. [Último acceso: 22 Abril 2025].

[7] V. Bolivia, «Administradora de peaje y pesaje,» [En línea]. Available: https://bit.ly/44LZ6iA. [Último acceso: 20 Abril 2025].

[8] F. Pérez Gutiérrez, «Detección de vehículos en circulación mediante visión artificial y redes neuronales convolucionales,» [En línea]. Available: https://bit.ly/435lVfU. [Último acceso: 26 Abril 2025].

[9] Á. García Gonzáles, «Desarrollo de un sistema de clasificación de vehículos en movimiento mediante LIDIAR 3D,» [En línea]. Available: https://bit.ly/3GvE5i2. [Último acceso: 28 Abril 2025].

[10] F. Y. Coanqui Apaza, R. F. Estofanero Yanapa y H. W. Mamani Condori, «Aplicación de inteligencia y visión artificial para la obtención del aforo vehicular,» [En línea]. Available: https://bit.ly/3SesYN4. [Último acceso: 28 Abril 2025].

[11] S. Ocampo Montoya, «MÉTODO PARA LA CLASIFICACIÓN DEL TIPO DE VEHÍCULO Y EL NÚMERO DE PLACA SEGÚN LAS CLASIFICACIONES DEL COBRO DE PEAJE EN COLOMBIA, EMPLEANDO TÉCNICAS DE INTELIGENCIA ARTIFICIAL,» [En línea]. Available: https://surl.li/uithvp. [Último acceso: 29 Abril 2025].

[12] J. T. Córdova Martillo y K. A. Mendieta Garcés, «DISEÑO E IMPLEMENTACIÓN DE

UN PROTOTIPO DE CONTEO VEHICULOAR MEDIANTE INTELIGENCIA ARTIFICIAL PARA REALIZAR EL CONTROL ADAPTATIVO DE REGULADORES DE TRÁNSITO LC2,» [En línea]. Available: https://surl.lu/vuklse. [Último acceso: 01 Abril 2025].

[13] J. C. Ponce Gallegos, A. Torres Soto, F. S. Quezada Aguilera, A. Silva Sprock, E. U. Martínez Flor, A. Casali, E. Scheihing, Y. J. Túpac Valdivia, M. D. Torres Soto, F. J. Ornelas Zapata, J. A. Hernández A., C. Zavala D., N. Vakhnia y O. Pedreño, Inteligencia Artificial, LATIn, 2014.

[14] R. Benítez, G. Escudero, S. Kanaan y D. Masip Rodó, INTELIGENCIA ARTIFICIAL AVANZADA, Barcelona: UOC, 2013.

[15] R. López de Mántaras Badia y P. Meseguer Gonzáles, ¿Qué sabemos de? Intelifencia Artificial, Madrid: Catarata, 2017.

[16] E. Soria Olivas, M. A. Sánchez Montañés Isla, R. Gamero Cruz, B. Castillo Caballero y P. Cano Michelena, Sistemas de aprendizaje automatico, Madrid: Ra-Ma, 2023.

[17] F. Sánchez Lasheras, C. Rodríguez Muiños y L. A. Menéndez García, Sistemas de aprendizaje automatico, Marcombo, 2022.

[18] J. Bobadilla, Machine learning y deep learning usando Python, Scikit y Keras, Ra-Ma, 2020.

[19] A. Bosch Rue, J. Casas Roma y T. Lozano Bagén, Deep Learning principios y fundamentos, Catalunya: UOC, 2019.

[20] J. Torres, Python deep learning. Introduccián práctica con keras y tensorflow 2, Marcombo, 2020.

[21] V. Yepes Piqueras, «El aprendizaje profundo (deep learning) en la optimización de estructuras,» [En línea]. Available: https://tinylink.info/11cam. [Último acceso: 10 Abril 2025].

[22] C. A. Ruiz y M. S. Basualdo, «Redes Neuronales: Conceptos Básicos y Aplicaciones.,» de *Informática Aplicada a la Ingeniería de Procesos – Orientación I*, Rosario, 2001.

[23] F. Lara Rosano, «FUNDAMENTOS DE REDES NEURONALES ARTIFICIALES».

[24] M. Franco Guzmán, M. A. Romero Romo, M. E. Palomar Pardavé, J. Á. Cobos Murcia, G. A. Álvarez Romero y D. Hernández Ramírez, «De neuronas biológicas a neuronas artificiales, el fascinante mundo de las redes neuronales,» [En línea]. Available:

https://short-url.org/11cao. [Último acceso: 20 Abril 2025].

[25] F. A. Incio Flores, D. L. Capuñay Sanchez y R. O. Estela Urbina, «Modelo de red neuronal artificial para predecir resultados académicos en la asignatura Matemática II,» [En línea]. Available: https://shorturl.at/ZzWia. [Último acceso: 26 Abril 2025].

[26] J. R. Hilera González y V. J. Martínez Hernando, Redes neuronales artificiales : fundamentos, modelos y aplicaciones, Madrid: Ra-Ma, 1995.

[27] N. Sánchez Anzola, «Máquinas de soporte vectorial y redes neuronales artificiales en la predicción del movimiento USD/COP spot intradiario,» [En línea]. Available: https://shorturl.at/IN3V6. [Último acceso: 26 Abril 2025].

[28] X. Basogain Olabe, REDES NEURONALES ARTIFICIALES Y SUS APLICACIONES, Bilbao: EPV-EHU.

[29] Mathworks, «¿Qué son las redes neuronales convolucionales?,» [En línea]. Available: https://shorturl.at/XtdlX. [Último acceso: 28 Abril 2025].

[30] D. Calvo, «Convolución,» [En línea]. Available: https://shorturl.at/Hj60t. [Último acceso: 27 Abril 2025].

[31] IBM, «¿Qué son las redes neuronales convolucionales?,» [En línea]. Available: https://acortar.link/hCbWOz. [Último acceso: 16 Abril 2025].

[32] J. Barrios, «Redes Neuronales Convolucionales. Inteligencia Artificial en Salud,» [En línea]. Available: https://acortar.link/VXAduo. [Último acceso: 10 Abril 2025].

[33] T. c. reader, «The Convolution/Pooling Operation for RGB images,» [En línea]. Available: https://shorturl.at/bv8EX. [Último acceso: 10 Abril 2025].

[34] IBM, «Guía de CRISP-DM de IBM SPSS Modeler,» [En línea]. Available: https://tinyurl.com/2qqkq7pg. [Último acceso: 14 05 2025].

[35] J. Fernández Gonzáles, Introducción a las metodologías ágiles, Catalunya: UOC.

[36] E. E. Gallardo Echenique, «Metodologia de la investigación,» [En línea]. Available: https://shorturl.at/ltiS8. [Último acceso: 16 Abril 2025].

Anexos

Anexo 1:

Instalación de YOLOv12s

```
# crear entorno virtual para tarea
python -m venv yolov12-env

# activar entorno virtual
yolov12-env\Scripts\activate

# actualizar pip
pip install --upgrade pip

# instalar librerias ultralytics (Yolo)
pip install ultralytics

# desinstalar completamente PyTorch y sus modulos relacionados
pip uninstall torch torchvision torchaudio -y

# instalar PyTorch con soporte para CUDA 11.8
pip install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cu118
```

Anexo 2:

Entrenamiento del modelo

```
# entrenar modelo
yolo task=detect mode=train model=yolov12s.pt data=data.yaml epochs=150 imgsz=640 patience=20
device=0 batch=16
```

Anexo 3:

Codigo post procesamiento

```
import os, cv2, numpy as np
from ultralytics import YOLO

# parametros
RUTA_MODELO, RUTA_IMEGENES, RUTA_SALIDA = "...", "...", "..."
CLASE_EJE, CLASE_EJE_LEV, CLASE_LIG, CLASE_PES = 0, 1, 2, 3
DIST_MAX = 250

os.makedirs(RUTA_SALIDA, exist_ok=True)
model = YOLO(RUTA_MODELO)

def centro(box): return [(box[0]+box[2])/2, (box[1]+box[3])/2]

def asociar(puntos, vehiculos, key):
        for p in puntos:
                dist_min, v_cercano = float('inf'), None
                for v in vehiculos:
                        d = np.linalg.norm(np.array(p) - np.array(centro(v['box'])))
                        if d < dist_min and d < DIST_MAX: dist_min, v_cercano = d, v
                if v_cercano: v_cercano[key] += 1

for f in ok.listdir(RUTA_IMEGENES):
        if not f.lower().endswith((".jpg", ".jpeg", ".png")): continue
        img = cv2.imread(ruta:=os.path.join(RUTA_IMEGENES,f))
        res = model(ruta, conf=0.4)[0]
        boxes, clases = res.boxes.xyxy.cpu().numpy(), res.boxes.cls.cpu().numpy()

        vehiculos, ejes, ejes_lev = [], [], []
        for b,c in zip(boxes, clases.astype(int)):
                if c in [CLASE_LIG, CLASE_PES]:
                        vehiculos.append({'box': b, 'tipo': "ligero" if c == CLASE_LIG else "pesado",
'ejes': 0, 'lev': 0})
                elif c == CLASE_EJE: ejes.append(centro(b))
                elif c == CLASE_EJE_LEV: ejes_lev.append((centro(b))

        asociar(ejes, vehiculos, 'ejes')
        asociar(ejes_lev, vehiculos, 'lev')

        for v in vehiculos:
                x1, y1, x2, y2 = map(int, v['box'])
                total = v['ejes'] + v['lev']
                label = f"{v['tipo']} {total}" + (f" ({v['lev']} levantado)" if v['lev'] else "")
                color = (0,255,0) if v['tipo'] == "ligero" else (0,165,255)
                cv2.rectangle(img, (x1, y1), (x2, y2), color, 2)
                (tw, th),_ = cv2.getTextSize(label, cv2.FONT_HERSHEY_SIMPLEX, 0.6, 1)
                cv2.rectangle(img, (x1, y1-25), (x1 + tw + 5, y1), color, -1)
                cv2. putText(img, label, (x1 + 2, y1 - 7), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 0,
0). 2)

        cv2.imwrite(os.path.join(RUTA_SALIDA, f), img)
```

Anexo 4:

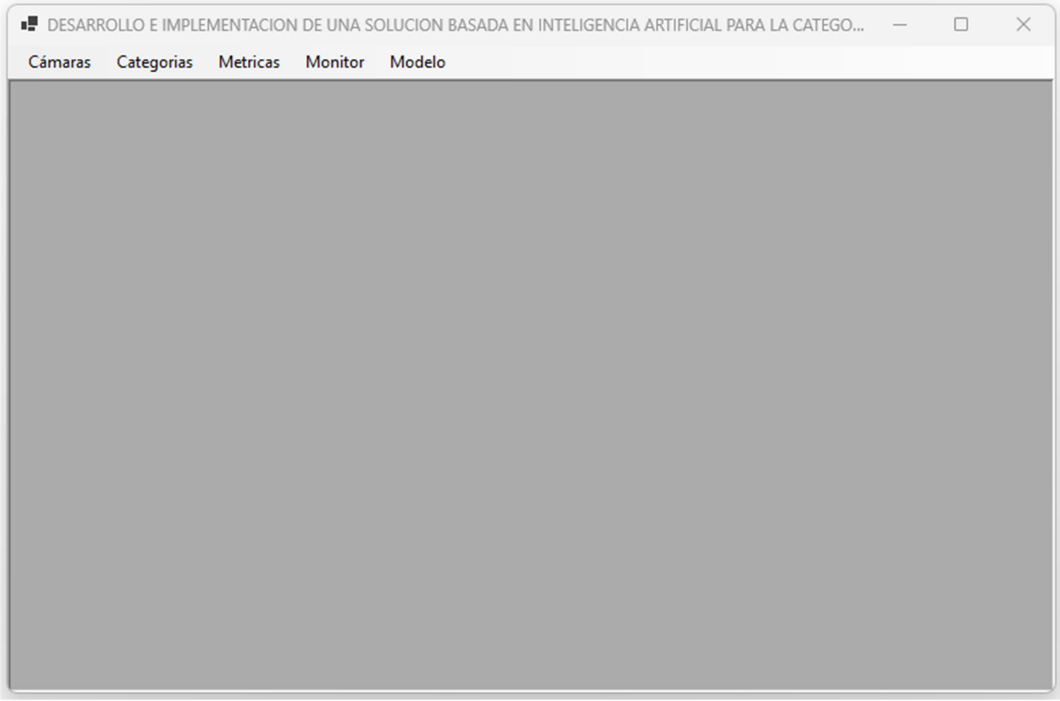
Migrar modelo a formato onnx

```
# migrar modelo a ONNX
yolo export model=best-s-rgb.pt format=onnx opset=12 dynamic=False simplify=False
```

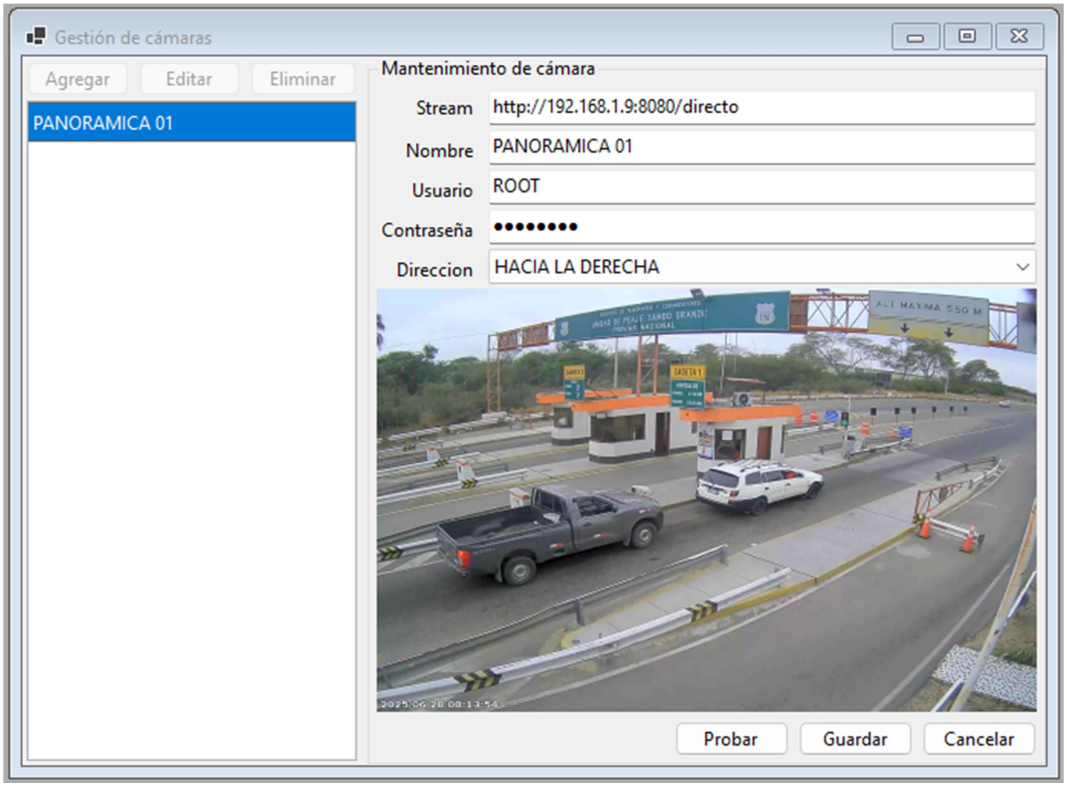
Anexo 5:

Interfaz carga de modelo

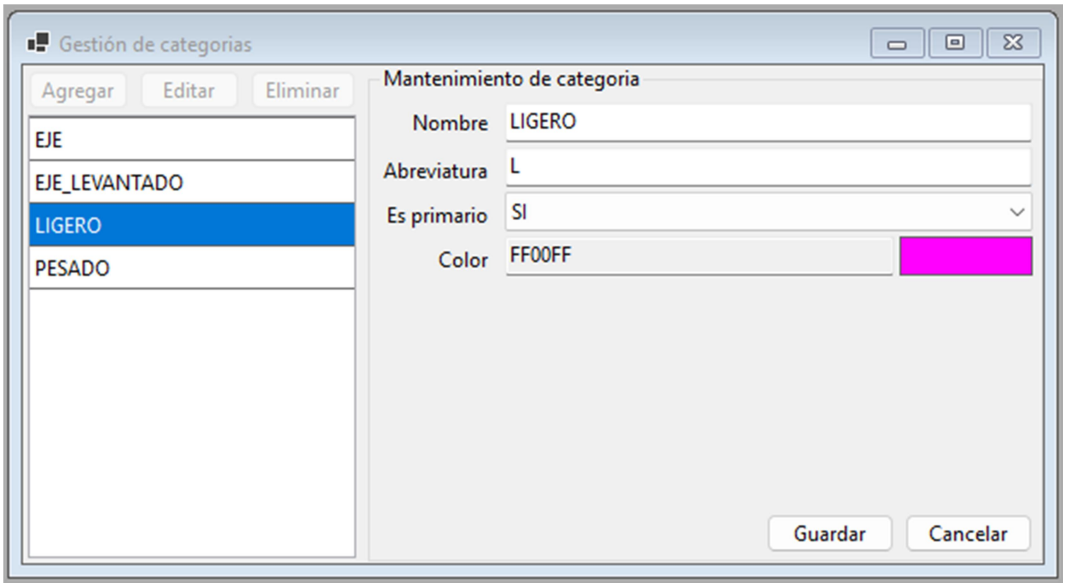
Interfaz pantalla principal



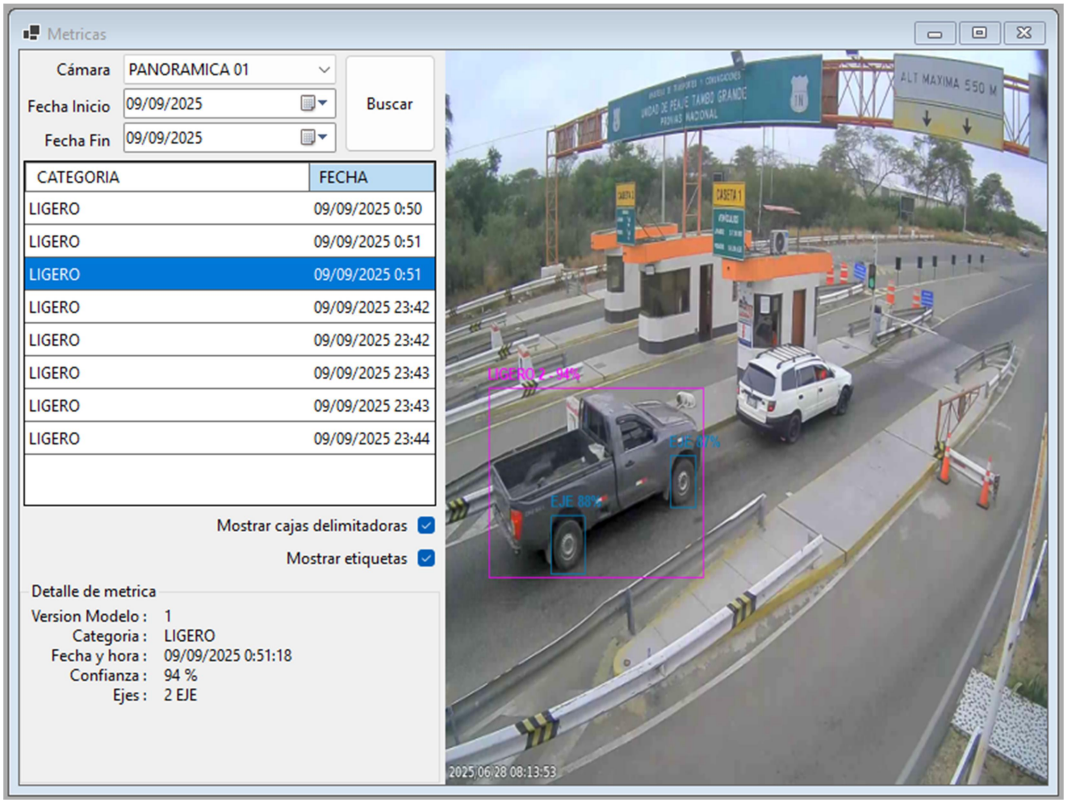
Interfaz gestion de camaras



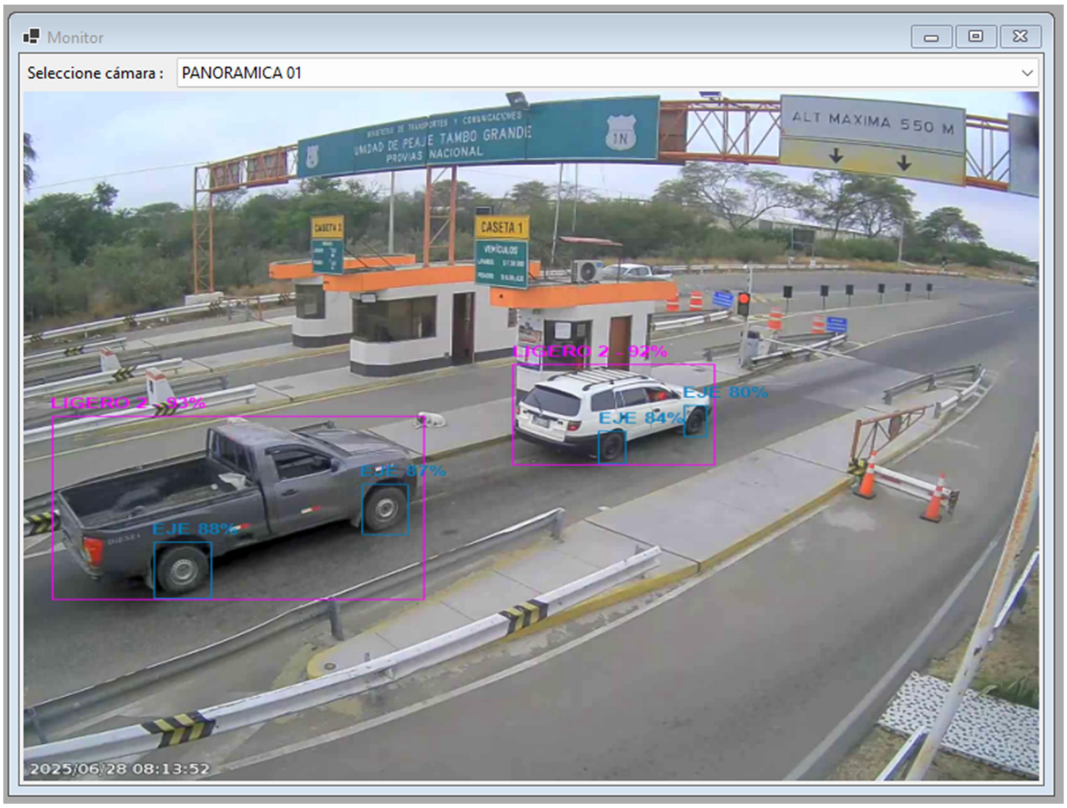
Interfaz gestion de categorias



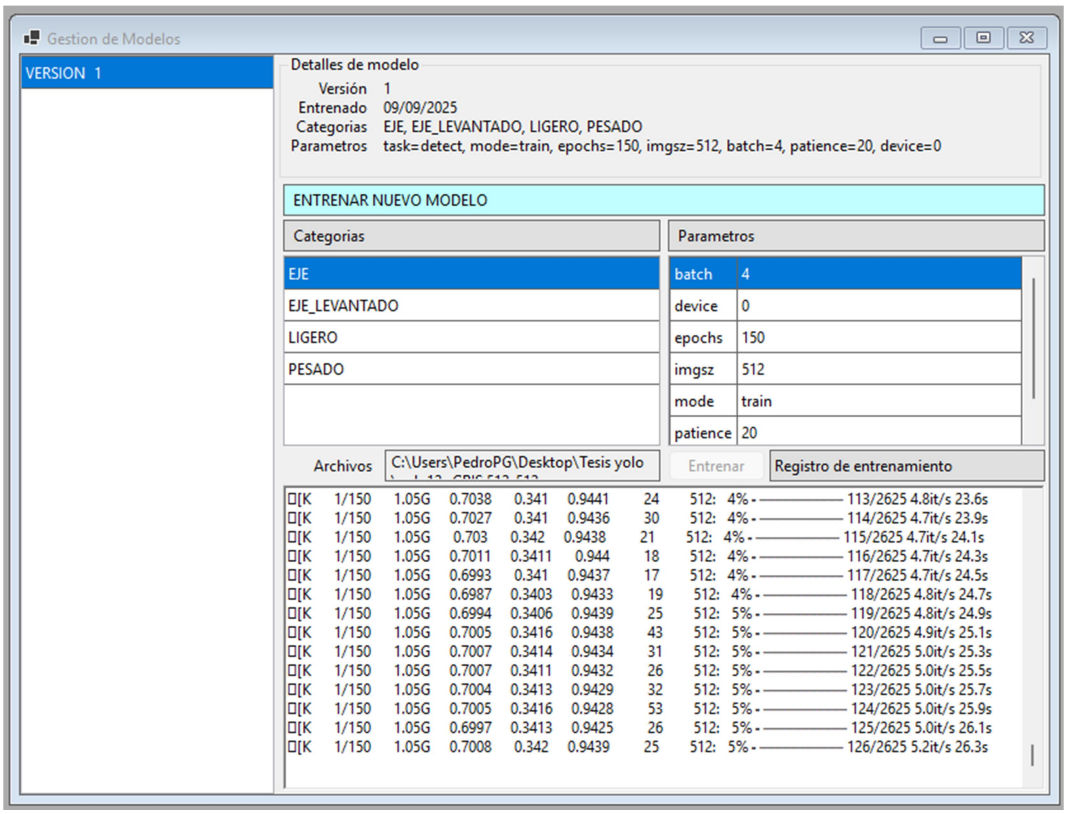
Interfaz metricas



Interfaz monitor



Interfaz gestion de modelo



Anexo 6:

Fragmento representativo de código para el módulo gestión de cámaras.

```
CREATE PROCEDURE SP__CAMARA___GUARDAR
   @ID INT OUTPUT,
   @NOMBRE VARCHAR(50),
   @STREAM VARCHAR(250),
   @USUARIO VARCHAR(100),
   @CLAVE VARCHAR(100),
   @DIRECCION INT
AS
BEGIN TRY
BEGIN TRANSACTION
   SET NOCOUNT ON;

   SET @NOMBRE = NULLIF(@NOMBRE, '');
   SET @STREAM = NULLIF(@STREAM, '');
   SET @USUARIO = NULLIF(@USUARIO, '');
   SET @CLAVE = NULLIF(@CLAVE, '');

   UPDATE    TB__CAMARA
   SET       NOMBRE = @NOMBRE,
   STREAM = @STREAM,
   USUARIO = @USUARIO,
   CLAVE = @CLAVE,
   DIRECCION = @DIRECCION
   WHERE    ID = @ID;

   IF (@@ROWCOUNT = 0)
   BEGIN
      DECLARE @TEMPORAL TABLE (ID INT);

      INSERT INTO TB__CAMARA (NOMBRE, STREAM, USUARIO, CLAVE, DIRECCION)
      OUTPUT INSERTED.ID INTO @TEMPORAL
      VALUES (@NOMBRE, @STREAM, @USUARIO, @CLAVE, @DIRECCION);

      SET @ID = (SELECT TOP 1 ID FROM @TEMPORAL);
   END

   COMMIT;
END TRY
BEGIN CATCH
   SET @ID = 0;
   DECLARE @MENSAJE AS VARCHAR(MAX) = ERROR_MESSAGE();
   RAISERROR (@MENSAJE, 16, 1);

   IF @@TRANCOUNT > 0 ROLLBACK TRANSACTION;
END CATCH
GO
```

```
namespace Tesis.Entidades {
  public class CamaraEntidad {
    public int Id { get; set; }
    public string Nombre { get; set; }
    public string Stream { get; set; }
    public string Usuario { get; set; }
    public string Clave { get; set; }
    public int Direccion { get; set; }
  }
```

```
}

namespace Tesis.Negocio {
  public abstract class CamaraNegocio {
    public static void Guardar(CamaraEntidad registro) {
      using (var con = new
SqlConnection(ConfigurationManager.ConnectionStrings["CON"].ConnectionString)) {
        con.Open();

        var parametros = new SqlParameter[] {
          new SqlParameter() { ParameterName = "@ID", Value = registro.Id },
          new SqlParameter() { ParameterName = "@NOMBRE", Value = registro.Nombre },
          new SqlParameter() { ParameterName = "@STREAM", Value = registro.Stream },
          new SqlParameter() { ParameterName = "@USUARIO", Value = registro.Usuario },
          new SqlParameter() { ParameterName = "@CLAVE", Value = registro.Clave },
          new SqlParameter() { ParameterName = "@DIRECCION", Value = registro.Direccion },
        };

        using (var cmd = new SqlCommand()) {
          cmd.Connection = con;
          cmd.CommandText = "SP__CAMARA___GUARDAR";
          cmd.CommandType = System.Data.CommandType.StoredProcedure;
          cmd.Parameters.AddRange(parametros);

          cmd.ExecuteNonQuery();
        }
      }
    }
    }
}

namespace Tesis.Negocio {
  public abstract class CamaraNegocio {
    public static void Guardar(CamaraEntidad registro) {
      CamaraDato.Guardar(registro);
    }
  }
}

namespace Tesis.Aplicacion {
  public partial class frmGestionCamara : Form {
    private void btnGuardar_Click(object sender, EventArgs e) {
      try {
        CamaraActiva.Stream = txtStream.Text;
        CamaraActiva.Nombre = txtNombre.Text;
        CamaraActiva.Usuario = txtUsuario.Text;
        CamaraActiva.Clave = txtClave.Text;
        CamaraActiva.Direccion = cboDireccion.SelectedIndex;

        CamaraNegocio.Guardar(CamaraActiva);
        CargaRegistros();
        btnCancelar.PerformClick();
      } catch (Exception ex) {
        MessageBox.Show(
          ex.Message.ToString(),
          "Error al intentar Guardar",
          MessageBoxButtons.OK,
          MessageBoxIcon.Exclamation
        );
      }
```

```
    }
  }
}
```

Anexo 7:

Fragmento representativo de código modulo gestión de categorías.

```
CREATE PROCEDURE SP__CATEGORIA___GUARDAR
   @ID INT OUTPUT,
   @NOMBRE VARCHAR(50),
   @ABREVIATURA VARCHAR(50),
   @COLOR VARCHAR(250),
   @ES_CONTENEDOR BIT
AS
BEGIN TRY
BEGIN TRANSACTION
   SET NOCOUNT ON;

   DECLARE @ORDEN TINYINT;

   SET @NOMBRE = NULLIF(@NOMBRE, '');
   SET @ABREVIATURA = NULLIF(@ABREVIATURA, '');
   SET @COLOR = NULLIF(@COLOR, '');
   SET @ORDEN = (SELECT COALESCE(MAX(ORDEN), 0) FROM TB__CATEGORIA);

   UPDATE    TB__CATEGORIA
   SET       NOMBRE = @NOMBRE,
   ABREVIATURA = @ABREVIATURA,
   COLOR = @COLOR,
   ES_CONTENEDOR = @ES_CONTENEDOR
   WHERE    ID = @ID;

   IF (@@ROWCOUNT = 0)
   BEGIN
      DECLARE @TEMPORAL TABLE (ID INT);

      INSERT INTO TB__CATEGORIA (
         NOMBRE, ABREVIATURA, ORDEN, ENTRENADO, COLOR, ES_CONTENEDOR
      )
      OUTPUT INSERTED.ID INTO @TEMPORAL
      VALUES (@NOMBRE, @ABREVIATURA, @ORDEN, 0, @COLOR, @ES_CONTENEDOR);

      SET @ID = (SELECT TOP 1 ID FROM @TEMPORAL);
   END

   COMMIT;
END TRY
BEGIN CATCH
   SET @ID = 0;
   DECLARE @MENSAJE AS VARCHAR(MAX) = ERROR_MESSAGE();
   RAISERROR (@MENSAJE, 16, 1);

   IF @@TRANCOUNT > 0 ROLLBACK TRANSACTION;
END CATCH
GO
```

```
namespace Tesis.Entidades {
  public class CategoriaEntidad {
    public int Id { get; set; }
    public string Nombre { get; set; }
    public string Abreviatura { get; set; }
    public byte Orden { get; set; }
    public bool Entrenado { get; set; }
```

```
    public string Color { get; set; }
    public bool Es_Contenedor { get; set; }
  }
}

namespace Tesis.Negocio {
  public abstract class CategoriaNegocio {
    public static void Guardar(CategoriaEntidad registro) {
      using (var con = new
SqlConnection(ConfigurationManager.ConnectionStrings["CON"].ConnectionString)) {
        con.Open();

        var parametros = new SqlParameter[] {
          new SqlParameter() { ParameterName = "@ID", Value = registro.Id },
          new SqlParameter() { ParameterName = "@NOMBRE", Value = registro.Nombre },
          new SqlParameter() { ParameterName = "@ABREVIATURA", Value = registro.Abreviatura },
          new SqlParameter() { ParameterName = "@COLOR", Value = registro.Color },
          new SqlParameter() { ParameterName = "@ES_CONTENEDOR", Value =
registro.Es_Contenedor },
        };

        using (var cmd = new SqlCommand()) {
          cmd.Connection = con;
          cmd.CommandText = "SP__CATEGORIA___GUARDAR";
          cmd.CommandType = System.Data.CommandType.StoredProcedure;
          cmd.Parameters.AddRange(parametros);

          cmd.ExecuteNonQuery();
        }
      }
    }
  }
}

namespace Tesis.Negocio {
  public abstract class CategoriaNegocio {
    public static void Guardar(CategoriaEntidad registro) {
      CategoriaDato.Guardar(registro);
    }
  }
}

namespace Tesis.Aplicacion {
  public partial class frmGestionCamara : Form {
    private void btnGuardar_Click(object sender, EventArgs e) {
      try {
        CategoriaActiva.Nombre = txtNombre.Text;
        CategoriaActiva.Abreviatura = txtAbreviatura.Text;
        CategoriaActiva.Es_Contenedor = Convert.ToBoolean(cboEsContenedor.SelectedIndex);
        CategoriaActiva.Color = txtColor.Text;

        CategoriaNegocio.Guardar(CategoriaActiva);
        CargaRegistros();
        btnCancelar.PerformClick();
      } catch (Exception ex) {
        MessageBox.Show(
          ex.Message.ToString(),
          "Error al intentar Guardar",
          MessageBoxButtons.OK,
          MessageBoxIcon.Exclamation
```

```
      );
    }
  }
 }
}
```

Anexo 8:

Fragmento representativo de código modulo de métricas

```
CREATE PROCEDURE SP__METRICA___REGISTRO____OBTENER_POR_ID
   @ID INT
AS
BEGIN
   SELECT
   VW__METRICA.ID,
   VW__METRICA.IDMETRICA,
   VW__METRICA.IDCATEGORIA,
   VW__METRICA.IDCAMARA,
   VW__METRICA.IDMODELO,
   VW__METRICA.FECHA,
   VW__METRICA.CONFIABILIDAD,
   VW__METRICA.X,
   VW__METRICA.Y,
   VW__METRICA.W,
   VW__METRICA.H,
   VW__METRICA.IMAGEN,
   JSON_QUERY(CAT.CATEGORIA, '$[0]') AS CATEGORIA,
   JSON_QUERY(CAM.CAMARA, '$[0]') AS CAMARA,
   JSON_QUERY(M.MODELO, '$[0]') AS MODELO,
   METRICA.EJES
   FROM
   VW__METRICA
   CROSS APPLY (
     SELECT    *
     FROM    VW__CATEGORIA
     WHERE    ID = VW__METRICA.IDCATEGORIA
     FOR JSON PATH
   ) CAT (CATEGORIA)
   CROSS APPLY (
     SELECT    *
     FROM    VW__CAMARA
     WHERE    ID = VW__METRICA.IDCAMARA
     FOR JSON PATH
   ) CAM (CAMARA)
   CROSS APPLY (
     SELECT    *
     FROM    VW__MODELO
     WHERE    ID = VW__METRICA.IDMODELO
     FOR JSON PATH
   ) M (MODELO)
   CROSS APPLY (
     SELECT  TB__METRICA.*,
       JSON_QUERY(C.CATEGORIA, '$[0]') AS CATEGORIA
     FROM    TB__METRICA
       CROSS APPLY (
         SELECT    *
         FROM    TB__CATEGORIA
         WHERE    ID = TB__METRICA.IDCATEGORIA
         FOR JSON PATH
       ) C (CATEGORIA)
     WHERE    IDMETRICA = VW__METRICA.ID
     FOR JSON PATH
   ) METRICA (EJES)
   WHERE
   VW__METRICA.ID = @ID
   AND VW__METRICA.IDMETRICA IS NULL
```

```
END
GO

CREATE PROCEDURE
SP__METRICA_RECURSIVA___REGISTRO____OBTENER_POR_IDMETRICA
   @IDMETRICA INT
AS
BEGIN
   SELECT
   VW__METRICA.CONFIABILIDAD,
   VW__METRICA.X,
   VW__METRICA.Y,
   VW__METRICA.H,
   VW__METRICA.W,
   VW__CATEGORIA.NOMBRE,
   VW__CATEGORIA.COLOR
   FROM
   VW__METRICA
   INNER JOIN VW__CATEGORIA ON ( VW__METRICA.IDCATEGORIA = VW__CATEGORIA.ID )
   WHERE
   VW__METRICA.IDMETRICA = @IDMETRICA
END
GO
```

```
namespace Tesis.Entidades {
  public class MetricaEntidad {
    public int Id { get; set; }
    public MetricaEntidad Metrica { get; set; }
    public CategoriaEntidad Categoria { get; set; }
    public CamaraEntidad Camara { get; set; }
    public ModeloEntidad Modelo { get; set; }
    public DateTime Fecha { get; set; }
    public float Confiabilidad { get; set; }
    public float X { get; set; }
    public float Y { get; set; }
    public float W { get; set; }
    public float H { get; set; }
    public byte[] Imagen { get; set; }
    public List<MetricaEntidad> Ejes { get; set; }

    // Rastreo
    public int rastreo { get; set; }
    public bool registrado_en_bd { get; set; }
    public int FramesPerdidos { get; set; }

    // Direccion
    public float UltimaX { get; set; }
    public float UltimaY { get; set; }
    public Direccion Orientacion { get; set; }

    public enum Direccion {
      HaciaLaIzquierda = 0,
      HaciaLaDerecha = 1,
    }

    public MetricaEntidad() {
      Categoria = new CategoriaEntidad();
      Camara = new CamaraEntidad();
      Modelo = new ModeloEntidad();
      Ejes = new List<MetricaEntidad>();
```

```
    }
  }
}

namespace Tesis.Datos {
  public abstract class MetricaDato {
    public static MetricaEntidad ObtenerPorId(int id) {
      var registro = new MetricaEntidad();

      using (var con = new
SqlConnection(ConfigurationManager.ConnectionStrings["CON"].ConnectionString)) {
        con.Open();

        var parametros = new SqlParameter[] {
          new SqlParameter() { ParameterName = "@ID", Value = id },
        };

        using (var cmd = new SqlCommand()) {
          cmd.Connection = con;
          cmd.CommandText = "SP__METRICA___REGISTRO____OBTENER_POR_ID";
          cmd.CommandType = System.Data.CommandType.StoredProcedure;
          cmd.Parameters.AddRange(parametros);

          cmd.ExecuteNonQuery();

          using (var reader = cmd.ExecuteReader()) {
            while (reader.Read()) {
              registro.Fecha = reader.GetDateTime(reader.GetOrdinal("FECHA"));
              registro.Confiabilidad = reader.GetFloat(reader.GetOrdinal("CONFIABILIDAD"));
              registro.X = reader.GetFloat(reader.GetOrdinal("X"));
              registro.Y = reader.GetFloat(reader.GetOrdinal("Y"));
              registro.W = reader.GetFloat(reader.GetOrdinal("W"));
              registro.H = reader.GetFloat(reader.GetOrdinal("H"));
              registro.Imagen = (byte[])reader.GetValue(reader.GetOrdinal("IMAGEN"));

              var categoria_json = reader.GetString(reader.GetOrdinal("CATEGORIA"));
              var camara_json = reader.GetString(reader.GetOrdinal("CAMARA"));
              var modelo_json = reader.GetString(reader.GetOrdinal("MODELO"));
              var ejes_json = reader.GetString(reader.GetOrdinal("EJES"));

              if (!string.IsNullOrEmpty(categoria_json)) {
                registro.Categoria = JsonConvert.DeserializeObject<CategoriaEntidad>(categoria_json);
              }

              if (!string.IsNullOrEmpty(camara_json)) {
                registro.Camara = JsonConvert.DeserializeObject<CamaraEntidad>(camara_json);
              }

              if (!string.IsNullOrEmpty(modelo_json)) {
                registro.Modelo = JsonConvert.DeserializeObject<ModeloEntidad>(modelo_json);
              }

              if (!string.IsNullOrEmpty(ejes_json)) {
                registro.Ejes = JsonConvert.DeserializeObject<List<MetricaEntidad>>(ejes_json);
              }
            }
          }
        }
      }
    }
```

```
      return registro;
    }

    public static List<MetricaEntidad> ObtenerPorIdMetrica(int idmetrica) {
      var lista = new List<MetricaEntidad>();

      using (var con = new
SqlConnection(ConfigurationManager.ConnectionStrings["CON"].ConnectionString)) {
        con.Open();

        var parametros = new SqlParameter[] {
          new SqlParameter() { ParameterName = "@IDMETRICA", Value = idmetrica },
        };

        using (var cmd = new SqlCommand()) {
          cmd.Connection = con;
          cmd.CommandText =
"SP__METRICA_RECURSIVA___REGISTRO____OBTENER_POR_IDMETRICA";
          cmd.CommandType = System.Data.CommandType.StoredProcedure;
          cmd.Parameters.AddRange(parametros);

          cmd.ExecuteNonQuery();

          using (var reader = cmd.ExecuteReader()) {
            while (reader.Read()) {
              lista.Add(new MetricaEntidad() {
                Confiabilidad = reader.GetFloat(reader.GetOrdinal("CONFIABILIDAD")),
                X = reader.GetFloat(reader.GetOrdinal("X")),
                Y = reader.GetFloat(reader.GetOrdinal("Y")),
                W = reader.GetFloat(reader.GetOrdinal("W")),
                H = reader.GetFloat(reader.GetOrdinal("H")),
                Categoria = new CategoriaEntidad() {
                  Nombre = reader.GetString(reader.GetOrdinal("NOMBRE")),
                  Color = reader.GetString(reader.GetOrdinal("COLOR")),
                }
              });
            }
          }
        }
      }

      return lista;
    }
    }
}

namespace Tesis.Negocio {
  public abstract class MetricaNegocio {
    public static MetricaEntidad ObtenerPorId(int id) {
      return MetricaDato.ObtenerPorId(id);
    }

    public static List<MetricaEntidad> ObtenerPorIdMetrica(int idmetrica) {
      return MetricaDato.ObtenerPorIdMetrica(idmetrica);
    }
    }
}

namespace Tesis.Aplicacion {
  public partial class frmMetrica : Form {
```

```
    private void dgvDatos_CellEnter(object sender, DataGridViewCellEventArgs e) {
      var metrica = MetricaNegocio.ObtenerPorId(Convert.ToInt32(dgvDatos[ID.Index,
e.RowIndex].Value));
      var ejes = MetricaNegocio.ObtenerPorIdMetrica(Convert.ToInt32(dgvDatos[ID.Index,
e.RowIndex].Value));

      var imagen = Resources._64ximagen;
      var modo_visualizacion_imagen = PictureBoxSizeMode.CenterImage;

      if (metrica.Imagen != null && metrica.Imagen.Length > 0) {
        imagen = ConvierteBytesABitmap(metrica.Imagen);
        modo_visualizacion_imagen = PictureBoxSizeMode.StretchImage;

        if (chkMostrarCajaDelimitadora.Checked) {
          imagen = DibujaCajaDelimitadora(metrica, imagen, chkMostrarEtiquetas.Checked);
        }
      }

      picImagen.Image = imagen;
      picImagen.SizeMode = modo_visualizacion_imagen;

      lblCategoria.Text = metrica.Categoria.Nombre;
      lblFechaHora.Text = Convert.ToString(metrica.Fecha);
      lblConfianza.Text = metrica.Confiabilidad.ToString("P0");
      lblVersionModelo.Text = Convert.ToString(metrica.Modelo.Version);
      lblEjes.Text = string.Join(
        Environment.NewLine,
        ejes.GroupBy(x => x.Categoria.Nombre).Select(g => $"{g.Count()} {g.Key}")
      );
    }
  }
}
```

Anexo 9:

Fragmento representativo de código modulo monitor.

```
namespace Tesis.Aplicacion {
  public partial class frmMonitor : Form {
    private async Task IniciaInferencia() {
      foreach (var camara in Program.Camaras) {
        _ = Task.Run(() => {
          var parametros = new List<NamedOnnxValue>();
          using var fuente = new VideoCapture(camara.Stream);

          if (!fuente.IsOpened()) {
            return;
          }

          using var mat = new Mat();

          while (!cts.Token.IsCancellationRequested) {
            if (fuente.Read(mat) && !mat.Empty()) {
              var metricas = new List<MetricaEntidad>();
              var tensor_entrada = GeneraTensor(mat);

              parametros.Clear();
              parametros.Add(NamedOnnxValue.CreateFromTensor("images", tensor_entrada));

              using (var resultado = Program.Sesion.Run(parametros)) {
                var salida = resultado.First().AsTensor<float>();
                var numero_batch = salida.Dimensions[0];
                var numero_canales = salida.Dimensions[1];
                var numero_predicciones = salida.Dimensions[2];
                var numero_clases = Program.Categorias.Count;

                for (int i = 0; i < numero_predicciones; i++) {
                  float x = salida[0, 0, i];
                  float y = salida[0, 1, i];
                  float w = salida[0, 2, i];
                  float h = salida[0, 3, i];

                  //clases
                  int clsId = -1;
                  float clsConf = 0;

                  for (int c = 0; c < numero_clases; c++) {
                    float score = salida[0, 4 + c, i];

                    if (score > clsConf) {
                      clsConf = score;
                      clsId = c;
                    }
                  }

                  if (clsConf > ConfThreshold) {
                    metricas.Add(new MetricaEntidad {
                      X = x - w / 2,
                      Y = y - h / 2,
                      W = w,
                      H = h,
                      Confiabilidad = clsConf,
                      Categoria = Program.Categorias[clsId]
                    });
```

```
            }
          }

          // NMS
          metricas = NMS(metricas, IouThreshold);

          // Agrupar detecciones hijas dentro de padres
          metricas = AgruparDetecciones(metricas);

          // Eliminar rezagados
          metricas = EliminarDeteccionesRezagadas(metricas);

          // Eliminar detecciones parciales por clase
          metricas = EliminarDeteccionesParcialesPorClase(metricas,
            Program.Categorias,
            640,
            640,
            Program.Categorias.Where(c => c.Es_Contenedor).Select(c => c.Nombre).ToArray(),
            minVisibleRatio: 1f,
            borderMarginPx: 1
          );

          // Rastreo y seguimiento
          metricas = SeguimientoDetecciones(metricas);

          // ELiminar vehiculos con 1 solo eje
          metricas = EliminarConUnEje(metricas);

          // Eliminar segun movimiento
          metricas = metricas.Where(
            m => m.Orientacion == (Direccion)(Convert.ToInt32(camara.Direccion))
          ).ToList();
        }

        var img = AjustaImagen(BitmapConverter.ToBitmap(mat));

        // registrar en BD
        foreach (var metrica in FiltrarSoloCentro(metricas, 640, 640, 0.5f)) {
          var objPersistente = RastreoObjetosActivos.FirstOrDefault(o => o.rastreo ==
metrica.rastreo);

          if (objPersistente != null && !objPersistente.registrado_en_bd) {
            metrica.Camara = camara;
            metrica.Modelo = Program.Modelo;
            metrica.Fecha = DateTime.Now;
            metrica.Imagen = frmMetrica.ConvierteImagenABytes(img);

            var hijos = metrica.Ejes.Select(item => new {
              IDCATEGORIA = 1,
              CONFIABILIDAD = item.Confiabilidad,
              X = item.X,
              Y = item.Y,
              W = item.W,
              H = item.H,
            }).ToList<dynamic>();

            MetricaNegocio.Guardar(metrica, hijos);
            objPersistente.registrado_en_bd = true;
          }
        }
```

```
            // mostrar
            BeginInvoke((MethodInvoker)(() => {
              if (Visible && camara.Id == Convert.ToInt32(cboCamara.SelectedValue)) {
                using var g = Graphics.FromImage(img);

                foreach (var metrica in metricas) {
                  if (metrica != null) {
                    frmMetrica.DibujaCajaDelimitadora(metrica, g, true);
                  }
                }

                pictureBox.Image?.Dispose();
                pictureBox.Image = img;
              }
            }));

            Task.Delay(100, cts.Token);
          }
        }
      });
    }
  }
 }
}
```

Anexo 10:

Fragmento representativo de código modulo de modelo

```
namespace Tesis.Aplicacion {
  public partial class frmModelo : Form {
    private async void btnEntrenar_Click(object sender, EventArgs e) {
      btnEntrenar.Enabled = false;
      dgvDatos.Enabled = false;

      txtSalida.Clear();

      try {
        var path = Path.GetDirectoryName(Assembly.GetExecutingAssembly().Location);
        var version_nueva = ModeloActivo.Version + 1;

        var args_entrenamiento = new List<string>() {
          $@"model=""{path}\Modelo\Versiones\{ModeloActivo.Version}\best.pt""",
          $@"data=""{txtArchivos.Text}\data.yaml""",
          $@"project=""{path}\Modelo\Versiones\{version_nueva}""",
          $"name=entrenamiento",
        };

        var args_genera_onnx = new List<string>() {
          $@"model=""{path}\Modelo\Versiones\{version_nueva}\best.pt""",
          $@"format=onnx",
          $@"opset=12",
          $@"dynamic=True",
          $@"simplify=False",
          $"export",
        };

        foreach (DataGridViewRow row in dgvParametros.Rows) {
          var nombre = row.Cells[PARAMETRO_NOMBRE.Index].Value;
          var valor = row.Cells[PARAMETRO_VALOR.Index].Value;
          args_entrenamiento.Add($"{nombre}={valor}");
        }

        await ExecuteProcess("yolo", args_entrenamiento);

        var sourcePath =
$@"""{path}\Modelo\Versiones\{version_nueva}\entrenamiento\weights\best.pt""";
        var destFolder = $@"""{path}\Modelo\Versiones\{version_nueva}""";
        var destPath = Path.Combine(destFolder, "best.pt");

        File.Copy(sourcePath, destPath, true);

        await ExecuteProcess("yolo", args_genera_onnx);

        var modelo = new ModeloEntidad() {
          Version = version_nueva,
          Modelo = new ModeloEntidad() {
            Id = ModeloActivo.Id
          },
        };

        foreach (DataGridViewRow row in dgvCategorias.Rows) {
          modelo.ModeloCategoria.Add(new ModeloCategoriaEntidad() {
            Categoria = new CategoriaEntidad() {
              Id = Convert.ToInt32(row.Cells[CATEGORIA_ID.Index].Value),
            }
```

```
        });
      }

      foreach (DataGridViewRow row in dgvParametros.Rows) {
        modelo.ModeloParametro.Add(new ModeloParametroEntidad() {
          Valor = Convert.ToString(row.Cells[PARAMETRO_VALOR.Index].Value),
          Parametro = new ParametroEntidad() {
            Id = Convert.ToInt32(row.Cells[PARAMETRO_ID.Index].Value),
          },
        });
      }

      ModeloNegocio.Guardar(modelo);
      CargaRegistros();
    }
    catch (Exception ex) {
      MessageBox.Show("Error: " + ex.Message.ToString());
    } finally {
      btnEntrenar.Enabled = true;
      dgvDatos.Enabled = true;
    }
  }
 }
}
```