



UNIVERSIDAD JUÁREZ AUTÓNOMA DE TABASCO

**DIVISIÓN ACADÉMICA DE CIENCIAS Y TECNOLOGÍAS DE LA
INFORMACIÓN**

**MODELO DE PLATAFORMA DE APRENDIZAJE PROFUNDO PARA EL
CONTROL DE ACCESO VEHICULAR**

**TESIS PARA OBTENER EL GRADO DE:
DOCTORADO EN CIENCIAS DE LA COMPUTACIÓN**

PRESENTA:

MANUEL ANTONIO GARCÍA HERRERA

BAJO LA DIRECCIÓN DE:

DR. OSCAR ALBERTO CHÁVEZ BOSQUEZ

CUNDUACÁN, TABASCO, A: SEPTIEMBRE 2026



UNIVERSIDAD JUÁREZ AUTÓNOMA DE TABASCO

**DIVISIÓN ACADÉMICA DE CIENCIAS Y TECNOLOGÍAS DE LA
INFORMACIÓN**

**MODELO DE PLATAFORMA DE APRENDIZAJE PROFUNDO PARA EL
CONTROL DE ACCESO VEHICULAR**

**TESIS PARA OBTENER EL GRADO DE:
DOCTORADO EN CIENCIAS DE LA COMPUTACIÓN**

PRESENTA:

MANUEL ANTONIO GARCÍA HERRERA

BAJO LA DIRECCIÓN DE:

DR. OSCAR ALBERTO CHÁVEZ BOSQUEZ

Declaración de Autoría y Originalidad

En la Ciudad de Cunduacán el día Treinta del mes de Septiembre del año 2026, el que suscribe **Manuel Antonio García Herrera**, alumno del Programa de **Doctorado en Ciencias de la Computación** con número de matrícula **231H18003**, adscrito a la **División Académica de Ciencias y Tecnologías de la Información**, de la Universidad Juárez Autónoma de Tabasco, como autor de la Tesis presentada para la obtención de Grado de Doctorado y titulada **Modelo de plataforma de aprendizaje profundo para el control de acceso vehicular**, dirigida por el Dr. Oscar Alberto Chávez Bosquez

DECLARO QUE: La Tesis es una obra original que no infringe los derechos de propiedad intelectual ni los derechos de propiedad industrial u otros, de acuerdo con el ordenamiento jurídico vigente, en particular, la LEY FEDERAL DEL DERECHO DE AUTOR (Decreto por el que se reforman y adicionan diversas disposiciones de la Ley Federal del Derecho de Autor del 01 de Julio de 2020 regularizando, aclarando y armonizando las disposiciones legales vigentes sobre la materia), en particular, las disposiciones referidas al derecho de cita. Del mismo modo, asumo frente a la Universidad cualquier responsabilidad que pudiera derivarse de la autoría o falta de originalidad o contenido de la Tesis presentada de conformidad con el ordenamiento jurídico vigente.

Cunduacán, Tabasco a 30 de Septiembre de 2026.



Estudiante: Manuel Antonio García Herrera



UJAT

UNIVERSIDAD JUÁREZ
AUTÓNOMA DE TABASCO

"ESTUDIO EN LA DUDA. ACCIÓN EN LA FE"



DIVISIÓN ACADÉMICA DE
CIENCIAS Y TECNOLOGÍAS
DE LA INFORMACIÓN



2026
año de
Margarita
Maza

Cunduacán, Tabasco, a 10 de agosto de 2026
Oficio No. 1982/2026/DACYTI/D

Asunto: Autorización de impresión de Tesis

C. Manuel Antonio García Herrera

Egresado del Doctorado en Ciencias de la Computación

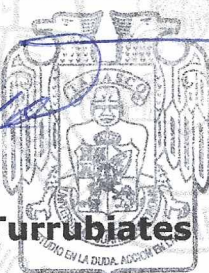
En virtud de que cumple satisfactoriamente los requisitos establecidos en el Reglamento General de Estudios de Posgrado vigente en la Universidad, informo a Usted que se autoriza la impresión del trabajo recepcional **"Modelo de plataforma de aprendizaje profundo para el control de acceso vehicular"**, para presentar examen y obtener el Grado de Doctor en Ciencias de la Computación.

Sin otro particular, aprovecho la ocasión para enviarle un afectuoso saludo.

Atentamente

Dra. Laura Beatriz Vidal Turrubiates

Directora



DIVISIÓN ACADÉMICA DE
CIENCIAS Y TECNOLOGÍAS
DE LA INFORMACIÓN

C.c.p. Dr. Eduardo Hernández de la Cruz. - Encargado del despacho de la Coordinación de Posgrado.
Alumno.

DRA.*LBVT/EHC

Av. Universidad s/n, Zona de la Cultura, Col. Magisterial,
Villahermosa, Centro, Tabasco, Mex. C.P. 86040.
Tel (993) 358 15 00 e-Mail: rectoria@ujat.mx

www.ujat.mx

Carta de Cesión de Derechos

Villahermosa, Tabasco a 30 de Septiembre de 2026.

Por medio de la presente manifiesto haber colaborado como AUTOR en la producción, creación y/o realización de la obra denominada: **Modelo de plataforma de aprendizaje profundo para el control de acceso vehicular.**

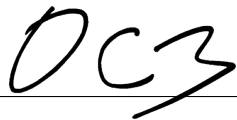
Con fundamento en el artículo 83 de la Ley Federal del Derecho de Autor y toda vez que, la creación y/o realización de la obra antes mencionada se realizó bajo la comisión de la Universidad Juárez Autónoma de Tabasco; entendemos y aceptamos el alcance del artículo en mención de que tenemos el derecho al reconocimiento como autores de la obra, y a la Universidad Juárez Autónoma de Tabasco mantendrá en un 100 % la titularidad de los derechos patrimoniales por un período de 20 años sobre la obra en la que colaboramos, por lo anterior, cedemos el derecho patrimonial exclusivo en favor de la Universidad.

COLABORADOR



Estudiante: Manuel Antonio García Herrera

TESTIGOS



Dr. Oscar Alberto Chávez Bosquez



Dra. Maricela García Avalos

Índice general

Índice de Tablas	VI
Índice de Figuras	VIII
Resumen	XI
Abstract	XII
1. Protocolo de Tesis	1
1.1. Introducción	1
1.2. Marco Conceptual	3
1.2.1. Cómputo en el borde	3
1.2.2. Aprendizaje automático	4
1.2.3. Aprendizaje profundo	5
1.2.4. <i>Transfer learning</i>	6
1.2.5. Visión computacional	6
1.2.5.1. Detección de objetos	7
1.2.5.2. Seguimiento de objetos	7
1.2.5.3. Reconocimiento óptico de caracteres (OCR)	8
1.2.5.4. Reconocimiento automático de placas vehiculares (ANPR)	8
1.2.6. <i>Single-Board Computer</i> (SBC)	9
1.2.6.1. Raspberry Pi	9
1.2.6.2. Cámara Raspberry Pi Camera Module 3	10
1.3. Estado del arte	11

1.3.1. Aprendizaje profundo aplicado en la <i>Edge computing</i>	11
1.3.2. Aprendizaje profundo aplicado en la detección de objetos para el control de acceso vehicular	13
1.3.3. Seguimiento de objetos y algoritmos de <i>tracking</i>	15
1.3.4. Reconocimiento automático de placas vehiculares (ANPR)	17
1.4. Planteamiento del problema	19
1.5. Preguntas de investigación	21
1.6. Hipótesis	21
1.7. Objetivos	21
1.7.1. Objetivo general	21
1.7.2. Objetivos específicos	21
1.8. Alcances y limitaciones	22
1.9. Justificación	22
1.10. Método	23
1.10.1. Descripción de Etapas de la Investigación	25
1.11. Contribución y resultados esperados	26
1.12. Cronograma	27
2. Modelo personalizado para la detección de vehículos	38
2.1. Introducción	40
2.2. Materiales y método	42
2.2.1. Modelo SSDMobilenetV2FPNlite	42
2.2.2. Bibliotecas y herramientas	42
2.2.3. Método	43
2.2.4. Dataset UJAT	43
2.2.5. Pre-procesamiento	44
2.2.6. Etiquetado	44
2.2.7. Aprendizaje por transferencia	46
2.2.8. Pruebas iniciales del modelo SSDMobileNetV2FPNlite	47
2.2.9. Re-entrenamiento del modelo (<i>Transfer learning</i>)	48

2.3. Pruebas y resultados	49
2.3.1. Evaluación del modelo original SSDMobileNetV2FPNlite	50
2.3.2. Evaluación del modelo optimizado	51
2.4. Conclusiones	53
3. Detección y seguimiento de objetos con YOLO11-ByteTrack	55
3.1. Introducción	57
3.2. Trabajos relacionados	58
3.3. Metodología	59
3.3.1. Materiales	59
3.3.2. Método	60
3.4. Conjunto de datos UJAT	60
3.4.1. Preprocesamiento de video	62
3.5. Diseño experimental	63
3.5.1. Conteo de objetos en la Región de Interés (RoI)	63
3.5.2. Conteo de objetos cruzando la línea (CTL)	65
3.6. Resultados y evaluación	66
3.6.1. Métricas de evaluación del rendimiento	67
3.7. Discusión	70
3.8. Conclusiones	73
4. Evaluación de motores OCR para el reconocimiento de placas	79
4.1. Introducción	81
4.2. Trabajo relacionado	82
4.3. Metodología	83
4.3.1. Materiales	83
4.3.2. Método	84
4.4. Conjunto de datos	85
4.4.1. Preprocesamiento de video	86
4.4.2. Registro y anotación de placas vehiculares	87
4.5. Diseño experimental	87

4.5.1. Métricas de evaluación para detección	88
4.5.2. Métrica de evaluación para OCR	89
4.6. Resultados y evaluación	90
4.6.1. Detección de objetos	90
4.6.2. Desempeño del OCR	91
4.7. Discusión	93
4.8. Conclusiones	94
5. Ajuste fino de TrOCR para el reconocimiento automático de placas vehiculares	97
5.1. Introducción	99
5.2. Trabajos relacionados	100
5.3. Conjunto de datos personalizado	101
5.3.1. Conjunto de datos de vehículos (video)	101
5.3.2. Conjunto de datos de placas (imagen)	102
5.4. Evaluación base de OCR	103
5.4.1. Resultados de detección de placas	104
5.4.2. Comparación base ANPR	105
5.5. Reentrenamiento de TrOCR	107
5.5.1. Estrategia de ajuste fino	108
5.5.2. Evaluación base de TrOCR	109
5.5.3. Resultados del TrOCR reentrenado	110
5.6. Conclusiones	112
6. Implementación y validación del sistema ANPR	117
6.1. Arquitectura del sistema	118
6.1.1. Descripción del hardware	118
6.1.2. Descripción del software	118
6.1.3. Diseño de la base de datos	119
6.1.4. Metodología experimental	121
6.2. Resultados	123
6.2.1. Protocolo de pruebas	123

6.2.2. Discusión	126
6.2.3. Limitaciones y alcance de la validación	126
6.3. Conclusiones	127
7. Conclusiones y recomendaciones	129
7.1. Conclusiones generales	129
7.1.1. Cumplimiento de los objetivos de investigación	130
7.2. Recomendaciones generales	131
Repositorio institucional	133

Índice de tablas

2.1. Captura de las imágenes de motocicletas.	44
2.2. Nomenclatura de imágenes y sus anotaciones.	46
2.3. Matriz de confusión del modelo SSDMobileNetV2FPNlite original.	50
2.4. Matriz de confusión del modelo SSDMobileNetV2FPNlite optimizado.	51
2.5. Comparación entre modelos.	51
3.1. Detalles de captura de video	61
3.2. Resultados del seguimiento de vehículos. Mejor resultado en negrita	70
3.3. Matriz de confusión	70
3.4. Métricas de rendimiento en cada experimento (<i>ground truth</i> : 92 vehículos). Mejores valores en negrita	71
4.1. Configuración de EasyOCR	88
4.2. Configuración de TrOCR	88
4.3. Configuración de Tesseract	88
4.4. Matriz de confusión para la detección de placas vehiculares	90
4.5. Métricas de desempeño de detección	91
4.6. Ejemplo de salidas OCR para un experimento individual	91
4.7. Desempeño OCR. Tasa de error de caracteres (CER)	92
5.1. Nomenclatura de nombres de archivo y anotaciones	102
5.2. Configuración de Tesseract	104
5.3. Configuración de EasyOCR	104

5.4. Configuración de TrOCR	104
5.5. Matriz de confusión para detección de placas	105
5.6. Métricas de desempeño para detección de placas	106
5.7. Ejemplo de salidas OCR para tres experimentos	106
5.8. Desempeño OCR (CER)	108
5.9. Parámetros de reentrenamiento de TrOCR	108
5.10. Ejemplo de salidas OCR para tres experimentos	110
5.11. Ejemplo de salidas de TrOCR con ajuste finamente	110
5.12. CER antes y después del ajuste fino	111
5.13. Ejemplos representativos de fallos OCR	111
6.1. Especificaciones del hardware utilizado.	118
6.2. Componentes de software del sistema.	120
6.3. Resultados del protocolo de pruebas experimentales y cálculo del CER por experimento.	123

Índice de figuras

1.1. Descripción general la identificación de placas vehiculares (Tecnoseguro, 2024).	3
1.2. Aprendizaje profundo ejecutado en dispositivos <i>edge</i> (J. Chen y Ran, 2015).	4
1.3. Taxonomía de Aprendizaje automático.	5
1.4. Pasos del <i>Transfer Learning</i>	7
1.5. Taxonomía de métodos de aceleración de inferencia en <i>EdgeComputing</i> (2015).	12
1.6. Aprendizaje profundo aplicado en la <i>Edge computing</i> (F. Wang et al., 2020).	13
1.7. Algunos casos que provocan fallos en los sistemas (VMMR, por las siglas en inglés de Vehicle Model and Make Recognition) basados en reconocimiento de matrículas (Lee et al., 2019).	14
1.8. Rendimiento comparativo de todos los métodos de clasificación aplicados al modelo VGG16 (Naseer et al., 2020).	14
1.9. Diagrama de la arquitectura del modelo híbrido propuesto por (Khan et al., 2023).	16
1.10. Diagrama de bloques del método LPD propuesto por (Tourani et al., 2020).	17
1.11. Diagrama típico del sistema ANPR analizado por (Kumawat et al., 2024).	18
1.12. Descripción general del control de acceso vehicular (Tecnoseguro, 2024).	22
1.13. Diagrama de las etapas de la investigación.	24
2.1. Ejemplo de motocicletas incluidas en el COCO dataset. Fuente: (COCO, 2024).	41
2.2. Diseño experimental de la investigación. Fuente: Elaboración propia.	43
2.3. Ejemplos de imágenes de motocicletas capturadas. Fuente: Elaboración propia.	44
2.4. Pasos del pre-procesamiento realizado a cada imagen del dataset. Fuente: Elaboración propia.	45

2.5. Ejemplo de la ejecución del modelo base. Fuente: Elaboración propia.	47
2.6. Metodología para el reentrenamiento del modelo. Fuente: Elaboración propia. . . .	48
2.7. Métricas durante el re-entrenamiento del modelo. Fuente: Elaboración propia. . . .	50
2.8. Ejemplo de resultados. Fuente: Elaboración propia.	52
3.1. Seguimiento de un solo objeto (SOT) y seguimiento de múltiples objetos (MOT) . . .	57
3.2. Metodología del proyecto	60
3.3. Entrada del Campus Chontalpa	61
3.4. Esquema del preprocesamiento de datos	62
3.5. Diseño experimental	64
3.6. Rol rectangular (izquierda), Rol poligonal (derecha)	65
3.7. Posición de línea horizontal. Mitad del frame (izquierda), cuarto del frame (derecha)	66
3.8. Posición de línea diagonal: 45° (izquierda), 67.5° (derecha)	67
3.9. Posición de línea vertical. Mitad del frame (izquierda), cuarto del frame (derecha) . .	68
3.10. Número de vehículos detectados por experimento. Fila #1: Rol rectangular (50 y 100 fotogramas), fila #2: Rol poligonal (50 y 100 fotogramas), fila #3: CtL horizontal (mitad y cuarto del frame), fila #4: CtL diagonal (45° y 67.5°), fila #5: CtL vertical (mitad y cuarto del frame)	69
3.11. Valores de Precisión, Recall y F1-score	71
3.12. Mejores configuraciones: CtL vertical (izquierda) y Rol rectangular (derecha)	72
3.13. Vehículos estacionados contados como tráfico activo en pruebas Rol	72
3.14. Vehículo tipo <i>buggy</i> detectado como motocicleta	73
3.15. Autobús de gran tamaño excluido del conteo final en todas las configuraciones . . .	74
3.16. Bicicleta cercana a la banqueta	74
4.1. Pipeline simplificado del sistema ANPR mostrando las etapas principales de proce- samiento.	85
4.2. Entrada del Campus Chontalpa	86
4.3. Ejemplo de redimensionamiento y preprocesamiento de video.	87
4.4. Flujo del diseño experimental para detección y reconocimiento de placas vehiculares.	89
4.5. Ejemplo de salida de EasyOCR para un experimento individual	92

5.1. Pasos de preprocesamiento de video.	102
5.2. Pasos de procesamiento de imágenes.	103
5.3. Ejemplos de placas vehiculares estandarizadas.	103
5.4. Flujo de trabajo del análisis comparativo.	105
5.5. Casos de falso positivo (derecha) y falso negativo (izquierda).	106
5.6. Ejemplos de salidas OCR para tres experimentos.	107
5.7. Flujo de trabajo para la evaluación del ajuste fino de TrOCR.	109
5.8. Ejemplo de salidas de TrOCR preentrenado vs. TrOCR con ajuste finamente.	115
5.9. Curva de pérdida durante el entrenamiento.	116
5.10. Estadísticas relevantes del desempeño de TrOCR: a) Histograma, b) Boxplot, c) Percentiles, d) Tasa exacta de coincidencia completa.	116
6.1. Hardware utilizado.	119
6.2. Arquitectura del sistema: conectividad entre el hardware (cámara y Raspberry Pi), el software embebido (detección y modelo TrOCR) y el servidor remoto que aloja la base de datos, enlazados mediante una red WiFi.	121
6.3. Diagrama de la metodología experimental.	122

Resumen

La presente tesis propone una plataforma inteligente para el control de acceso vehicular basada en técnicas de Aprendizaje profundo y *Edge computing*. El objetivo principal fue desarrollar e implementar una solución capaz de realizar la detección y seguimiento de vehículos junto con el reconocimiento automático de placas vehiculares en tiempo real mediante dispositivos de cómputo en el borde, es decir, de bajo costo y consumo energético.

La investigación se desarrolló en cuatro etapas. Inicialmente, se optimizó un modelo SSD MobileNetV2 FPNLite mediante aprendizaje por transferencia para mejorar la detección de motocicletas en escenarios reales. Posteriormente, se implementó un sistema de seguimiento y conteo vehicular utilizando YOLOv11 y ByteTrack, obteniendo resultados satisfactorios en precisión y estabilidad durante el monitoreo del flujo vehicular. En una tercera etapa se realizó una evaluación comparativa de motores OCR, incluyendo Tesseract, EasyOCR y TrOCR. Los resultados evidenciaron un mejor desempeño de los enfoques basados en aprendizaje profundo frente a los métodos tradicionales. A partir de ello, se efectuó el ajuste fino del modelo TrOCR utilizando un conjunto de datos propio con placas vehiculares del estado de Tabasco, mejorando la precisión del reconocimiento. Finalmente, el sistema fue implementado sobre una plataforma Raspberry Pi equipada con un módulo de cámara, validando la ejecución integrada de los procesos de captura, detección, seguimiento y reconocimiento de placas. Los resultados demostraron la viabilidad de la plataforma para aplicaciones de detección automática de placas vehiculares con dispositivos de baja latencia y procesamiento local eficiente.

Palabras clave: Aprendizaje profundo, Edge computing, YOLOv11, ByteTrack, TrOCR, ANPR, Reconocimiento de placas vehiculares, Visión computacional.

Abstract

This thesis proposes an intelligent vehicle access control platform based on Deep learning and Edge computing. The main objective was to develop and implement a solution capable of performing vehicle detection, object tracking, and automatic license plate recognition in real time using low-cost edge computing devices.

The research was conducted in four stages. First, an SSD MobileNetV2 FPNLite model was optimized through transfer learning to improve motorcycle detection in real-world scenarios. Subsequently, a vehicle tracking and counting system was implemented using YOLOv11 and ByteTrack, achieving satisfactory results in terms of accuracy and stability during traffic monitoring tasks. In a third stage, a comparative evaluation of OCR engines, including Tesseract, EasyOCR, and TrOCR, was carried out. The results showed that deep learning-based approaches outperformed traditional methods. Based on these findings, the TrOCR model was fine-tuned using a specialized dataset of vehicle license plates from the state of Tabasco, improving recognition accuracy. Finally, the complete system was deployed on a Raspberry Pi platform equipped with a camera module, validating the integrated execution of image acquisition, vehicle detection, tracking, and license plate recognition processes. Experimental results demonstrated the feasibility of the platform for automatic license plate applications with low-latency operation and efficient local processing devices.

Keywords: Deep learning, Edge Computing, YOLOv11, ByteTrack, TrOCR, ANPR, license plate recognition, computer vision.

Capítulo 1

Protocolo de Tesis

1.1. Introducción

La Inteligencia Artificial (IA) ha influido en diversos ámbitos de la sociedad así como en la vida diaria de las personas. El crecimiento de la IA ha creado varios escenarios de prueba para evaluar la manera en la que humanos se conectan a Internet a través de dispositivos que están presentes en todos los ámbitos de la sociedad, hogares inteligentes, automóviles, cámaras autónomas, en el transporte, entre otras.

Una de las tecnologías que está definiendo el paradigma de la forma en que los humanos se conectan a Internet es la *Edge computing* (también conocida como *Computación en el borde*), la cual habilita servicios de IA que escalan con volúmenes de datos y dispositivos finales en rápido crecimiento. La Computación en el borde se produce cerca de fuentes de datos, como terminales inteligentes, almacenando y procesando datos en el borde de las redes. En términos de procesamiento de datos, es más rápido, en tiempo real y seguro. También puede resolver el problema del consumo excesivo de energía en la computación en la nube, reducir costos y reducir la presión sobre el ancho de banda de la red. Actualmente hay una serie de factores que harán que este tipo de paradigmas sea aún más fácil implementarlos, uno de ellos es el coste más bajo de los dispositivos y sensores (Cao et al., 2020).

Otra tecnología relevante actualmente es el Aprendizaje profundo, la cual ha revolucionado diversos campos, desde el reconocimiento de voz hasta la conducción de vehículos autónoma basado en redes neuronales artificiales. La capacidad de las redes neuronales profundas de

aprender representaciones complejas de datos ha impulsado la necesidad de implementar modelos de aprendizaje profundo en dispositivos en el borde. El Aprendizaje profundo en el borde (*Deep Learning on the Edge*) se ejecuta en una variedad de dispositivos perimetrales, que van desde servidores perimetrales equipados con una GPU hasta teléfonos inteligentes con procesadores móviles y dispositivos *Single-Board Computer* (SBC) (J. Chen y Ran, 2015).

En ese sentido, La recopilación de datos es uno de los pasos o procesos primarios del Aprendizaje profundo y es de suma importancia para el Cómputo en el borde, pero se puede volver costoso y difícil. De esta manera surge una forma novedosa de transferir conocimiento entre modelos. El *Transfer learning* tiene como objetivo reutilizar los elementos de un modelo de Aprendizaje automático previamente entrenado en un nuevo modelo. Si ambos están diseñados para realizar tareas similares, se puede compartir conocimiento entre ellos (Hosna et al., 2022).

La idea del *Transfer learning* está impulsada por el hecho de que se pueden relacionar efectivamente habilidades para resolver problemas que requieren Aprendizaje profundo de manera más rápida y precisa. Una de las posibles aplicaciones es en el control de acceso vehicular a través del *Deep Learning on the Edge*, con el cual se podría evitar la entrada de personas no autorizadas desde un vehículo por medio del reconocimiento de su placa vehicular. Esto no solo para garantizar la seguridad, sino como una necesidad dentro de instalaciones que cuenten con controles de acceso.

Es posible que la identificación de vehículos a través de las imágenes capturadas de las placas vehiculares se lleve a cabo de forma automática sin necesidad de un humano. Para ello, los sistemas deben tomar en cuenta primero los diferentes niveles de luminosidad que se pueden presentar a diferentes horas del día, la diferente posición de los vehículos, las condiciones de clima, el deterioro de las placas, y en general todas las variables que pueden ser algún factor relevante en el uso de los sistemas de control de acceso. Las cámaras de alta resolución con visión artificial se complementan al software de control y a los sistemas electromecánicos para poder realizar un acceso vehicular seguro y personalizado, como se puede observar en la Figura 1.1.

En el Campus Chontalpa de la UJAT existe la problemática planteada con respecto el control de acceso vehicular. Aunque en el estado del arte existen modelos que resuelven esta problemática, se propone usar la *Edge computing* para aproximarnos o encontrar una solución viable para

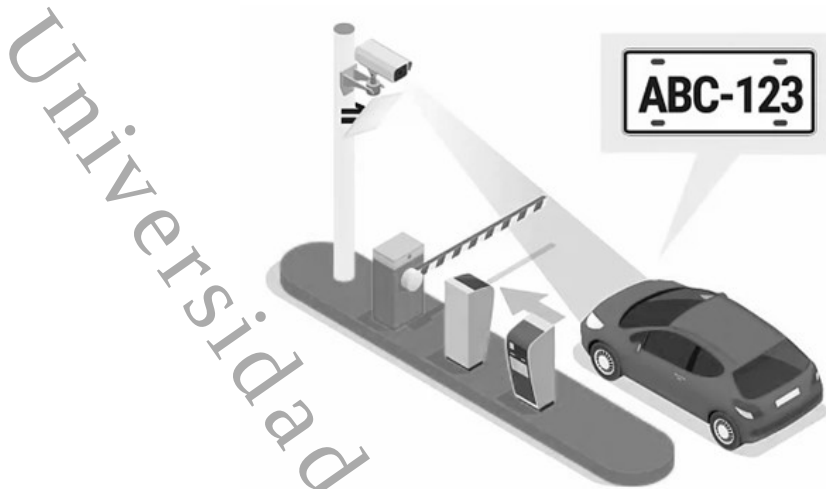


Figura 1.1. Descripción general la identificación de placas vehiculares (2024).

abordar los desafíos de latencia, escalabilidad y privacidad de datos.

1.2. Marco Conceptual

1.2.1. Cómputo en el borde

La idea básica de la *Edge Computing* es que la capacidad de computación debe implementarse cerca de la fuente de datos para el procesamiento de datos, en lugar de transmitir los datos a lugares con potencia de computación. De esta manera, se implementa la cantidad necesario de servidores en el borde de la red y las tareas en los dispositivos finales se pueden descargar a los servidores perimetrales para su procesamiento instantáneo, como se observa en la Figura 1.2.

La *Edge computing* permite servicios en tiempo real resolviendo desafíos de latencia, añadiendo proximidad a las fuentes de datos en los dispositivos finales. Para abordar los desafíos de escalabilidad, una arquitectura jerárquica de dispositivos finales, nodos de computación de borde y centros de datos en la nube pueden proporcionar recursos y escalar con la cantidad del usuario, evitando cuellos de botella en la red en una ubicación central. Para abordar los desafíos de la privacidad, permite analizar los datos cerca de la fuente, tal vez mediante un servidor local confiable, evitando así el acceso a la Internet pública y reduciendo la exposición a ataques a la privacidad y la seguridad.

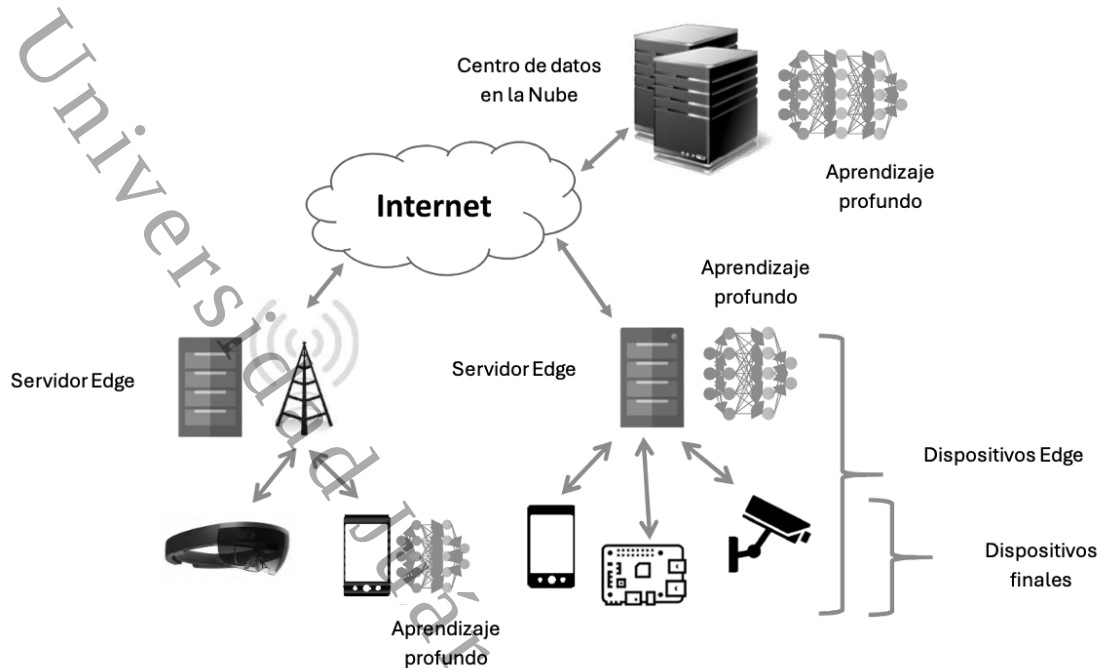


Figura 1.2. Aprendizaje profundo ejecutado en dispositivos *edge* (2015).

1.2.2. Aprendizaje automático

El *Machine learning* es la ciencia (y el arte) de hacer que las computadoras puedan aprender de los datos sin estar programadas explícitamente. De acuerdo con (Edgar y Manz, 2017), el Aprendizaje automático es un campo de investigación que utiliza algoritmos computacionales para transformar datos empíricos en modelos utilizables.

Los modelos de Aprendizaje automático se pueden clasificar según la cantidad y tipo de supervisión que reciben durante el entrenamiento:

- **Aprendizaje supervisado:** en éste los algoritmos generan modelos predictivos basados en datos de entrada y de salida.
- **Aprendizaje no supervisado:** los algoritmos funcionan de manera muy similar a los algoritmos de Aprendizaje supervisado, excepto que solo adaptan un modelo predictivo basado en los datos de entrada, independientemente de los datos de salida.
- **Aprendizaje por refuerzo:** Los algoritmos definen modelos y funciones que se centran en maximizar un cierto nivel de retribución en función de las acciones y el entorno en el que opera un agente inteligente (Géron, 2019).

Esta clasificación se describe visualmente en la Figura 1.3.

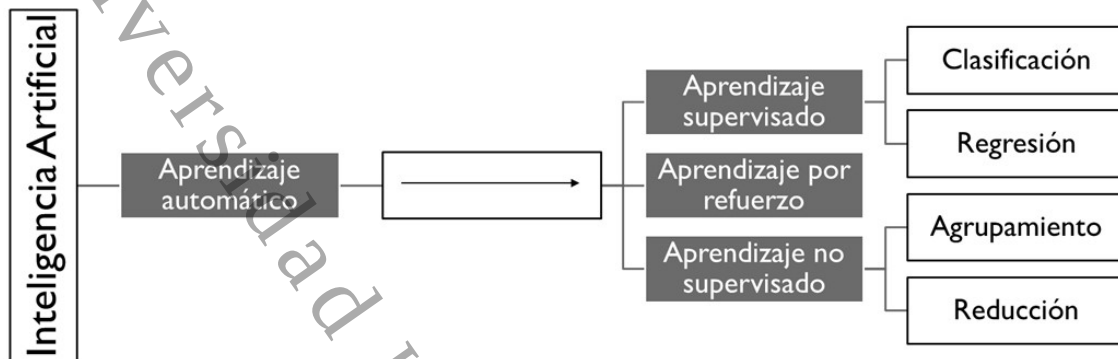


Figura 1.3. Taxonomía de Aprendizaje automático.

1.2.3. Aprendizaje profundo

El *Deep learning* es una forma de Aprendizaje automático cuya base principal es aprender características de una manera estratificada. La diferencia esencial entre el Aprendizaje profundo y otras técnicas es su capacidad de abstracción y de aprender jerarquías en diferentes niveles de representación mediante redes neuronales profundas. Las técnicas de Aprendizaje profundo buscan automatizar el proceso de abstracción, aprendiendo nuevas abstracciones a partir de las ya existentes (Berzal, 2019).

De acuerdo con (LeCun et al., 2015), las redes neuronales profundas funcionan como extractor de características y también como clasificador. el aspecto clave del Aprendizaje profundo es que las capas de características no están diseñadas por ingenieros humanos, sino son aprendidas a partir de los datos mediante un procedimiento de aprendizaje de propósito general. El Aprendizaje profundo ofrece muchas ventajas gracias a su utilidad en diversos tipos de aplicaciones en el mundo real, una de las más importantes es el poder trabajar con grandes volúmenes de datos (*Big Data*). Las áreas más abarcadas mediante el Aprendizaje profundo son: las pruebas adaptativas, clasificación de imágenes, visión computacional, procesamiento de lenguaje natural, detección de objetos, reconocimiento facial, reconocimiento de escritura, reconocimiento de voz.

El Aprendizaje automático aprovecha los datos estructurados (números, cadenas de texto, datos nominales) y clasificados para realizar predicciones, lo que significa que las características específicas se definen a partir de los datos de entrada para el modelo y se organizan de manera tabular. Esto no significa necesariamente que no utilice datos no estructurados, solo significa que si lo hace, generalmente pasan por un procesamiento previo para organizarlos en un formato estructurado. Por otro lado, el Aprendizaje profundo elimina parte del procesamiento previo de datos que normalmente conlleva el Aprendizaje automático, haciendo de manera particular que se puedan introducir y procesar datos no estructurados, como texto en lenguaje natural, imágenes y videos, lo que automatiza la extracción de características, eliminando parte de la dependencia de expertos humanos (IBM, 2024a).

1.2.4. *Transfer learning*

(Hosna et al., 2022) definen al *Transfer learning* como una técnica empleada para entrenar un modelo de Aprendizaje profundo con la reutilización de modelos preexistentes para resolver nuevos problemas. Los datos de entrenamiento preexistentes se utilizan para mejorar el desempeño en curso, por lo que no es necesario desarrollar la solución desde cero. Es diferente al enfoque tradicional de Aprendizaje automático, en el que los datos se aíslan en función de tareas específicas y cada actividad se desarrolla desde cero con conocimientos limitados para reconocerse entre sí. El *Transfer learning* esta integrado en cuatro pasos, como se puede observar en la Figura 1.4

1.2.5. *Visión computacional*

Se trata de un campo de la IA que actualmente utiliza el Aprendizaje automático y las redes neuronales para ayudar a las computadoras y los sistemas a obtener información significativa de imágenes digitales, videos y otras entradas visuales. (IBM, 2024b).

Uno de los campos más importantes y básicos para las aplicaciones de Visión computacional son las imágenes. Desarrollos recientes en Aprendizaje profundo han creado 8 modelos básicos en muchos campos de aplicación: AlexNet, VGGNet, GoogLeNet/Inception, ResNet, DenseNet, MobileNets, EfficientNet y RegNet. Reduciéndose a 4 aplicaciones principales: reconocimiento,

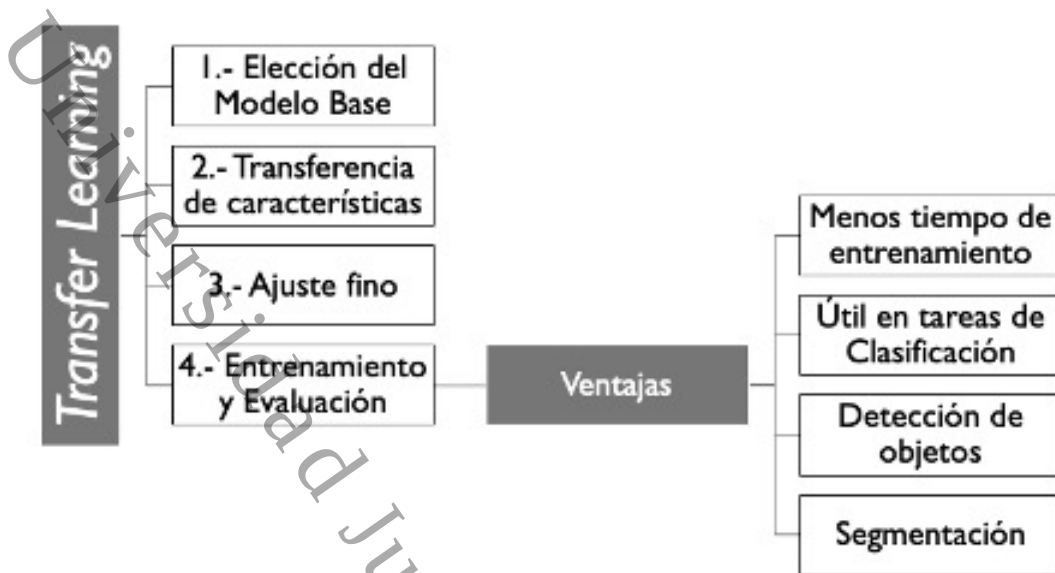


Figura 1.4. Pasos del *Transfer Learning*.

seguimiento visual, segmentación semántica y restauración de imágenes (Chai et al., 2021).

1.2.5.1. Detección de objetos

Según (MathWorks, 2024), la detección de objetos es el proceso de localizar elementos presentes en imágenes. En el caso del Aprendizaje profundo, la detección de objetos forma parte del reconocimiento de objetos, que no solo identifica el objetivo, sino que lo localiza en una imagen. Esto permite identificar y localizar varios objetos en la misma imagen. La tarea de detección de elementos en una imagen generalmente implica dibujar cuadros, delimitadores y etiquetas para cada elemento, es decir, clasificación y localización para cada objeto, con lo que es necesario aplicar redes neuronales a diferentes elementos en la imagen (Du y Bobkov, 2023a).

1.2.5.2. Seguimiento de objetos

Se trata de una aplicación de Aprendizaje profundo en la que el sistema toma un conjunto inicial de detecciones de objetos y desarrolla una identificación única para cada una de las detecciones iniciales y luego rastrea los objetos detectados a medida que se mueven por los fotogramas de un vídeo. En otras palabras, el seguimiento de objetos es la tarea de identificar automáticamente los elementos en un vídeo e interpretarlos como un conjunto de trayectorias con precisión. El seguimiento de objetos se utiliza para una variedad de casos de uso que involucran

diferentes tipos de material de archivo de entrada. Por lo tanto, los sistemas de seguimiento de vídeo pueden procesar imágenes en vivo y en tiempo real y también archivos de vídeo grabados (viso.ai, 2024).

1.2.5.3. Reconocimiento óptico de caracteres (OCR)

La tecnología de reconocimiento óptico de caracteres (OCR, por las siglas en Inglés de *Optical Character Recognition*) es un proceso eficiente que extrae y reutiliza datos de documentos escaneados, imágenes de cámaras y archivos de solo imágenes. El software OCR selecciona letras en la imagen, las convierte en palabras y luego las convierte en oraciones, lo que permite acceder y editar el contenido original. También elimina la necesidad de introducir datos manualmente.

El software OCR puede aprovechar la IA para implementar métodos más avanzados de reconocimiento inteligente de caracteres (ICR, por las siglas en Inglés de *Intelligent Character Recognition*), como identificar idiomas o estilos de escritura. El proceso de OCR se utiliza comúnmente para convertir documentos legales o históricos impresos en documentos PDF para que los usuarios puedan editar, formatear y buscar documentos como si se hubieran creado con un procesador de textos (IBM, 2024c).

1.2.5.4. Reconocimiento automático de placas vehiculares (ANPR)

El ANPR, por las siglas en Inglés de *Automatic Number Plate Recognition*, es una tecnología útil para capturar imágenes de vehículos utilizando el reconocimiento de caracteres para leer las matrículas de las placas y ayudar a obtener información adicional del vehículo, como la dirección y conteo. Es una tecnología que ha sido adoptada en múltiples aplicaciones, como el cobro de peajes, sistemas de gestión y control del tráfico, entregas, seguimiento y sistemas de gestión de aparcamiento (Kumawat et al., 2024).

El ANPR es una tarea que consta de varias etapas: captura, segmentación y preprocesamiento de imágenes, así como el reconocimiento y verificación de caracteres. Los métodos convencionales actuales se dividen principalmente en un enfoque de una etapa por ejemplo SSD (por las siglas en Inglés de *Single Shot MultiBox Detector*), YOLO (por las siglas en Inglés *You Only Look Once*) y un enfoque de dos etapas, por ejemplo la serie R-CNN (por las siglas en Inglés

Region-based Convolutional Neural Network). El enfoque de dos etapas genera en primer lugar un conjunto disperso del cuadro delimitador de la imagen. Luego hace correcciones basadas en la región del cuadro delimitador para mejorar los resultados de detección finales. Por el contrario, el enfoque de una sola etapa calcula directamente la imagen y genera resultados de detección (Chai et al., 2021).

1.2.6. **Single-Board Computer (SBC)**

El *chipset* de una computadora es un circuito integrado que se encarga de la comunicación entre la CPU, la RAM, el almacenamiento y otros periféricos. El chipset determina cuántos componentes de alta velocidad o dispositivos USB puede admitir la tarjeta madre. Los chipsets generalmente se componen de uno a cuatro chips y cuentan con controladores para periféricos de uso común, como el teclado, el mouse o el monitor.

Una computadora de placa única (SBC, por las siglas en Inglés de *Single Board Computer*) es una computadora completa construida sobre una sola placa de circuito, con microprocesador, memoria, comunicación de entrada y salidas, y otras características requeridas de una computadora funcional. Suelen ser pequeñas, livianas, confiables y con un buen manejo de la potencia eléctrica que los computadores de múltiples tarjetas (Braun et al., 1986).

De acuerdo con Intel Un SoC (*System on a Chip*), integra casi todos estos componentes (características del chipset) en un solo chip de silicio (Intel, 2024). Junto con un procesador, el SoC generalmente contiene una GPU (procesador de gráficos), memoria, controlador USB, circuitos de administración de energía y radios inalámbricas. Debido a que un SoC incluye tanto el hardware como el software, utiliza menos energía, tiene mejor desempeño, requiere menos espacio y es más confiable que los sistemas multichip.

1.2.6.1. **Raspberry Pi**

La Raspberry Pi (Raspberry Pi Foundation, 2026) es una computadora de bajo costo y con un tamaño compacto, del porte de una tarjeta de crédito, que puede ser conectada a un monitor de computadora o una TV, y usarse con un mouse y teclado estándar. Es una pequeña computadora que ejecuta un sistema operativo Linux capaz de hacer la mayoría de las tareas típicas de

una computadora de escritorio, desde navegar en Internet, reproducir videos en alta resolución, manipular documentos de ofimática, hasta reproducir juegos. Fue creada en febrero del 2012 por la Raspberry Pi Fundación, originalmente pensado para promover y enseñar las ciencias básicas de la computación en las escuelas y universidades de Reino Unido.

Los modelos de Aprendizaje profundo se han vuelto cada vez más populares para una amplia gama de aplicaciones, incluida la Visión computacional, procesamiento del lenguaje natural y reconocimiento del habla. Sin embargo, estos modelos normalmente requieren grandes cantidades de recursos computacionales, lo que hace que sea difícil ejecutarlos. Dispositivos de bajo consumo, como la Raspberry Pi, han abordado estos desafíos utilizando técnicas para reducir el tamaño de los modelos eliminando pesos y conexiones sin relevancia del modelo, lo que resulta en un modelo más pequeño y eficiente. Otro enfoque es optimizar los modelos de Aprendizaje profundo específicamente para la arquitectura Raspberry Pi, incluyendo la optimización de la arquitectura y parámetros del modelo para aprovechar las capacidades de hardware de Raspberry Pi, como su CPU y GPU. (Ameen et al., 2023) estudiaron el efecto de la poda y técnicas de optimización de la compresión en imágenes profundas a través de modelos de clasificación al ejecutar inferencias en tiempo real en una Raspberry Pi, demostrando la eficacia de estas técnicas para permitir el despliegue de modelos de Aprendizaje profundo en dispositivos de bajo consumo como la Raspberry Pi.

1.2.6.2. Cámara Raspberry Pi Camera Module 3

La Raspberry Pi Camera Module 3 (Foundation, 2026) es una cámara diseñada para trabajar junto con la Raspberry Pi. Este dispositivo permite capturar imágenes y videos digitales, por lo que es ampliamente utilizado en proyectos relacionados con visión computacional, monitoreo y automatización.

La cámara se conecta directamente a la Raspberry Pi mediante un puerto dedicado, lo que facilita su instalación y uso. Entre sus características principales se encuentran una mejor calidad de imagen y la capacidad de enfocar automáticamente, permitiendo obtener fotografías más nítidas en diferentes condiciones de uso.

Gracias a su integración con la Raspberry Pi, esta cámara se ha convertido en una herramién-

ta accesible para el desarrollo de proyectos académicos y de investigación. En aplicaciones de visión computacional, la cámara actúa como el dispositivo encargado de adquirir las imágenes que posteriormente son procesadas por algoritmos de análisis y reconocimiento, permitiendo la detección e identificación de objetos en tiempo real.

1.3. Estado del arte

En esta sección se encuentran las investigaciones más importantes que se han llevado a cabo en los últimos años, enfocadas en desarrollar métodos y aplicaciones que utilizan el Aprendizaje profundo en la *Edge computing*, con la finalidad de abordar los desafíos relacionados con el rendimiento de los sistemas, la tecnología y la gestión de las redes, evaluando y comparando las diferentes técnicas utilizadas.

1.3.1. Aprendizaje profundo aplicado en la *Edge computing*

El Aprendizaje profundo se utiliza ampliamente en diversas aplicaciones, como la visión por computadora y el procesamiento del lenguaje natural. Los dispositivos finales, como los teléfonos inteligentes y los sensores de Internet de las cosas, generan datos que pueden analizarse en tiempo real o usarse para entrenar modelos de Aprendizaje profundo. Sin embargo, la inferencia y el entrenamiento del Aprendizaje profundo requieren grandes cantidades de recursos computacionales para funcionar adecuadamente. El *Edge Computing* es una forma viable de colocar una red detallada de nodos cerca de los dispositivos finales para cumplir con los requisitos de alta complejidad computacional y baja latencia del Aprendizaje profundo en dispositivos *edge*, al tiempo que proporciona protección de datos, eficiencia y también ofrece servicios adicionales en términos de ancho de banda y escalabilidad.

(J. Chen y Ran, 2015) proporcionan una descripción general completa del estado del arte inicial del Aprendizaje profundo y la Computación en el borde. En particular, proporciona una descripción general de las aplicaciones y describe varios enfoques para ejecutar rápidamente la inferencia de Aprendizaje profundo en una combinación de dispositivos finales, servidores perimetrales y la nube. Describe cómo entrenar el Aprendizaje profundo abordando desafíos abiertos

relacionados con el rendimiento del sistema, la tecnología, la gestión de redes, la evaluación comparativa y la protección de datos, como se observa en la Figura 1.5.

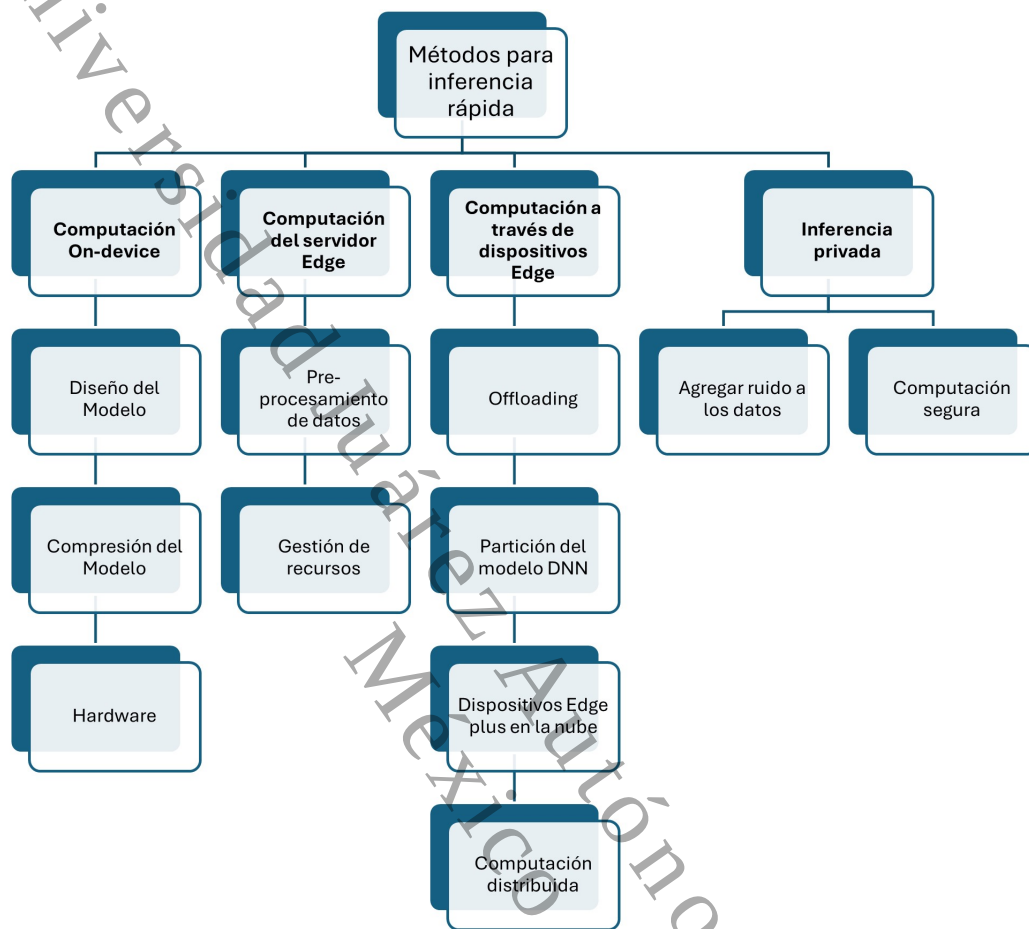


Figura 1.5. Taxonomía de métodos de aceleración de inferencia en *EdgeComputing* (2015).

La computación en la nube tradicional y la computación en los dispositivos no puede abordar suficientemente el problema de la alta latencia y la limitada capacidad de cálculo, respectivamente. La Computación en el borde impulsa el procesamiento de datos desde el núcleo de la red remota al borde de la red local, reduciendo notablemente la latencia y mejorar la eficiencia. Además, los recientes avances en el Aprendizaje profundo han facilitado la capacidad de procesamiento de datos, permitiendo un desarrollo de aplicaciones novedosas, como la videovigilancia y la conducción autónoma.

Se cree en la convergencia de la Computación en el borde y el Aprendizaje profundo para aportar nuevas posibilidades tanto a las investigaciones interdisciplinarias como a las aplicaciones industriales. Es el caso de la investigación de (F. Wang et al., 2020) donde basaron un estudio

completo donde ofrecen información sobre cómo aprovechar los avances del Aprendizaje profundo para facilitar su dominio, es decir, multimedia inteligente, transporte inteligente, ciudad inteligente e industria inteligente (Figura 1.6). También destacan los desafíos clave de la investigación y las direcciones de investigación prometedoras.

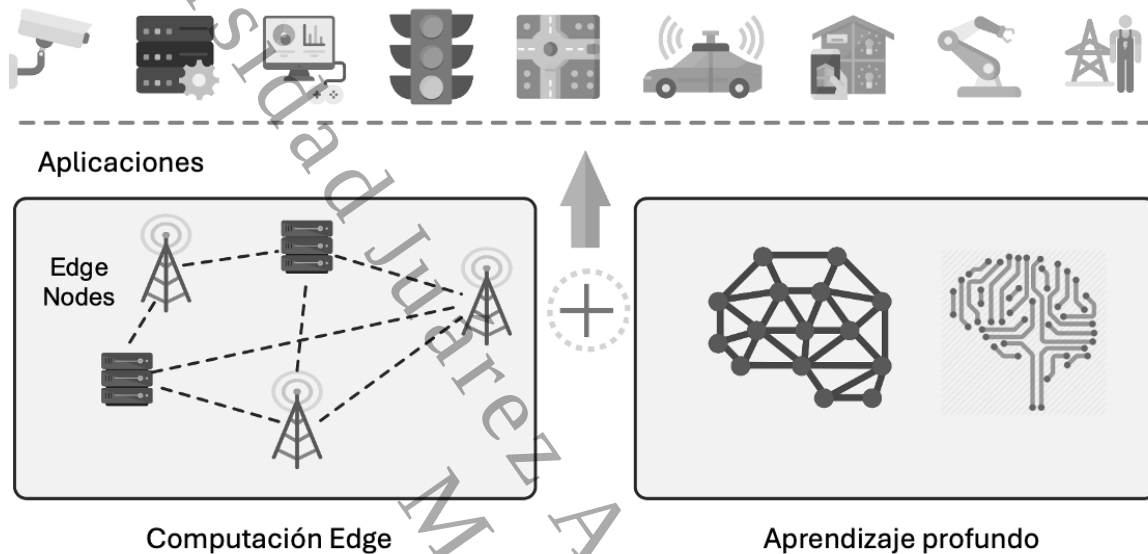


Figura 1.6. Aprendizaje profundo aplicado en la *Edge computing* (2020).

1.3.2. Aprendizaje profundo aplicado en la detección de objetos para el control de acceso vehicular

El reconocimiento de marcas y modelos de vehículos (VMMR, por las siglas en Inglés de *Vehicle Mark Model Recognition*) es importante en los sistemas basados en visión computacional ya que en algunos casos existen fallos en los sistemas VMMR basados en reconocimiento de matrículas (como se observa en la Figura 1.7). En este sentido, (Lee et al., 2019) estudiaron el problema y proponen un nuevo enfoque de Aprendizaje profundo para el reconocimiento de marca y modelo utilizando la arquitectura SqueezeNet (Iandola, 2016). Las vistas frontales de las imágenes del vehículo se extrajeron y se introdujeron en una red profunda para entrenamiento y prueba, con conexiones de derivación entre los módulos dentro de la arquitectura SqueezeNet, lo que hizo que el sistema VMMR fuera más eficiente. Los resultados experimentales en los conjuntos de datos recopilados, indicaron que el modelo propuesto logra una tasa de reconocimiento del

96.3 %. Para tareas de inferencia, el modelo implementado requiere menos de 5 MB de espacio en disco, por lo tanto, tiene viabilidad en aplicaciones en tiempo real.

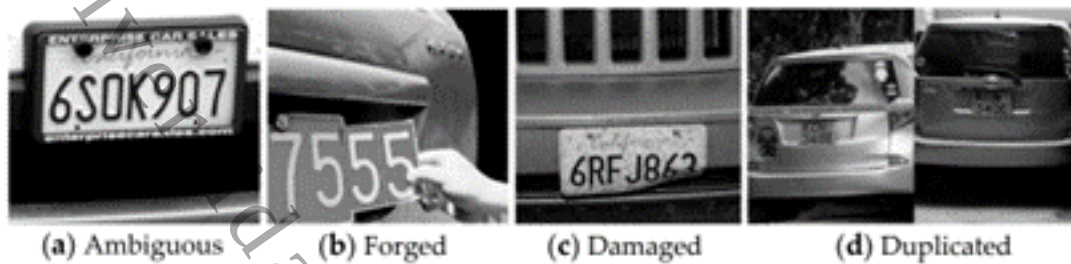


Figura 1.7. Algunos casos que provocan fallos en los sistemas (VMMR, por las siglas en inglés de Vehicle Model and Make Recognition) basados en reconocimiento de matrículas (2019).

La propuesta de (Naseer et al., 2020) consiste en un marco VMMR basado en la extracción profunda de características del modelo de red neuronal convolucional VGG16 (Simonyan y Zisserman, 2014), seguida de la reducción y clasificación de características. Un análisis comparativo con métodos de última generación muestra una mejora significativa del modelo propuesto (Figura 1.8).

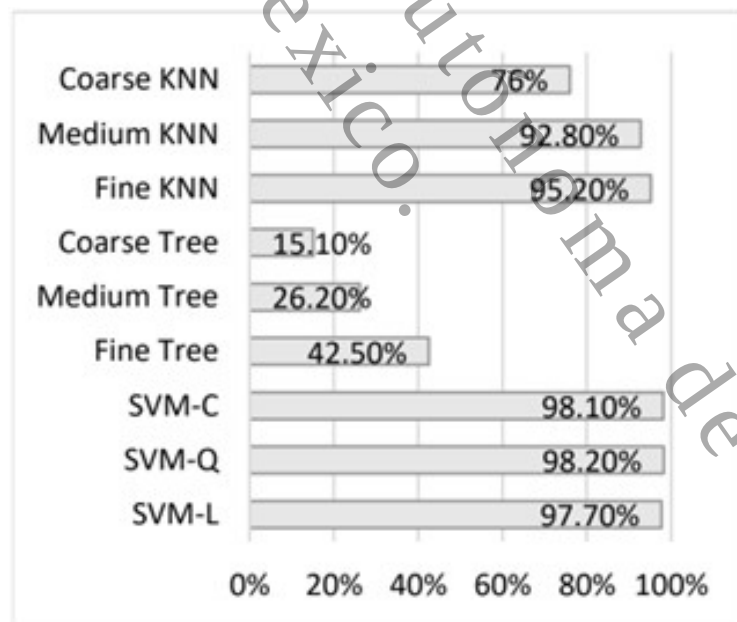


Figura 1.8. Rendimiento comparativo de todos los métodos de clasificación aplicados al modelo VGG16 (2020).

El nivel de precisión de las redes neuronales profundas en tareas de percepción visual computacional permite captar información crucial del entorno, como los vehículos autónomos y las ciudades inteligentes. Una de las posibilidades que permitiría este tipo de sistemas es el control y

seguimiento de determinados vehículos sospechosos. Teniendo en cuenta el uso de esta tecnología por parte de la policía, se facilitaría el seguimiento de determinados coches bajo investigación. Con este enfoque, (Ramajo Ballester et al., 2022) presentan una revisión del estado del arte actual, de los métodos y el desarrollo de sistemas que resuelvan dos tareas de forma eficiente: la caracterización visual y reidentificación de vehículos y la segmentación de matrículas y reconocimiento de caracteres. Esta doble identificación puede adaptarse a las condiciones ambientales, a la distancia del objetivo y a las capacidades y resolución de las cámaras. Para probar y validar este sistema, crearon un conjunto de datos personalizado para minimizar la diferencia entre el laboratorio y el entorno real.

Los avances recientes en la IA han mejorado las capacidades de detección de objetos de los vehículos autónomos, especialmente utilizando redes neuronales convolucionales (CNN, por las siglas en Inglés de *Convolutional Neural Network*). Sin embargo, lograr altos niveles de precisión y velocidad simultáneamente en entornos vehiculares sigue siendo un desafío. Por lo tanto, (Khan et al., 2023) presentan un enfoque híbrido que incorpora las características de dos modelos de detección de objetos: *You Only Look Once* (YOLO) y *Faster Region CNN* (Faster R-CNN). El modelo híbrido fue entrenado en un conjunto de datos local de 10,000 imágenes de tráfico etiquetadas y recopiladas durante escenarios de conducción, además mejorando su precisión. Los hallazgos de esta investigación tienen implicaciones prácticas para la integración de tecnologías de IA, como se observa en la Figura 1.9.

1.3.3. Seguimiento de objetos y algoritmos de *tracking*

Con el desarrollo de las tecnologías de la información, la detección y el seguimiento basados en la visión computacional tienen un nuevo auge. En los últimos años, han aparecido varios algoritmos de seguimiento, como el de (Du y Bobkov, 2023a), que plantean la comparación de algunos de los algoritmos clásicos en esta área de detección y seguimiento que han aparecido recientemente.

El seguimiento de objetos mediante Aprendizaje profundo es una dirección de investigación actual dentro de la visión inteligente. Uno de los desafíos clave en el seguimiento de objetos es predecir con precisión el movimiento del objeto, teniendo en cuenta la confiabilidad de los

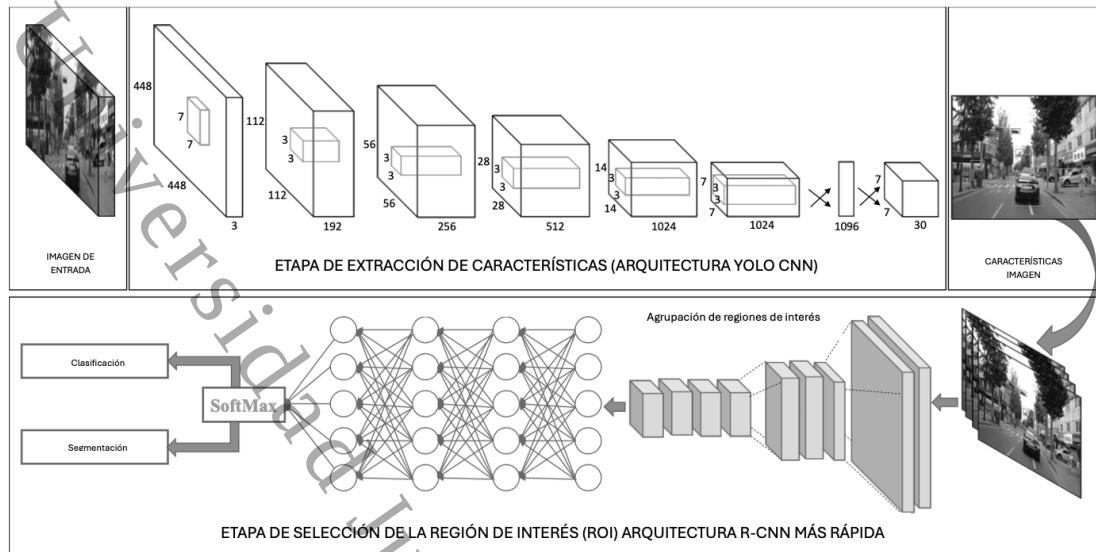


Figura 1.9. Diagrama de la arquitectura del modelo híbrido propuesto por (2023).

resultados de seguimiento. (Su et al., 2023) proponen un algoritmo de seguimiento de objetos que aprovecha tanto la dirección del movimiento como la información de series de tiempo. Una función de pérdida que guía el seguimiento para aprender la dirección del movimiento del objeto entre cuadros consecutivos, lo que resulta en una mejor precisión de localización de objetos. Los experimentos y el análisis cualitativo demuestran la efectividad del módulo propuesto y la función de pérdida.

Por otro lado, el análisis de vídeo multicanal en tiempo real es importante para el seguimiento de vehículos. (Yang, Tang, Chen et al., 2020a) presentan un método de seguimiento de vehículos para escenas de tráfico basado en un marco de seguimiento basado en detección (DBT, por las siglas en Inglés de *Detection Based Tracking*). Utilizaron un modelo (YOLO) combinando dos restricciones, incluida la información de atributos del objeto y la intersección sobre unión (IoU por las siglas en Inglés de *Intersection over Union*), para modificar el cuadro de detección de vehículos. Este enfoque mejora la precisión de la detección de vehículos.

El seguimiento de vehículos presenta varios desafíos, incluidas escenas complejas con frecuentes transiciones entre primer plano y fondo, movimientos de vehículos rápidos y no lineales, y la presencia de cuadros de detección limitada. (Gao et al., 2024a) proponen el AM-Vehicle-Track, siguiendo el paradigma probado y efectivo de seguimiento por detección *Tracking By Detection* (TBD). Los resultados experimentales muestran el rendimiento altamente competitivo del enfoque,

logrando un rendimiento de seguimiento a gran velocidad.

1.3.4. Reconocimiento automático de placas vehiculares (ANPR)

El proceso de detección de matrículas de vehículos, además de reconocer los caracteres que lo componen, siempre ha sido un reto debido a diversas condiciones. Estas condiciones incluyen el ambiente y la iluminación, el ruido en la adquisición de datos y algunos otros escenarios, como la demanda de rendimiento en tiempo real. (Tourani et al., 2020) proponen un método para la detección de placas de vehículos (LPD, por las siglas en Inglés de *License Plate Detection*) y el reconocimiento de sus caracteres como una aplicación unificada que presenta una precisión significativa y un rendimiento en tiempo real, como se observa en la Figura 1.10. El sistema fue diseñado para matrículas de vehículos iraníes, que tienen características de diferente resolución y diseño, dígitos/caracteres escasos, varios colores de fondo y diferentes tamaños de fuente. El sistema pudo realizar la tarea de LPD en un promedio de 119.73 milisegundos para datos de la vida real, lo que mostró un rendimiento en tiempo real para el sistema y aplicabilidad utilizable.

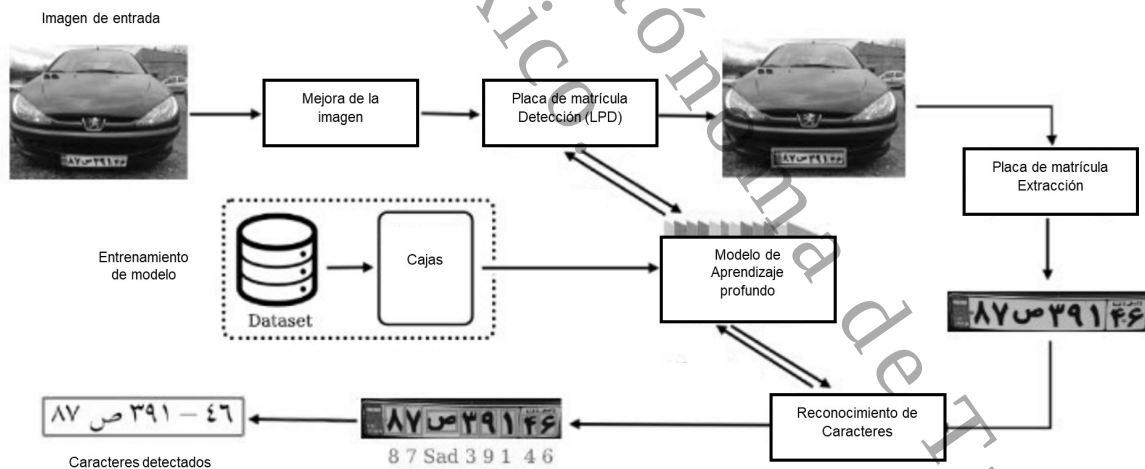


Figura 1.10. Diagrama de bloques del método LPD propuesto por (2020).

(Kumawat et al., 2024) presentan un análisis de sistemas ANPR y una comparación en cada etapa en el aspecto de tecnologías y algoritmos que involucran visión por computadora, como se observa en la Figura 1.11. Esta investigación compara algoritmos para el reconocimiento de matrículas en varias etapas. Los sistemas ANPR se utilizan generalmente en diferentes aplicacio-

nes como vigilancia de seguridad, gestión del tráfico y cobro de peajes eléctricos, aunque varios factores pueden reducir el rendimiento de los sistemas ANPR. Estos factores pueden provocar imprecisiones en el reconocimiento de placas o causar que el sistema no pueda identificar las matrículas correctamente. Algunos factores comunes que pueden socavar la ANPR incluyen la mala calidad de imagen, placas no estándar, condiciones climáticas, velocidad del vehículo, condiciones de iluminación y limitaciones basadas en hardware. Esta investigación tuvo como objetivo mejorar el estado de conocimiento de la ANPR, que incluye varios algoritmos y análisis de pasos ANPR para la detección de matrículas a través de trabajos anteriores relevantes.

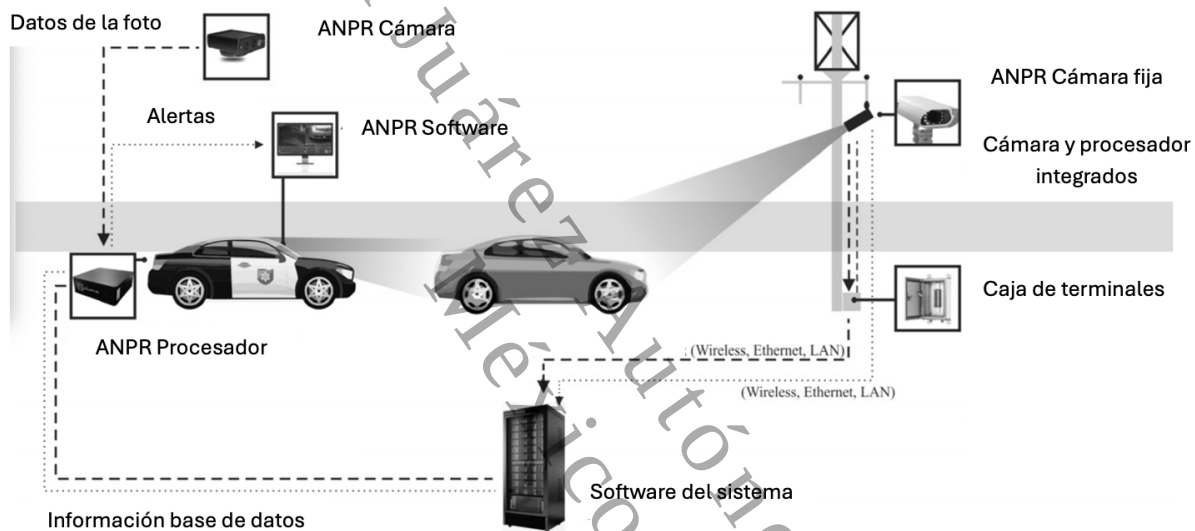


Figura 1.11. Diagrama típico del sistema ANPR analizado por (2024).

1.4. Planteamiento del problema

Los sistemas de control de acceso vehicular tienen como objetivo identificar vehículos de forma automática, precisa y rápida, sin necesidad de que intervenga el ser humano. El Aprendizaje profundo ha tenido recientemente mucho éxito en una variedad de dominios de aplicaciones, incluyendo la visión computacional, procesamiento del lenguaje natural, análisis de grandes volúmenes de datos, entre otros. Los métodos de Aprendizaje profundo han superado consistentemente a los métodos tradicionales de visión computacional para el reconocimiento y detección de objetos.

Sin embargo, la alta precisión de los modelos Aprendizaje profundo llega a expensas de altos requisitos computacionales y de memoria, tanto para las fases de entrenamiento como de inferencia. Entrenar un modelo requiere mucho espacio y cálculo debido a millones de parámetros que deben perfeccionarse de forma iterativa durante múltiples períodos de tiempo. La inferencia es computacionalmente costosa debido a la alta dimensionalidad de los datos de entrada (por ejemplo, imágenes o video) y a los millones de cálculos que deben realizarse con los datos de entrada.

Para cumplir con los requisitos computacionales del Aprendizaje profundo, un enfoque común es aprovechar la computación en la nube. Para utilizar los recursos de la nube, los datos deben trasladarse desde la ubicación de la fuente de datos en el borde de la red a una ubicación centralizada en la nube. Esta posible solución de mover los datos desde la fuente a la nube presenta varios desafíos, como:

- **Latencia:** la inferencia en tiempo real es fundamental para muchas aplicaciones. Por ejemplo, las imágenes tomadas de los vehículos deben procesarse rápidamente para detectar, analizar y comprender la consulta del usuario y devolver una respuesta. Sin embargo, el envío de datos a la nube para inferencia o entrenamiento puede generar retrasos en la propagación desde la red y no puede satisfacerse. Se necesitan requisitos de baja latencia de extremo a extremo para aplicaciones interactivas en tiempo real.
- **Escalabilidad:** el envío de datos desde las fuentes a la nube introduce problemas de escalabilidad, ya que el acceso de la red a la nube puede convertirse en un cuello de botella a

medida que aumenta el número de dispositivos conectados. Cargar todos los datos en la nube también es ineficiente en términos de recursos de la red, especialmente si el modelo de Aprendizaje profundo no necesita todos los datos de todas las fuentes.

- Privacidad: el envío de información a la nube conlleva riesgos de privacidad. Los usuarios pueden desconfiar de cargar su información confidencial en la nube (por ejemplo, imágenes de matrículas vehiculares) (J. Chen y Ran, 2015).

De esta manera, el control de acceso vehicular puede ayudar a prevenir la entrada de personas no autorizadas desde un vehículo, no solo para optimizar el uso de las instalaciones y garantizar la seguridad, sino también como una necesidad dentro de las instalaciones que cuenten con controles de acceso.

La gestión del acceso vehicular es un problema particularmente importante en los campus universitarios, con una población estudiantil y de personal significativo. Tal es el caso del Campus Chontalpa de la UJAT, que alberga tres divisiones académicas: la División Académica de Ciencias Básicas (DACB), la División Académica de Ciencias y Tecnologías de la Información (DACyTI), y la División Académica de Ingeniería y Arquitectura (DAIA).

La DACB cuenta con 1164 estudiantes de licenciatura y posgrado y 140 profesores, investigadores, y técnicos académicos (de Ciencias Básicas, 2024). Por otro lado, la DAIA tiene una mayor población de 2868 estudiantes de licenciatura y posgrado y 187 profesores, investigadores, y técnicos académicos (de Ingeniería y Arquitectura, 2024). Finalmente, la DACyTI alberga a 828 estudiantes de licenciatura y posgrado y 105 profesores, investigadores, y técnicos académicos (de Ciencias y de la Información, 2024). A estos números se les agrega el personal de intendencia, administrativos y comerciantes que laboran en el campus.

El Campus solamente cuenta con un acceso disponible vigilado por uno o dos miembros del personal de logística de la institución, lo que genera filas vehiculares durante los periodos de tiempo de alta cantidad de ingreso. Esto incrementa la problemática en el control de acceso y representa un reto para encontrar formas accesibles y económicas con respecto a hardware y software de aportar al control de acceso vehicular, con una solución práctica para problemas como la demora en la transferencia de datos, la capacidad de manejar grandes volúmenes de información y la protección de la privacidad de los usuarios.

1.5. Preguntas de investigación

¿Pueden ejecutarse modelos de Aprendizaje profundo en dispositivos SBC con un rendimiento aceptable en las condiciones particulares del campus para el control de acceso vehicular?

¿Cuál es la arquitectura de redes neuronales que, combinada con algoritmos de visión computacional, permite la detección de placas vehiculares de manera confiable, de acuerdo con las métricas de evaluación de rendimiento tradicionales en el Campus Chontalpa de la UJAT?

¿Es posible construir una plataforma para el control de acceso vehicular eficiente con dispositivos económicos disponibles en el mercado, de tal manera que sea más económica que una solución comercial?

1.6. Hipótesis

Una plataforma basada en dispositivos de Cómputo en el borde permitirá dar seguimiento y conteo a los vehículos que ingresan al Campus Chontalpa de la UJAT, mediante la lectura de la placa con al menos un 80 % de confiabilidad.

1.7. Objetivos

1.7.1. Objetivo general

Diseñar una plataforma basada en Cómputo en el borde para el control de acceso vehicular utilizando Aprendizaje profundo con componentes de hardware y software de bajo costo.

1.7.2. Objetivos específicos

- Crear un modelo para la detección de vehículos.
- Crear un modelo para la detección y reconocimiento de placas de Tabasco.
- Evaluar la plataforma en el dispositivo *edge* empleando métricas de evaluación de rendimiento utilizadas en la literatura.

1.8. Alcances y limitaciones

- La evaluación de la plataforma se llevó a cabo en la Unidad Chontalpa de la UJAT.
- El control de acceso vehicular se realizó a través de las imágenes capturadas de las placas vehiculares, como se puede ejemplifica en la Figura 6.1.
- Se utilizó la infraestructura de red y el hardware con el que cuenta el Campus Chontalpa.



Figura 1.12. Descripción general del control de acceso vehicular (2024).

1.9. Justificación

Enfoque computacional. Al utilizar modelos actuales de Aprendizaje profundo se garantiza una detección rápida y precisa de los vehículos que acceden al Campus. Se espera que el modelo funcione en las condiciones complejas de la localidad (ubicación, clima, infraestructura). Además, el desarrollo de un modelo específico para la identificación de placas vehiculares de Tabasco mediante *Transfer learning* asegura una adaptación al contexto del Campus

universitario. Se espera que este modelo pueda servir como base para el diseño de otras plataformas a la medida en otras instituciones.

Enfoque tecnológico. Aportar un sistema simple con tecnología de vanguardia pero con equipos de bajo costo en términos de hardware y software. Para el desarrollo de la plataforma se utilizó exclusivamente software libre.

Enfoque social. Procurar la seguridad y el orden entre los vehículos acceden a la institución, apegándose al marco legal de uso de datos personales de los usuarios dentro de la institución.

1.10. Método

Esta investigación tiene un enfoque cuantitativo, ya que su objetivo es la detección e identificación de vehículos a través de las imágenes capturadas de sus placas, con el uso de métricas que permitan evaluar y validar los resultados. Este estudio está compuesto por una serie de etapas basado en (Khan et al., 2023) (ver Figura 1.13).

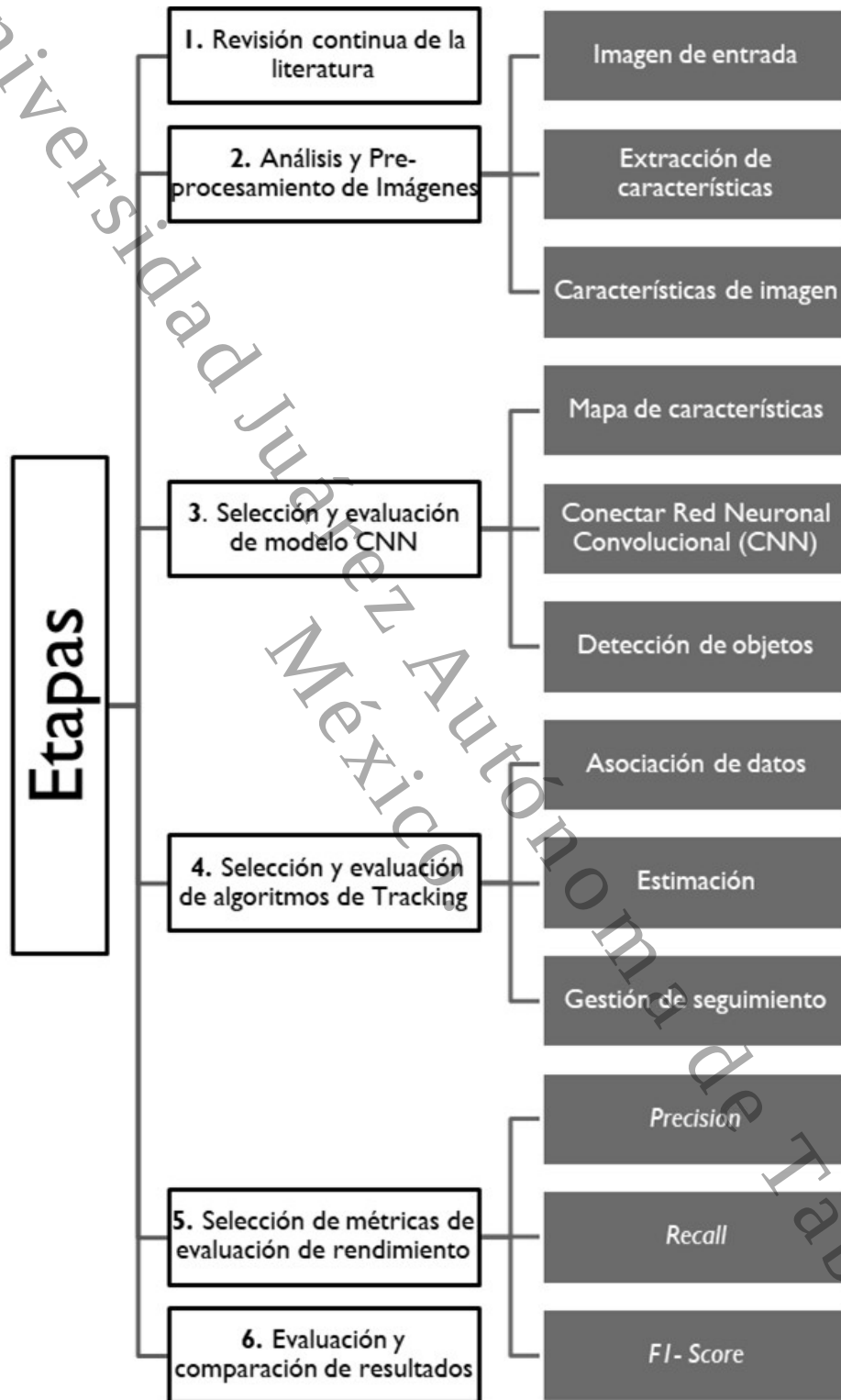


Figura 1.13. Diagrama de las etapas de la investigación.

1.10.1. Descripción de Etapas de la Investigación

1. Esta etapa es un proceso dinámico que implica la búsqueda, evaluación y análisis de la literatura relevante relacionada con el tema de estudio. Este enfoque se realiza a lo largo de todo el proceso de investigación, desde la fase inicial de formulación de las preguntas de investigación hasta la redacción de los resultados finales. La revisión continua de la literatura permite mantenerse al tanto de los últimos avances y descubrimientos en el estado del arte.
2. En esta fase el proceso comienza con la recopilación de imágenes capturadas de las placas en el control de acceso vehicular en el sitio. Luego las imágenes se preprocesan para extraer las características que se utilizarán en el modelo. Esta etapa es fundamental para preparar los datos de entrada de manera que sean adecuados para el entrenamiento del modelo para la detección de objetos.
3. Esta etapa dentro del modelo de la GNN es crucial en el desarrollo del modelo de Aprendizaje profundo para las tareas de detección y reconocimiento de imágenes, objetos y segmentación.
4. Esta sección es clave en la implementación de los algoritmos de seguimiento, ya que llevará el control del conteo de los fotogramas con mayor índice de confianza que permiten seguir el movimiento de un objeto a través de una secuencia de imágenes, lo que garantizaría la coherencia y robustez del seguimiento.
5. En este segmento se utilizan las métricas de evaluación del rendimiento del seguimiento de objetos. Cada métrica proporciona información sobre diferentes aspectos del rendimiento de los modelos.
6. En esta etapa se busca comprender el desempeño del modelo y su capacidad para reconocer placas y obtener de los resultados información para entender su rendimiento, identificar áreas de mejora y tomar decisiones sobre su implementación en aplicaciones del mundo real.

1.11. Contribución y resultados esperados

1. Un modelo de Aprendizaje profundo personalizado para la identificación de placas vehiculares de Tabasco. Se espera que el modelo sea eficiente, que cuente con un rendimiento aceptable, que pueda ejecutarse en entornos con baja capacidad de procesamiento y en condiciones climáticas adversas. Para ello, se optimizará el modelo para su ejecución en un dispositivo de bajo consumo.
2. Una plataforma basada en *Edge computing* para el control de acceso vehicular que sea económica, fácil de instalar y de configurar en el entorno del Campus. Se espera publicar el software y la arquitectura de la plataforma como software libre.

1.12. Cronograma

No.	Actividad	Sem. 3	Sem. 4	Sem. 5	Sem. 6
1	Revisión continua de literatura	✓	✓	✓	✓
2	Análisis y preprocesamiento de imágenes	✓	✓		
	2.1 Imagen de entrada	✓	✓		
	2.2 Extracción de características	✓	✓		
	2.3 Características de imagen	✓	✓		
3	Selección y evaluación de modelo CNN	✓	✓	✓	
	3.1 Mapa de características	✓	✓	✓	
	3.2 Entrenar la CNN	✓	✓	✓	
	3.3 Detección de objetos	✓	✓	✓	
4	Selección y evaluación de algoritmos de <i>Tracking</i>	✓	✓	✓	
	4.1 Asociación de datos	✓	✓	✓	
	4.2 Estimación	✓	✓	✓	
	4.3 Gestión de seguimiento	✓	✓	✓	
5	Selección de métricas de evaluación de rendimiento		✓	✓	
	5.1 <i>Precision</i>		✓	✓	
	5.2 <i>Recall</i>		✓	✓	
	5.3 <i>F1-Score</i>		✓	✓	
6	Evaluación y comparación de resultados			✓	✓
7	Redacción de artículos	✓	✓	✓	✓
8	Redacción de documento de tesis	✓	✓	✓	✓

Referencias del capítulo

- Alfarano, A., Maiano, L., Papa, L., & Amerini, I. (2024). Estimating optical flow: A comprehensive review of the state of the art [<https://doi.org/10.1016/j.cviu.2024.104160>]. *Computer Vision and Image Understanding*, 249(1), 104160. <https://doi.org/https://doi.org/10.1016/j.cviu.2024.104160>
- Ameen, S., Siriwardana, K., & Theodoridis, T. (2023). Optimizing Deep Learning Models For Raspberry Pi. *arXiv*, 1 (1), 1-6.
- Anwar, N., Khan, T., & Mollah, A. F. (2022a). Text Detection from Scene and Born Images: How Good is Tesseract? En *Recent Trends in Communication and Intelligent Systems* (pp. 115-122). https://doi.org/10.1007/978-981-19-1324-2_13
- Anwar, N., Khan, T., & Mollah, A. F. (2022b). Text Detection from Scene and Born Images: How Good is Tesseract? [https://doi.org/10.1007/978-981-19-1324-2_13]. En A. K. S. Pundir,

- N. Yadav, H. Sharma & S. Das (Eds.), *Recent Trends in Communication and Intelligent Systems* (pp. 115-122). Springer Nature Singapore. https://doi.org/10.1007/978-981-19-1324-2_13
- Awwad, S., Igried, B., Wedyan, M., & Alshira'H, M. (2021). Hybrid Features for Object Detection in RGB-D Scenes [<https://doi.org/10.11591/ijeecs.v23.i2.pp1073-1083>]. *Indonesian Journal of Electrical Engineering and Computer Science*, 23(2), 1073-1083. <https://doi.org/10.11591/ijeecs.v23.i2.pp1073-1083>
- Bakhtan, M., Abdullah, M., & Rahman, A. (2016). A review on License Plate Recognition system algorithms [<https://doi.org/10.1109/ICICTM.2016.7890782>]. *Proceedings of the International Conference on Information and Communication Technology Management (ICICTM)*, 84-89. <https://doi.org/10.1109/ICICTM.2016.7890782>
- Berzal, F. (2019). Redes neuronales & deep learning: Volumen II. *Independently published*, 1, 15-25.
- Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020a). YOLOv4: Optimal Speed and Accuracy of Object Detection [<https://arxiv.org/abs/2004.10934>]. *arXiv preprint arXiv:2004.10934*.
- Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020b). YOLOv4: Optimal Speed and Accuracy of Object Detection [<https://arxiv.org/abs/2004.10934>]. *arXiv preprint arXiv:2004.10934*.
- Bolaños-Fernández, C., & Bacca-Cortes, E. B. (2024). Mobile Application for Recognizing Colombian Currency with Audio Feedback for Visually Impaired People [<https://doi.org/10.14483/23448393.21408>]. *Ingeniería*, 29(2), e21408. <https://doi.org/10.14483/23448393.21408>
- Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.
- Bradski, G., & Kaehler, A. (2008). *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media.
- Braun, P. A., Moss, R. H., & Stigal, P. D. (1986). The single board computer: A practical alternative to a custom designed microsystem. *Computers & Electrical Engineering*, 12 (3), 109-118.
- Cao, K., Liu, Y., Meng, G., & Sun, Q. (2020). An Overview on Edge Computing Research. *IEEE Access*, 13(8), 85714-85728.
- Chai, J., Zeng, H., Li, A., & Ngai, E. W. (2021). Deep learning in computer vision: A critical review of emerging techniques and application scenarios. *Machine Learning with Applications*, 6 (), 1-13.

- Chen, J., & Ran, X. (2015). Deep Learning With Edge Computing: A Review. *Proceedings of the IEEE*, 107(8), 1655-1674.
- Chen, S.-H., Wang, C.-W., Tai, I.-H., Ken-Pen Weng, Y.-H. C., & Hsieh, K.-S. (2021). Modified YOLOv4-DenseNet Algorithm for Detection of Ventricular Septal Defects in Ultrasound Images [<https://doi.org/10.9781/ijimai.2021.06.001>]. *International Journal of Interactive Multimedia and Artificial Intelligence*, 6(7), 1-8. <https://doi.org/10.9781/ijimai.2021.06.001>
- Chen, Y., Liu, X., Li, C., Zhu, J., Wu, M., & Su, X. (2024). UAV Swarm Centroid Tracking for Edge Computing Applications Using GRU-Assisted Multi-Model Filtering [<https://doi.org/10.3390/electronics13061054>]. *Electronics*, 13(6), 1-15. <https://doi.org/2079-9292>
- Chiu, S.-Y., Chiu, C.-C., & Xu, S. (2018). A Background Subtraction Algorithm in Complex Environments Based on Category Entropy Analysis [<https://doi.org/10.3390/app8060885>]. *Applied Sciences*, 8(6), 1-15. <https://doi.org/2076-3417>
- COCO. (2024). *COCO Dataset* [Accedido: 30 de septiembre de 2024]. <https://cocodataset.org/#home>
- D K, T., James, J., Gopinath, D. P., & Ashraf K, M. (2024). Advocating Character Error Rate for Multilingual ASR Evaluation [<https://doi.org/10.48550/arXiv.2410.07400>]. *arXiv preprint arXiv:2410.07400*. <https://doi.org/10.48550/arXiv.2410.07400>
- de Ciencias, D. A., & de la Información, T. (2024). *Informe de actividades 2022-2023*. Consultado el 4 de junio de 2024, desde <https://archivos.ujat.mx/2023/div-dacyti/InformeDACYTlv-2022-2023.pdf>
- de Ciencias Básicas, D. A. (2024). *Informe de actividades 2022-2023*. Consultado el 4 de junio de 2024, desde https://archivos.ujat.mx/2023/div-dacb/informe/INFORME_CIENCIAS%20BASICAS.web.pdf
- de Ingeniería, D. A., & Arquitectura. (2024). *Informe de actividades 2022-2023*. Consultado el 4 de junio de 2024, desde https://archivos.ujat.mx/2023/div-daia/INFORME%202022-2023_presentacion.pdf
- Dhyani, S., & Kumar, V. (2023). Real-Time License Plate Detection and Recognition System Using YOLOv7x and EasyOCR. *2023 Global Conference on Information Technologies and Communications (GCITC)*, 1-5. <https://doi.org/10.1109/GCITC60406.2023.10425814>

- Drobac, S., & Lindén, K. (2020). Optical character recognition with neural networks and post-correction with finite state methods [<https://doi.org/10.1007/s10032-020-00359-9>]. *International Journal on Document Analysis and Recognition (IJДАР)*, 23, 279-295. <https://doi.org/10.1007/s10032-020-00359-9>
- Du, K., & Bobkov, A. (2023a). An Overview of Object Detection and Tracking Algorithms. *Engineering Proceedings*, 33(22), 1-6.
- Du, K., & Bobkov, A. (2023b). An Overview of Object Detection and Tracking Algorithms [<https://doi.org/10.3390/engproc2023033022>]. *Engineering Proceedings*, 33(22), 1-6. <https://doi.org/10.3390/engproc2023033022>
- Edgar, T. W., & Manz, D. O. (2017). Machine Learning. *Syngress*, 6, 153-173.
- Foundation, R. P. (2026). *Raspberry Pi Camera Module 3*. Consultado el 25 de junio de 2026, desde <https://www.raspberrypi.com/products/camera-module-3/>
- Galvez, R. L., Bandala, A. A., Dadios, E. P., Vicerra, R. R. P., & Maningo, J. M. Z. (2018). Object Detection Using Convolutional Neural Networks. *Proceedings of TENCON 2018 - 2018 IEEE Region 10 Conference*. <https://doi.org/10.1109/TENCON.2018.8650517>
- Gao, J., Han, G., Zhu, H., & Liao, L. (2024a). Multiple Moving Vehicles Tracking Algorithm with Attention Mechanism and Motion Model. *Electronics*, 1(13), 1-17.
- Gao, J., Han, G., Zhu, H., & Liao, L. (2024b). Multiple Moving Vehicles Tracking Algorithm with Attention Mechanism and Motion Model [<https://doi.org/10.3390/electronics13010242>]. *Electronics* 2024, 13(242), 1-12. <https://doi.org/10.3390/electronics13010242>
- Géron, A. (2019, junio). *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow* (2nd). O'Reilly Media, Inc.
- Google. (2024). *Google Colaboratory* [Accedido: 30 de diciembre de 2024]. <https://cloud.google.com/colab/docs>
- Google. (2025). *Colaboratory Documentation* [<https://cloud.google.com/colab/docs?hl=es-419>].
- Goswami, S. (2020). False detection (positives and negatives) in object detection [<https://doi.org/10.48550/arXiv.2008.06986>]. *arXiv preprint arXiv:2008.06986*, 1(1), 1-15. <https://doi.org/10.48550>
- Guntuboina, C., Aditya Porwal abd, P. J., & Shingrakhia, H. (2021). Deep Learning Based Automated Sports Video Summarization Using Yolo [<https://doi.org/10.5565/rev/elcvia.1286>].

- Electronic Letters on Computer Vision and Image Analysis*, 20(1), 99-116. <https://doi.org/10.5565/rev/elcvia.1286>
- Hamood Bakhtan, M. A., Abdullah, M., & Rahman, A. A. (2016). A Review on License Plate Recognition System Algorithms. *Proceedings of the International Conference on Information and Communication Technology Management (ICICTM)*, 84-89. <https://doi.org/10.1109/ICICTM.2016.7890782>
- Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., ... Oliphant, T. E. (2020). Array programming with NumPy [<https://doi.org/10.1038/s41586-020-2649-2>]. *Nature*, 585(7825), 357-362. <https://doi.org/10.1038/s41586-020-2649-2>
- Hosna, A., Merry, E., Gyalmo, J., & Z., A. (2022). Transfer learning: a friendly introduction. *Journal of Big Data*, 102(9), 2196-1115.
- Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). Mobilenets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv preprint arXiv:1704.04861*.
- Hu, X., Zhao, J., Hui, Y., Li, S., & You, S. (2023). SiamHSFT: A Siamese Network-Based Tracker with Hierarchical Sparse Fusion and Transformer for UAV Tracking [<https://doi.org/10.3390/s23218666>]. *Sensors*, 23(21), 1-15. <https://doi.org/10.3390/s23218666>
- Iandola, F. N. (2016). *SqueezeNet: AlexNet-level accuracy with 50x fewer parameters* [Repositorio de GitHub. Consultado el 25 de junio de 2026]. <https://github.com/forresti/SqueezeNet>
- IBM. (2024a). *¿Qué es el deep learning?* Consultado el 16 de abril de 2024, desde <https://www.ibm.com/mx-es/topics/deep-learning/>
- IBM. (2024b). *What is computer vision?* Consultado el 11 de abril de 2024, desde <https://www.ibm.com/topics/computer-vision/>
- IBM. (2024c). *What Is Optical Character Recognition (OCR)?* Consultado el 11 de abril de 2024, desde <https://www.ibm.com/blog/optical-character-recognition/>
- Intel. (2024). *Referencia de chipset y sistema en un chip (SoC) para Intel® NUC*. Consultado el 1 de abril de 2024, desde <https://www.intel.la/content/www/xl/es/support/articles/>

- 000056236/intel-nuc.html#:~:text=Un%20SoC%2C%20o%20System%2Don,de%20energ%C3%ADa%20y%20radios%20inal%C3%A1mblicas./
- Jaied AI. (2020). *EasyOCR*. Consultado el 2 de octubre de 2025, desde <https://github.com/JaiedAI/EasyOCR>
- Jocher, G., & Qiu, J. (2024a). *Ultralytics YOLO11*. <https://github.com/ultralytics/ultralytics>
- Jocher, G., & Qiu, J. (2024b). *Ultralytics YOLO11* [<https://github.com/ultralytics/ultralytics>] (11.0.0).
- Khan, S. A., Lee, H. J., & Lim. (2023). Enhancing Object Detection in Self-Driving Cars Using a Hybrid Approach. *Electronics*, 12(2768), 1-12.
- Kumawat, K., Jain, A., & Tiwari, N. (2024). Relevance of Automatic Number Plate Recognition Systems in Vehicle Theft Detection. *Engineering Proceedings*, 59(185), 1-11.
- Laroca, R., Zanlorensi, L. A., Gonçalves, G. R., Todt, E., Schwartz, W. R., & Menotti, D. (2021). An Efficient and Layout-Independent Automatic License Plate Recognition System Based on the YOLO Detector. *IET Intelligent Transportation Systems*, 15(4), 483-503. <https://doi.org/10.1049/itr2.12030>
- Laroca, R., Zanlorensi, L. A., Gonçalves, G. R., Todt, E., Schwartz, W. R., & Menotti, D. (2019). An Efficient and Layout-Independent Automatic License Plate Recognition System Based on the YOLO Detector [<https://arxiv.org/abs/1909.01754>]. *arXiv preprint arXiv:1909.01754*.
- LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, 521(7553), 436-444.
- Lee, H. J., Ullah, I., Wan, W., Gao, Y., & Fang, Z. (2019). Real-Time Vehicle Make and Model Recognition with the Residual SqueezeNet Architecture. *Sensors*, 19(5), 982.
- Li, M., Lv, T., Chen, H., Cui, L., Lu, Y., Florencio, D., Zhang, C., Li, Z., & Wei, F. (2023). TrOCR: Transformer-Based Optical Character Recognition with Pre-Trained Models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(11), 13094-13102. <https://doi.org/10.1609/aaai.v37i11.26538>
- Li, M., Zhang, Y., Li, L., Zhou, L., Yang, N., Zhou, M., & Wei, F. (2021). TrOCR: Transformer-based Optical Character Recognition with Pre-trained Models [<https://arxiv.org/abs/2109.10282>]. *arXiv preprint arXiv:2109.10282*.
- Lin, T. Y., et al. (2014). Microsoft COCO: Common Objects in Context. *Computer Vision – ECCV 2014*, 740-755.

- Lin, T. Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal Loss for Dense Object Detection. *arXiv preprint arXiv:1708.02002*.
- Liu, L., Ouyang, W., Wang, X., Fieguth, P., Chen, J., Liu, X., & Pietikäinen, M. (2020). Deep Learning for Generic Object Detection: A Survey. *International Journal of Computer Vision*, 128(2), 261-318. <https://doi.org/10.1007/s11263-019-01247-4>
- Liu, L., Sun, Y., & Ge, X. (2025). A Hybrid Multi-Person Fall Detection Scheme Based on Optimized YOLO and ST-GCN [<http://dx.doi.org/10.9781/ijimai.2024.09.003>]. *International Journal of Interactive Multimedia and Artificial Intelligence*, 9(2), 26-38. <https://doi.org/10.9781/ijimai.2024.09.003>
- Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. *Computer Vision – ECCV 2016*, 21-37.
- López Flórez, S., González-Briones, A., Hernández, G., Ramos, C., & de la Prieta, F. (2023). Automatic Cell Counting With YOLOv5: A Fluorescence Microscopy Approach [<https://doi.org/10.9781/ijimai.2023.08.001>]. *International Journal of Interactive Multimedia and Artificial Intelligence*, 8(3), 1-8. <https://doi.org/10.9781/ijimai.2023.08.001>
- MathWorks. (2024). *Procesamiento de imágenes y Visión artificial*. Consultado el 11 de abril de 2024, desde <https://la.mathworks.com/solutions/image-video-processing.html/>
- Matplotlib. (2024). *Matplotlib Quick Start* [Accedido: 30 de septiembre de 2024]. https://matplotlib.org/stable/users/explain/quick_start.html
- Meinhardt, T., Kirillov, A., Leal-Taixe, L., & Feichtenhofer, C. (2022). Trackformer: Multi-object tracking with transformers [<https://doi.org/10.48550/arXiv.2101.02702>]. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 8844-8854. <https://doi.org/10.48550>
- Naseer, S., Shah, S. M. A., Aziz, S., Khan, M. U., & Iqtidar, K. (2020). Vehicle Make and Model Recognition using Deep Transfer Learning and Support Vector Machines. *2020 IEEE 23rd International Multitopic Conference (INMIC)*, 1(1), 1-6.
- NumPy. (2024). *NumPy Documentation* [Accedido: 30 de septiembre de 2024]. <https://numpy.org/doc/stable/>

- Padilla, R., Passos, W. L., Dias, T. L. B., Netto, S. L., & da Silva, E. A. B. (2021). A Comparative Analysis of Object Detection Metrics with a Companion Open-Source Toolkit. *Electronics*, 10(3), 279. <https://doi.org/10.3390/electronics10030279>
- Pereira, R., Carvalho, G., Garrote, L., & Nunes, U. (2022). Sort and Deep-SORT Based Multi-Object Tracking for Mobile Robotics: Evaluation with New Data Association Metrics [<https://doi.org/10.3390/app12031319>]. *Applied Sciences*, 12(3), 1-15. <https://doi.org/2076-3417>
- Peres Barbosa, J., Dalla Rosa, F., & Rieder, R. (2024). Pothole Detection Web App: an approach for detecting potholes in asphalt pavements using YOLO [<https://doi.org/10.5335/rbca.v16i3.15851>]. *Revista Brasileira de Computação Aplicada*, 16(3), 25-36. <https://doi.org/10.5335/rbca.v16i3.15851>
- Pérez-Aguilar, D., Pérez-Aguilar, J., Pérez-Aguilar, A., Risco-Ramos, R., & Malpica- Rodríguez, M. (2024). Electric substation inspection: YOLOv5 in hotspot detection through thermal imaging [<https://doi.org/10.17163/ings.n31.2024.04>]. *Ingenius, Revista de Ciencia y Tecnología*, 1(31), 43-54. <https://doi.org/10.17163/ings.n31.2024.04>
- Ramajo Ballester, Á., González Cepeda, J., Armingol Moreno, J., & Escalera Hueso, A. d. I. (2022). Reidentificación de vehículos mediante técnicas de deep learning. *XLIII Jornadas de Automática: libro de actas*, 1(1), 1031-1039.
- Raspberry Pi Foundation. (2026). *Raspberry Pi 4 model B*. Consultado el 25 de junio de 2026, desde <https://www.raspberrypi.com/products/raspberry-pi-4-model-b/>
- Ren, Y., Li, Y., & Gao, X. (2024). An MRS-YOLO Model for High-Precision Waste Detection and Classification [<https://doi.org/10.3390/s24134339>]. *Sensors*, 24(13).
- Reyes Zambrano, J., & Moreira Santos, M. (2025). Computer Vision in the field of productive control and health in Farm Animals. *Revista de Tecnologías de la Informática y las Comunicaciones*, 9(1), 16-29. <https://doi.org/10.33936/isrtic.v9.i1.7156>
- Roboflow. (2024). *Supervision* [<https://roboflow.github.io/supervision/>].
- Rondón, C. N., Castro Casadiego, S., & Medina Delgado, B. (2021). Background subtraction and yolo algorithm: two methods for the detection of people in uncontrolled environments. *Revista Ingeniería Solidaria*, 17(1), 1-25. <https://doi.org/2357-6014.2021.01.06>
- S. M. Silva & C. R. Jung. (2018). License Plate Detection and Recognition in Unconstrained Scenarios. *Proceedings of the European Conference on Computer Vision (ECCV)*, 580-596.

- Satya, B., Manongga, D., Hendry & Aminuddin, A. (2025). Optimized YOLOv8 for Automatic License Plate Recognition on Resource Constrained Devices. *Engineering, Technology & Applied Science Research*, 15(2), 21976-21981. <https://doi.org/10.48084/etasr.9983>
- Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv preprint arXiv:1409.1556*. <https://arxiv.org/abs/1409.1556>
- Sonnara, F., Chihoui, H., & Filali, F. (2025). Efficient Real-Time License Plate Recognition Using Deep Learning on Edge Devices. *Journal of Real-Time Image Processing*, 22, 159. <https://doi.org/10.1007/s11554-025-01738-3>
- Sporici, D., Cuşnir, E., & Boiană, C.-A. (2020). Improving the Accuracy of Tesseract 4.0 OCR Engine Using Convolution-Based Preprocessing [<https://www.mdpi.com/2073-8994/12/5/715>]. *Symmetry*, 12(5), 715. <https://doi.org/10.3390/sym12050715>
- Spruck, A., Hawesch, M., Maier, A., Riess, C., Seiler, J., & Kaup, A. (2022). 3D Rendering Framework for Data Augmentation in Optical Character Recognition [<https://arxiv.org/abs/2209.14970>]. *arXiv preprint arXiv:2209.14970*.
- Ströbel, P. B., Clematide, S., Volk, M., & Hodel, T. (2022). Transformer-Based HTR for Historical Documents. *arXiv preprint arXiv:2203.11008*. <https://arxiv.org/abs/2203.11008>
- Su, J., Wu, C., & Yang, S. (2023). Object-Tracking Algorithm Combining Motion Direction and Time Series. *Applied Sciences*, 13(8), 1-14.
- Tavares, R. A. (2024). Comparison of Image Preprocessing Techniques for Vehicle License Plate Recognition Using OCR: Performance and Accuracy Evaluation. *arXiv preprint arXiv:2410.13622*. <https://doi.org/10.48550/arXiv.2410.13622>
- Tavares, R., & Silva, L. (2024). Comparison of Image Preprocessing Techniques for Vehicle License Plate Recognition Using OCR: Performance and Accuracy Evaluation [<https://arxiv.org/abs/2410.13622>]. *arXiv preprint arXiv:2410.13622*.
- Tecnoseguro. (2024). *Qué es LPR, cómo funciona y sus aplicaciones*. Consultado el 20 de marzo de 2024, desde <https://www.tecnoseguro.com/faqs/que-es-lpr-como-funciona-y-sus-aplicaciones/>
- TensorFlow. (2024). *tf.keras.models.load_model* [Accedido: 30 de diciembre de 2024]. https://www.tensorflow.org/api_docs/python/tf/keras/models/load_model

- Tourani, A., Shahbahrami, A., Soroori, S., Khazaee, S., & Suen, C. Y. (2020). A Robust Deep Learning Approach for Automatic Iranian Vehicle License Plate Detection and Recognition for Surveillance Systems. *IEEE Access*, 8(1), 201317-201330.
- viso.ai. (2024). *Object Tracking in Computer Vision (2024 Guide)* Read more at: <https://viso.ai/deep-learning/object-tracking/>. Consultado el 11 de abril de 2024, desde <https://viso.ai/deep-learning/object-tracking/>
- Wang, C. Y., Bochkovskiy, A., & Liao, H. Y. M. (2022). YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors. *arXiv preprint arXiv:2207.02696*. <https://doi.org/10.48550/arXiv.2207.02696>
- Wang, C.-Y., Bochkovskiy, A., & Liao, H.-Y. M. (2022). YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors [<https://arxiv.org/abs/2207.02696>]. *arXiv preprint arXiv:2207.02696*.
- Wang, F., Zhang, M., Wang, X., Ma, X., & Liu, (2020). Deep Learning for Edge Computing Applications: A State-of-the-Art Survey. *IEEE Access*, 1(8), 58322-58336.
- Wang, Y., & Mariano, V. (2024). A Multi Object Tracking Framework Based on YOLOv8s and ByteTrack Algorithm [<https://doi.org/10.1109/ACCESS.2024.3450370>]. *IEEE Access*, 12, 120711-120719. <https://doi.org/10.1109/ACCESS.2024.3450370>
- Yang, B., Tang, M., Chen, S., et al. (2020a). A vehicle tracking algorithm combining detector and tracker. *J Image Video Proc.*, 2020(17), 1-20.
- Yang, B., Tang, M., Chen, S., Wang, G., Tan, Y., & Li, B. (2020b). A vehicle tracking algorithm combining detector and tracker [<https://doi.org/10.1186/s13640-020-00505-7>]. *J Image Video Proc.* 2020, 1(17), 1-10. <https://doi.org/10.1186/s13640-020-00505-7>
- Younis, A., Shixin, L., Shelembi, J. N., & Hai, Z. (2020). Real-Time Object Detection Using Pre-Trained Deep Learning Models MobileNet-SSD. *Proceedings of the Conference*, 1-6. <https://doi.org/10.1145/3379247.3379264>
- Zhang, Y., Sun, P., Jiang, Y., Yu, D., Weng, F., Yuan, Z., Luo, P., Liu, W., & Wang, X. (2022). ByteTrack: Multi-Object Tracking by Associating Every Detection Box [https://doi.org/10.1007/978-3-031-19830-4_1]. *European Conference on Computer Vision*, 1(1), 1-21. <https://doi.org/10.1007>

- Zherzdev, S., & Gruzdev, A. (2018). LPRNet: License Plate Recognition via Deep Neural Networks. *arXiv preprint arXiv:1806.10447*. <https://doi.org/10.48550/arXiv.1806.10447>
- Zherzdev, S., & Gruzdev, A. (2019). Deep Learning for Vehicle License Plate Recognition: A Survey [<https://doi.org/10.1109/ICCS162669.2024.10799401>]. *Pattern Recognition and Image Analysis*, 29(3), 606-617. <https://doi.org/10.1134/S1054661819030124>
- Zulko, Z. (2015). *MoviePy - Video editing with Python* [<https://github.com/Zulko/moviepy>].

Capítulo 2

Modelo personalizado para la detección de vehículos empleando el modelo MobileNetSSDv2FPNlite

Portada interior

Nombre del artículo: Mejora en la detección de motocicletas en un entorno real usando el modelo MobileNetSSDv2FPNlite

Autores: Manuel Antonio García-Herrera, José Hernández-Torruco, Betania Hernández-Ocaña, Oscar Chávez-Bosquez

Institución de adscripción: División Académica de Ciencias y Tecnologías de la Información, Universidad Juárez Autónoma de Tabasco

Resumen

En este trabajo se utiliza el modelo de Aprendizaje Profundo MobileNetSSDv2FPNlite en un conjunto de 320 imágenes de motocicletas preprocesadas y etiquetadas, las cuales fueron capturadas en el punto de acceso de una universidad con distintas condiciones climáticas y horarios (dataset UJAT). El modelo original mostró dificultades para distinguir motocicletas, confundiéndolas con bicicletas o detectando únicamente a los conductores, obteniendo un rendimiento del 46.25%. Por ello, se ajustaron los hiperparámetros del modelo original y se utilizó la técnica de Transfer Learning para mejorar el rendimiento del modelo en la detección de motocicletas. Después de reentrenar el modelo, se alcanzó un 100 % en la detección utilizando las métricas Accuracy, Precision y Recall. Este enfoque demuestra que el modelo optimizado presenta una mejora significativa respecto al modelo original en todas las métricas clave, lo que lo convierte en una solución efectiva para aplicaciones de detección de objetos en un entorno real.

Palabras clave: aprendizaje por transferencia, conjunto de datos, detección de objetos, redes neuronales convolucionales, visión computacional.

Artículo publicado en la revista UIS Ingenierías: <https://doi.org/10.18273/revuin.v25n2-2026004>

2.1. Introducción

La detección de objetos usando redes neuronales convolucionales (CNN, por las siglas en Inglés de Convolutional Neural Network) es actualmente un componente fundamental en los sistemas de visión por computadora. Con el avance de las redes neuronales profundas, se han desarrollado modelos que mejoran significativamente la precisión y la velocidad de detección, lo que permite a las máquinas interpretar su entorno de manera más efectiva (Galvez et al., 2018).

Esto cobra relevancia con el surgimiento de modelos de Aprendizaje profundo pre-entrenados para la detección de objetos, los cuales cuentan con licencia libre y son lo suficientemente robustos para ser utilizados en un entorno de producción. MobileNet es uno de estos modelos que ha sido diseñado con el objetivo de ser ligero y eficiente, lo que lo convierte en una opción adecuada para aplicaciones móviles y embebidas que requieren de soluciones de detección de objetos sin comprometer el rendimiento. En particular, la combinación de la arquitectura del modelo MobileNet (Howard et al., 2017) con el modelo de detección de objetos SSD (por las siglas en Inglés de Single Shot Multi-Box Detector) (W. Liu et al., 2016) es crucial debido a su capacidad para optimizar el rendimiento en dispositivos con recursos limitados.

La integración del SSD en este modelo permite realizar la detección de objetos a través de una única red neuronal profunda, lo que facilita un procesamiento rápido adecuado para aplicaciones en tiempo real. Esta combinación permite mantener una buena precisión en la detección de una amplia variedad de objetos en diferentes escenarios de la vida real, como los sistemas de control de acceso vehicular y el reconocimiento de vehículos (Younis et al., 2020). La fusión de estos modelos posibilita la localización y clasificación de múltiples objetos en una sola ejecución, mejorando notablemente la velocidad de procesamiento.

Por otro lado, el conjunto de datos COCO (por las siglas en Inglés de *Common Objects in Context*) (Lin et al., 2014) es una extensa colección de imágenes utilizada para la detección y segmentación de objetos. Contiene más de 330,000 imágenes, con más de 2.5 millones de anotaciones en formato XML, abarcando 80 categorías de objetos comunes en contextos variados, como personas, vehículos y animales.

Debido a estas capacidades, los modelos pre-entrenados con el dataset COCO muestran un buen rendimiento en diversas aplicaciones, mejorando tanto la precisión como la eficiencia en

tareas de visión computacional. En la Figura 2.1 se ilustran ejemplos de las imágenes etiquetadas en el dataset (COCO, 2024).



Figura 2.1. Ejemplo de motocicletas incluidas en el COCO dataset. Fuente: (2024).

El propósito de este artículo es implementar un modelo pre-entrenado de detección de objetos, específicamente MobileNetSSDv2FPNlite (Lin et al., 2017), para identificar motocicletas en la entrada del Campus Chontalpa de la Universidad Juárez Autónoma de Tabasco, México, en tiempo real.

Sin embargo, en un primer acercamiento, hemos observado que este modelo tiene dificultad para identificar motocicletas en el entorno de prueba, a menudo confundiendo éstas con bicicletas o detectando únicamente a los conductores de las motocicletas.

Esta confusión puede deberse a varias razones. En primer lugar, las características visuales de las motocicletas y bicicletas en nuestro contexto pueden ser similares en ciertas condiciones, como en ángulos o distancias específicas. Además, el modelo puede no estar suficientemente entrenado en ejemplos representativos de motocicletas de nuestra región, lo que limita su capacidad para generalizar adecuadamente.

Para abordar este problema, fue necesario recopilar un conjunto de imágenes que incluya una mayor diversidad de muestras de motocicletas y, con ello, reentrenar el modelo con el fin de mejorar significativamente su rendimiento.

2.2. Materiales y método

2.2.1. Modelo SSDMobileNetV2FPNlite

El modelo SSDMobileNetV2 es una arquitectura de detección de objetos creada por Andrew G. Howard y sus colaboradores, que combina la eficiencia de MobileNet (Howard et al., 2017) con el enfoque de “*Single Shot Detector*” (SSD) (W. Liu et al., 2016). Está diseñado para detectar y localizar objetos en imágenes de manera rápida y precisa, siendo ideal para dispositivos móviles y entornos con recursos limitados. Además, utiliza FPNLite, una versión optimizada y más ligera de FPN (*Feature Pyramid Network*), una técnica comúnmente empleada en el Aprendizaje profundo para tareas de detección y segmentación de objetos. Entrenado con el conjunto de datos COCO, MobileNetSSDv2FPNlite puede identificar las 80 categorías incluidas en el dataset y puede detectar y clasificar múltiples objetos en una sola imagen.

El modelo pre-entrenado se encuentra disponible con licencia libre en: http://download.tensorflow.org/models/object_detection/tf2/20200711/ssd_mobilenet_v2_fpn-lite_320x320_coco17_tpu-8.tar.gz Específicamente, el modelo consta de los siguientes parámetros:

- `model_name: ssd_mobilenet_v2_fpn-lite_320x320_coco17_tpu-8`
- `base_pipeline_file: ssd_mobilenet_v2_fpn-lite_320x320_coco17_tpu-8.config`
- `pretrained_checkpoint: ssd_mobilenet_v2_fpn-lite_320x320_coco17_tpu-8.tar.gz`

2.2.2. Bibliotecas y herramientas

- OpenCV: biblioteca de código abierto para procesamiento de imágenes y visión por computadora (Bradski y Kaehler, 2008).
- NumPy: proporciona soporte para arreglos multidimensionales y funciones matemáticas de alto rendimiento (NumPy, 2024).
- Matplotlib: permite crear gráficos estáticos, animados e interactivos (Matplotlib, 2024).
- TensorFlow: biblioteca de aprendizaje automático utilizada para cargar y usar modelos pre-entrenados (TensorFlow, 2024).

- Google Colab: entorno de ejecución en la nube con acceso gratuito a GPU y TPU (Google, 2024).

2.2.3. Método

La Figura 2.2 muestra los pasos realizados durante esta investigación.

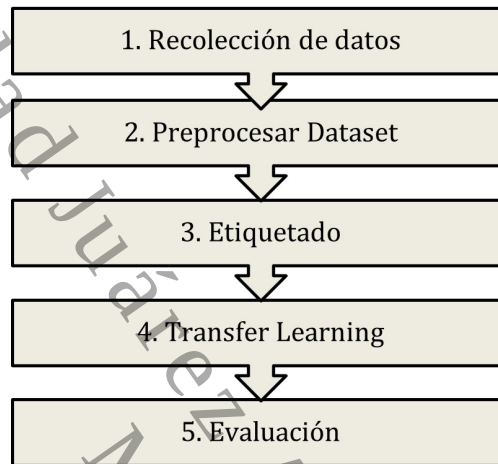


Figura 2.2. Diseño experimental de la investigación. Fuente: Elaboración propia.

El diseño experimental se basó en la recolección de 320 imágenes de motocicletas en la entrada del Campus Chontalpa de la UJAT. Durante el pre-procesamiento, se utilizaron técnicas de recorte y redimensionamiento para ajustar las imágenes a un formato cuadrado que enfocara el área de interés y eliminara ruido. Posteriormente se llevó a cabo el uso de la técnica Transfer Learning con el modelo pre-entrenado MobileNetSSDv2FPNlite. Finalmente, se evaluó el rendimiento del modelo reentrenado mediante las métricas Accuracy, Precision y Recall.

2.2.4. Dataset UJAT

Para el entrenamiento, se construyó un dataset con 320 imágenes a color en formato PNG de motocicletas capturadas en dos días de una misma semana en la entrada del Campus Chontalpa de la Universidad Juárez Autónoma de Tabasco. Las imágenes fueron tomadas en diferentes horarios, desde las 07:00 hasta las 15:00 horas, bajo diversas condiciones climáticas: un día nublado y otro día soleado. En la Tabla 2.1 se muestra la estructura de los tiempos de recolección del dataset UJAT.

Tabla 2.1. Captura de las imágenes de motocicletas.

Día	Clima	Motocicletas	Imágenes
1	Nublado	90	180
2	Soleado	70	140
Total	–	160	320

Este proceso dio lugar a un dataset estructurado que contiene imágenes de motos en diferentes ángulos y condiciones.



Figura 2.3. Ejemplos de imágenes de motocicletas capturadas. Fuente: Elaboración propia.

2.2.5. Pre-procesamiento

En esta fase, se recortaron todas las imágenes del conjunto de datos, dando enfoque al sujeto de detección, y fueron escaladas a una resolución de 320×320 píxeles usando la biblioteca OpenCV de Python.

2.2.6. Etiquetado

La siguiente etapa consistió en crear un conjunto de anotaciones en formato XML para las imágenes del dataset UJAT, compatible con los requerimientos del dataset COCO. Cada anotación de instancia de objeto incluye un identificador de categoría y un cuadro delimitador con coordenadas desde la esquina superior izquierda hasta la esquina inferior derecha. Tomando

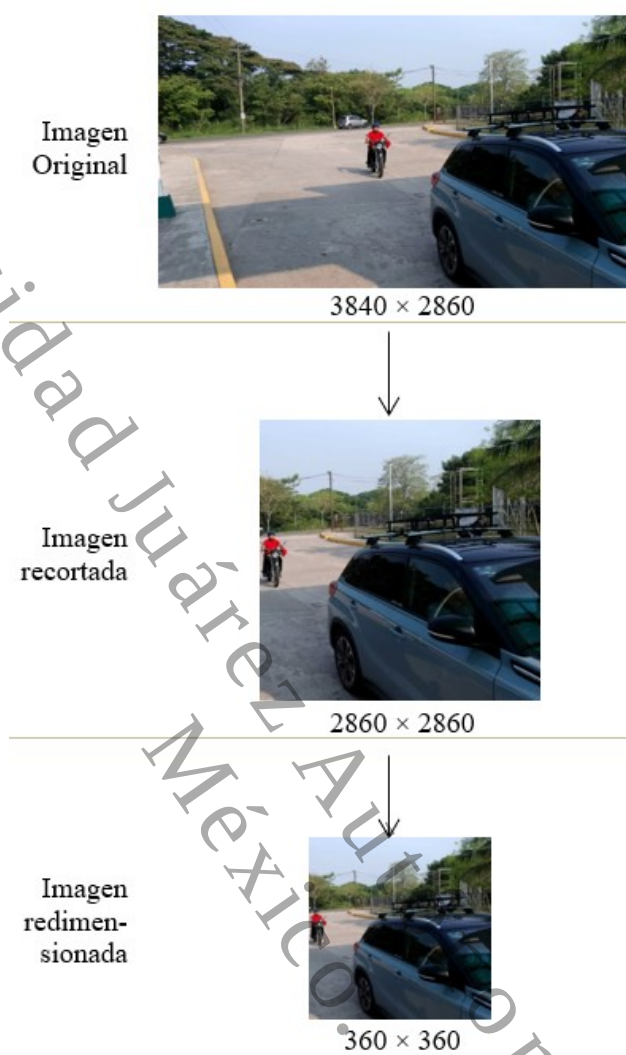


Figura 2.4. Pasos del pre-procesamiento realizado a cada imagen del dataset. Fuente: Elaboración propia.

en cuenta estas consideraciones, se automatizó el proceso de etiquetado mediante un script en Python con los siguientes pasos:

- Cargar y mostrar imágenes.
- Anotación mediante interacción con el mouse.
- Guardar las anotaciones y la imagen anotada.

A continuación se muestra un ejemplo del contenido de la anotación en formato XML:

```
<?xml version='1.0' encoding='utf-8'?>
```

```

<annotation>
  <folder>images</folder>
  <filename>img1.jpg</filename>
  <size>
    <width>320</width>
    <height>320</height>
    <depth>3</depth>
  </size>
  <segmented>0</segmented>
  <object>
    <name>motorcycle</name>
    <bndbox>
      <xmin>148</xmin>
      <ymin>166</ymin>
      <xmax>204</xmax>
      <ymax>229</ymax>
    </bndbox>
  </object>
</annotation>

```

En la Tabla 2.2 se muestran los resultados de la nomenclatura de imágenes y sus anotaciones.

Tabla 2.2. Nomenclatura de imágenes y sus anotaciones.

Imágenes	Anotaciones	Total
img001.jpg	img001.xml	1
img002.jpg	img002.xml	2
...	...	...
img320.jpg	img320.xml	320

2.2.7. Aprendizaje por transferencia

El transfer learning en redes neuronales convolucionales implica entrenar un modelo en un conjunto de datos a gran escala y posteriormente ajustarlo finamente en un conjunto de datos

más pequeño (L. Liu et al., 2020).

2.2.8. Pruebas iniciales del modelo SSDMobileNetV2FPNlite

La primera prueba consistió en probar el modelo base SSDMobileNetV2FPNlite con las 320 imágenes del dataset. Los resultados iniciales mostraron un rendimiento deficiente del 46.25 %, detectando únicamente 148 de las 320 motocicletas.



Figura 2.5. Ejemplo de la ejecución del modelo base. Fuente: Elaboración propia.

2.2.9. Re-entrenamiento del modelo (*Transfer learning*)

Para mejorar el rendimiento del modelo, se utilizó la técnica de aprendizaje por transferencia, ajustando el modelo MobileNetSSDv2FPNlite pre-entrenado con las imágenes y anotaciones específicas del Dataset UJAT.

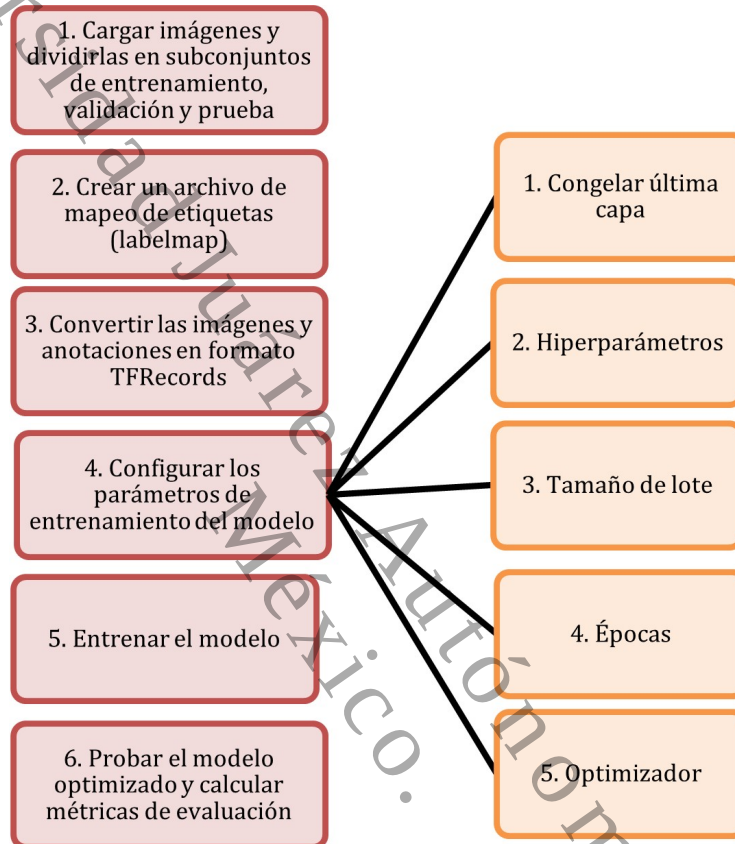


Figura 2.6. Metodología para el reentrenamiento del modelo. Fuente: Elaboración propia.

El dataset fue dividido de la siguiente manera:

- Entrenamiento: 80 %
- Validación: 10 %
- Prueba: 10 %

La configuración final de los parámetros del modelo quedó de la siguiente manera:

- Congelación de capas: capa final.
- Número de clases: 1.

- Escala de cajas:
 - `x_scale`: 10.0
 - `y_scale`: 10.0
 - `width_scale`: 5.0
 - `height_scale`: 5.0
- Pérdida de localización: Smooth L1.
- `batch_size`: 16.
- `num_steps`: 40000.
- Épocas: 1000.
- Optimizador: Momentum Optimizer con Cosine Decay.

2.3. Pruebas y resultados

La evaluación de los resultados implicó medir el desempeño de ambos modelos utilizando las métricas clásicas: Accuracy, Precision y Recall (Padilla et al., 2021).

El Accuracy indica cuántas predicciones del modelo fueron correctas en relación con el total de predicciones:

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN}$$

La Precision muestra cuántas de las predicciones positivas fueron realmente correctas:

$$Precision = \frac{TP}{TP + FP}$$

El Recall evalúa la capacidad del modelo para detectar todas las instancias positivas:

$$Recall = \frac{TP}{TP + FN}$$

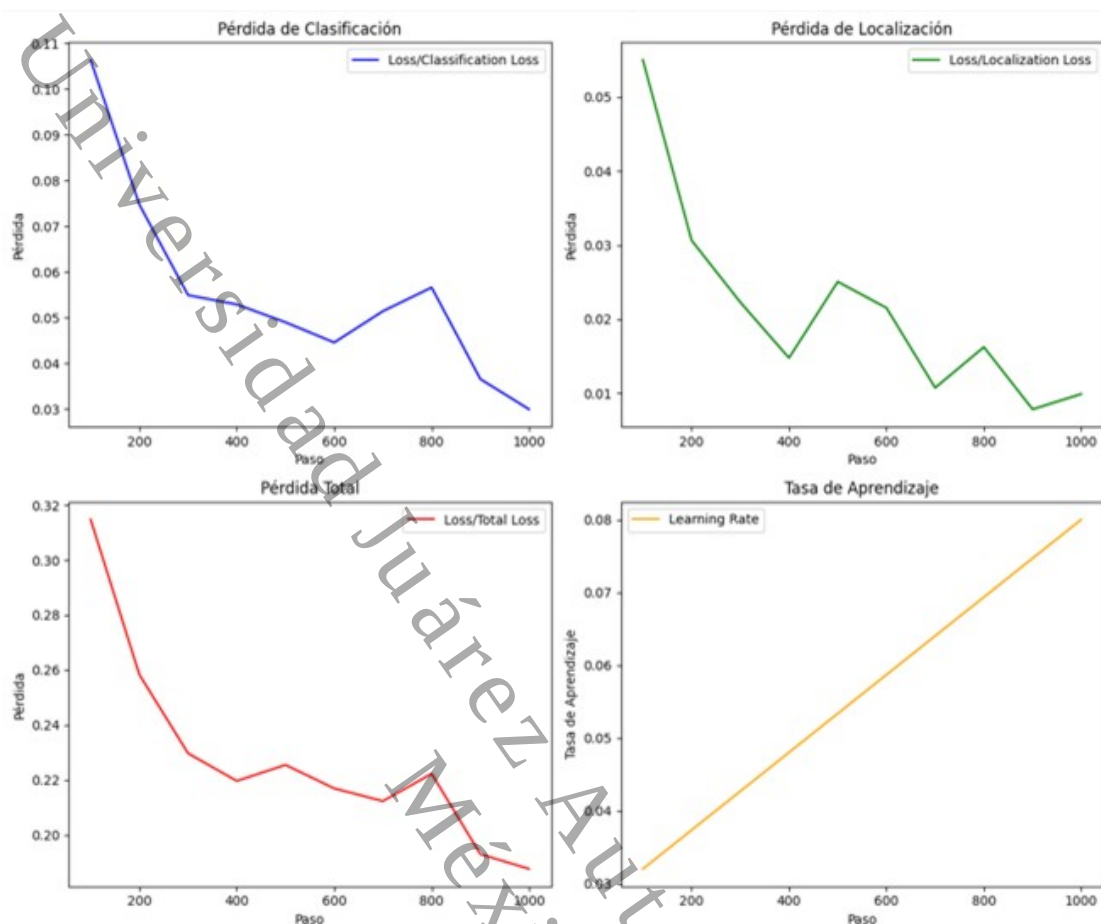


Figura 2.7. Métricas durante el re-entrenamiento del modelo. Fuente: Elaboración propia.

2.3.1. Evaluación del modelo original SSDMobileNetV2FPNlite

Los resultados de evaluar el modelo original fueron:

- 148 TP.
- 49 FN.
- 123 FP.

Tabla 2.3. Matriz de confusión del modelo SSDMobileNetV2FPNlite original.

	Predicción: No motocicleta	Predicción: Motocicleta
Real: No motocicleta	– (TN)	123 (FP)
Real: Motocicleta	49 (FN)	148 (TP)

Con los valores de la matriz de confusión, se calcularon las métricas:

- Accuracy: 0.4625
- Precision: 0.5461
- Recall: 0.7513

2.3.2. Evaluación del modelo optimizado

Los resultados de evaluar el modelo re-entrenado fueron:

- 32 TP.
- 0 FN.
- 0 FP.

Tabla 2.4. Matriz de confusión del modelo SSDMobileNetV2FPNlite optimizado.

	Predicción: No motocicleta	Predicción: Motocicleta
Real: No motocicleta	– (TN)	0 (FP)
Real: Motocicleta	0 (FN)	32 (TP)

Con los valores de la matriz de confusión, se calcularon las métricas:

- Accuracy: 1.000
- Precision: 1.000
- Recall: 1.000

Tabla 2.5. Comparación entre modelos.

Métrica	Modelo original	Modelo optimizado
Accuracy	0.4625	1.000
Precision	0.5461	1.000
Recall	0.7513	1.000

En la Figura 2.8 se muestran un par de ejemplos de detecciones correctas obtenidas con el modelo reentrenado.



Figura 2.8. Ejemplo de resultados. Fuente: Elaboración propia.

2.4. Conclusiones

En esta investigación se evaluaron los resultados de dos modelos de detección de motocicletas utilizando las métricas clave Accuracy, Precision y Recall, las cuales proporcionaron una visión clara de cómo la configuración y el ajuste del modelo pueden impactar significativamente en su rendimiento.

Los resultados demuestran que el modelo optimizado presenta una mejora significativa respecto al modelo original en todas las métricas clave, logrando resultados perfectos en términos de exactitud.

Este comportamiento se debe a la reutilización de un modelo pre-entrenado que optimiza la capacidad de detección en tareas específicas; en este caso, la detección de motocicletas en un entorno real.

Referencias del capítulo

- Bradski, G., & Kaehler, A. (2008). *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media.
- COCO. (2024). *COCO Dataset* [Accedido: 30 de septiembre de 2024]. <https://cocodataset.org/#home>
- Galvez, R. L., Bandala, A. A., Dadios, E. P., Vicerra, R. R. P., & Maningo, J. M. Z. (2018). Object Detection Using Convolutional Neural Networks. *Proceedings of TENCON 2018 - 2018 IEEE Region 10 Conference*. <https://doi.org/10.1109/TENCON.2018.8650517>
- Google. (2024). *Google Colaboratory* [Accedido: 30 de diciembre de 2024]. <https://cloud.google.com/colab/docs>
- Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). Mobilenets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv preprint arXiv:1704.04861*.
- Lin, T. Y., et al. (2014). Microsoft COCO: Common Objects in Context. *Computer Vision – ECCV 2014*, 740-755.

- Lin, T. Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal Loss for Dense Object Detection. *arXiv preprint arXiv:1708.02002*.
- Liu, L., Ouyang, W., Wang, X., Fieguth, P., Chen, J., Liu, X., & Pietikäinen, M. (2020). Deep Learning for Generic Object Detection: A Survey. *International Journal of Computer Vision*, 128(2), 261-318. <https://doi.org/10.1007/s11263-019-01247-4>
- Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. *Computer Vision – ECCV 2016*, 21-37.
- Matplotlib. (2024). *Matplotlib Quick Start* [Accedido: 30 de septiembre de 2024]. https://matplotlib.org/stable/users/explain/quick_start.html
- NumPy. (2024). *NumPy Documentation* [Accedido: 30 de septiembre de 2024]. <https://numpy.org/doc/stable/>
- Padilla, R., Passos, W. L., Dias, T. L. B., Netto, S. L., & da Silva, E. A. B. (2021). A Comparative Analysis of Object Detection Metrics with a Companion Open-Source Toolkit. *Electronics*, 10(3), 279. <https://doi.org/10.3390/electronics10030279>
- TensorFlow. (2024). *tf.keras.models.load_model* [Accedido: 30 de diciembre de 2024]. https://www.tensorflow.org/api_docs/python/tf/keras/models/load_model
- Younis, A., Shixin, L., Shelembi, J. N., & Hai, Z. (2020). Real-Time Object Detection Using Pre-Trained Deep Learning Models MobileNet-SSD. *Proceedings of the Conference*, 1-6. <https://doi.org/10.1145/3379247.3379264>

Capítulo 3

Detección y seguimiento de objetos con YOLO11-ByteTrack

Portada interior

Nombre del artículo: Detección y seguimiento de vehículos con YOLO11-ByteTrack

Autores: Manuel Antonio García-Herrera, Oscar Chávez-Bosquez, José Hernández-Torruco,
Betania Hernández-Ocaña

Institución de adscripción: División Académica de Ciencias y Tecnologías de la Información,
Universidad Juárez Autónoma de Tabasco

Resumen

Este artículo presenta un modelo para el seguimiento y conteo de vehículos en tiempo real en la entrada de un campus universitario, integrando el modelo de detección YOLO11n con el algoritmo ByteTrack. Se evalúan dos enfoques de seguimiento: Región de Interés (RoI) y Cruce de Línea (CtL), bajo múltiples configuraciones que incluyen RoI rectangular y poligonal, distintos umbrales de persistencia en cuadros y diferentes posiciones de línea. Se construyó un conjunto de datos de 25 videos capturados bajo diversas condiciones climáticas y de tráfico, considerando cuatro clases: automóviles, motocicletas, bicicletas y autobuses. El desempeño del sistema se evalúa mediante las métricas de precisión, recall y F1-score. Los resultados muestran que la configuración CtL vertical obtuvo el mejor desempeño con un F1-score de 0.7733, seguida por la RoI rectangular con 0.7651. Entre las principales limitaciones se encuentran la confusión de vehículos estacionados como flujo activo y la baja detección de objetos de gran tamaño o de pequeña escala.

Palabras clave: visión por computadora, aprendizaje profundo, redes neuronales, detección de objetos, seguimiento de objetos.

Artículo en revisión en la revista *Electronic Letters on Computer Vision and Image Analysis*.

3.1. Introducción

El seguimiento de objetos, en el contexto de la visión por computadora, se refiere al proceso de localizar y seguir un objeto en una secuencia de video o imagen a lo largo del tiempo. A partir de la definición anterior, se distinguen dos categorías: el seguimiento de un solo objeto (SOT) y el seguimiento de múltiples objetos (MOT). La Figura 3.1 muestra un diagrama de ejemplo que separa conceptualmente los dos tipos de seguimiento. El SOT se centra en seguir un único objeto, mientras que el MOT implica seguir la trayectoria de múltiples objetos en la escena. Es claro que la mayor complejidad del MOT amplía su aplicabilidad, pero al mismo tiempo enfrenta desafíos adicionales como oclusión, cambios de escala y coincidencia de identidad entre múltiples fotogramas (Y. Wang y Mariano, 2024).

El MOT comienza con la detección de un conjunto inicial de objetos, a los cuales se les asigna una identidad única, y luego se sigue su movimiento a través de los fotogramas, interpretando cada objeto como una trayectoria individual. Esta tecnología tiene una amplia gama de aplicaciones, tanto en imágenes en tiempo real como en videos previamente grabados, particularmente en áreas como el control de acceso vehicular (Du y Bobkov, 2023b). En este sentido, la detección de objetos representa el núcleo del seguimiento de objetos, ya que identifica y localiza objetos mediante cajas delimitadoras en fotogramas individuales de una imagen o video.

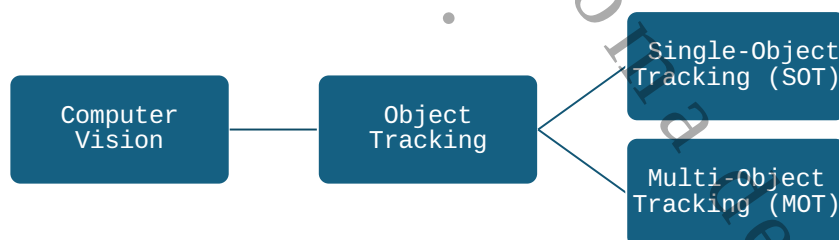


Figura 3.1. Seguimiento de un solo objeto (SOT) y seguimiento de múltiples objetos (MOT)

Existen varios algoritmos comúnmente utilizados para el seguimiento de objetos, que van desde algoritmos tradicionales de visión por computadora, por ejemplo: sustracción de fondo (Chiu et al., 2018), flujo óptico (Alfarano et al., 2024), rastreador por centroides (Y. Chen et al., 2024); hasta enfoques de aprendizaje profundo: Deep SORT (Pereira et al., 2022), rastreadores basados en redes siamesas (Hu et al., 2023), seguimiento de objetos con transformadores (Meinhardt et al., 2022). Sin embargo, cada técnica tiene sus propias ventajas y desventajas, dependiendo

principalmente del escenario.

En este trabajo, integramos el algoritmo de seguimiento de múltiples objetos ByteTrack (Y. Wang y Mariano, 2024) con el modelo de aprendizaje profundo YOLO (*You Only Look Once*) (Jocher y Qiu, 2024b) y probamos nuestra implementación en un entorno real, comparando dos técnicas: seguimiento por Región de Interés (RoI) y seguimiento por cruce de línea (CtL). El propósito de este artículo es mostrar la mejor configuración de seguimiento de objetos para detectar cuatro tipos de vehículos (autos, motocicletas, bicicletas y autobuses) en la entrada de un campus de la Universidad más grande de Tabasco, México, con el fin de contribuir al monitoreo efectivo del acceso vehicular y mejorar la seguridad de la comunidad académica.

3.2. Trabajos relacionados

El modelo de detección de objetos YOLO (*You Only Look Once*) ha demostrado ser fundamental debido a su velocidad y rendimiento, permitiendo una identificación eficiente de objetos y mejorando la precisión del seguimiento continuo (Yang, Tang, Chen et al., 2020b). Al combinar la detección de objetos con algoritmos de seguimiento, el rendimiento se optimiza en distintos escenarios, abordando desafíos como movimientos rápidos, oclusión de objetos y variabilidad en las condiciones de iluminación (Gao et al., 2024b).

Específicamente, el detector YOLO ha sido utilizado en diversos dominios, como la detección y clasificación de residuos (Ren et al., 2024), la extracción automática de eventos clave y resumen de videos deportivos mediante la detección de marcadores (Guntuboina et al., 2021), y sistemas de procesamiento de señales para la detección de personas en áreas públicas concurridas (Rondón et al., 2021). También ha sido aplicado mediante una aplicación web para detectar baches en carreteras de asfalto (Peres Barbosa et al., 2024), en la gestión reproductiva y de salud de animales de granja (Reyes Zambrano y Moreira Santos, 2025), en la inspección de subestaciones eléctricas mediante imágenes térmicas (Pérez-Aguilar et al., 2024), en una aplicación móvil para el reconocimiento de monedas y billetes colombianos sin conexión a Internet (Bolaños-Fernández y Bacca-Cortes, 2024), en escenarios de detección de caídas de múltiples personas (L. Liu et al., 2025), en el reconocimiento y conteo automático de células mediante imágenes de microscopio (López Flórez et al., 2023), y en la detección de defectos del tabique

ventricular en imágenes de ultrasonido (S.-H. Chen et al., 2021).

3.3. Metodología

3.3.1. Materiales

- **Modelo Yolo11n:** Ultralytics YOLO11 es la versión más reciente del modelo de detección de objetos en tiempo real y segmentación de imágenes de última generación. Su diseño particular lo hace adecuado para una amplia gama de aplicaciones y permite una fácil adaptación a diversas plataformas de hardware, desde dispositivos de borde hasta APIs en la nube (Jocher y Qiu, 2024b).
- **Algoritmo ByteTrack:** Supervision ByteTrack es un algoritmo de seguimiento confiable, especialmente en escenas complejas donde los objetos suelen perderse (Zhang et al., 2022). Asigna una identidad única a cada objeto, permitiendo un seguimiento confiable a lo largo del tiempo y manejando desafíos como la oclusión y la reidentificación en entornos dinámicos (Roboflow, 2024).
- **OpenCV:** La biblioteca de código abierto más utilizada para la manipulación de imágenes y el procesamiento de visión por computadora. Proporciona herramientas para tareas como lectura, escritura y transformación de imágenes, así como técnicas como filtrado, detección de bordes y transformaciones geométricas. Sus funciones permiten ajustar brillo, contraste y aplicar diversos filtros, además de realizar operaciones como escalado y rotación. OpenCV se integra eficientemente con otras bibliotecas como NumPy, lo que facilita el procesamiento de datos (Bradski, 2000).
- **NumPy:** Proporciona soporte para trabajar con arreglos multidimensionales y realizar cálculos matemáticos complejos de manera eficiente. Las operaciones vectorizadas permiten procesar grandes volúmenes de datos, facilitando tareas como el análisis de imágenes y cálculos numéricos avanzados (Harris et al., 2020).
- **MoviePy:** Utilizado para la edición básica de video. Permite cortar, unir y combinar videos con texto o efectos. También se usa para crear animaciones simples. Es compatible con

formatos comunes de audio y video (Zulko, 2015).

- Google Colaboratory: Colab es una plataforma basada en la nube en la que se crea y ejecuta código Python bajo un modelo de computación interactiva. Proporciona acceso gratuito a recursos computacionales como GPUs y TPUs para proyectos de aprendizaje automático y análisis de datos. Su interfaz amigable y compatibilidad con bibliotecas populares como NumPy y OpenCV lo convierten en una herramienta accesible para investigadores y desarrolladores (Google, 2025).

3.3.2. Método

Esta propuesta emplea un modelo de detección de objetos preentrenado, con un algoritmo de seguimiento aplicado para contar las clases detectadas en videos en tiempo real. La etapa final consiste en evaluar el rendimiento del algoritmo de seguimiento para mejorar la precisión del rastreo y el conteo de objetos.

La Figura 3.2 muestra la metodología del proyecto.

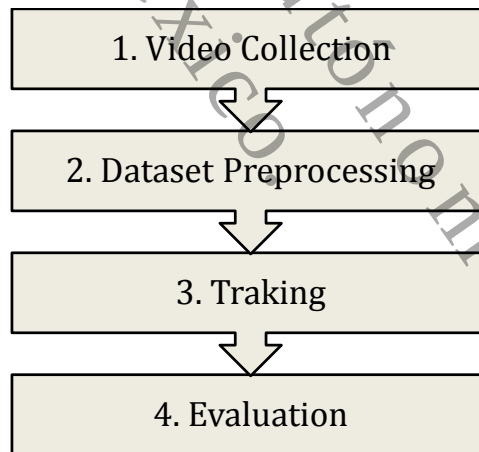


Figura 3.2. Metodología del proyecto

3.4. Conjunto de datos UJAT

En este experimento, se creó un conjunto de datos compuesto por 25 videos en formato MP4 para capturar las 4 clases seleccionadas (automóviles, motocicletas, bicicletas y autobuses).

Los videos fueron grabados durante dos días diferentes de la misma semana, en la entrada del Campus Chontalpa de la Universidad Juárez Autónoma de Tabasco. Los videos fueron grabados en diferentes intervalos de tiempo, entre las 07:00 y las 15:00 horas, bajo distintas condiciones climáticas: un día nublado y un día soleado. Para optimizar la recolección de datos, se priorizaron las horas pico de tráfico, típicamente a las 08:00, 10:00 y 12:00 horas. La Tabla 3.1 muestra los detalles de captura de video del conjunto de datos UJAT.

Tabla 3.1. Detalles de captura de video

Día	Condiciones climáticas	Número de videos
1	Nublado	10
2	Soleado	15
Total		25

Cada video fue grabado desde una ubicación preestablecida por la administración del campus, por lo que la cámara tuvo que mantenerse en una posición fija. Cabe destacar que los videos fueron capturados desde el lado izquierdo de la entrada del campus, considerando la posición del sol en cada hora del día. Los videos se grabaron con una resolución de 3840×2160 píxeles, lo que ayudó a mejorar la calidad visual y facilitar el análisis posterior. La Figura 3.3 muestra el acceso al Campus Chontalpa.



Figura 3.3. Entrada del Campus Chontalpa

3.4.1. Preprocesamiento de video

En esta fase, todos los videos del conjunto de datos fueron recortados a una relación de aspecto cuadrada 1:1 y luego redimensionados a una resolución de 848×848 píxeles. Este ajuste optimiza el uso de memoria y reduce el tiempo de procesamiento, mejorando el rendimiento del modelo en tareas de detección y seguimiento. La Figura 4.3 muestra un ejemplo del esquema del preprocesamiento de datos.

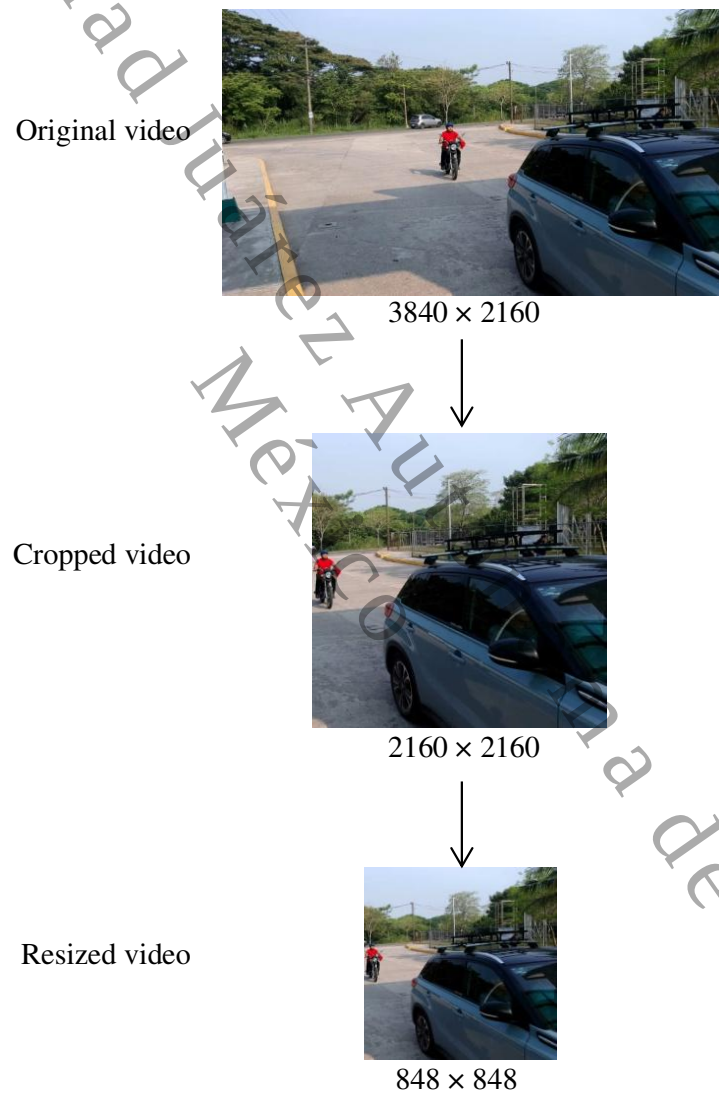


Figura 3.4. Esquema del preprocesamiento de datos

3.5. Diseño experimental

Las primeras etapas del proceso de configuración son las siguientes:

1. Carga del modelo: Se carga el modelo YOLO preentrenado para detectar las 4 clases (automóviles, motocicletas, bicicletas y autobuses) en el conjunto de videos.
2. Detección de objetos: El video de entrada se carga y los fotogramas se extraen para realizar la detección de objetos. El modelo YOLO se utiliza para identificar y localizar objetos dentro de cada fotograma del video.
3. Filtrado de detecciones: Las detecciones realizadas por el modelo YOLO se filtran para seleccionar únicamente los objetos que pertenecen a las clases de interés (automóvil, motocicleta, bicicleta y autobús).
4. Inicialización del rastreador: El algoritmo ByteTrack se inicializa y se utiliza para rastrear los objetos a través de los fotogramas del video. El rastreador se actualiza con las detecciones filtradas para mantener el seguimiento, asegurando que los objetos detectados en cada fotograma estén correctamente asociados.
5. Anotación de fotogramas: En esta etapa, se agregan anotaciones al fotograma para mostrar los objetos detectados. Se dibujan trayectorias, cajas delimitadoras, etiquetas (con ID del rastreador, clase y nivel de confianza) y la línea de zona para visualizar el conteo en el video.

Figura 3.5 muestra el diseño experimental propuesto.

3.5.1. Conteo de objetos en la Región de Interés (RoI)

Este proceso tuvo como objetivo realizar un seguimiento y conteo preciso de los objetos detectados dentro de una Región de Interés (RoI) en un video, filtrando únicamente los vehículos que permanecen dentro de la RoI durante un cierto periodo de tiempo, es decir, un número fijo de fotogramas. En este experimento se probaron 2 formas de RoI:

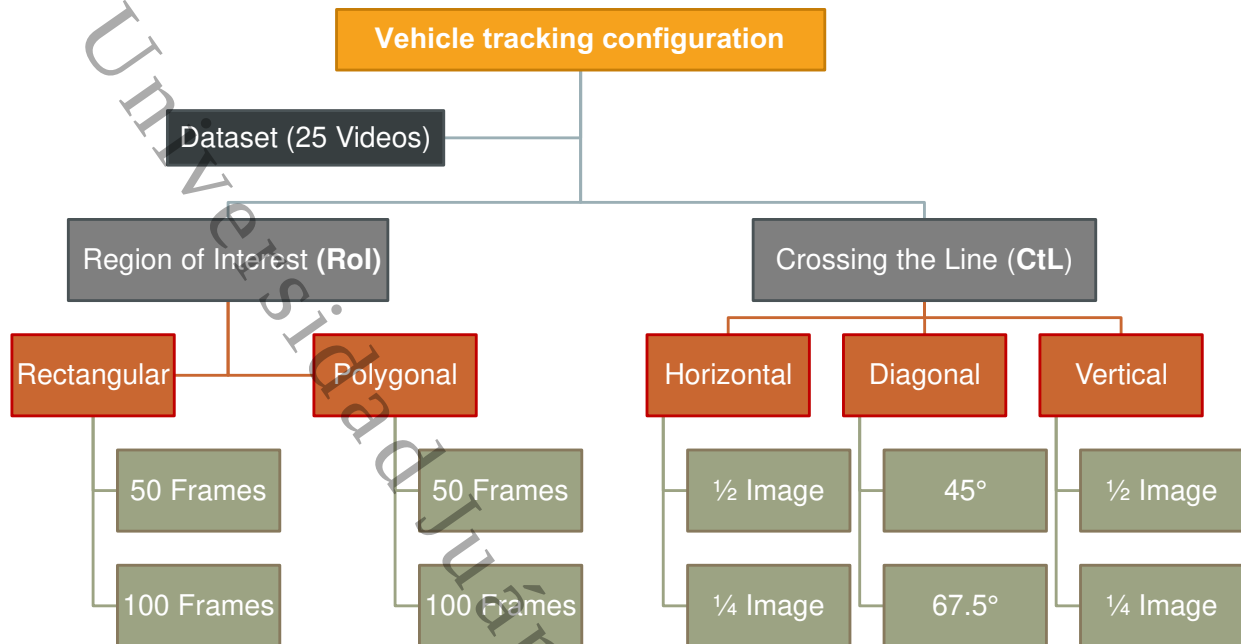


Figura 3.5. Diseño experimental

RoI rectangular. Se definió un área específica dentro del video para el seguimiento de objetos, conocida como Región de Interés (RoI). En los videos, la RoI se resalta para su análisis, con su posición y tamaño definidos por las coordenadas (x, y) y las dimensiones (*ancho*, *alto*). Parámetros:

- $\text{RoI} = (260, 340, 400, 300)$
- $(x = 260, y = 340)$ representa la esquina superior izquierda de la RoI
- $(\text{ancho} = 400, \text{alto} = 300)$ es el tamaño del rectángulo RoI

La Figura 3.6 (izquierda) muestra un ejemplo de RoI rectangular.

RoI poligonal En esta configuración, la RoI se representa como un polígono personalizado. Los puntos del polígono son:

- $p_1 = (50, 750)$ — esquina inferior izquierda
- $p_2 = (50, 350)$ — esquina superior izquierda
- $p_3 = (270, 350)$ — esquina superior interna izquierda
- $p_4 = (636, 430)$ — esquina superior interna derecha



Figura 3.6. Rol rectangular (izquierda), Rol poligonal (derecha)

- $p_5 = (640, 750)$ — esquina inferior derecha

La Figura 3.6 (derecha) muestra un ejemplo de Rol poligonal.

Posteriormente, se definieron dos umbrales mínimos de fotogramas para contar vehículos dentro de la Rol:

1. 50 fotogramas: Si un vehículo detectado permanece dentro de la Rol durante 50 fotogramas o más, se cuenta como un vehículo que ingresa al campus. Como la tasa de fotogramas del video es de 30 fps, 50 fotogramas representan aproximadamente 1.5 segundos.
2. 100 fotogramas: En esta configuración, se evalúa un mínimo de 100 fotogramas dentro de la Rol para contar vehículos, lo que representa aproximadamente 3 segundos.

En resumen, solamente los vehículos que permanecieron dentro de la Rol durante un número suficiente de fotogramas (50 o 100) fueron contabilizados. Los objetos que no cumplieron este criterio fueron excluidos del conteo final.

3.5.2. Conteo de objetos cruzando la línea (CtL)

Este proceso tiene como objetivo contabilizar los vehículos detectados únicamente cuando cruzan una línea fija. En este experimento se probaron tres ubicaciones diferentes de la línea:

Línea horizontal. Se dibuja una línea horizontal que cubre todo el ancho del frame desde el píxel 0 hasta 848 en el eje x . Se probaron 2 alturas: una dividiendo el frame a la mitad y otra a un cuarto. La Figura 3.7 muestra estas configuraciones.

Línea diagonal. Se dibuja una línea diagonal que divide el frame en dos, desde la esquina inferior izquierda hasta la esquina superior derecha, formando un ángulo de 45° . También se probó otra configuración desde un cuarto de altura, formando un ángulo de 67.5° . La Figura 3.8 muestra ambos casos.

Línea vertical. Se dibuja una línea vertical desde el píxel 0 hasta 848 en el eje y . Se probaron 2 posiciones: una en el centro del frame y otra en un cuarto del ancho. La Figura 3.9 muestra estas configuraciones.



Figura 3.7. Posición de línea horizontal. Mitad del frame (izquierda), cuarto del frame (derecha)

3.6. Resultados y evaluación

A continuación se presentan los resultados obtenidos mediante el conteo automático de vehículos, utilizando cinco enfoques diferentes: Rol rectangular (50 y 100 fotogramas), Rol poligonal (50 y 100 fotogramas), CtL horizontal (mitad y cuarto del frame), CtL diagonal (45° y 67.5°), y CtL vertical (mitad y cuarto del frame).

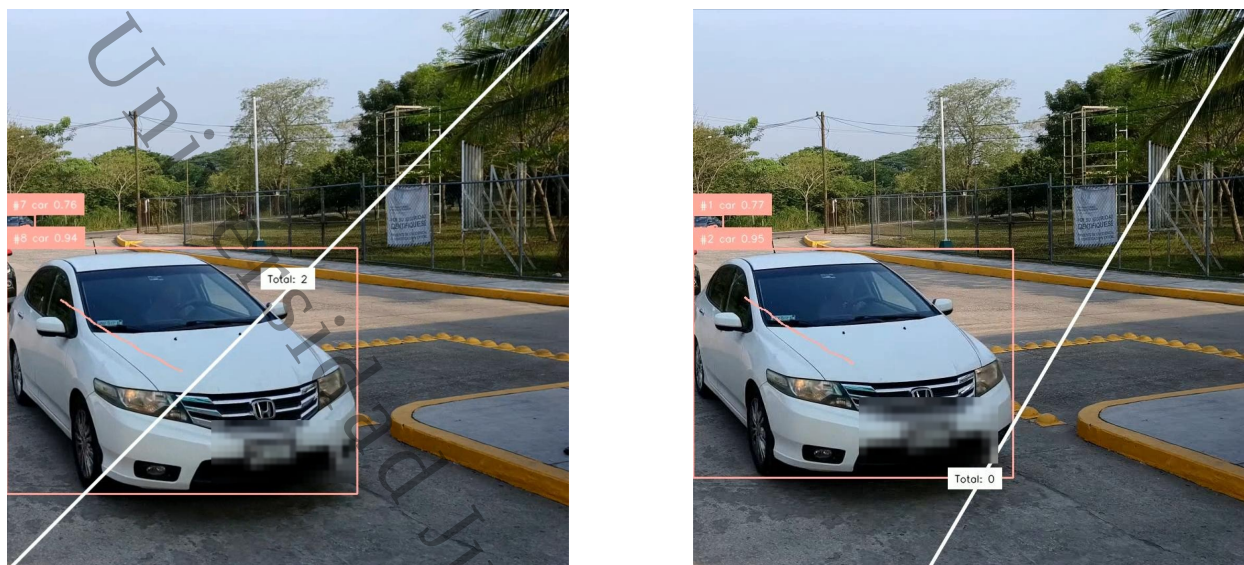


Figura 3.8. Posición de línea diagonal: 45° (izquierda), 67.5° (derecha)

La evaluación del modelo se realizó comparando los resultados del conteo automático con un conteo manual de referencia en un conjunto de 25 videos, aplicando los cinco métodos a cada video en dos configuraciones diferentes (250 evaluaciones en total).

La Tabla 3.2 presenta el conteo total obtenido por cada método automático en comparación con el conteo manual (*ground truth*). Cada fila representa un video, y los valores reflejan el número total de vehículos detectados (suma de automóviles, motocicletas, bicicletas y autobuses), así como el total de vehículos en la última columna. Se analizaron un total de 250 evaluaciones: 25 videos $\times$ 5 métodos de conteo $\times$ 2 configuraciones.

La Figura 3.10 muestra el número de vehículos detectados por experimento, representado para cada técnica.

3.6.1. Métricas de evaluación del rendimiento

La matriz de confusión sintetiza globalmente el comportamiento del sistema en términos de aciertos y errores, comparando los conteos manuales con las predicciones obtenidas en las 250 evaluaciones automáticas.

La Tabla 3.3 resume los siguientes valores:

- Verdaderos positivos (TP): vehículos correctamente rastreados.



Figura 3.9. Posición de línea vertical. Mitad del frame (izquierda), cuarto del frame (derecha)

- Falsos positivos (FP): detecciones que no corresponden a vehículos reales.
- Falsos negativos (FN): vehículos reales no detectados por el modelo.
- Nota: los verdaderos negativos (TN) no se consideran en este contexto, ya que solo se evalúan conteos reales vs. detecciones (Goswami, 2020).

A continuación, se calculan las métricas principales derivadas de la matriz de confusión (Awad et al., 2021):

Precisión. Mide cuántos vehículos detectados son realmente correctos:

$$\text{Precisión} = \frac{TP}{TP + FP} \quad (3.1)$$

Recall. Mide cuántos vehículos reales fueron detectados correctamente:

$$\text{Recall} = \frac{TP}{TP + FN} \quad (3.2)$$

F1-score. Combina precisión y recall en una sola métrica:

$$\text{F1-score} = \frac{2 \cdot \text{Precisión} \cdot \text{Recall}}{\text{Precisión} + \text{Recall}} \quad (3.3)$$

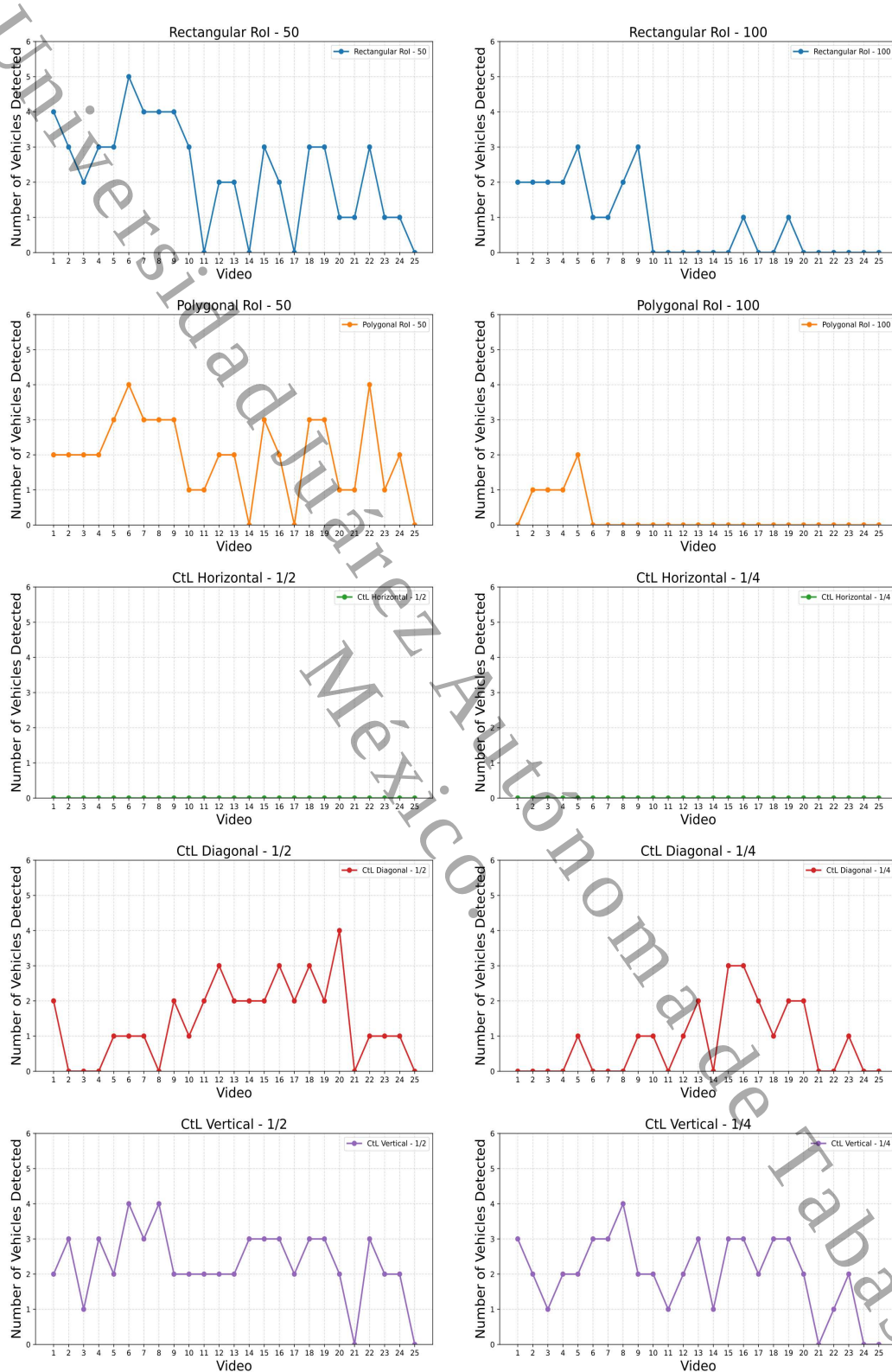


Figura 3.10. Número de vehículos detectados por experimento. Fila #1: RoI rectangular (50 y 100 fotogramas), fila #2: RoI poligonal (50 y 100 fotogramas), fila #3: CtL horizontal (mitad y cuarto del frame), fila #4: CtL diagonal (45° y 67.5°), fila #5: CtL vertical (mitad y cuarto del frame)

Tabla 3.2. Resultados del seguimiento de vehículos. Mejor resultado en **negrita**

Video	Rol rectangular		Rol poligonal		CtL horizontal		CtL diagonal		CtL vertical		Verdad real
	50	100	50	100	1/2	1/4	1/2	1/4	1/2	1/4	
1	4	2	2	0	0	0	2	0	2	3	5
2	3	2	2	1	0	0	0	0	3	2	3
3	2	2	2	1	0	0	0	0	1	1	3
4	3	2	2	1	0	0	0	0	3	2	5
5	3	3	3	2	0	0	1	1	2	2	3
6	5	1	4	0	0	0	1	0	4	3	6
7	4	1	3	0	0	0	1	0	3	3	5
8	4	2	3	0	0	0	0	0	4	4	7
9	4	3	3	0	0	0	2	1	2	2	4
10	3	0	1	0	0	0	1	1	2	2	3
11	0	0	1	0	0	0	2	0	2	1	2
12	2	0	2	0	0	0	3	1	2	2	4
13	2	0	2	0	0	0	2	2	2	3	3
14	0	0	0	0	0	0	2	0	3	1	3
15	3	0	3	0	0	0	2	3	3	3	5
16	2	1	2	0	0	0	3	3	3	3	3
17	0	0	0	0	0	0	2	2	2	2	2
18	3	0	3	0	0	0	3	1	3	3	4
19	3	1	3	0	0	0	2	2	3	3	5
20	1	0	1	0	0	0	4	2	2	2	4
21	1	0	1	0	0	0	0	0	0	0	2
22	3	0	4	0	0	0	1	0	3	1	4
23	1	0	1	0	0	0	1	1	2	2	3
24	1	0	2	0	0	0	1	0	2	0	3
25	0	0	0	0	0	0	0	0	0	0	1
Total	57	20	50	5	0	0	36	20	58	50	92

La Tabla 3.4 muestra los resultados de estas métricas para cada técnica evaluada. La Figura 3.11 muestra la comparación de precisión, recall y F1-score entre técnicas.

3.7. Discusión

La integración del modelo YOLO11n con el algoritmo ByteTrack demostró proporcionar una solución confiable y eficiente para la detección y el seguimiento de vehículos en tiempo real en la entrada del Campus Chontalpa.

En particular, los mejores resultados se obtuvieron utilizando las técnicas de Conteo por Cruce

Tabla 3.3. Matriz de confusión

		Predicciones	
		No	Sí
Vehículos reales	No	–	FP
	Sí	FN	TP

Tabla 3.4. Métricas de rendimiento en cada experimento (*ground truth*: 92 vehículos). Mejores valores en **negrita**

Técnica	Configuración	TP	FP	FN	Precisión	Recall	F1-score
Rol rectangular	50 frames	57	1	34	0.9827	0.6263	0.7651
	100 frames	19	1	72	0.9500	0.2088	0.3424
Rol poligonal	50 frames	50	1	41	0.9803	0.5494	0.7042
	100 frames	4	1	87	0.8000	0.0440	0.0834
CtL horizontal	Mitad del frame	0	0	0	0.0000	0.0000	0.0000
	Cuarto del frame	0	0	0	0.0000	0.0000	0.0000
CtL diagonal	45°	36	0	56	1.0000	0.3913	0.5625
	67.5°	20	0	72	1.0000	0.2174	0.3571
CtL vertical	Mitad del frame	58	0	34	1.0000	0.6304	0.7733
	Cuarto del frame	50	0	42	1.0000	0.5435	0.7042

de Línea Vertical (CtL vertical) y Región de Interés rectangular (Rol rectangular) (Figura 3.12), las cuales superaron de manera constante a las demás configuraciones. CtL vertical (mitad del frame) alcanzó un F1-score de 0.7733, mientras que Rol rectangular (50 fotogramas) alcanzó 0.7651.

Sin embargo, se identificó una limitación en el uso de Rol: los vehículos estacionados también fueron contados como tráfico activo, lo que puede generar un conteo adicional del flujo vehicular. Este problema fue especialmente evidente en escenarios con movimiento limitado, como se observa en la Figura 3.13. En este caso, una motocicleta permaneció dentro de la Rol durante suficientes fotogramas, por lo que el modelo la contó como un vehículo ingresando al campus.

La Rol poligonal, a pesar de su aparente flexibilidad, no superó a la rectangular. La mayor

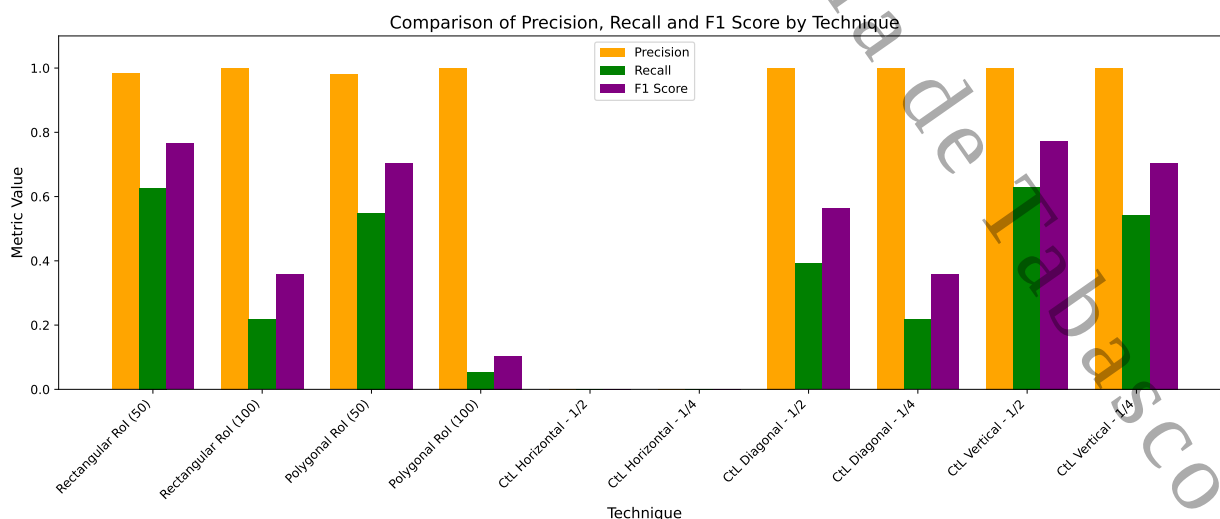


Figura 3.11. Valores de Precisión, Recall y F1-score



Figura 3.12. Mejores configuraciones: CtL vertical (izquierda) y Rol rectangular (derecha)

complejidad en el cálculo de límites redujo la efectividad del seguimiento y la confianza en la detección. Por lo tanto, se deduce que las formas geométricas simples resultan más efectivas en entornos de vigilancia fijos.

El peor desempeño se obtuvo con la configuración CtL horizontal, ya que no logró detectar vehículos, principalmente debido a la mala alineación con el flujo vehicular. Otros resultados deficientes se debieron a la posición fija de la cámara, la cual no pudo modificarse por las restricciones de ubicación.

La precisión fue de 1.000 en las configuraciones CtL vertical y CtL horizontal, lo que indica la

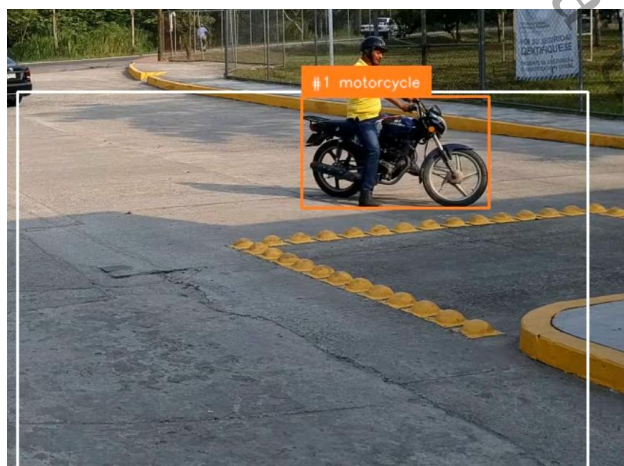


Figura 3.13. Vehículos estacionados contados como tráfico activo en pruebas Rol

ausencia de falsos positivos (FP). En las configuraciones RoI, solo se registró un FP. Sin embargo, el Recall varió significativamente según la configuración, lo que sugiere que algunos vehículos no fueron detectados debido a limitaciones en la generalización del modelo. Se identificó un caso particular en el que un vehículo tipo *buggy* fue detectado mayormente como motocicleta, lo que refleja una limitación en la diferenciación de clases (Figura 3.14).



Figura 3.14. Vehículo tipo *buggy* detectado como motocicleta

También se observó que el único autobús presente en el conjunto de datos no fue rastreado. Esto puede deberse al gran tamaño del objeto, que excede la escala manejada por el modelo. Como resultado, el autobús fue excluido del conteo final, como se muestra en la Figura 3.15.

Además, las motocicletas y bicicletas que circulaban muy cerca de la orilla de la vía no fueron detectadas en la mayoría de los casos. Su proximidad al borde del frame y su pequeño tamaño pudieron contribuir a su omisión en el conteo, como se ilustra en la Figura 3.16.

3.8. Conclusiones

En este estudio se demostró la efectividad de integrar un modelo de aprendizaje profundo de última generación como YOLO11 con un algoritmo de seguimiento robusto como ByteTrack para la detección y conteo de vehículos en tiempo real. Un conjunto de datos personalizado, diseñado bajo condiciones variadas de iluminación y clima, permitió una evaluación realista del rendimiento.

El mejor desempeño de conteo se obtuvo mediante el método de Cruce de Línea (CTL) en posición vertical en la mitad del frame. Otra técnica con buenos resultados fue el conteo de 50



Figura 3.15. Autobús de gran tamaño excluido del conteo final en todas las configuraciones

fotogramas dentro de una Región de Interés (RoI). Sin embargo, en casos donde un vehículo se detiene dentro de la RoI, es incluido en el conteo, generando un falso positivo en situaciones poco frecuentes. Además, las RoI poligonales introdujeron complejidad innecesaria, reduciendo la confianza en la detección y el seguimiento. Las configuraciones CtL horizontales tuvieron un bajo desempeño debido a su mala alineación con la cámara fija, la cual no pudo ajustarse por restricciones del entorno.

Se observó variabilidad en precisión y recall, especialmente en escenas con oclusiones, mo-

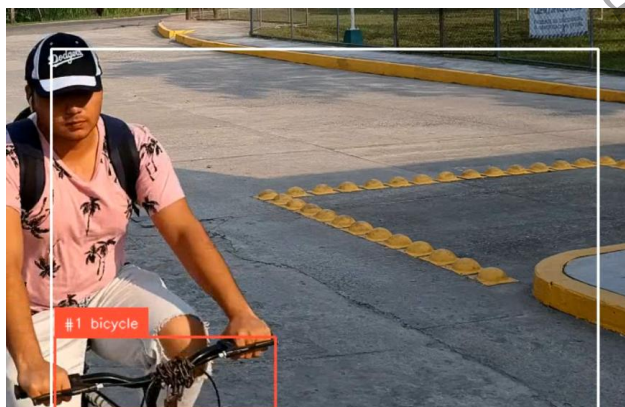


Figura 3.16. Bicicleta cercana a la banqueta

vimientos rápidos o mala iluminación. Esto resalta la necesidad de mejorar la generalización del modelo para asegurar una detección correcta en todos los tipos de vehículos.

Finalmente, el conjunto de datos utilizado fue diseñado para reflejar condiciones reales del campus, incorporando dos patrones climáticos y varios intervalos de tiempo, lo que añadió robustez a la evaluación.

Referencias del capítulo

- Alfarano, A., Maiano, L., Papa, L., & Amerini, I. (2024). Estimating optical flow: A comprehensive review of the state of the art [<https://doi.org/10.1016/j.cviu.2024.104160>]. *Computer Vision and Image Understanding*, 249(1), 104160. <https://doi.org/10.1016/j.cviu.2024.104160>
- Awwad, S., Igried, B., Wedyan, M., & Alshira'H, M. (2021). Hybrid Features for Object Detection in RGB-D Scenes [<https://doi.org/10.11591/ijeecs.v23.i2.pp1073-1083>]. *Indonesian Journal of Electrical Engineering and Computer Science*, 23(2), 1073-1083. <https://doi.org/10.11591/ijeecs.v23.i2.pp1073-1083>
- Bolaños-Fernández, C., & Bacca-Cortes, E. B. (2024). Mobile Application for Recognizing Colombian Currency with Audio Feedback for Visually Impaired People [<https://doi.org/10.14483/23448393.21408>]. *Ingeniería*, 29(2), e21408. <https://doi.org/10.14483/23448393.21408>
- Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.
- Chen, S.-H., Wang, C.-W., Tai, I.-H., Ken-Pen Weng, Y.-H. C., & Hsieh, K.-S. (2021). Modified YOLOv4-DenseNet Algorithm for Detection of Ventricular Septal Defects in Ultrasound Images [<https://doi.org/10.9781/ijimai.2021.06.001>]. *International Journal of Interactive Multimedia and Artificial Intelligence*, 6(7), 1-8. <https://doi.org/10.9781/ijimai.2021.06.001>
- Chen, Y., Liu, X., Li, C., Zhu, J., Wu, M., & Su, X. (2024). UAV Swarm Centroid Tracking for Edge Computing Applications Using GRU-Assisted Multi-Model Filtering [<https://doi.org/10.3390/electronics13061054>]. *Electronics*, 13(6), 1-15. <https://doi.org/10.3390/electronics13061054>
- Chiu, S.-Y., Chiu, C.-C., & Xu, S. (2018). A Background Subtraction Algorithm in Complex Environments Based on Category Entropy Analysis [<https://doi.org/10.3390/app8060885>]. *Applied Sciences*, 8(6), 1-15. <https://doi.org/10.3390/app8060885>

- Du, K., & Bobkov, A. (2023b). An Overview of Object Detection and Tracking Algorithms [<https://doi.org/10.3390/engproc2023033022>]. *Engineering Proceedings*, 33(22), 1-6. <https://doi.org/10.3390/engproc2023033022>
- Gao, J., Han, G., Zhu, H., & Liao, L. (2024b). Multiple Moving Vehicles Tracking Algorithm with Attention Mechanism and Motion Model [<https://doi.org/10.3390/electronics13010242>]. *Electronics* 2024, 13(242), 1-12. <https://doi.org/10.3390/electronics13010242>
- Google. (2025). *Colaboratory Documentation* [<https://cloud.google.com/colab/docs?hl=es-419>].
- Goswami, S. (2020). False detection (positives and negatives) in object detection [<https://doi.org/10.48550/arXiv.2008.06986>]. *arXiv preprint arXiv:2008.06986*, 1(1), 1-15. <https://doi.org/10.48550>
- Guntuboina, C., Aditya Porwal abd, P. J., & Shingrakhia, H. (2021). Deep Learning Based Automated Sports Video Summarization Using Yolo [<https://doi.org/10.5565/rev/elcvia.1286>]. *Electronic Letters on Computer Vision and Image Analysis*, 20(1), 99-116. <https://doi.org/10.5565/rev/elcvia.1286>
- Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., ... Oliphant, T. E. (2020). Array programming with NumPy [<https://doi.org/10.1038/s41586-020-2649-2>]. *Nature*, 585(7825), 357-362. <https://doi.org/10.1038/s41586-020-2649-2>
- Hu, X., Zhao, J., Hui, Y., Li, S., & You, S. (2023). SiamHSFT: A Siamese Network-Based Tracker with Hierarchical Sparse Fusion and Transformer for UAV Tracking [<https://doi.org/10.3390/s23218666>]. *Sensors*, 23(21), 1-15. <https://doi.org/1424-8220>
- Jocher, G., & Qiu, J. (2024b). *Ultralytics YOLO11* [<https://github.com/ultralytics/ultralytics>] (11.0.0).
- Liu, L., Sun, Y., & Ge, X. (2025). A Hybrid Multi-Person Fall Detection Scheme Based on Optimized YOLO and ST-GCN [<http://dx.doi.org/10.9781/ijimai.2024.09.003>]. *International Journal of Interactive Multimedia and Artificial Intelligence*, 9(2), 26-38. <https://doi.org/10.9781/ijimai.2024.09.003>
- López Flórez, S., González-Briones, A., Hernández, G., Ramos, C., & de la Prieta, F. (2023). Automatic Cell Counting With YOLOv5: A Fluorescence Microscopy Approach [<https://>

- doi.org/10.9781/ijimai.2023.08.001]. *International Journal of Interactive Multimedia and Artificial Intelligence*, 8(3), 1-8. <https://doi.org/10.9781/ijimai.2023.08.001>
- Meinhardt, T., Kirillov, A., Leal-Taixe, L., & Feichtenhofer, C. (2022). Trackformer: Multi-object tracking with transformers [<https://doi.org/10.48550/arXiv.2101.02702>]. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 8844-8854. <https://doi.org/10.48550>
- Pereira, R., Carvalho, G., Garrote, L., & Nunes, U. (2022). Sort and Deep-SORT Based Multi-Object Tracking for Mobile Robotics: Evaluation with New Data Association Metrics [<https://doi.org/10.3390/app12031319>]. *Applied Sciences*, 12(3), 1-15. <https://doi.org/2076-3417>
- Peres Barbosa, J., Dalla Rosa, F., & Rieder, R. (2024). Pothole Detection Web App: an approach for detecting potholes in asphalt pavements using YOLO [<https://doi.org/10.5335/rbca.v16i3.15851>]. *Revista Brasileira de Computação Aplicada*, 16(3), 25-36. <https://doi.org/10.5335/rbca.v16i3.15851>
- Pérez-Aguilar, D., Pérez-Aguilar, J., Pérez-Aguilar, A., Risco-Ramos, R., & Malpica- Rodríguez, M. (2024). Electric substation inspection: YOLOv5 in hotspot detection through thermal imaging [<https://doi.org/10.17163/ings.n31.2024.04>]. *Ingenius, Revista de Ciencia y Tecnología*, 1(31), 43-54. <https://doi.org/10.17163/ings.n31.2024.04>
- Ren, Y., Li, Y., & Gao, X. (2024). An MRS-YOLO Model for High-Precision Waste Detection and Classification [<https://doi.org/10.3390/s24134339>]. *Sensors*, 24(13).
- Reyes Zambrano, J., & Moreira Santos, M. (2025). Computer Vision in the field of productive control and health in Farm Animals. *Revista de Tecnologías de la Informática y las Comunicaciones*, 9(1), 16-29. <https://doi.org/10.33936/isrtic.v9.i1.7156>
- Roboflow. (2024). *Supervision* [<https://roboflow.github.io/supervision/>].
- Rondón, C. N., Castro Casadiego, S., & Medina Delgado, B. (2021). Background subtraction and yolo algorithm: two methods for the detection of people in uncontrolled environments. *Revista Ingeniería Solidaria*, 17(1), 1-25. <https://doi.org/2357-6014.2021.01.06>
- Wang, Y., & Mariano, V. (2024). A Multi Object Tracking Framework Based on YOLOv8s and ByteTrack Algorithm [<https://doi.org/10.1109/ACCESS.2024.3450370>]. *IEEE Access*, 12, 120711-120719. <https://doi.org/10.1109/ACCESS.2024.3450370>

- Yang, B., Tang, M., Chen, S., Wang, G., Tan, Y., & Li, B. (2020b). A vehicle tracking algorithm combining detector and tracker [<https://doi.org/10.1186/s13640-020-00505-7>]. *J Image Video Proc.* 2020, 1(17), 1-10. <https://doi.org/10.1186/s13640-020-00505-7>
- Zhang, Y., Sun, P., Jiang, Y., Yu, D., Weng, F., Yuan, Z., Luo, P., Liu, W., & Wang, X. (2022). ByteTrack: Multi-Object Tracking by Associating Every Detection Box [https://doi.org/10.1007/978-3-031-19830-4_1]. *European Conference on Computer Vision*, 1(1), 1-21. <https://doi.org/10.1007>
- Zulko, Z. (2015). *MoviePy - Video editing with Python* [<https://github.com/Zulko/moviepy>].

Capítulo 4

Evaluación de motores OCR para el reconocimiento automático de placas vehiculares

Portada interior

Nombre del artículo: Evaluación de motores OCR para el reconocimiento automático de placas vehiculares usando YOLOv11

Autores: Manuel Antonio García-Herrera, Betania Hernández-Ocaña, José Hernández-Torruco, Oscar Chávez-Bosquez

Institución de adscripción: División Académica de Ciencias y Tecnologías de la Información, Universidad Juárez Autónoma de Tabasco

Resumen

El Reconocimiento Automático de Placas Vehiculares (ANPR) desempeña un papel importante en el transporte inteligente, la supervisión del tráfico y la seguridad pública. A pesar de los avances en la detección de placas, la etapa de Reconocimiento Óptico de Caracteres (OCR) sigue siendo un cuello de botella importante en implementaciones del mundo real debido a la variabilidad ambiental y la distorsión de los caracteres. Este estudio tiene como objetivo evaluar y comparar el rendimiento de tres motores OCR: Tesseract, EasyOCR y TrOCR, cuando se integran en un sistema ANPR basado en el modelo de detección de objetos YOLOv11. Para las pruebas, se compiló un conjunto de datos compuesto por 25 videos de tráfico en condiciones reales bajo diversas condiciones de iluminación y clima. YOLOv11 se utilizó para la localización de la placa vehicular, mientras que cada motor OCR se aplicó sobre las regiones detectadas. El rendimiento se evaluó utilizando métricas estándar: precisión, *recall* y tasa de error de caracteres (CER). Los resultados mostraron que YOLOv11 alcanzó un alto desempeño en detección (precisión: 0.9380, *recall*: 0.9950), mientras que TrOCR superó significativamente a los otros motores OCR con un CER de 0.4605. EasyOCR fue el siguiente con un CER de 0.6260, y Tesseract presentó la mayor tasa de error con 1.0271.

Palabras clave: ANPR, OCR, YOLOv11, Tesseract, EasyOCR, TrOCR.

Artículo publicado en los *proceedings* del INTERNATIONAL CONGRESS ON SMART MACHINES AND DATA ENGINEERING (SMaDE 2025): <https://eddsac.com/SMaDE/proceedings2025.pdf>

4.1. Introducción

El Reconocimiento Automático de Placas Vehiculares (ANPR, por sus siglas en inglés) puede considerarse una tecnología novedosa aplicada en diversos dominios, incluyendo la seguridad pública, los sistemas de transporte inteligente y la infraestructura urbana inteligente. Con la creciente necesidad de una identificación eficiente de vehículos en áreas como el cobro de peajes, el control de acceso y la vigilancia del tráfico, la importancia de sistemas ANPR robustos ha sido ampliamente estudiada (Bakhtan et al., 2016). Generalmente, el proceso ANPR se divide en dos etapas principales: (1) la localización y detección de la placa dentro de una imagen dada y (2) el reconocimiento de los caracteres alfanuméricos a partir de la región de la placa mediante técnicas de Reconocimiento Óptico de Caracteres (OCR).

En los últimos años, los avances en el aprendizaje profundo han mejorado significativamente la precisión tanto en las tareas de detección como de reconocimiento dentro de los sistemas ANPR. Las Redes Neuronales Convolucionales (CNN) y los *frameworks* de detección de objetos como la familia YOLO (*You Only Look Once*) se han convertido en estándares para la detección en tiempo real y de alta precisión. En particular, YOLOv11, un modelo de detección de objetos de última generación, ha sido adoptado en este estudio por su velocidad y fiabilidad para identificar regiones de placas vehiculares bajo diversas condiciones ambientales (Bochkovskiy et al., 2020a).

Una vez detectadas las placas, se aplican motores OCR para extraer los caracteres contenidos en ellas. Aunque se han propuesto e implementado varios modelos OCR con éxito, su desempeño puede variar significativamente dependiendo de la calidad de la imagen, las condiciones de iluminación, el ángulo de captura y la presencia de ruido u oclusión. Entre los motores OCR más utilizados, Tesseract sigue siendo una opción clásica de código abierto, mientras que EasyOCR introduce mejoras basadas en redes neuronales. Recientemente, se han propuesto modelos basados en transformers como TrOCR, que utilizan aprendizaje secuencial para mejorar la robustez en tareas de reconocimiento de texto (Li et al., 2021).

Este estudio se enfoca en evaluar y comparar Tesseract, EasyOCR y TrOCR cuando se integran en un sistema ANPR que utiliza YOLOv11 para la detección (Laroca et al., 2019). Se considera un escenario de prueba realista que incorpora variaciones en iluminación, desenfoque por movimiento, ángulo de visión y oclusión parcial de las placas. Se utilizan métricas objetivas

como precisión, *recall* y tasa de error de caracteres (CER) para cuantificar el desempeño de los OCR (R. Tavares y Silva, 2024).

4.2. Trabajo relacionado

Numerosos estudios han sido realizados para explorar, desarrollar y probar sistemas de Reconocimiento Automático de Placas Vehiculares (ANPR); la mayoría de ellos utilizan metodologías de aprendizaje profundo tanto para la detección como para el reconocimiento. Diversos trabajos han destacado los beneficios del uso de detectores basados en redes convolucionales y arquitecturas OCR neuronales para mejorar el rendimiento global del sistema (Zherzdev y Gruzdev, 2019). En particular, la división modular del ANPR en detección y reconocimiento de caracteres ha permitido optimizar y evaluar cada componente de forma independiente, contribuyendo a soluciones más robustas y escalables.

Para la detección de placas vehiculares, los modelos basados en YOLO han sido ampliamente adoptados debido a su capacidad de equilibrar la inferencia en tiempo real con una alta precisión de localización. Desde YOLOv4 hasta versiones más recientes como YOLOv7 y YOLOv11, las mejoras en las arquitecturas backbone, el diseño de anclas y las estrategias de entrenamiento han incrementado el rendimiento en diversas condiciones (C.-Y. Wang et al., 2022). Estos avances demuestran la idoneidad de YOLO para aplicaciones donde la velocidad y la precisión son críticas.

En el campo del OCR, motores clásicos como Tesseract han sido utilizados para la extracción de caracteres en placas vehiculares. EasyOCR introdujo mejoras basadas en redes convolucionales (CNN), incrementando la robustez bajo distorsiones moderadas. Más recientemente, modelos basados en transformers como TrOCR han emergido, utilizando arquitecturas preentrenadas y mecanismos de atención para lograr resultados de alto rendimiento en diversas tareas de reconocimiento de texto, incluyendo texto en escenas, texto impreso y escritura manual.

Investigaciones recientes han buscado permitir que los sistemas ANPR operen bajo restricciones de recursos computacionales, manteniendo alta precisión. Por ejemplo, en estudios como “Efficient Real-Time License Plate Recognition using Deep Learning on Edge Devices”, se proponen arquitecturas compactas capaces de operar en hardware de bajo consumo energético. Otros trabajos han optimizado versiones de YOLO para dispositivos con recursos limitados, logrando un

desempeño en tiempo real incluso bajo condiciones adversas como baja iluminación u oclusiones parciales.

Adicionalmente, se han explorado técnicas de aumento de datos para simular condiciones ambientales adversas, así como métodos de segmentación para corrección de perspectiva y análisis de robustez ante ataques adversariales. Estos enfoques han demostrado que la confiabilidad de los sistemas ANPR no depende únicamente de la precisión del modelo, sino también de su estabilidad en entornos reales complejos.

4.3. Metodología

4.3.1. Materiales

Con el objetivo de realizar un análisis comparativo del desempeño de motores OCR dentro de un sistema de Reconocimiento Automático de Placas Vehiculares (ANPR), se utilizaron diversos componentes y herramientas. Estos incluyen los siguientes *frameworks* de software y protocolos de evaluación:

- **Modelo de detección:** YOLOv11, una versión reciente de la arquitectura *You Only Look Once*, fue utilizada como detector principal de objetos para localizar placas vehiculares dentro de los fotogramas de video. Su alta velocidad de inferencia y su capacidad mejorada de extracción de características lo hacen especialmente adecuado para aplicaciones en tiempo real (Bochkovskiy et al., 2020b).
- **Motores OCR:** Se emplearon tres sistemas de reconocimiento óptico de caracteres:
 - Tesseract OCR, un motor clásico de código abierto desarrollado por Google, ampliamente utilizado en tareas de extracción de texto, incluyendo el reconocimiento de placas vehiculares (Anwar et al., 2022b).
 - EasyOCR, un sistema OCR basado en aprendizaje profundo que utiliza redes neuronales convolucionales y recurrentes, muestra mejores resultados en condiciones de imagen no ideales (Jaided AI, 2020).

- TrOCR, un modelo moderno basado en transformers que integra modelos de lenguaje preentrenados para mejorar la comprensión contextual del texto (Li et al., 2021).

- **Herramientas y métricas:** Se utilizó un conjunto de datos basado en video para simular condiciones reales de ANPR. La evaluación se realizó mediante métricas ampliamente aceptadas: precisión, *recall* y tasa de error de caracteres (CER), utilizadas en estudios previos para evaluar la calidad de reconocimiento en entornos ruidosos y dinámicos (D K et al., 2024).

4.3.2. Método

El sistema ANPR propuesto fue diseñado para procesar video de entrada y evaluar el desempeño de los motores OCR, como se ilustra en la Fig. 4.1. La metodología sigue un flujo de procesamiento por etapas:

1. Extracción de fotogramas: Se extrajeron frames individuales de secuencias de video continuas, simulando condiciones de tráfico en tiempo real. Esto permitió introducir variaciones controladas en calidad de imagen, desenfoque por movimiento y condiciones de iluminación.
2. Detección de placas vehiculares: Se utilizó YOLOv11 para identificar y localizar placas dentro de cada fotograma. La detección se realizó mediante cajas delimitadoras generadas por el modelo, las cuales fueron utilizadas para recortar las Regiones de Interés (RoI).
3. Extracción de RoI: Las regiones detectadas fueron recortadas del video original y preparadas para el OCR. Este paso incluyó preprocesamiento básico como redimensionamiento y normalización para asegurar consistencia entre motores OCR.
4. Aplicación de OCR: Cada región RoI fue procesada de manera independiente por los tres motores OCR: Tesseract, EasyOCR y TrOCR. Se registraron salidas a nivel de carácter y de placa completa para su posterior análisis.

Después de este flujo, se realizó una evaluación cuantitativa. El desempeño de detección se midió mediante precisión y *recall*, mientras que el reconocimiento de texto se evaluó mediante la

tasa de error de caracteres (CER), que cuantifica la distancia de edición entre la predicción y el texto real de la placa.

Todos los experimentos fueron ejecutados utilizando aceleración por GPU a través de Google Colab, una plataforma en la nube que permite ejecutar código Python con acceso a hardware acelerado (Google, 2025). Este entorno permitió garantizar condiciones de evaluación consistentes entre los motores OCR, asegurando una comparación justa.

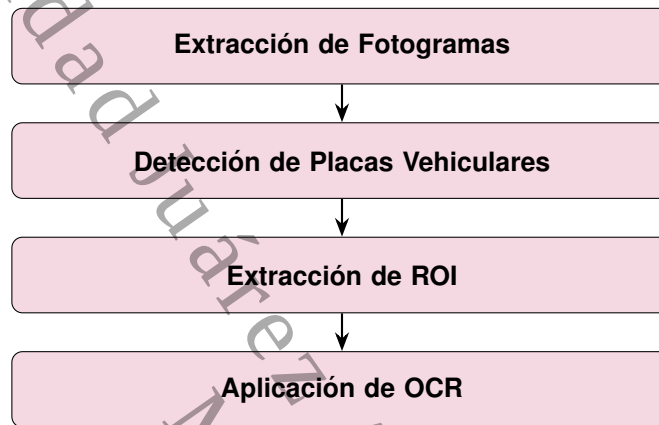


Figura 4.1. Pipeline simplificado del sistema ANPR mostrando las etapas principales de procesamiento.

4.4. Conjunto de datos

Se recopiló un conjunto de datos compuesto por 25 videos en formato MP4 durante dos días distintos dentro de la misma semana, con variaciones ambientales intencionales para asegurar diversidad y representatividad. Las grabaciones se realizaron en condiciones de clima soleado y nublado, en diferentes horarios entre las 07:00 y las 15:00 horas. Se prestó especial atención a las horas de mayor tráfico (08:00, 10:00 y 12:00) para capturar variaciones en iluminación y densidad vehicular.

Se registraron un total de 198 vehículos al pasar por el campo de visión de la cámara. Se incluyeron múltiples escenarios visuales, como condiciones de iluminación variable (sombras y reflejos), ángulos de visión oblicuos debido al movimiento de los vehículos y oclusiones parciales causadas por otros automóviles u objetos.

Estas variaciones permiten que el conjunto de datos represente condiciones reales de tráfico, proporcionando una base sólida para evaluar la efectividad del sistema ANPR.

Todos los videos fueron capturados desde una posición fija en la entrada principal del Campus Chontalpa de la Universidad Juárez Autónoma de Tabasco, en el estado de Tabasco, México. Esta ubicación fue seleccionada considerando la posición del sol en diferentes momentos del día para capturar variaciones de sombra y reflejo.

La Fig. 4.2 muestra el entorno de captura utilizado.



Figura 4.2. Entrada del Campus Chontalpa

4.4.1. Preprocesamiento de video

Todos los videos fueron estandarizados mediante un proceso de preprocesamiento. Los videos originales fueron recortados a una relación de aspecto 1:1 y redimensionados a 848×848 píxeles. Este paso se implementó para optimizar el uso de memoria, reducir la carga computacional y mejorar el rendimiento de los algoritmos de detección y seguimiento.

El proceso de preprocesamiento se muestra en la Fig. 4.3.

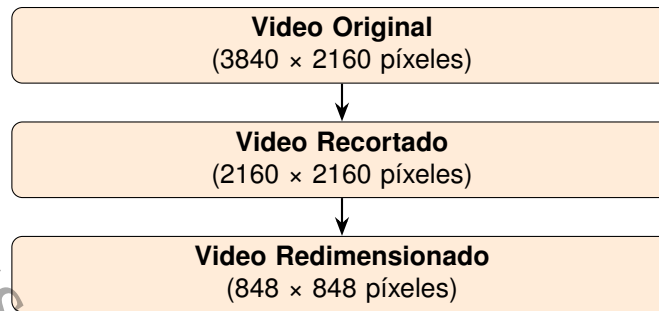


Figura 4.3. Ejemplo de redimensionamiento y preprocesamiento de video.

4.4.2. Registro y anotación de placas vehiculares

Como parte de la etapa de preprocesamiento, se realizó la anotación manual de las placas de los 198 vehículos. Esta tarea se llevó a cabo revisando cada video del conjunto de datos. Los vehículos fueron identificados al pasar por el campo de visión de la cámara, y sus placas fueron transcritas manualmente desde los fotogramas.

4.5. Diseño experimental

El diseño experimental fue estructurado cuidadosamente para evaluar el desempeño del sistema ANPR de manera integral. El proceso de evaluación se dividió en dos etapas principales: la detección de placas vehiculares y la evaluación de la precisión del Reconocimiento Óptico de Caracteres (OCR).

En primer lugar, las placas vehiculares fueron detectadas en cada fotograma del video utilizando el algoritmo de detección YOLO. Se seleccionó YOLOv11 debido a su rendimiento de última generación en tareas de detección en tiempo real, especialmente en objetos pequeños y complejos como las placas vehiculares.

La evaluación del modelo de detección se realizó utilizando dos métricas estándar: precisión y *recall*. Estas métricas permiten medir la capacidad del modelo para identificar correctamente las placas, minimizando al mismo tiempo las falsas detecciones.

Una vez localizadas las placas, los tres motores OCR (Tesseract, EasyOCR y TrOCR) fueron aplicados de manera independiente sobre las regiones recortadas. Las Tablas 4.1, 4.2 y 4.3 muestran los principales parámetros utilizados en cada uno de los motores OCR. Esto permitió

realizar un análisis comparativo del desempeño de cada método en la tarea de reconocimiento de caracteres. El flujo completo del experimento se resume en la Figura 4.4.

Tabla 4.1. Configuración de EasyOCR

Parámetro	Valor	Descripción
lang_list	['en']	Lista de idiomas
gpu	True	Uso de GPU si está disponible
recog_network	standard	Modelo de reconocimiento
detector	True	Activar detección de texto
recognizer	True	Activar reconocimiento

Tabla 4.2. Configuración de TrOCR

Parámetro	Valor	Descripción
trocr_processor	Requerido	Preprocesador de imagen
trocr_model	Requerido	Modelo de reconocimiento
device	'cuda'	Dispositivo de ejecución
batch_decode	True	Decodificación sin tokens especiales

Tabla 4.3. Configuración de Tesseract

Parámetro	Valor	Descripción
OEM	OEM_DEFAULT	Selección automática del motor
PSM	PSM_AUTO (3)	Segmentación automática de texto
Language Data	*.traineddata	Diccionario de idioma
Preprocesamiento	Activado	Binarización y corrección de inclinación

4.5.1. Métricas de evaluación para detección

Precisión (*Precision*) se define como la proporción de placas vehiculares correctamente detectadas (verdaderos positivos) respecto al total de detecciones realizadas (verdaderos positivos más falsos positivos). Indica qué tan precisas son las predicciones positivas del modelo de detección (Sporici et al., 2020).

$$\text{Precisión} = \frac{TP}{TP + FP} \quad (4.1)$$

donde TP representa los verdaderos positivos y FP los falsos positivos.

Recall se define como la proporción de placas correctamente detectadas respecto al total

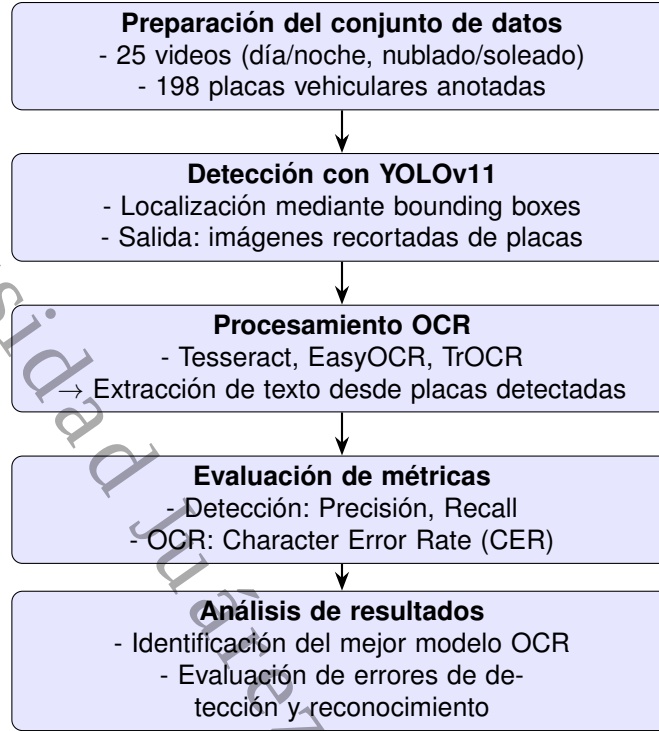


Figura 4.4. Flujo del diseño experimental para detección y reconocimiento de placas vehiculares.

de placas reales presentes en el conjunto de datos (verdaderos positivos más falsos negativos) (Spruck et al., 2022).

$$\text{Recall} = \frac{TP}{TP + FN} \quad (4.2)$$

donde FN representa los falsos negativos.

4.5.2. Métrica de evaluación para OCR

Para la evaluación del OCR se utilizó la Tasa de Error de Caracteres (CER), la cual mide la proporción de errores a nivel de carácter en comparación con el texto real. Esta métrica considera sustituciones, inserciones y eliminaciones (Drobac y Lindén, 2020). El CER se calcula como:

$$\text{CER} = \frac{S + D + I}{N} \quad (4.3)$$

donde:

- S : sustituciones,

- D : eliminaciones,
- I : inserciones,
- N : número total de caracteres del texto de referencia.

Un valor de CER más bajo indica un mejor desempeño del sistema OCR.

En resumen, la detección y el reconocimiento fueron evaluados de manera independiente para diferenciar su contribución al rendimiento global del sistema ANPR. La precisión y el *recall* permiten analizar la calidad de la detección, mientras que el CER mide la efectividad del reconocimiento de caracteres bajo condiciones reales.

4.6. Resultados y evaluación

4.6.1. Detección de objetos

Para el proceso de evaluación y la obtención de las métricas, se presenta la matriz de confusión en la Tabla 4.4. Esta matriz ilustra la relación entre las predicciones del modelo y los valores reales en la tarea de detección.

Tabla 4.4. Matriz de confusión para la detección de placas vehiculares

Real	Predicción	
	Positivo (1)	Negativo (0)
Positivo (1)	197 Verdadero Positivo (TP)	1 Falso Negativo (FN)
Negativo (0)	13 Falso Positivo (FP)	—

El desempeño del modelo YOLOv11 en la detección de placas vehiculares fue evaluado utilizando métricas estándar. Los resultados muestran que, de un total de 198 placas reales, el modelo detectó correctamente 197 y omitió 1. Además, el modelo generó 13 falsas detecciones, correspondientes a elementos como logotipos de vehículos o calcomanías.

Con base en estos valores, se calcularon las métricas de desempeño mostradas en la Tabla 4.5.

Tabla 4.5. Métricas de desempeño de detección

Métrica	Valor
Precisión	0.9380
Recall	0.9950

La precisión obtenida fue de 0.9380, lo que indica que el 93.8 % de las detecciones fueron correctas. El *recall* fue de 0.9950, lo que demuestra que casi todas las placas vehiculares presentes fueron detectadas correctamente.

Estos resultados indican que el modelo YOLOv11 presenta una alta robustez y fiabilidad en la tarea de detección, incluso bajo condiciones complejas, como variaciones de iluminación y oclusiones parciales.

4.6.2. Desempeño del OCR

La segunda etapa del sistema consistió en aplicar los motores OCR a las regiones de interés (RoI) previamente extraídas de las placas vehiculares detectadas. Como ejemplo ilustrativo, la Tabla 4.6 muestra los resultados de un experimento individual.

Tabla 4.6. Ejemplo de salidas OCR para un experimento individual

Campo	Valor
Id de experimento	21
Número de placa	1
Texto real	XX-####-X
Salida Tesseract	...
Salida EasyOCR	XX.####.
Salida TrOCR	XX.####.X

El desempeño de cada motor OCR fue evaluado mediante la Tasa de Error de Caracteres (CER), la cual mide la proporción de errores a nivel de carácter respecto al texto real. Los valores de CER obtenidos se presentan en la Tabla 4.7.



Figura 4.5. Ejemplo de salida de EasyOCR para un experimento individual

Tabla 4.7. Desempeño OCR. Tasa de error de caracteres (CER)

Motor OCR	CER
Tesseract	1.0271
EasyOCR	0.6260
TrOCR	0.4605

Entre los tres motores evaluados, TrOCR obtuvo el menor CER (0.4605), lo que indica el mejor desempeño en reconocimiento de caracteres. EasyOCR presentó un rendimiento intermedio, mientras que Tesseract obtuvo el mayor índice de error. Estos resultados sugieren que los modelos basados en transformers, como TrOCR, ofrecen una mayor robustez frente a condiciones difíciles como desenfoque, baja iluminación y oclusiones parciales.

Se observaron patrones recurrentes de error en todos los modelos, especialmente confusiones entre caracteres visualmente similares, como la letra 'O' y el número '0', o la letra 'B' y el número '8'. Además, se identificaron errores de inserción y eliminación en casos donde los caracteres estaban parcialmente ocultos o distorsionados.

En términos de tiempo de ejecución, se observaron diferencias importantes entre los motores OCR incluso utilizando aceleración GPU en Google Colab. EasyOCR tardó entre 3 y 5 minutos por video; TrOCR entre 4 y 6 minutos; mientras que Tesseract fue el más rápido, con tiempos de

1 a 2 minutos.

En resumen, aunque YOLOv11 proporciona una detección altamente confiable, el rendimiento global del sistema ANPR depende en gran medida del motor OCR utilizado.

4.7. Discusión

Los resultados obtenidos en este estudio resaltan la importancia de la selección del motor OCR dentro de sistemas de Reconocimiento Automático de Placas Vehiculares (ANPR). Aunque el modelo YOLOv11 mostró un desempeño sólido en la etapa de detección, con una precisión de 0.9380 y un recall de 0.9950, la variabilidad en el rendimiento del sistema completo estuvo principalmente determinada por la etapa de reconocimiento de texto.

TrOCR obtuvo el mejor desempeño en términos de la *Character Error Rate* (CER), lo que indica una mayor capacidad para interpretar imágenes degradadas o ambiguas. Este resultado se atribuye a su arquitectura basada en transformers, la cual permite modelar dependencias contextuales entre caracteres mediante mecanismos de atención. Esta propiedad resulta especialmente útil en escenarios con distorsión, oclusión parcial o ruido visual.

Por otro lado, EasyOCR presentó un rendimiento intermedio, con errores frecuentes en caracteres visualmente similares. Tesseract mostró la menor robustez ante condiciones adversas, lo que resultó en el mayor CER entre los tres motores evaluados. Esto sugiere que los métodos OCR tradicionales pueden ser insuficientes en entornos dinámicos donde la calidad de imagen no está garantizada.

Además, se observó que factores externos como la iluminación variable, el desenfoque por movimiento y los ángulos de captura influyen significativamente en el desempeño del sistema. Estas condiciones son comunes en entornos reales de monitoreo vehicular, lo que representa un desafío importante para la robustez de los sistemas ANPR.

En general, los resultados indican que la integración de modelos modernos de OCR es esencial para mejorar el desempeño global del sistema. Sin embargo, incluso los modelos más avanzados aún presentan limitaciones frente a condiciones altamente degradadas, lo que abre la puerta a futuras mejoras en adaptabilidad y robustez.

4.8. Conclusiones

En este estudio se desarrolló y evaluó un sistema ANPR utilizando YOLOv11 para la detección de placas y tres motores OCR (Tesseract, EasyOCR y TrOCR) para la extracción de texto.

El modelo YOLOv11 mostró un desempeño confiable en la detección de placas vehiculares, alcanzando un *recall* de 0.9950 y una precisión de 0.9380, lo que confirma su efectividad en escenarios con variabilidad de iluminación y oclusión parcial.

Sin embargo, el desempeño global del sistema dependió fuertemente del motor OCR utilizado. TrOCR obtuvo el mejor resultado con un CER de 0.4605, seguido de EasyOCR, mientras que Tesseract presentó el peor desempeño con un CER de 1.0271. Esto evidencia la superioridad de los modelos basados en transformers para tareas de reconocimiento de texto en condiciones complejas.

A pesar de estas mejoras, el sistema aún enfrenta limitaciones relacionadas con condiciones reales, como desenfoque por movimiento, baja iluminación y oclusiones parciales. Por lo tanto, aunque los modelos de detección actuales son altamente efectivos, la etapa de OCR continúa siendo el principal cuello de botella en los sistemas ANPR.

En trabajos futuros, se recomienda explorar técnicas de mejora de imagen, modelos OCR más robustos y estrategias de entrenamiento específicas para dominios de placas vehiculares, con el fin de aumentar la precisión y estabilidad del sistema en entornos reales.

Referencias del capítulo

- Anwar, N., Khan, T., & Mollah, A. F. (2022b). Text Detection from Scene and Born Images: How Good is Tesseract? [https://doi.org/10.1007/978-981-19-1324-2_13]. En A. K. S. Pundir, N. Yadav, H. Sharma & S. Das (Eds.), *Recent Trends in Communication and Intelligent Systems* (pp. 115-122). Springer Nature Singapore. https://doi.org/10.1007/978-981-19-1324-2_13
- Bakhtan, M., Abdullah, M., & Rahman, A. (2016). A review on License Plate Recognition system algorithms [<https://doi.org/10.1109/ICICTM.2016.7890782>]. *Proceedings of the Interna-*

- tional Conference on Information and Communication Technology Management (ICICTM)*, 84-89. <https://doi.org/10.1109/ICICTM.2016.7890782>
- Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020a). YOLOv4: Optimal Speed and Accuracy of Object Detection [<https://arxiv.org/abs/2004.10934>]. *arXiv preprint arXiv:2004.10934*.
- Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020b). YOLOv4: Optimal Speed and Accuracy of Object Detection [<https://arxiv.org/abs/2004.10934>]. *arXiv preprint arXiv:2004.10934*.
- D K, T., James, J., Gopinath, D. P., & Ashraf K, M. (2024). Advocating Character Error Rate for Multilingual ASR Evaluation [<https://doi.org/10.48550/arXiv.2410.07400>]. *arXiv preprint arXiv:2410.07400*. <https://doi.org/10.48550/arXiv.2410.07400>
- Drobac, S., & Lindén, K. (2020). Optical character recognition with neural networks and post-correction with finite state methods [<https://doi.org/10.1007/s10032-020-00359-9>]. *International Journal on Document Analysis and Recognition (IJ DAR)*, 23, 279-295. <https://doi.org/10.1007/s10032-020-00359-9>
- Google. (2025). *Colaboratory Documentation* [<https://cloud.google.com/colab/docs?hl=es-419>].
- Jaied AI. (2020). *EasyOCR*. Consultado el 2 de octubre de 2025, desde <https://github.com/JaiedAI/EasyOCR>
- Laroca, R., Zanlorensi, L. A., Gonçalves, G. R., Todt, E., Schwartz, W. R., & Menotti, D. (2019). An Efficient and Layout-Independent Automatic License Plate Recognition System Based on the YOLO Detector [<https://arxiv.org/abs/1909.01754>]. *arXiv preprint arXiv:1909.01754*.
- Li, M., Zhang, Y., Li, L., Zhou, L., Yang, N., Zhou, M., & Wei, F. (2021). TrOCR: Transformer-based Optical Character Recognition with Pre-trained Models [<https://arxiv.org/abs/2109.10282>]. *arXiv preprint arXiv:2109.10282*.
- Sporici, D., Cuşnir, E., & Boiangiu, C.-A. (2020). Improving the Accuracy of Tesseract 4.0 OCR Engine Using Convolution-Based Preprocessing [<https://www.mdpi.com/2073-8994/12/5/715>]. *Symmetry*, 12(5), 715. <https://doi.org/10.3390/sym12050715>
- Spruck, A., Hawesch, M., Maier, A., Riess, C., Seiler, J., & Kaup, A. (2022). 3D Rendering Framework for Data Augmentation in Optical Character Recognition [<https://arxiv.org/abs/2209.14970>]. *arXiv preprint arXiv:2209.14970*.

- Tavares, R., & Silva, L. (2024). Comparison of Image Preprocessing Techniques for Vehicle License Plate Recognition Using OCR: Performance and Accuracy Evaluation [<https://arxiv.org/abs/2410.13622>]. *arXiv preprint arXiv:2410.13622*.
- Wang, C.-Y., Bochkovskiy, A., & Liao, H.-Y. M. (2022). YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors [<https://arxiv.org/abs/2207.02696>]. *arXiv preprint arXiv:2207.02696*.
- Zherzdev, S., & Gruzdev, A. (2019). Deep Learning for Vehicle License Plate Recognition: A Survey [<https://doi.org/10.1109/ICCS162669.2024.10799401>]. *Pattern Recognition and Image Analysis*, 29(3), 606-617. <https://doi.org/10.1134/S1054661819030124>

Capítulo 5

Ajuste fino de TrOCR para el reconocimiento automático de placas vehiculares

Portada interior

Nombre del artículo: Ajuste fino de TrOCR para el reconocimiento automático de placas vehiculares

Autores: Manuel Antonio García-Herrera, Betania Hernández-Ocaña, José Hernández-Torruco, Oscar Chávez-Bosquez

Institución de adscripción: División Académica de Ciencias y Tecnologías de la Información, Universidad Juárez Autónoma de Tabasco

Resumen

Los sistemas de reconocimiento automático de placas vehiculares (ANPR, por sus siglas en inglés) generalmente involucran dos etapas: la localización de la placa en una imagen y el reconocimiento de sus caracteres mediante técnicas de reconocimiento óptico de caracteres (OCR). Aunque el aprendizaje profundo ha mejorado significativamente la precisión de detección, la etapa OCR continúa siendo un problema abierto, especialmente bajo condiciones complejas como desenfoque por movimiento, baja iluminación u oclusión parcial. Este trabajo evalúa y compara tres motores OCR: Tesseract, EasyOCR y TrOCR, utilizando YOLOv11 para la detección en un entorno real dentro de un campus universitario. La principal contribución consiste en el ajuste fino de TrOCR utilizando un conjunto de datos personalizado de 651 imágenes de placas, diseñado para capturar patrones visuales específicos del dominio y ruido característico. El desempeño se cuantifica utilizando métricas de detección y exactitud de reconocimiento (Character Error Rate, CER). YOLOv11 mostró un alto desempeño de detección, mientras que TrOCR superó a los otros motores (CER: 0.461) frente a EasyOCR (0.626) y Tesseract (1.027). Después del ajuste fino, el CER de TrOCR mejoró aproximadamente un 44 % hasta 0.258, confirmando que el reentrenamiento de un modelo robusto puede incrementar la precisión de reconocimiento de caracteres en escenarios reales complejos.

Palabras clave: Visión por computadora, Aprendizaje profundo, Arquitectura Transformer.

Artículo publicado en la revista International Journal of Combinatorial Optimization Problems and Informatics (IJCOPi): <https://doi.org/10.61467/2007.1558.2026.v17i4.1294>

5.1. Introducción

El reconocimiento automático de placas vehiculares (ANPR) representa una tecnología innovadora aplicable a diversos sectores, tales como seguridad pública, sistemas inteligentes de transporte e infraestructura urbana inteligente. A medida que crece la demanda de identificación eficiente de vehículos en áreas como cobro de peaje, control de acceso y monitoreo del tráfico, el desarrollo de sistemas ANPR confiables ha sido objeto de una extensa investigación (Hamood Bakhtan et al., 2016).

Generalmente, el flujo de trabajo ANPR consta de dos fases principales: la localización y detección de la placa vehicular dentro de una imagen, y el reconocimiento de los caracteres alfanuméricos desde la región extraída de la placa utilizando técnicas OCR.

Recientemente, los avances en Aprendizaje profundo han mejorado sustancialmente la precisión tanto de las tareas de detección como de reconocimiento dentro de los flujos de trabajo ANPR. Las Redes Neuronales Convolucionales (CNN) y los marcos de detección de objetos, como la familia YOLO (*You Only Look Once*), son actualmente estándares para lograr la detección de placas en tiempo real y con alta precisión (Laroca et al., 2021). Específicamente, este estudio utiliza YOLOv11, un modelo de detección de objetos de última generación, seleccionado por su velocidad y confiabilidad para identificar regiones de placas bajo diversas condiciones ambientales (Jocher y Qiu, 2024a).

Después de la detección, se emplean motores OCR para extraer los caracteres de las placas. Aunque varios modelos de OCR han sido propuestos e implementados exitosamente, su desempeño puede variar según la calidad de entrada, la iluminación, los ángulos de visión y la presencia de ruido u oclusiones. Entre los motores OCR más utilizados, Tesseract continúa siendo una opción tradicional de código abierto, mientras que EasyOCR introduce mejoras basadas en redes neuronales. Más recientemente, modelos basados en Transformers como TrOCR han sido propuestos utilizando aprendizaje secuencia a secuencia para incrementar la robustez en tareas de reconocimiento de texto (Li et al., 2023). Sin embargo, el desempeño de modelos OCR preentrenados suele estar limitado cuando se aplican a datos como placas vehiculares, donde los patrones visuales y las características de ruido difieren significativamente de los encontrados en conjuntos de texto genéricos.

Este estudio se enfoca en evaluar y comparar Tesseract, EasyOCR y TrOCR integrados dentro de un sistema ANPR que utiliza YOLOv11 para la detección en un escenario real en uno de nuestros campus universitarios. Adicionalmente, el propósito de este trabajo es incorporar y evaluar un modelo TrOCR ajustado finamente dentro de la metodología ANPR propuesta, entrenado con un conjunto de datos personalizado creado específicamente para dicho propósito. Las métricas de evaluación utilizadas incluyen precisión, *recall* y tasa de error de caracteres (CER) (R. A. Tavares, 2024). Específicamente, las contribuciones de este trabajo incluyen:

- Diseño de un caso de prueba ANPR en condiciones ambientales no controladas en la entrada de un campus universitario.
- Análisis comparativo entre las versiones base de Tesseract, EasyOCR y TrOCR.
- Implementación y evaluación de una estrategia de ajuste fino específica para el dominio sobre TrOCR.
- Incorporación de una exploración estadística (media, mediana, percentiles y categorización de fallos) para medir el desempeño más allá de métricas promedio globales.
- Análisis empírico de errores de reconocimiento bajo condiciones complejas reales (desenfoco por movimiento, baja iluminación, oclusión y confusión de caracteres).

5.2. Trabajos relacionados

Numerosos estudios han explorado sistemas ANPR utilizando predominantemente metodologías de aprendizaje profundo que se benefician de tuberías modulares de detección y reconocimiento (Zherzdev y Gruzdev, 2018). Para la detección, los modelos basados en YOLO son ampliamente favorecidos debido a su inferencia en tiempo real y precisión de localización, con mejoras en arquitecturas *backbone* y estrategias de entrenamiento que incrementan el desempeño entre versiones (C. Y. Wang et al., 2022).

En el campo del OCR, los enfoques han evolucionado desde motores clásicos como Tesseract (Anwar et al., 2022a) hacia soluciones basadas en CNN como EasyOCR (Dhyani y Kumar, 2023),

y recientemente hacia modelos basados en Transformers como TrOCR, los cuales aprovechan mecanismos de atención para alcanzar resultados de estado del arte en múltiples *benchmarks*.

Investigaciones recientes se enfocan en optimizar ANPR para entornos con recursos limitados, manteniendo alta precisión. Por ejemplo, la arquitectura *Light-Edge* fue validada sobre hardware de bajo consumo y demostró robustez frente a vistas oblicuas, desenfoque y reflejos (Sonnara et al., 2025). De manera similar, estrategias optimizadas de entrenamiento sobre YOLOv8 han permitido desempeño confiable bajo baja iluminación y oclusión en dispositivos *edge* (Satya et al., 2025).

5.3. Conjunto de datos personalizado

Se recolectó un conjunto de videos durante dos días en la misma semana, introduciendo deliberadamente diversas condiciones ambientales para garantizar la representatividad. Las sesiones de grabación ocurrieron tanto bajo cielo nublado como soleado, entre las 07:00 y 15:00 horas.

Se prestó especial atención a los horarios pico de tráfico (específicamente 08:00, 10:00 y 12:00) para capturar variaciones naturales de densidad vehicular e iluminación. Todo el material fue grabado desde un punto fijo localizado al lado izquierdo de la entrada principal de la Universidad Juárez Autónoma de Tabasco, Campus Chontalpa, Tabasco, México.

5.3.1. Conjunto de datos de vehículos (video)

Se compiló un total de 25 videos MP4, donde 198 vehículos fueron registrados atravesando el campo visual de la cámara. El conjunto priorizó capturar placas bajo una amplia gama de escenarios visuales, incluyendo reflejos, sombras, ángulos oblicuos y oclusiones parciales. Todos los videos fueron estandarizados mediante una tubería de preprocesamiento. Los videos originales fueron recortados a una relación de aspecto 1:1 y redimensionados a 848×848 píxeles para optimizar el uso de memoria y reducir la carga computacional. La Figura 5.1 muestra los pasos de preprocesamiento de video.

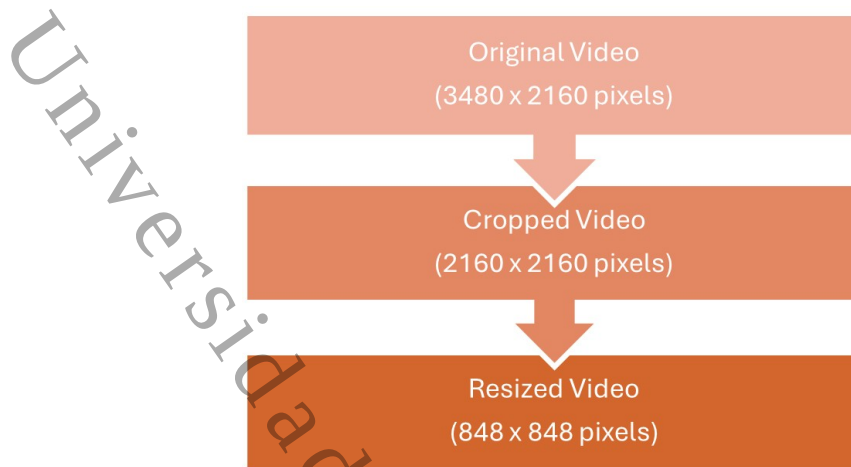


Figura 5.1. Pasos de preprocesamiento de video.

5.3.2. Conjunto de datos de placas (imagen)

Un total de 651 imágenes de placas fue utilizado para el ajuste fino. Se incluyeron entre 2 y 4 instancias de cada placa del conjunto de vehículos, capturando variaciones en distancia, ángulo e iluminación. El conjunto final fue dividido en entrenamiento (90 %) y prueba (10 %). Todas las imágenes fueron redimensionadas y convertidas a RGB. La Figura 5.2 muestra los pasos de preprocesamiento de imágenes. La Figura 5.3 muestra ejemplos de placas vehiculares estandarizadas.

La estructura del conjunto de datos fue organizada de manera que cada nombre de archivo correspondiera a una etiqueta textual dentro del archivo de anotaciones, como se ejemplifica en la Tabla 5.1.

Tabla 5.1. Nomenclatura de nombres de archivo y anotaciones

Imagen	Anotación
001.jpg	XXX-###-X
002.jpg	XXX-###-X
003.jpg	XXX-###-X
004.jpg	XXX-###-X
...	...
651.jpg	XXX-###-X

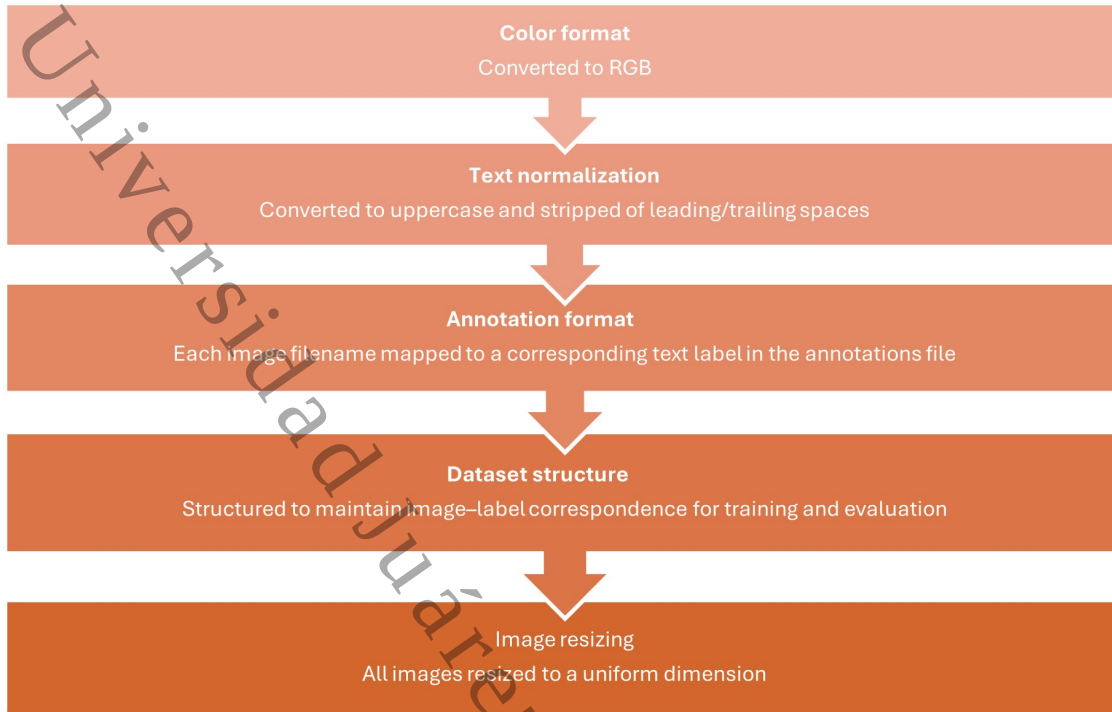


Figura 5.2. Pasos de procesamiento de imágenes.



Figura 5.3. Ejemplos de placas vehiculares estandarizadas.

5.4. Evaluación base de OCR

La configuración experimental fue estructurada para evaluar el desempeño de los modelos OCR de manera integral. El proceso de evaluación fue dividido en dos etapas principales: detección de placas y reconocimiento óptico de caracteres.

Primero, las placas fueron detectadas en cada cuadro de video utilizando YOLOv11. Posteriormente, Tesseract, EasyOCR y TrOCR fueron aplicados independientemente sobre las regiones recortadas. Las tablas 2, 3 y 4 muestran los principales parámetros y valores de los 3 motores OCR (Tablas 5.2, 5.3 y 5.4). La Figura 5.4 muestra el flujo de trabajo del análisis comparativo.

Tabla 5.2. Configuración de Tesseract

Parámetro	Valor	Descripción
OEM	OEM DEFAULT	Selección automática del motor
PSM	PSM AUTO (3)	Distribución automática del texto
Language Data	*.traineddata	Reglas básicas del diccionario
Preprocessing	Enabled	Binarización + deskew

Tabla 5.3. Configuración de EasyOCR

Parámetro	Valor	Descripción
lang_list	['en']	Lista de idiomas
gpu	True	Uso de GPU
recog_network	standard	Modelo OCR
detector	True	Habilitar detección
recognizer	True	Habilitar reconocimiento

5.4.1. Resultados de detección de placas

Para medir el rendimiento del modelo de detección de objetos YOLOv11, se presenta una matriz de confusión en la Tabla 5.5. Esta matriz ilustra los resultados de la tarea de detección de matrículas.

El rendimiento del modelo YOLOv11 en la detección de matrículas en fotogramas de vídeo se evaluó mediante métricas estándar. Según la Tabla 5, los resultados muestran que el modelo detectó 197 de las 198 matrículas y no detectó solo una. Además, el modelo detectó 13 matrículas falsas, por ejemplo, marcas de coches o pegatinas en el parachoques de algunos vehículos. La Figura 5.5 muestra un ejemplo de falso positivo (izquierda) y el caso de falso negativo.

En función de estos valores, las métricas de rendimiento de detección se presentan en la Tabla 5.6.

Tabla 5.4. Configuración de TrOCR

Parámetro	Valor	Descripción
trocr_processor	Required	Preprocesador de imágenes
trocr_model	Required	Modelo OCR
device	cuda	Dispositivo de procesamiento
batch_decode	True	Omitir tokens especiales

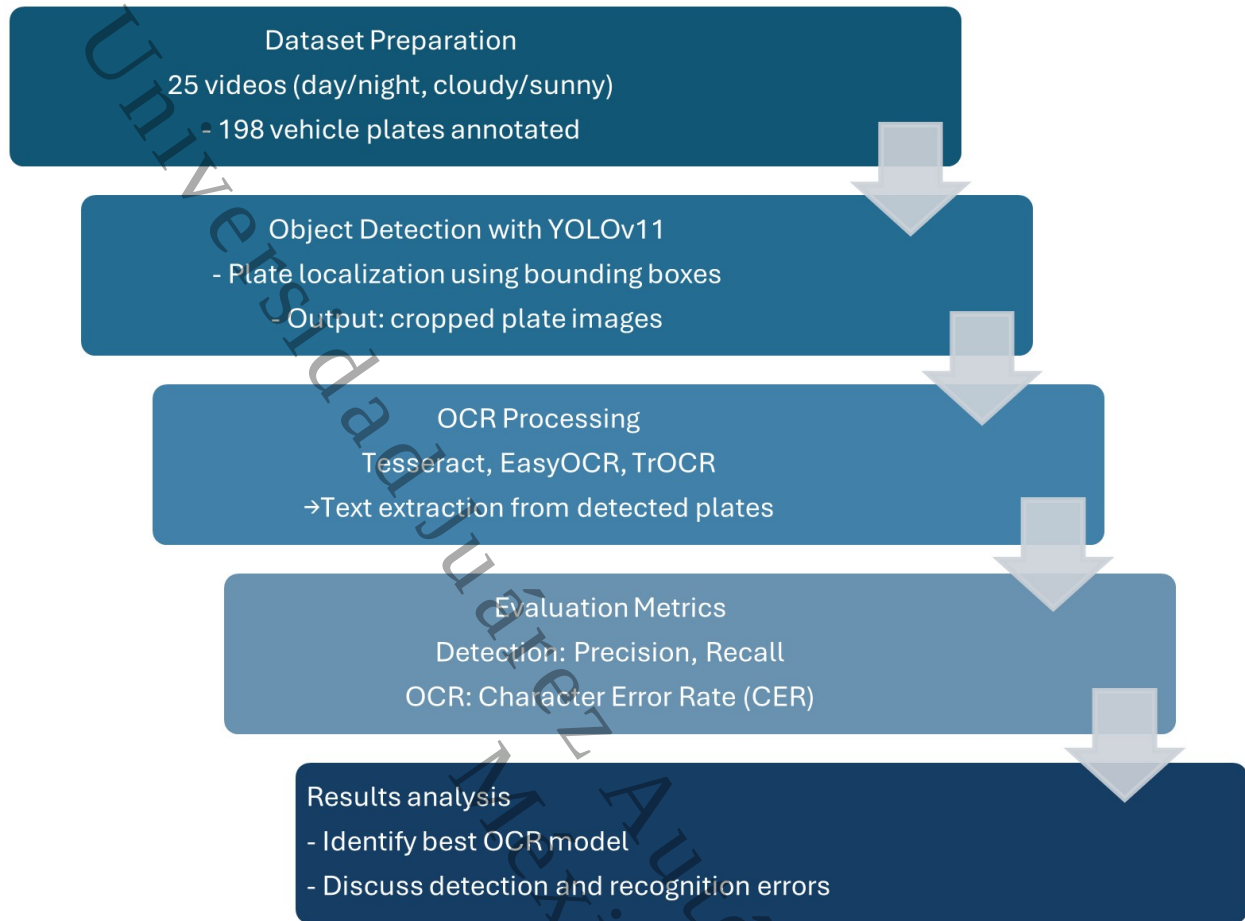


Figura 5.4. Flujo de trabajo del análisis comparativo.

5.4.2. Comparación base ANPR

El rendimiento se midió utilizando la tasa de error de caracteres (CER), que evalúa la precisión del texto reconocido midiendo el número de errores a nivel de carácter en relación con el número total de caracteres en la verdad fundamental. CER tiene en cuenta las inserciones, eliminaciones y sustituciones de caracteres. La tabla 5.7 muestra el Ejemplo de salidas OCR para tres experimentos.

Tabla 5.5. Matriz de confusión para detección de placas

		Predicción	
		Positivo (1)	Negativo (0)
Real	Positivo (1)	197 (TP)	1 (FN)
	Negativo (0)	13 (FP)	–

Tabla 5.6. Métricas de desempeño para detección de placas

Métrica	Valor
Precisión	0.9380
Recall	0.9950



Figura 5.5. Casos de falso positivo (derecha) y falso negativo (izquierda).

donde:

- S es el número de sustituciones,
- D es el número de eliminaciones,
- I es el número de inserciones,
- N es el número total de caracteres.

Tabla 5.7. Ejemplo de salidas OCR para tres experimentos

Video/Imagen	Placa real	Tesseract	CER	EasyOCR	CER	TrOCR
21.mp4 / 377.jpg	XX-####-X	...	1.000	XX.####.	0.333	XX.####.X
15.mp4 / 141.jpg	XXX-###-X	...	1.000	XXX.###.#	0.444	XXXXX.###.#
16.mp4 / 182.jpg	XXX-###-X	...	1.000	XXX.###.	0.667	XXX.###.#

Se comparó el número de matrícula real con los resultados producidos por cada motor OCR para evaluar su precisión a la hora de aislar y reconocer la secuencia alfanumérica correcta que se muestra en la Figura 5.6.

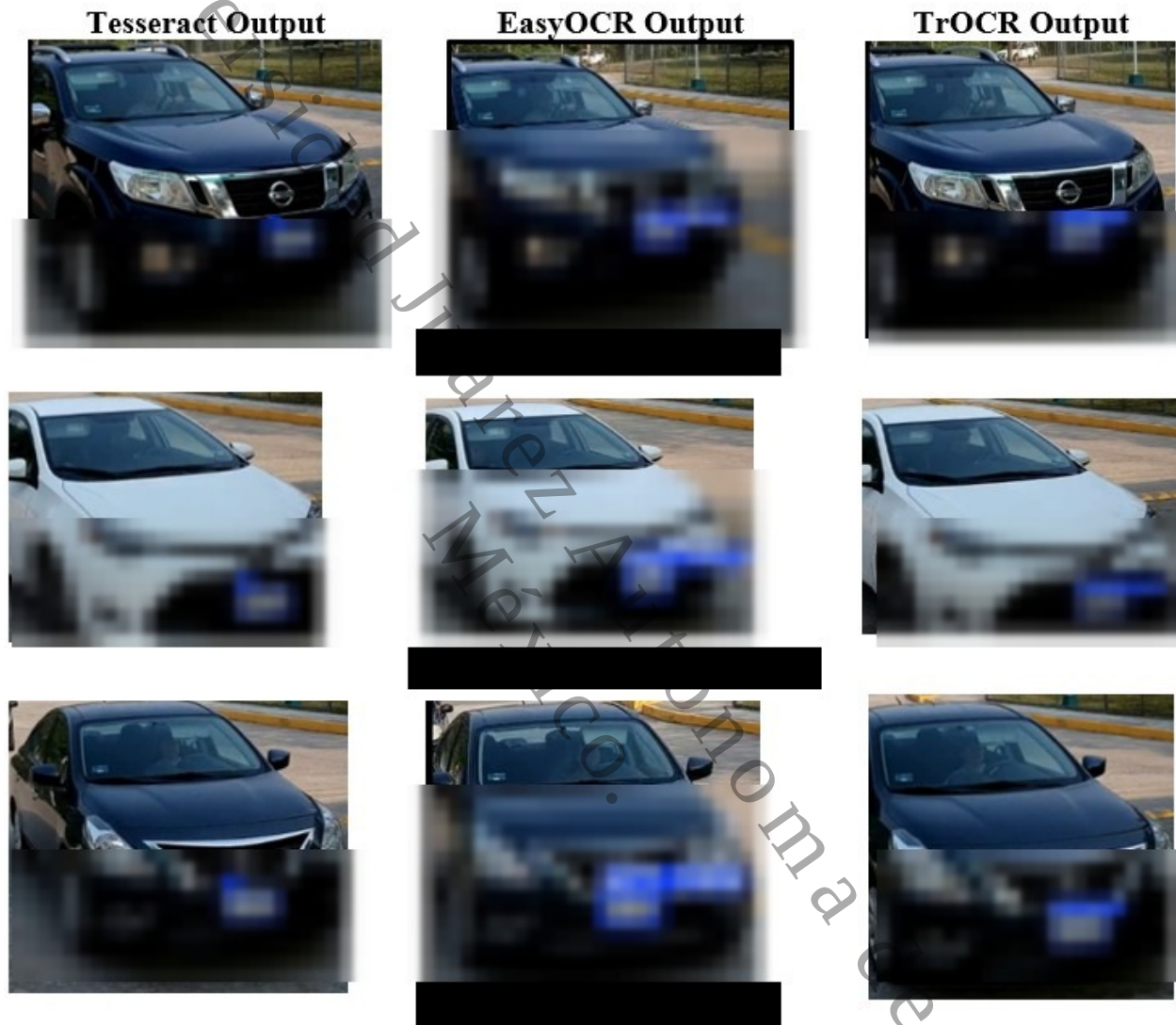


Figura 5.6. Ejemplos de salidas OCR para tres experimentos.

Los valores CER obtenidos para cada motor OCR se presentan en la Tabla 5.8.

5.5. Reentrenamiento de TrOCR

TrOCR, un moderno modelo OCR basado en Transformers que incorpora modelos de lenguaje preentrenados para una mejor comprensión contextual (Ströbel et al., 2022), fue utilizado en su

Tabla 5.8. Desempeño OCR (CER)

Motor OCR	CER
Tesseract	1.0271
EasyOCR	0.6260
TrOCR	0.4605

configuración base para realizar un ajuste fino (*fine tuning*) bajo condiciones reales de ANPR.

5.5.1. Estrategia de ajuste fino

La arquitectura base `microsoft/trocr-base-printed` fue reentrenada. La estructura codificador-decodificador fue preservada y únicamente los parámetros relevantes para la generación de secuencias fueron ajustados finamente. El modelo preentrenado se encuentra disponible bajo licencia libre en: <https://huggingface.co/microsoft/trocr-base-printed>

Los valores de píxeles de cada imagen del conjunto de datos fueron extraídos utilizando el procesador TrOCR, mientras que las anotaciones textuales fueron tokenizadas y rellenadas/truncadas a una longitud fija para el modelado secuencial utilizando el tokenizador TrOCR. El ajuste fino se realizó durante 5 épocas utilizando un tamaño de lote de 4, una tasa de aprendizaje de 5×10^{-5} , weight decay de 0.01 y 50 pasos de warmup. Las actualizaciones de gradiente fueron calculadas utilizando el optimizador AdamW y entrenamiento de precisión mixta (fp16) cuando existían recursos GPU disponibles. La tabla 5.9 muestra todos los parámetros de reentrenamiento.

Tabla 5.9. Parámetros de reentrenamiento de TrOCR

Parámetro	Valor / Descripción
Número de épocas	5
Batch size	4 (por dispositivo)
Learning rate	5×10^{-5}
Weight decay	0.01
Warmup steps	50
Optimizer	AdamW
Entrenamiento de precisión mixta	Activado (fp16)
Logging steps	50
Save steps	200
Máximo de checkpoints	2

El reentrenamiento fue ejecutado en un sistema con GPU habilitada mediante Google Colab utilizando PyTorch y Hugging Face Transformers, garantizando eficiencia computacional y reproducibilidad de resultados. La Figura 5.7 muestra el flujo de trabajo para la evaluación del ajuste fino de TrOCR

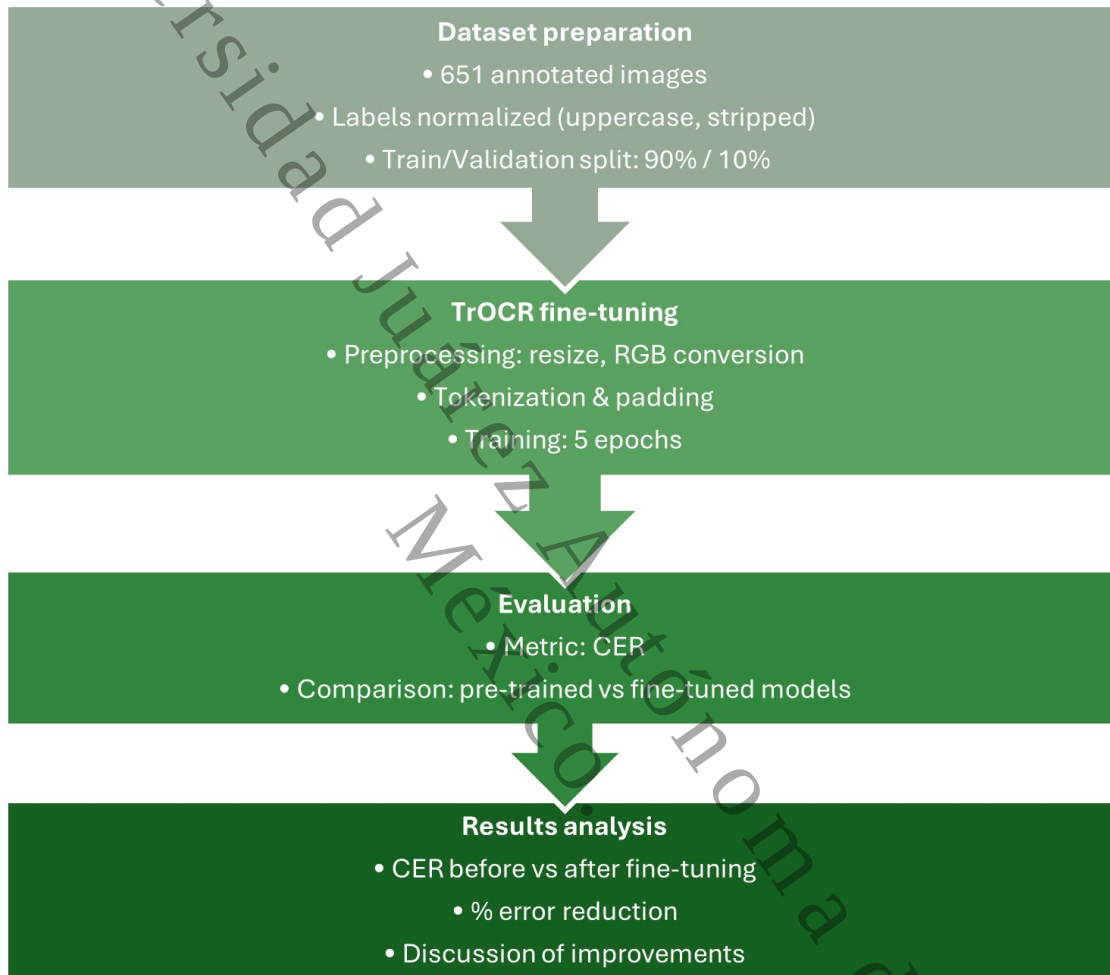


Figura 5.7. Flujo de trabajo para la evaluación del ajuste fino de TrOCR.

5.5.2. Evaluación base de TrOCR

La Tabla 5.10 presenta resultados representativos para tres experimentos de reconocimiento de placas utilizando TrOCR con parámetros por defecto.

El valor CER obtenido para TrOCR con parámetros por defecto sobre el conjunto completo fue de 0.4605, evaluando todos los videos en aproximadamente 6 minutos. Se identificó un

Tabla 5.10. Ejemplo de salidas OCR para tres experimentos

Video / Imagen	Número de placa	Salida TrOCR base	CER
22.mp4 / 064.jpg	XXX-###-X	XXX.###.#	0.667
16.mp4 / 146.jpg	XXX-###-X	XXX.###.#	0.556
15.mp4 / 127.jpg	XXX-###-X	XXXXX.###.#	0.556

patrón consistente de errores OCR caracterizado principalmente por sustituciones de símbolos visualmente similares. Las confusiones más frecuentes incluyeron la letra “O” reconocida como el número “0” y la letra “B” interpretada como “8”. Estos errores generalmente se originaron por iluminación deficiente o ángulos complicados de captura.

Además de las sustituciones, también se observó errores de inserción y eliminación, particularmente cuando los caracteres estaban parcialmente ocultos o distorsionados. Esto motivó la necesidad de ajustar finamente TrOCR con el objetivo de obtener un valor CER menor. Incluso con estas dificultades, TrOCR mostró una capacidad aceptable para identificar placas bajo ciertas condiciones, ventaja atribuida a su mecanismo de atención y decodificación consciente del contexto.

5.5.3. Resultados del TrOCR reentrenado

Para garantizar una comparación consistente, se aplicó el mismo procedimiento de evaluación OCR descrito previamente al modelo TrOCR ajustado finamente.

La Tabla 5.11 presenta los resultados de tres experimentos de reconocimiento de placas realizados con ambas versiones del modelo TrOCR.

Tabla 5.11. Ejemplo de salidas de TrOCR con ajuste finamente

Video / Imagen	Número de placa	Salida TrOCR base	CER	Salida ajustada	CER
16.mp4 / 171.jpg	X##-XXX	COMPANY	0.857	X##.XXX	0.143
11.mp4 / 058.jpg	XXX-###-X	XXX.###.#	0.444	XXX.###.X	0.222
04.mp4 / 437.jpg	XXX-###-X	TOTAL..GSTRA	1.222	XXX.###.X	0.333

La Figura 5.8 muestra una comparación entre las diferencias relevantes en el reconocimiento

de placas entre el modelo base y el modelo ajustado finamente. El desempeño global del modelo ajustado finamente fue evaluado utilizando CER, manteniendo el mismo procedimiento que se utilizó en la sección anterior.

El modelo base obtuvo un CER global de 0.4605, mientras que el modelo ajustado redujo el CER hasta 0.2576. Esta mejora demuestra un incremento sustancial en la precisión de reconocimiento como resultado de la adaptación específica al dominio. La Tabla 5.12 muestra el CER antes y después del ajuste fino

Tabla 5.12. CER antes y después del ajuste fino

Modelo OCR	CER
TrOCR por defecto	0.4605
TrOCR ajustado finamente	0.2576

La reducción del CER fue calculada aproximadamente en un 44 %, destacando la efectividad del procedimiento de ajuste fino para mejorar el reconocimiento de caracteres bajo condiciones complejas, incluyendo bajo contraste, desenfoque y oclusiones parciales. El modelo ajustado no solo redujo errores a nivel de carácter, sino que también mostró mayor robustez al reconocer símbolos visualmente ambiguos. La Figura 5.9 muestra la Curva de pérdida durante el entrenamiento.

A pesar de la mejora obtenida tras el reentrenamiento, la baja iluminación, el movimiento vehicular, las oclusiones parciales y los caracteres intercambiados continuaron siendo las causas más comunes de errores OCR. La Tabla 5.13 muestra Ejemplos representativos de fallos OCR.

Tabla 5.13. Ejemplos representativos de fallos OCR

Condición	Video / Imagen	Número de placa	Salida TrOCR	CER
Baja iluminación	04.mp4 / 023.jpg	XXX-###-X	XXX.###.X	0.333
Desenfoque por movimiento	03.mp4 / 011.jpg	X-###-XXX	X.###.X	0.556
Oclusión parcial	19.mp4 / 135.jpg	XXX-####-X	XX.####.X	0.200
Confusión C / 0	13.mp4 / 078.jpg	XXX-###-X	XXX.####.#	0.333

En el Experimento 19, la placa “XXX-####-X” fue parcialmente reconocida como “XX.####.X”. El primer carácter (una letra) fue omitido y el guion ‘-’ fue interpretado como un punto ‘.’. En el Experimento 13, una confusión típica del OCR entre el ‘0’ y la ‘C’ ilustra los problemas genera-

dos por símbolos visualmente similares durante el reconocimiento de placas. Errores comunes también incluyeron sustituciones entre 'W' y 'M', 'O' y '0', así como entre '8' y 'B'.

La Figura 5.10 presenta resultados relevantes del análisis estadístico del desempeño del OCR.

Se observó una mediana CER de 0.222 para el modelo ajustado finamente. El análisis de percentiles reveló valores CER de 0.222 (p50), 0.333 (p75), 0.333 (p90) y 0.556 (p95), indicando que la mayoría de las placas fueron reconocidas con tasas de error relativamente bajas. Adicionalmente, se obtuvo una tasa exacta de coincidencia completa de placa del 4.65 %.

5.6. Conclusiones

En este estudio se desarrolló una tubería ANPR completa utilizando YOLOv11 + TrOCR para la detección y reconocimiento de placas vehiculares, evaluando el desempeño mediante precisión, *recall* y *Character Error Rate* (CER).

YOLOv11 alcanzó alta confiabilidad en la detección (*recall*: 0.9950, precisión: 0.9380), confirmando su efectividad bajo condiciones variables. Sin embargo, TrOCR con su configuración por defecto alcanzó un desempeño moderado (CER: 0.4605), razón por la cual se propuso una estrategia de ajuste fino utilizando un conjunto personalizado de placas vehiculares.

Los beneficios del ajuste fino fueron evidentes, ya que el modelo TrOCR reentrenado sobre 651 imágenes produjo secuencias de caracteres considerablemente más precisas comparadas con la versión base, particularmente bajo condiciones complejas como baja iluminación, desenfoque por movimiento, oclusiones parciales y ángulos inclinados.

El modelo reentrenado alcanzó un CER de 0.2576, representando una mejora aproximada del 44 %.

En aplicaciones reales, el modelo ajustado permite el reconocimiento vehicular más confiable bajo condiciones desafiantes. La incorporación de ajuste fino demuestra que el OCR basado en Transformers puede adaptarse efectivamente a necesidades operativas específicas.

Como trabajo futuro se propone expandir el conjunto de datos con más placas y utilizar estrategias de división por grupos, garantizando que todas las instancias de una misma placa o vehículo pertenezcan exclusivamente al conjunto de entrenamiento o prueba.

Asimismo, incrementar el número de placas diferentes fortalecería la validez del modelo. Fi-

nalmente, un acoplamiento más estrecho entre YOLOv11 y TrOCR podría mejorar el desempeño en velocidad y precisión.

Referencias del capítulo

- Anwar, N., Khan, T., & Mollah, A. F. (2022a). Text Detection from Scene and Born Images: How Good is Tesseract? En *Recent Trends in Communication and Intelligent Systems* (pp. 115-122). https://doi.org/10.1007/978-981-19-1324-2_13
- Dhyani, S., & Kumar, V. (2023). Real-Time License Plate Detection and Recognition System Using YOLOv7x and EasyOCR. *2023 Global Conference on Information Technologies and Communications (GCITC)*, 1-5. <https://doi.org/10.1109/GCITC60406.2023.10425814>
- Hamood Bakhtan, M. A., Abdullah, M., & Rahman, A. A. (2016). A Review on License Plate Recognition System Algorithms. *Proceedings of the International Conference on Information and Communication Technology Management (ICICTM)*, 84-89. <https://doi.org/10.1109/ICICTM.2016.7890782>
- Jocher, G., & Qiu, J. (2024a). *Ultralytics YOLO11*. <https://github.com/ultralytics/ultralytics>
- Laroca, R., Zanlorensi, L. A., Gonçalves, G. R., Todt, E., Schwartz, W. R., & Menotti, D. (2021). An Efficient and Layout-Independent Automatic License Plate Recognition System Based on the YOLO Detector. *IET Intelligent Transportation Systems*, 15(4), 483-503. <https://doi.org/10.1049/itr2.12030>
- Li, M., Lv, T., Chen, H., Cui, L., Lu, Y., Florencio, D., Zhang, C., Li, Z., & Wei, F. (2023). TrOCR: Transformer-Based Optical Character Recognition with Pre-Trained Models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(11), 13094-13102. <https://doi.org/10.1609/aaai.v37i11.26538>
- Satya, B., Manongga, D., Hendry & Aminuddin, A. (2025). Optimized YOLOv8 for Automatic License Plate Recognition on Resource Constrained Devices. *Engineering, Technology & Applied Science Research*, 15(2), 21976-21981. <https://doi.org/10.48084/etasr.9983>
- Sonnara, F., Chihaoui, H., & Filali, F. (2025). Efficient Real-Time License Plate Recognition Using Deep Learning on Edge Devices. *Journal of Real-Time Image Processing*, 22, 159. <https://doi.org/10.1007/s11554-025-01738-3>

- Ströbel, P. B., Clematide, S., Volk, M., & Hodel, T. (2022). Transformer-Based HTR for Historical Documents. *arXiv preprint arXiv:2203.11008*. <https://arxiv.org/abs/2203.11008>
- Tavares, R. A. (2024). Comparison of Image Preprocessing Techniques for Vehicle License Plate Recognition Using OCR: Performance and Accuracy Evaluation. *arXiv preprint arXiv:2410.13622*. <https://doi.org/10.48550/arXiv.2410.13622>
- Wang, C. Y., Bochkovskiy, A., & Liao, H. Y. M. (2022). YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors. *arXiv preprint arXiv:2207.02696*. <https://doi.org/10.48550/arXiv.2207.02696>
- Zherzdev, S., & Gruzdev, A. (2018). LPRNet: License Plate Recognition via Deep Neural Networks. *arXiv preprint arXiv:1806.10447*. <https://doi.org/10.48550/arXiv.1806.10447>

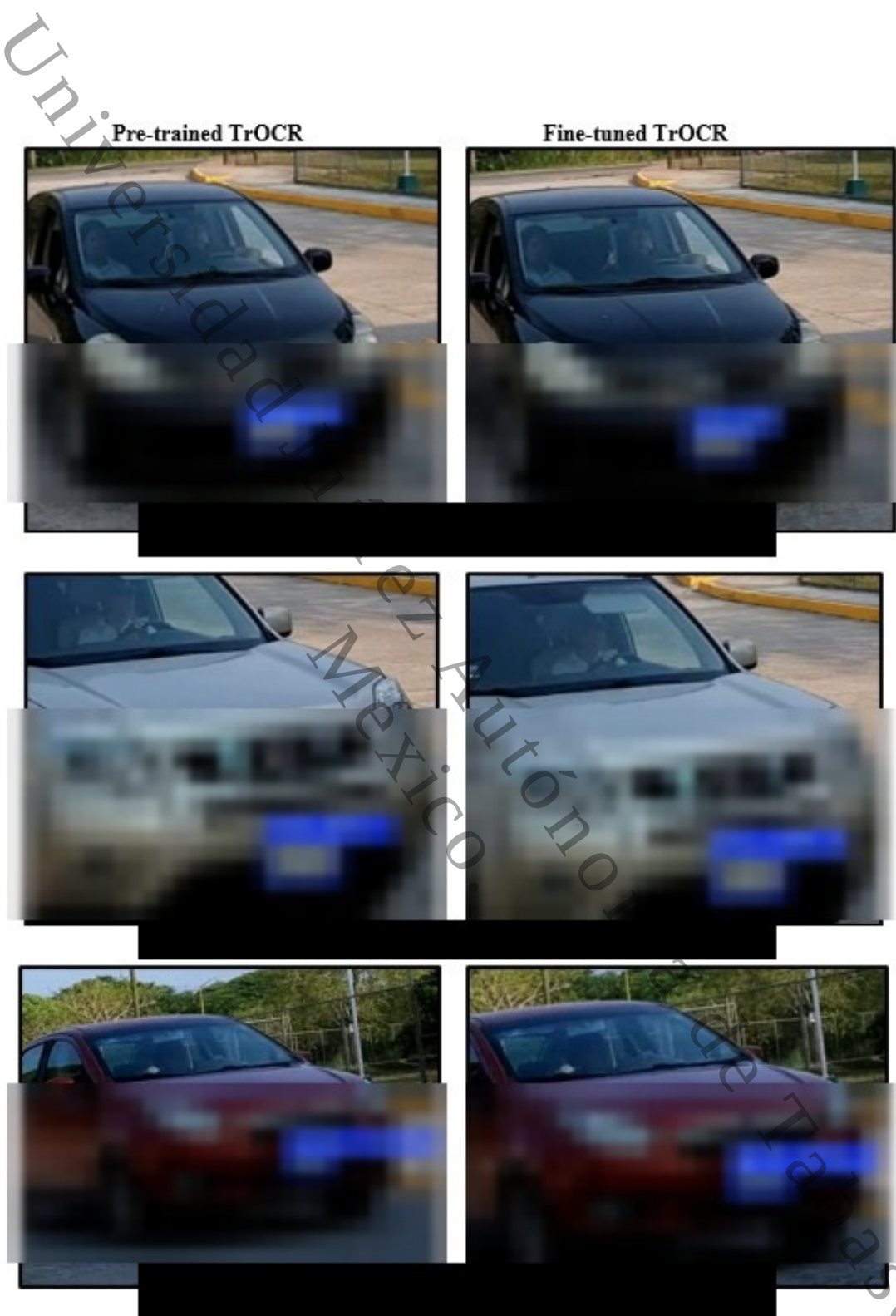


Figura 5.8. Ejemplo de salidas de TrOCR preentrenado vs. TrOCR con ajuste finamente.

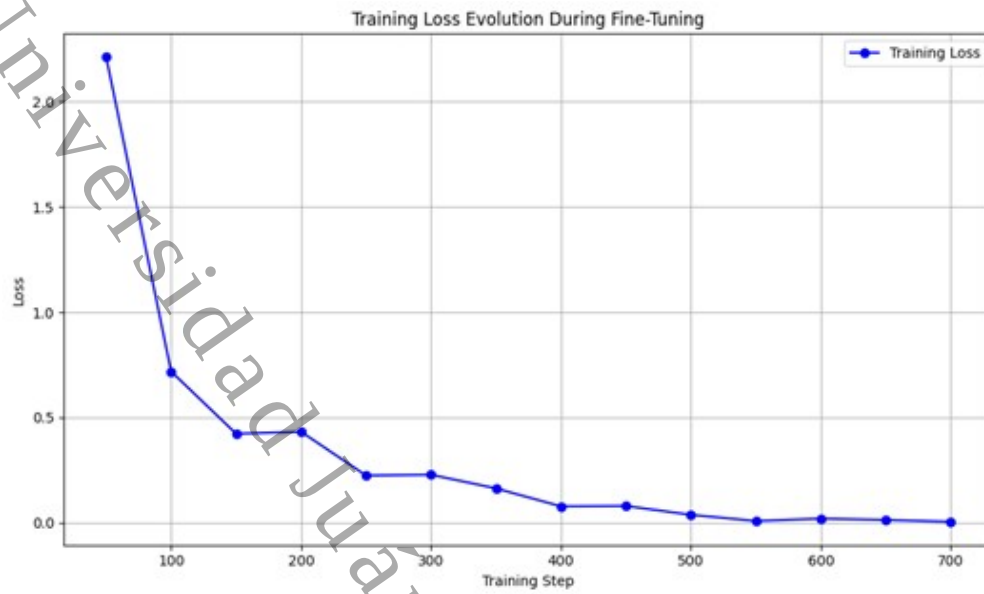


Figura 5.9. Curva de pérdida durante el entrenamiento.

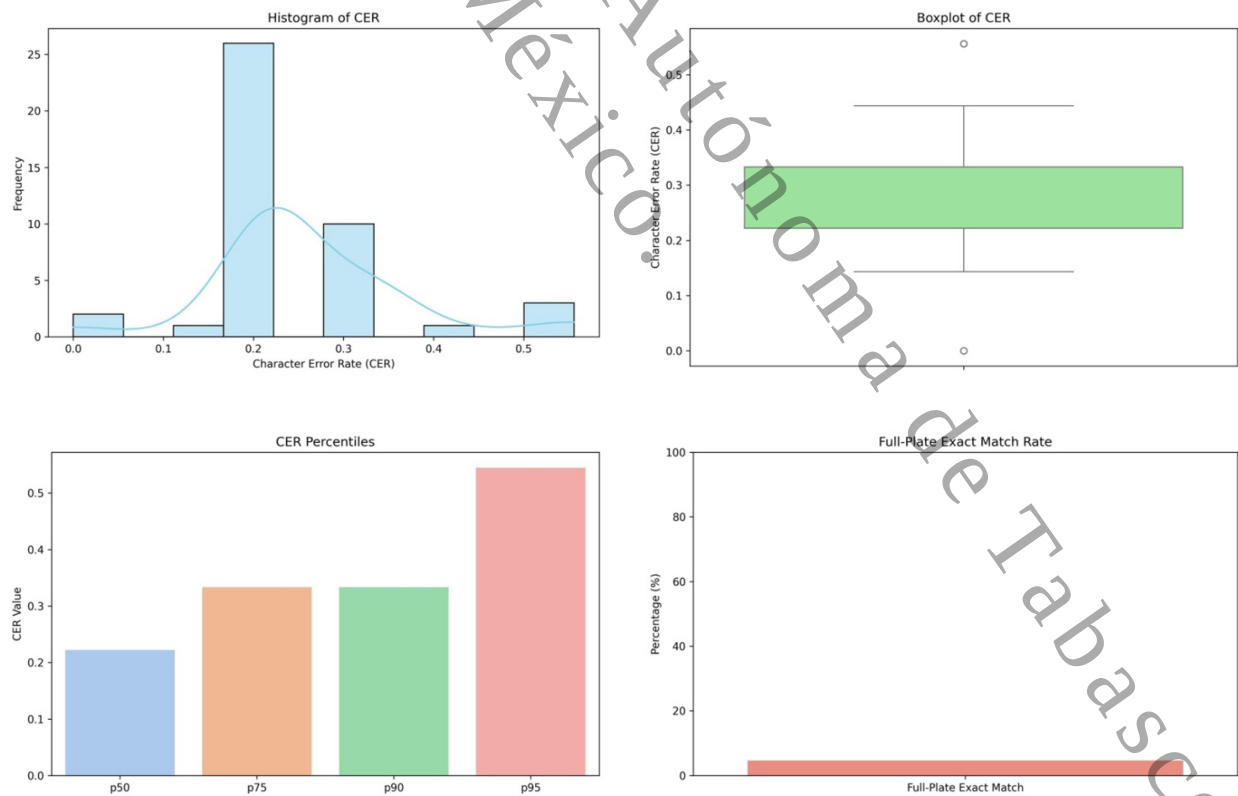


Figura 5.10. Estadísticas relevantes del desempeño de TrOCR: a) Histograma, b) Boxplot, c) Percentiles, d) Tasa exacta de coincidencia completa.

Capítulo 6

Implementación y validación del sistema ANPR en una plataforma *Edge Computing*

Una vez desarrolladas las etapas de detección vehicular, seguimiento de objetos y reconocimiento automático de placas mediante aprendizaje profundo, se llevó a cabo la implementación integral del sistema propuesto sobre una plataforma en el borde. El objetivo de esta fase consistió en validar la viabilidad de los modelos desarrollados en condiciones reales de funcionamiento, utilizando hardware de bajo costo y de consumo energético reducido.

La implementación fue realizada sobre una plataforma Raspberry Pi 4 modelo B (Raspberry Pi Foundation, 2026) equipada con un módulo de cámara (Foundation, 2026), permitiendo la captura, procesamiento y análisis de video en tiempo real. Esta etapa representa la validación final de la arquitectura propuesta, demostrando la capacidad de integrar modelos avanzados de visión computacional en dispositivos de cómputo *edge* para aplicaciones de control de acceso vehicular.

La arquitectura implementada incorpora los modelos desarrollados en capítulos anteriores, incluyendo la detección vehicular mediante YOLO11 (Jocher y Qiu, 2024b), el seguimiento de objetos utilizando ByteTrack (Zhang et al., 2022) y el reconocimiento automático de caracteres mediante la versión ajustada de TrOCR (Li et al., 2021) entrenada específicamente para placas

vehiculares del estado de Tabasco.

6.1. Arquitectura del sistema

6.1.1. Descripción del hardware

El sistema fue implementado utilizando componentes de bajo costo orientados a cómputo en el borde. La Tabla 6.1 resume las características técnicas relevantes de cada componente.

Tabla 6.1. Especificaciones del hardware utilizado.

Componente	Descripción
Raspberry Pi 4 Model B	SoC Broadcom BCM2711, CPU Cortex-A72 (ARM v8) de 64 bits a 1.5 GHz (quad-core), 8 GB de RAM LPDDR4, conectividad Wi-Fi 802.11ac de doble banda (2.4/5 GHz) y Bluetooth 5.0, puerto CSI-2 para cámara, GPIO de 40 pines.
Raspberry Pi Camera Module 3	Sensor Sony IMX708 de 12 MP, enfoque automático por detección de fase (PDAF), campo de visión estándar de 75°, interfaz CSI-2, resolución máxima de captura de 4608 × 2592 píxeles, captura de video hasta 1080p a 50 fps.
Fuente de alimentación	Fuente USB-C oficial de 5.1 V / 3 A.
Almacenamiento	Tarjeta microSD de 32 GB, clase A2, para el sistema operativo, instalación y configuración de modelos, y almacenamiento temporal de imágenes.
Carcasa y disipación	Gabinete con disipador pasivo/activo para evitar el <i>thermal throttling</i> durante la operación continua.
Enlace de red	Módulo WiFi integrado, utilizado para el envío de los registros a la base de datos remota.

La cámara se montó en la entrada del Campus con un ángulo fijo, a una altura y distancia definidas por las autoridades de la Universidad pero maximizando la región del cuadro capturado y minimizando la distorsión de perspectiva.

6.1.2. Descripción del software

La plataforma opera sobre el sistema operativo *Raspberry Pi OS* (64 bits, basado en Debian 12 “Bookworm”). La Tabla 6.2 resume todo el software empleado.

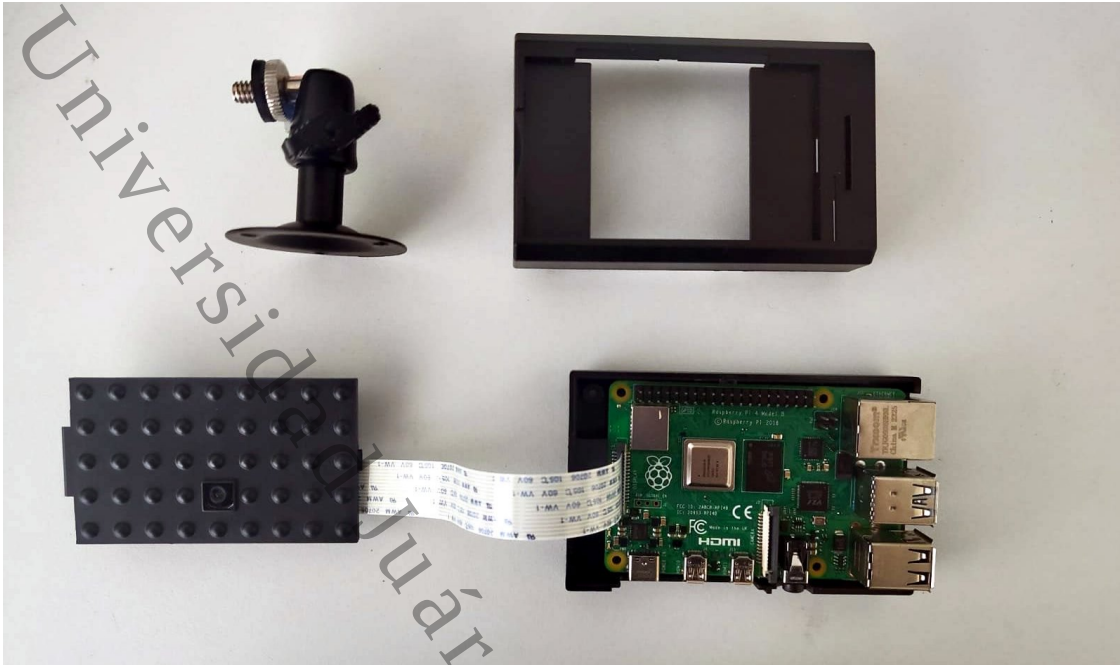


Figura 6.1. Hardware utilizado.

6.1.3. Diseño de la base de datos

Se diseñó una base de datos relacional para el almacenamiento de los eventos de reconocimiento. La tabla *acceso* almacena la placa reconocida, la fecha y hora del evento, y el nivel de confianza reportado por el modelo; en este caso, el *Character Error Rate* (CER). El Listado 6.1 muestra el script SQL de creación de la base de datos. Cabe mencionar que en el servidor se almacena un frame completo al momento de hacer la inferencia, es decir, una imagen del vehículo que ingresa en el momento de hacer el reconocimiento de la placa vehicular.

```

1  -- Creacion de la base de datos
2  CREATE DATABASE IF NOT EXISTS plataforma_edge
3      CHARACTER SET utf8mb4
4      COLLATE utf8mb4_unicode_ci;
5
6  USE plataforma_edge;
7
8  -- Tabla principal de registros de acceso vehicular
9  CREATE TABLE IF NOT EXISTS acceso (
10      id          INT AUTO_INCREMENT PRIMARY KEY,
11      placa       VARCHAR(15) NOT NULL,

```

Tabla 6.2. Componentes de software del sistema.

Capa	Descripción
Sistema operativo	Raspberry Pi OS (64-bit), kernel Linux.
Captura de imagen	Picamera2 (basado en libcamera), interfaz Python para el control del sensor IMX708.
Detección de placas	Modelo de detección de objetos YOLOv11n encargado de segmentar la región de interés (ROI) que contiene la placa.
Reconocimiento de caracteres	Modelo TrOCR reentrenado (<i>fine-tuning</i>) con un conjunto de datos de placas vehiculares locales.
Framework de inferencia	PyTorch modo CPU, con optimización mediante cuantización dinámica int8 para reducir la latencia en el hardware ARM.
Pipeline	Script en Python 3.13.7 que integra captura, detección, inferencia y persistencia de datos.
Capa de persistencia	Biblioteca mysql-connector-python / mediante conexión TCP/IP sobre la red WiFi local.
Base de datos	Servidor MariaDB 15.1 alojado en un servidor remoto dentro de la red UJAT.

```

12     fecha_hora    DATETIME    NOT NULL DEFAULT CURRENT_TIMESTAMP ,
13     cer_ocr       FLOAT       DEFAULT NULL ,
14     dispositivo  VARCHAR(30)  NOT NULL DEFAULT 'rpi4-entrada-01' ,
15     arch_imagen   VARCHAR(255) DEFAULT NULL ,
16 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

```

Listing 6.1. Esquema de base de datos.

La Figura 6.2 presenta la arquitectura completa del sistema, ilustrando la conectividad entre los componentes de hardware y software, así como el enlace inalámbrico hacia el servidor remoto donde se encuentra la base de datos.

Como se observa en la Figura 6.2, todo el procesamiento de visión por computadora (detección de la placa e inferencia del modelo TrOCR) se ejecuta localmente en la Raspberry Pi 4B, de acuerdo con el paradigma de *Edge Computing*; únicamente el resultado textual (placa reconocida, fecha, hora y metadatos) se transmite a través de la red WiFi hacia el servidor remoto, minimizando el ancho de banda requerido en comparación con el envío continuo de fotogramas de video para su procesamiento completo del lado del servidor.

Nodo Edge – Entrada del Campus

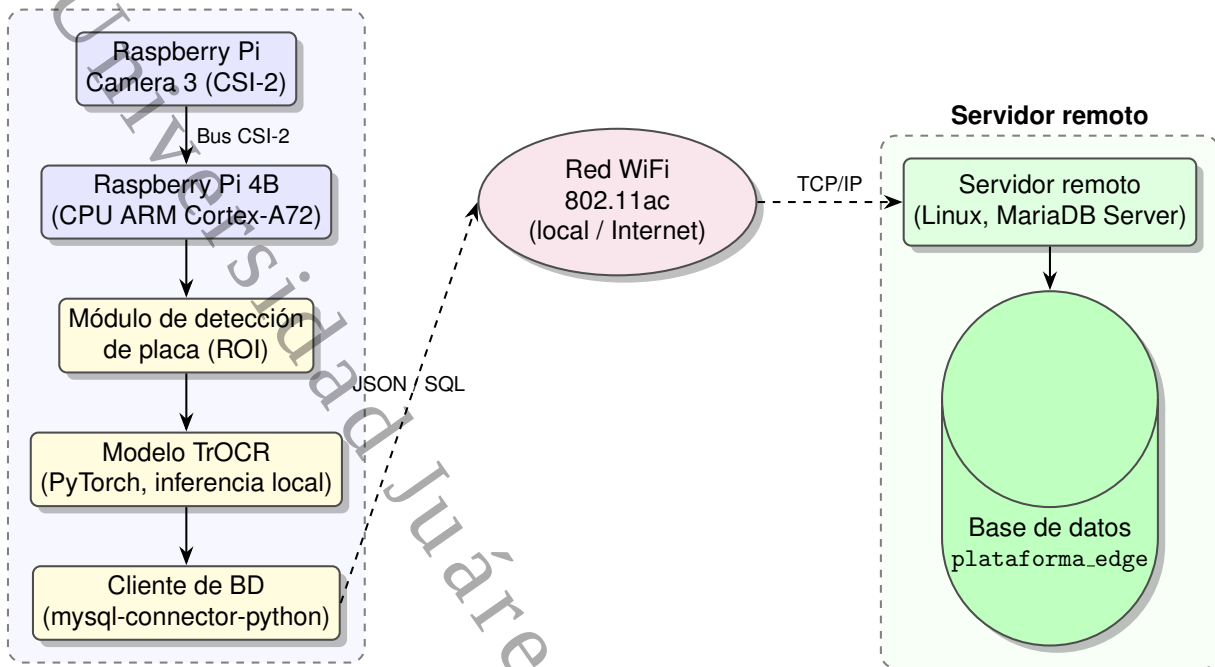


Figura 6.2. Arquitectura del sistema: conectividad entre el hardware (cámara y Raspberry Pi), el software embebido (detección y modelo TrOCR) y el servidor remoto que aloja la base de datos, enlazados mediante una red WiFi.

6.1.4. Metodología experimental

El sistema implementado sigue una arquitectura secuencial compuesta por múltiples etapas de procesamiento. Inicialmente, la cámara captura continuamente secuencias de video del flujo vehicular. Posteriormente, cada fotograma es procesado por el detector de objetos para localizar los vehículos presentes en la escena.

La Figura 6.3 describe el flujo metodológico seguido para el desarrollo y la evaluación del sistema, desde la adquisición de la imagen hasta el almacenamiento remoto del resultado.

El protocolo experimental se ejecutó de la siguiente manera:

1. Se capturaron vehículos de distintos tipos (automóvil, SUV, autobús y *pickup*) frente a la cámara, representando condiciones reales de entrada al Campus.
2. Se capturó la imagen y se ejecutó el *pipeline* completo descrito en la Figura 6.3.
3. Se registró la salida cruda del modelo TrOCR para cada evento.
4. Se comparó la cadena reconocida contra la placa real (*ground truth*) mediante la métrica CER.

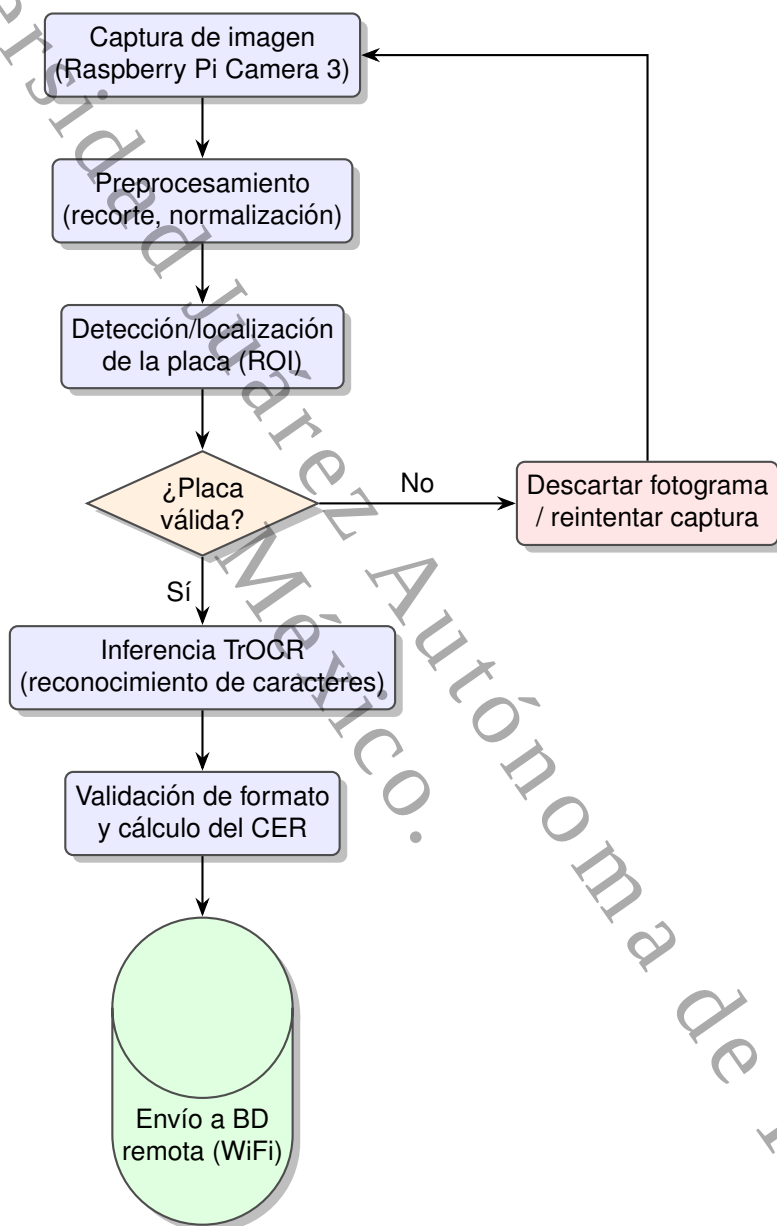


Figura 6.3. Diagrama de la metodología experimental.

6.2. Resultados

6.2.1. Protocolo de pruebas

La validación de la plataforma se realizó en escenarios reales de circulación vehicular dentro del Campus Chontalpa de la Universidad Juárez Autónoma de Tabasco. Se diseñó un protocolo de pruebas compuesto por 40 experimentos, cada uno correspondiente a un vehículo distinto ingresando al Campus en condiciones de iluminación normales. Este tamaño de muestra corresponde a una validación de prueba de concepto realizada durante una única sesión de recolección de datos en campo, y no representa un límite técnico del sistema. En la Sección 6.2.3 se detallan las razones de este alcance y se discute la manera en que el protocolo puede ampliarse en trabajo futuro.

Para cada experimento se registró el tipo de vehículo, la placa real (*ground truth*) y la cadena de texto producida por el modelo TrOCR tras ejecutar el *pipeline* completo descrito en la sección anterior. La Tabla 6.3 presenta los resultados obtenidos.

Para cada experimento, la Raspberry Pi 4B ejecutó localmente el *pipeline* de cuatro etapas:

- Captura del fotograma con la Raspberry Pi Camera 3 al detectar la presencia del vehículo.
- Localización de la región de la placa dentro del fotograma completo.
- Recorte y normalización (corrección de perspectiva y escalado a la resolución de entrada esperada por el modelo).
- Inferencia del modelo TrOCR, que genera la secuencia de caracteres carácter por carácter mediante su decodificador *Transformer*. El tiempo de inferencia promedio observado por fotograma, en modo CPU con cuantización `int8`, fue de aproximadamente 0.6–0.9 segundos, un valor compatible con el flujo vehicular de entrada del Campus en afluencia normal.

Tabla 6.3. Resultados del protocolo de pruebas experimentales y cálculo del CER por experimento.

No.	Tipo de vehículo	Placa real	Placa reconocida	Errores	Longitud (N)	CER
1	Automóvil	XXX-###-X	XXX.###.X	2	9	0.222

continúa en la siguiente página

(Tabla 6.3 continuación)

Exp.	Tipo de vehículo	Placa real	Placa reconocida	Errores	Longitud (N)	CER
2	Automóvil	XXX-###-X	XXX.###.X	2	9	0.222
3	Automóvil	XXX-###-X	XXX.###.X	4	9	0.444
4	Automóvil	XXX-###-X	XXX.###.X	2	9	0.222
5	Automóvil	XXX-##-##	XXX.##.##	2	9	0.222
6	Automóvil	XXX-###-X	XXX.###.X	4	9	0.444
7	Automóvil	XXX-###-X	XXX.###.X	2	9	0.222
8	Automóvil	XXX-###-X	XXX.###.X	3	9	0.333
9	SUV	XXX-###-X	XXX.###.X	2	9	0.222
10	SUV	XXX-###-X	XXX.###.X	2	9	0.222
11	Automóvil	XXX-##-##	XXX.##.##	2	9	0.222
12	Automóvil	XXX-###-X	XXX.###.X	2	9	0.222
13	Automóvil	XXX-###-X	XXX.###.X	2	9	0.222
14	Automóvil	XXX-###-X	XXX.###.#	3	9	0.333
15	Automóvil	XXX-###-X	XXX.###.X	2	9	0.222
16	Pick-up	XXX-###-X	XXX.###.X	2	9	0.222
17	Automóvil	XXX-###-X	XXX.###.X	2	9	0.222
18	Automóvil	XXX-###-X	XXX.###.X	2	9	0.222
19	Automóvil	XXX-###-X	XXX.###.X	2	9	0.222
20	Automóvil	XXX-###-X	XXX.###.X	2	9	0.222
21	Automóvil	X##-XXX	X##.XXX	1	7	0.142
22	Buggy	—	—	—	—	—
23	Automóvil	XXX-###-X	XXX.###.X	2	9	0.222
24	Automóvil	XXX-###-X	XXX.###.X	2	9	0.222
25	SUV	XXX-###-X	XXX.###.X	2	9	0.222
26	Automóvil	XXX-###-X	XXX.###.X	3	9	0.333
27	Automóvil	XXX-###-X	XXX.###.X	2	9	0.222
28	SUV	XXX-###-X	XXX.###.X	2	9	0.222
29	Pick-up	XXX-####-X	XX.####.X	3	10	0.300
30	Automóvil	XXX-###-X	XXX.###.X	2	9	0.222
31	SUV	XXX-###-X	XXX.###.X	2	9	0.222
32	Automóvil	XXX-###-X	XXX.###.X	2	9	0.222
33	Automóvil	XXX-###-X	XXX.###.X	4	9	0.444
34	Automóvil	XXX-###-X	XXX.###.X	3	9	0.333

continúa en la siguiente página

(Tabla 6.3 continuación)

Exp.	Tipo de vehículo	Placa real	Placa reconocida	Errores	Longitud (N)	CER
35	Pick-up	XX-####-X	XX.###.X	3	9	0.333
36	SUV	XXX-###-X	XXX.###.X	3	9	0.333
37	Automóvil	XXX-###-X	XXX.###.X	3	9	0.333
38	SUV	XXX-###-X	XXX.###.X	3	9	0.333
39	Autobús	##-XXX-##	X##.XXX.X	5	9	0.555
40	SUV	XXX-###-X	XXX.###.X	5	9	0.555
Total / CER global				98	350	0.280

Es importante mencionar que el Experimento 22 fue descartado del análisis por captura fallida de la placa (rechazo automático por ROI no válida, la ruta “No” de la Figura 6.3).

El CER es la métrica estándar para evaluar sistemas de reconocimiento de texto a nivel de carácter. Se define como:

$$CER = \frac{S + D + I}{N} \quad (6.1)$$

donde S es el número de sustituciones, D el número de eliminaciones, I el número de inserciones, y N es el número total de caracteres de la placa real (*ground truth*). El desglose completo de este cálculo para los 40 experimentos se presenta en la Tabla 6.3.

El CER global se calcula sumando el total de errores (sustituciones, eliminaciones e inserciones) y el total de caracteres de referencia de todos los experimentos:

$$CER_{\text{global}} = \frac{\sum_{i=1}^{40} (S_i + D_i + I_i)}{\sum_{i=1}^{40} N_i} = \frac{98}{350} = 0.280 \quad (6.2)$$

Los resultados obtenidos demostraron que la plataforma propuesta es capaz de ejecutar de manera satisfactoria el flujo completo de reconocimiento automático de placas vehiculares sobre hardware *edge*. Sin embargo, El CER_{global} se puede interpretar como una tasa de error del 28 %, o dicho de otra manera, un rendimiento del 72 %. Esto se puede considerar bajo para un despliegue, ya que en la literatura de sistemas OCR aplicados a documentos y placas vehiculares se considera que un CER inferior al 5–10 % es adecuado para aplicaciones de producción (S. M. Silva y C. R. Jung, 2018).

Un análisis de los resultados indica que la mayor cantidad de errores en la identificación de placas lo constituye el carácter guión '-', ya que el modelo lo reconoce como un punto '.'. Esto ocurre en los 39 casos de placas reconocidas y es el único error en 25 de ellas. Por lo anterior, en el script de identificación de caracteres se incluyó la instrucción `deteccion.replace(".", "-")` para sustituir cada punto '.' por un guión '-' en las placas detectadas por el modelo. Bajo esta normalización, se recalculó el CER_{global} , con lo cual el total de errores se reduce de 98 a 21, quedando:

$$CER_{normalizado} = \frac{21}{350} \approx 0.060 \quad (6.3)$$

De esta manera, la plataforma propuesta tiene un CER del 6 %, lo que se interpreta como un rendimiento del 94 %.

6.2.2. Discusión

El análisis carácter por carácter de los resultados revela que existe un patrón de error recurrente que aparece en prácticamente todos los experimentos: el modelo identifica al carácter '-' como un '.'. De hecho, en el 62.5 % de los experimentos, el error es únicamente de formato, ya que en todos estos casos el modelo sustituye los guiones por puntos. Se podría considerar esto como un error sintáctico y no semántico, al confundir el separador de los caracteres que representan el reconocimiento real.

Otro error importante lo constituye la confusión de la letra 'M' por 'W' (4 casos), seguida de 'U' por 'V' y viceversa. Esto se debe claramente a la tipografía utilizada, en la que estos caracteres son muy similares entre sí. Otras confusiones de un solo carácter son 'O' por 'D', 'C' por '0', y la pérdida de un dígito en secuencias numéricas largas, por ejemplo "7803" por "783".

6.2.3. Limitaciones y alcance de la validación

El protocolo experimental descrito fue diseñado como una validación de *prueba de concepto* del sistema ANPR integrado en la plataforma *edge*, y no como una evaluación exhaustiva del desempeño del sistema en condiciones reales. Se leccionó un tamaño de la muestra de 40 vehículos debido a la ventana de tiempo y a los recursos disponibles para la recolección de datos en vivo

en el Campus Chontalpa, realizada en el turno matutino bajo condiciones normales de tránsito. Es importante señalar que este valor no constituye un límite del sistema, ya que ni el hardware ni el software poseen restricciones sobre el número de vehículos que pueden procesarse.

En este sentido, todas las pruebas reportadas en este capítulo se realizaron bajo condiciones de iluminación diurna normales, con la cámara en una posición fija, frontal y a una distancia aproximadamente constante del punto de detección, sin presencia de lluvia, neblina, contraluz u obstrucciones parciales de la placa. Estas condiciones juegan a favor del sistema y no representan todos los posibles escenarios que un sistema ANPR podría enfrentar en un despliegue real.

6.3. Conclusiones

La implementación desarrollada confirma que los avances recientes en Aprendizaje profundo permiten migrar modelos complejos de visión computacional desde entornos de alto desempeño hacia plataformas *edge* de bajo costo.

Los resultados obtenidos también demuestran la importancia del ajuste fino de modelos OCR especializados para contextos regionales. La adaptación de TrOCR utilizando placas vehiculares de Tabasco permitió reducir errores de reconocimiento asociados a características particulares de tipografía, iluminación y condiciones locales de captura.

No obstante, los resultados presentados se enmarcan dentro de un escenario controlado de iluminación diurna, posición fija de cámara y bajo tránsito vehicular. Extender el protocolo de pruebas a un mayor número de vehículos y a condiciones ambientales adversas (iluminación nocturna, lluvia, distintos ángulos de captura) constituye la línea de trabajo futuro más relevante para acercar el sistema propuesto a un despliegue en producción.

Referencias del capítulo

- Foundation, R. P. (2026). *Raspberry Pi Camera Module 3*. Consultado el 25 de junio de 2026, desde <https://www.raspberrypi.com/products/camera-module-3/>
- Jocher, G., & Qiu, J. (2024b). *Ultralytics YOLO11* [<https://github.com/ultralytics/ultralytics>] (11.0.0).

- Li, M., Zhang, Y., Li, L., Zhou, L., Yang, N., Zhou, M., & Wei, F. (2021). TrOCR: Transformer-based Optical Character Recognition with Pre-trained Models [<https://arxiv.org/abs/2109.10282>]. *arXiv preprint arXiv:2109.10282*.
- Raspberry Pi Foundation. (2026). *Raspberry Pi 4 model B*. Consultado el 25 de junio de 2026, desde <https://www.raspberrypi.com/products/raspberry-pi-4-model-b/>
- S. M. Silva & C. R. Jung. (2018). License Plate Detection and Recognition in Unconstrained Scenarios. *Proceedings of the European Conference on Computer Vision (ECCV)*, 580-596.
- Zhang, Y., Sun, P., Jiang, Y., Yu, D., Weng, F., Yuan, Z., Luo, P., Liu, W., & Wang, X. (2022). ByteTrack: Multi-Object Tracking by Associating Every Detection Box [https://doi.org/10.1007/978-3-031-19830-4_1]. *European Conference on Computer Vision*, 1(1), 1-21. <https://doi.org/10.1007>

Capítulo 7

Conclusiones y recomendaciones

7.1. Conclusiones generales

La presente investigación tuvo como objetivo desarrollar una plataforma inteligente para el control de acceso vehicular, basada en técnicas de aprendizaje profundo y *Edge computing*. A través de las diferentes etapas del trabajo se diseñaron, entrenaron, evaluaron e implementaron modelos de visión computacional orientados a la detección, seguimiento y reconocimiento automático de placas vehiculares, logrando una solución integral capaz de operar en escenarios reales.

En la primera etapa de la investigación, se demostró que el uso de técnicas de *Transfer learning* constituye una estrategia eficiente para adaptar modelos de detección de objetos a contextos específicos. La optimización del modelo SSDMobileNetV2FPN Lite permitió mejorar la detección de motocicletas utilizando un conjunto de datos propio, reduciendo significativamente el tiempo y los recursos necesarios para el entrenamiento de modelos desde cero.

Posteriormente, la implementación de YOLO11 combinado con ByteTrack permitió desarrollar un sistema robusto de seguimiento y conteo vehicular. Los resultados obtenidos evidenciaron que la integración de algoritmos modernos de detección y seguimiento permite mantener la identidad de los vehículos durante su desplazamiento dentro de la escena, específicamente durante el ingreso al Campus Chontalpa de la Universidad.

Respecto al reconocimiento automático de placas vehiculares, el análisis comparativo realizado entre Tesseract, EasyOCR y TrOCR permitió identificar las ventajas de los enfoques basados

en aprendizaje profundo frente a métodos OCR tradicionales. Los resultados demostraron que los modelos neuronales presentan una mayor capacidad para enfrentar variaciones de iluminación, perspectiva, ruido visual y condiciones ambientales complejas.

Uno de los principales aportes de la investigación fue el ajuste fino del modelo TrOCR utilizando un conjunto de datos personalizado de placas vehiculares del estado de Tabasco. El rendimiento del modelo ajustado fue del 94 %, demostrando que la especialización del modelo permitió mejorar significativamente la precisión del reconocimiento y reducir la tasa de error de caracteres, con lo que se confirma la hipótesis planteada.

La implementación final del sistema sobre una plataforma de hardware libre como la Raspberry Pi 4B equipada con un módulo de cámara, permitió validar la viabilidad de la plataforma ANPR propuesta. Los resultados obtenidos demostraron que es posible ejecutar localmente procesos de captura de video, detección vehicular, seguimiento de objetos y reconocimiento de placas sin depender de infraestructura en la nube. La validación en tiempo real confirmó que la arquitectura propuesta puede servir efectivamente para el control de acceso vehicular, proporcionando tiempos de respuesta adecuados y manteniendo niveles satisfactorios de precisión en todas las etapas del procesamiento. Asimismo, la ejecución local de los modelos permitió reducir la latencia asociada a la transmisión de datos hacia servidores remotos, mejorar la privacidad de la información capturada y disminuir el consumo de ancho de banda de red; ventajas que representan beneficios importantes para sistemas de vigilancia inteligente y monitoreo vehicular.

Finalmente, los resultados obtenidos permiten concluir que la integración del Aprendizaje profundo con el reconocimiento automático de placas vehiculares y el *Edge computing* constituye una alternativa tecnológica viable para el desarrollo de sistemas inteligentes de control de acceso vehicular, con potencial de aplicación en instituciones educativas, instalaciones gubernamentales, estacionamientos inteligentes o zonas industriales.

7.1.1. Cumplimiento de los objetivos de investigación

Los objetivos planteados al inicio de la investigación fueron alcanzados satisfactoriamente. En particular:

- Se desarrolló un modelo de detección vehicular basado en Aprendizaje profundo capaz de

identificar vehículos en escenarios reales.

- Se implementó un sistema de seguimiento y conteo vehicular utilizando algoritmos de última generación para mantener la identidad de los objetos detectados durante su desplazamiento.
- Se evaluaron diferentes motores OCR para el reconocimiento automático de placas vehiculares, identificando sus fortalezas y limitaciones.
- Se realizó el ajuste fino del modelo TrOCR utilizando un conjunto de datos especializado de placas vehiculares de Tabasco, obteniendo mejoras significativas en su rendimiento.
- Se validó experimentalmente la integración completa del sistema mediante su despliegue sobre una plataforma en el borde basada en Raspberry Pi.
- Se generó evidencia experimental que demuestra la factibilidad de implementar soluciones ANPR inteligentes en tiempo real utilizando hardware de bajo costo.

7.2. Recomendaciones generales

Como líneas de trabajo futuro y recomendaciones para investigaciones posteriores, se propone:

- Ampliar el conjunto de datos incorporando placas vehiculares de diferentes estados de la República Mexicana. En particular, incluir las nuevas placas que se lanzaron en este 2026 en el estado de Tabasco.
- Evaluar el comportamiento del sistema bajo condiciones climáticas adversas (como lluvia intensa o niebla) y escenarios de tráfico de alta densidad.
- Implementar técnicas de cuantización, poda de redes neuronales y aceleración por hardware en los modelos de Aprendizaje profundo para incrementar el rendimiento en plataformas *edge*.
- Integrar tecnologías complementarias como RFID o códigos QR para sistemas de autenticación multimodal.

Finalmente, se sugiere promover la adopción de arquitecturas basadas en *Edge computing* en aplicaciones de seguridad y monitoreo vehicular, debido a sus ventajas en términos de latencia, privacidad de datos, disponibilidad del servicio y reducción de costos operativos, aspectos que serán cada vez más relevantes dentro del desarrollo de infraestructuras inteligentes y sistemas de movilidad del futuro.

Modelo de plataforma de aprendizaje profundo para el control de acceso vehicular	
Título de la tesis:	Modelo de plataforma de aprendizaje profundo para el control de acceso vehicular
Autor:	Manuel Antonio García Herrera
ORCID:	https://orcid.org/0009-0009-0635-5375
Resumen:	La presente tesis propone una plataforma inteligente para el control de acceso vehicular basada en técnicas de Aprendizaje profundo y <i>Edge computing</i>. El objetivo principal fue desarrollar e implementar una solución capaz de realizar la detección y seguimiento de vehículos junto con el reconocimiento automático de placas vehiculares en tiempo real mediante dispositivos de cómputo en el borde, es decir, de bajo costo y consumo energético. La investigación se desarrolló en cuatro etapas. Inicialmente, se optimizó un modelo SSD MobileNetV2 FPNLite mediante aprendizaje por transferencia para mejorar la detección de motocicletas en escenarios reales. Posteriormente, se implementó un sistema de seguimiento y conteo vehicular utilizando YOLOv11 y ByteTrack, obteniendo resultados satisfactorios en precisión y estabilidad durante el monitoreo del flujo vehicular. En una tercera etapa se realizó una evaluación comparativa de motores OCR, incluyendo Tesseract, EasyOCR y TrOCR. Los resultados evidenciaron un mejor desempeño de los enfoques basados en aprendizaje profundo frente a los métodos tradicionales. A partir de ello, se efectuó el ajuste fino del modelo TrOCR utilizando un conjunto de datos propio con placas vehiculares del estado de Tabasco, mejorando la precisión del reconocimiento. Finalmente, el sistema fue implementado sobre una plataforma Raspberry Pi equipada con un módulo de cámara, validando la ejecución integrada de los procesos de captura, detección, seguimiento y reconocimiento de placas. Los resultados demostraron la viabilidad de la plataforma para aplicaciones de detección automática de placas vehiculares con dispositivos de baja latencia y procesamiento local eficiente.
Palabras clave:	Aprendizaje profundo, Edge computing, YOLOv11, ByteTrack, TrOCR, ANPR, Reconocimiento de placas vehiculares, Visión computacional.
Referencias citadas:	En la siguiente página se muestran las referencias.

Bibliografía

- Alfarano, A., Maiano, L., Papa, L., & Amerini, I. (2024). Estimating optical flow: A comprehensive review of the state of the art [<https://doi.org/10.1016/j.cviu.2024.104160>]. *Computer Vision and Image Understanding*, 249(1), 104160. <https://doi.org/10.1016/j.cviu.2024.104160>
- Ameen, S., Siriwardana, K., & Theodoridis, T. (2023). Optimizing Deep Learning Models For Raspberry Pi. *arXiv*, 1 (1), 1-6.
- Anwar, N., Khan, T., & Mollah, A. F. (2022a). Text Detection from Scene and Born Images: How Good is Tesseract? En *Recent Trends in Communication and Intelligent Systems* (pp. 115-122). https://doi.org/10.1007/978-981-19-1324-2_13
- Anwar, N., Khan, T., & Mollah, A. F. (2022b). Text Detection from Scene and Born Images: How Good is Tesseract? [https://doi.org/10.1007/978-981-19-1324-2_13]. En A. K. S. Pundir, N. Yadav, H. Sharma & S. Das (Eds.), *Recent Trends in Communication and Intelligent Systems* (pp. 115-122). Springer Nature Singapore. https://doi.org/10.1007/978-981-19-1324-2_13
- Awwad, S., Igried, B., Wedyan, M., & Alshira'H, M. (2021). Hybrid Features for Object Detection in RGB-D Scenes [<https://doi.org/10.11591/ijeecs.v23.i2.pp1073-1083>]. *Indonesian Journal of Electrical Engineering and Computer Science*, 23(2), 1073-1083. <https://doi.org/10.11591/ijeecs.v23.i2.pp1073-1083>
- Bakhtan, M., Abdullah, M., & Rahman, A. (2016). A review on License Plate Recognition system algorithms [<https://doi.org/10.1109/ICICTM.2016.7890782>]. *Proceedings of the International Conference on Information and Communication Technology Management (ICICTM)*, 84-89. <https://doi.org/10.1109/ICICTM.2016.7890782>
- Berzal, F. (2019). Redes neuronales & deep learning: Volumen II. *Independently published*, 1, 15-25.
- Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020a). YOLOv4: Optimal Speed and Accuracy of Object Detection [<https://arxiv.org/abs/2004.10934>]. *arXiv preprint arXiv:2004.10934*.
- Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020b). YOLOv4: Optimal Speed and Accuracy of Object Detection [<https://arxiv.org/abs/2004.10934>]. *arXiv preprint arXiv:2004.10934*.
- Bolaños-Fernández, C., & Bacca-Cortes, E. B. (2024). Mobile Application for Recognizing Colombian Currency with Audio Feedback for Visually Impaired People [<https://doi.org/10.14483/23448393.21408>]. *Ingeniería*, 29(2), e21408. <https://doi.org/10.14483/23448393.21408>

- Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.
- Bradski, G., & Kaehler, A. (2008). *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media.
- Braun, P. A., Moss, R. H., & Stigal, P. D. (1986). The single board computer: A practical alternative to a custom designed microsystem. *Computers & Electrical Engineering*, 12 (3), 109-118.
- Cao, K., Liu, Y., Meng, G., & Sun, Q. (2020). An Overview on Edge Computing Research. *IEEE Access*, 13(8), 85714-85728.
- Chai, J., Zeng, H., Li, A., & Ngai, E. W. (2021). Deep learning in computer vision: A critical review of emerging techniques and application scenarios. *Machine Learning with Applications*, 6 (), 1-13.
- Chen, J., & Ran, X. (2015). Deep Learning With Edge Computing: A Review. *Proceedings of the IEEE*, 107(8), 1655-1674.
- Chen, S.-H., Wang, C.-W., Tai, I.-H., Ken-Pen Weng, Y.-H. C., & Hsieh, K.-S. (2021). Modified YOLOv4-DenseNet Algorithm for Detection of Ventricular Septal Defects in Ultrasound Images [https://doi.org/10.9781/ijimai.2021.06.001]. *International Journal of Interactive Multimedia and Artificial Intelligence*, 6(7), 1-8. https://doi.org/10.9781/ijimai.2021.06.001
- Chen, Y., Liu, X., Li, C., Zhu, J., Wu, M., & Su, X. (2024). UAV Swarm Centroid Tracking for Edge Computing Applications Using GRU-Assisted Multi-Model Filtering [https://doi.org/10.3390/electronics13061054]. *Electronics*, 13(6), 1-15. https://doi.org/2079-9292
- Chiu, S.-Y., Chiu, C.-C., & Xu, S. (2018). A Background Subtraction Algorithm in Complex Environments Based on Category Entropy Analysis [https://doi.org/10.3390/app8060885]. *Applied Sciences*, 8(6), 1-15. https://doi.org/2076-3417
- COCO. (2024). *COCO Dataset* [Accedido: 30 de septiembre de 2024]. https://cocodataset.org/#home
- D K, T., James, J., Gopinath, D. P., & Ashraf K, M. (2024). Advocating Character Error Rate for Multilingual ASR Evaluation [https://doi.org/10.48550/arXiv.2410.07400]. *arXiv preprint arXiv:2410.07400*. https://doi.org/10.48550/arXiv.2410.07400
- de Ciencias, D. A., & de la Información, T. (2024). *Informe de actividades 2022-2023*. Consultado el 4 de junio de 2024, desde https://archivos.ujat.mx/2023/div-dacyti/InformeDACYTlv-2022-2023.pdf
- de Ciencias Básicas, D. A. (2024). *Informe de actividades 2022-2023*. Consultado el 4 de junio de 2024, desde https://archivos.ujat.mx/2023/div-dacb/informe/INFORME_CIENCIAS%20BASICAS.web.pdf
- de Ingeniería, D. A., & Arquitectura. (2024). *Informe de actividades 2022-2023*. Consultado el 4 de junio de 2024, desde https://archivos.ujat.mx/2023/div-daia/INFORME%202022-2023_presentacion.pdf
- Dhyani, S., & Kumar, V. (2023). Real-Time License Plate Detection and Recognition System Using YOLOv7x and EasyOCR. *2023 Global Conference on Information Technologies and Communications (GCITC)*, 1-5. https://doi.org/10.1109/GCITC60406.2023.10425814

- Drobac, S., & Lindén, K. (2020). Optical character recognition with neural networks and post-correction with finite state methods [<https://doi.org/10.1007/s10032-020-00359-9>]. *International Journal on Document Analysis and Recognition (IJDAR)*, 23, 279-295. <https://doi.org/10.1007/s10032-020-00359-9>
- Du, K., & Bobkov, A. (2023a). An Overview of Object Detection and Tracking Algorithms. *Engineering Proceedings*, 33(22), 1-6.
- Du, K., & Bobkov, A. (2023b). An Overview of Object Detection and Tracking Algorithms [<https://doi.org/10.3390/engproc2023033022>]. *Engineering Proceedings*, 33(22), 1-6. <https://doi.org/10.3390/engproc2023033022>
- Edgar, T. W., & Manz, D. O. (2017). Machine Learning. *Syngress*, 6, 153-173.
- Foundation, R. P. (2026). *Raspberry Pi Camera Module 3*. Consultado el 25 de junio de 2026, desde <https://www.raspberrypi.com/products/camera-module-3/>
- Galvez, R. L., Bandala, A. A., Dadios, E. P., Vicerra, R. R. P., & Maningo, J. M. Z. (2018). Object Detection Using Convolutional Neural Networks. *Proceedings of TENCON 2018 - 2018 IEEE Region 10 Conference*. <https://doi.org/10.1109/TENCON.2018.8650517>
- Gao, J., Han, G., Zhu, H., & Liao, L. (2024a). Multiple Moving Vehicles Tracking Algorithm with Attention Mechanism and Motion Model. *Electronics*, 1(13), 1-17.
- Gao, J., Han, G., Zhu, H., & Liao, L. (2024b). Multiple Moving Vehicles Tracking Algorithm with Attention Mechanism and Motion Model [<https://doi.org/10.3390/electronics13010242>]. *Electronics* 2024, 13(242), 1-12. <https://doi.org/10.3390/electronics13010242>
- Géron, A. (2019, junio). *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow* (2nd). O'Reilly Media, Inc.
- Google. (2024). *Google Colaboratory* [Accedido: 30 de diciembre de 2024]. <https://cloud.google.com/colab/docs>
- Google. (2025). *Colaboratory Documentation* [<https://cloud.google.com/colab/docs?hl=es-419>].
- Goswami, S. (2020). False detection (positives and negatives) in object detection [<https://doi.org/10.48550/arXiv.2008.06986>]. *arXiv preprint arXiv:2008.06986*, 1(1), 1-15. <https://doi.org/10.48550>
- Guntuboina, C., Aditya Porwal abd, P. J., & Shingrakhia, H. (2021). Deep Learning Based Automated Sports Video Summarization Using Yolo [<https://doi.org/10.5565/rev/elcvia.1286>]. *Electronic Letters on Computer Vision and Image Analysis*, 20(1), 99-116. <https://doi.org/10.5565/rev/elcvia.1286>
- Hamood Bakhtan, M. A., Abdullah, M., & Rahman, A. A. (2016). A Review on License Plate Recognition System Algorithms. *Proceedings of the International Conference on Information and Communication Technology Management (ICICTM)*, 84-89. <https://doi.org/10.1109/ICICTM.2016.7890782>
- Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., ... Oliphant, T. E.

- (2020). Array programming with NumPy [<https://doi.org/10.1038/s41586-020-2649-2>]. *Nature*, 585(7825), 357-362. <https://doi.org/10.1038/s41586-020-2649-2>
- Hosna, A., Merry, E., Gyalmo, J., & Z., A. (2022). Transfer learning: a friendly introduction. *Journal of Big Data*, 102(9), 2196-1115.
- Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). Mobilenets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv preprint arXiv:1704.04861*.
- Hu, X., Zhao, J., Hui, Y., Li, S., & You, S. (2023). SiamHSFT: A Siamese Network-Based Tracker with Hierarchical Sparse Fusion and Transformer for UAV Tracking [<https://doi.org/10.3390/s23218666>]. *Sensors*, 23(21), 1-15. <https://doi.org/10.3390/s23218666>
- Iandola, F. N. (2016). *SqueezeNet: AlexNet-level accuracy with 50x fewer parameters* [Repositorio de GitHub. Consultado el 25 de junio de 2026]. <https://github.com/forresti/SqueezeNet>
- IBM. (2024a). *¿Qué es el deep learning?* Consultado el 16 de abril de 2024, desde <https://www.ibm.com/mx-es/topics/deep-learning/>
- IBM. (2024b). *What is computer vision?* Consultado el 11 de abril de 2024, desde <https://www.ibm.com/topics/computer-vision/>
- IBM. (2024c). *What Is Optical Character Recognition (OCR)?* Consultado el 11 de abril de 2024, desde <https://www.ibm.com/blog/optical-character-recognition/>
- Intel. (2024). *Referencia de chipset y sistema en un chip (SoC) para Intel® NUC*. Consultado el 1 de abril de 2024, desde <https://www.intel.la/content/www/xl/es/support/articles/000056236/intel-nuc.html#:~:text=Un%20SoC%2C%20o%20System%2Don,de%20energ%C3%ADa%20y%20radios%20inal%C3%A1mblicas/>
- Jaidev AI. (2020). *EasyOCR*. Consultado el 2 de octubre de 2025, desde <https://github.com/JaidevAI/EasyOCR>
- Jocher, G., & Qiu, J. (2024a). *Ultralytics YOLO11*. <https://github.com/ultralytics/ultralytics>
- Jocher, G., & Qiu, J. (2024b). *Ultralytics YOLO11* [<https://github.com/ultralytics/ultralytics>] (11.0.0).
- Khan, S. A., Lee, H. J., & Lim. (2023). Enhancing Object Detection in Self-Driving Cars Using a Hybrid Approach. *Electronics*, 12(2768), 1-12.
- Kumawat, K., Jain, A., & Tiwari, N. (2024). Relevance of Automatic Number Plate Recognition Systems in Vehicle Theft Detection. *Engineering Proceedings*, 59(185), 1-11.
- Laroca, R., Zanlorensi, L. A., Gonçalves, G. R., Todt, E., Schwartz, W. R., & Menotti, D. (2021). An Efficient and Layout-Independent Automatic License Plate Recognition System Based on the YOLO Detector. *IET Intelligent Transportation Systems*, 15(4), 483-503. <https://doi.org/10.1049/itr2.12030>
- Laroca, R., Zanlorensi, L. A., Gonçalves, G. R., Todt, E., Schwartz, W. R., & Menotti, D. (2019). An Efficient and Layout-Independent Automatic License Plate Recognition System Based on the YOLO Detector [<https://arxiv.org/abs/1909.01754>]. *arXiv preprint arXiv:1909.01754*.
- LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, 521(7553), 436-444.

- Lee, H. J., Ullah, I., Wan, W., Gao, Y., & Fang, Z. (2019). Real-Time Vehicle Make and Model Recognition with the Residual SqueezeNet Architecture. *Sensors*, 19(5), 982.
- Li, M., Lv, T., Chen, H., Cui, L., Lu, Y., Florencio, D., Zhang, C., Li, Z., & Wei, F. (2023). TrOCR: Transformer-Based Optical Character Recognition with Pre-Trained Models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(11), 13094-13102. <https://doi.org/10.1609/aaai.v37i11.26538>
- Li, M., Zhang, Y., Li, L., Zhou, L., Yang, N., Zhou, M., & Wei, F. (2021). TrOCR: Transformer-based Optical Character Recognition with Pre-trained Models [<https://arxiv.org/abs/2109.10282>]. *arXiv preprint arXiv:2109.10282*.
- Lin, T. Y., et al. (2014). Microsoft COCO: Common Objects in Context. *Computer Vision – ECCV 2014*, 740-755.
- Lin, T. Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal Loss for Dense Object Detection. *arXiv preprint arXiv:1708.02002*.
- Liu, L., Ouyang, W., Wang, X., Fieguth, P., Chen, J., Liu, X., & Pietikäinen, M. (2020). Deep Learning for Generic Object Detection: A Survey. *International Journal of Computer Vision*, 128(2), 261-318. <https://doi.org/10.1007/s11263-019-01247-4>
- Liu, L., Sun, Y., & Ge, X. (2025). A Hybrid Multi-Person Fall Detection Scheme Based on Optimized YOLO and ST-GCN [<http://dx.doi.org/10.9781/ijimai.2024.09.003>]. *International Journal of Interactive Multimedia and Artificial Intelligence*, 9(2), 26-38. <https://doi.org/10.9781/ijimai.2024.09.003>
- Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. *Computer Vision – ECCV 2016*, 21-37.
- López Flórez, S., González-Briones, A., Hernández, G., Ramos, C., & de la Prieta, F. (2023). Automatic Cell Counting With YOLOv5: A Fluorescence Microscopy Approach [<https://doi.org/10.9781/ijimai.2023.08.001>]. *International Journal of Interactive Multimedia and Artificial Intelligence*, 8(3), 1-8. <https://doi.org/10.9781/ijimai.2023.08.001>
- MathWorks. (2024). *Procesamiento de imágenes y Visión artificial*. Consultado el 11 de abril de 2024, desde <https://la.mathworks.com/solutions/image-video-processing.html/>
- Matplotlib. (2024). *Matplotlib Quick Start* [Accedido: 30 de septiembre de 2024]. https://matplotlib.org/stable/users/explain/quick_start.html
- Meinhardt, T., Kirillov, A., Leal-Taixe, L., & Feichtenhofer, C. (2022). Trackformer: Multi-object tracking with transformers [<https://doi.org/10.48550/arXiv.2101.02702>]. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 8844-8854. <https://doi.org/10.48550>
- Naseer, S., Shah, S. M. A., Aziz, S., Khan, M. U., & Iqtidar, K. (2020). Vehicle Make and Model Recognition using Deep Transfer Learning and Support Vector Machines. *2020 IEEE 23rd International Multitopic Conference (INMIC)*, 1(1), 1-6.
- NumPy. (2024). *NumPy Documentation* [Accedido: 30 de septiembre de 2024]. <https://numpy.org/doc/stable/>

- Padilla, R., Passos, W. L., Dias, T. L. B., Netto, S. L., & da Silva, E. A. B. (2021). A Comparative Analysis of Object Detection Metrics with a Companion Open-Source Toolkit. *Electronics*, 10(3), 279. <https://doi.org/10.3390/electronics10030279>
- Pereira, R., Carvalho, G., Garrote, L., & Nunes, U. (2022). Sort and Deep-SORT Based Multi-Object Tracking for Mobile Robotics: Evaluation with New Data Association Metrics [<https://doi.org/10.3390/app12031319>]. *Applied Sciences*, 12(3), 1-15. <https://doi.org/2076-3417>
- Peres Barbosa, J., Dalla Rosa, F., & Rieder, R. (2024). Pothole Detection Web App: an approach for detecting potholes in asphalt pavements using YOLO [<https://doi.org/10.5335/rbca.v16i3.15851>]. *Revista Brasileira de Computação Aplicada*, 16(3), 25-36. <https://doi.org/10.5335/rbca.v16i3.15851>
- Pérez-Aguilar, D., Pérez-Aguilar, J., Pérez-Aguilar, A., Risco-Ramos, R., & Malpica- Rodríguez, M. (2024). Electric substation inspection: YOLOv5 in hotspot detection through thermal imaging [<https://doi.org/10.17163/ings.n31.2024.04>]. *Ingenius, Revista de Ciencia y Tecnología*, 1(31), 43-54. <https://doi.org/10.17163/ings.n31.2024.04>
- Ramajo Ballester, Á., González Cepeda, J., Armingol Moreno, J., & Escalera Hueso, A. d. I. (2022). Reidentificación de vehículos mediante técnicas de deep learning. *XLIII Jornadas de Automática: libro de actas*, 1(1), 1031-1039.
- Raspberry Pi Foundation. (2026). *Raspberry Pi 4 model B*. Consultado el 25 de junio de 2026, desde <https://www.raspberrypi.com/products/raspberrypi-4-model-b/>
- Ren, Y., Li, Y., & Gao, X. (2024). An MRS-YOLO Model for High-Precision Waste Detection and Classification [<https://doi.org/10.3390/s24134339>]. *Sensors*, 24(13).
- Reyes Zambrano, J., & Moreira Santos, M. (2025). Computer Vision in the field of productive control and health in Farm Animals. *Revista de Tecnologías de la Informática y las Comunicaciones*, 9(1), 16-29. <https://doi.org/10.33936/isrtic.v9.i1.7156>
- Roboflow. (2024). *Supervision* [<https://roboflow.github.io/supervision/>].
- Rondón, C. N., Castro Casadiego, S., & Medina Delgado, B. (2021). Background subtraction and yolo algorithm: two methods for the detection of people in uncontrolled environments. *Revista Ingeniería Solidaria*, 17(1), 1-25. <https://doi.org/2357-6014.2021.01.06>
- S. M. Silva & C. R. Jung. (2018). License Plate Detection and Recognition in Unconstrained Scenarios. *Proceedings of the European Conference on Computer Vision (ECCV)*, 580-596.
- Satya, B., Manongga, D., Hendry & Aminuddin, A. (2025). Optimized YOLOv8 for Automatic License Plate Recognition on Resource Constrained Devices. *Engineering, Technology & Applied Science Research*, 15(2), 21976-21981. <https://doi.org/10.48084/etasr.9983>
- Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv preprint arXiv:1409.1556*. <https://arxiv.org/abs/1409.1556>
- Sonnara, F., Chihaoui, H., & Filali, F. (2025). Efficient Real-Time License Plate Recognition Using Deep Learning on Edge Devices. *Journal of Real-Time Image Processing*, 22, 159. <https://doi.org/10.1007/s11554-025-01738-3>

- Sporici, D., Cuşnir, E., & Boiangiu, C.-A. (2020). Improving the Accuracy of Tesseract 4.0 OCR Engine Using Convolution-Based Preprocessing [<https://www.mdpi.com/2073-8994/12/5/715>]. *Symmetry*, 12(5), 715. <https://doi.org/10.3390/sym12050715>
- Spruck, A., Hawesch, M., Maier, A., Riess, C., Seiler, J., & Kaup, A. (2022). 3D Rendering Framework for Data Augmentation in Optical Character Recognition [<https://arxiv.org/abs/2209.14970>]. *arXiv preprint arXiv:2209.14970*.
- Ströbel, P. B., Clematide, S., Volk, M., & Hodel, T. (2022). Transformer-Based HTR for Historical Documents. *arXiv preprint arXiv:2203.11008*. <https://arxiv.org/abs/2203.11008>
- Su, J., Wu, C., & Yang, S. (2023). Object-Tracking Algorithm Combining Motion Direction and Time Series. *Applied Sciences*, 13(8), 1-14.
- Tavares, R. A. (2024). Comparison of Image Preprocessing Techniques for Vehicle License Plate Recognition Using OCR: Performance and Accuracy Evaluation. *arXiv preprint arXiv:2410.13622*. <https://doi.org/10.48550/arXiv.2410.13622>
- Tavares, R., & Silva, L. (2024). Comparison of Image Preprocessing Techniques for Vehicle License Plate Recognition Using OCR: Performance and Accuracy Evaluation [<https://arxiv.org/abs/2410.13622>]. *arXiv preprint arXiv:2410.13622*.
- Tecnoseguro. (2024). *Qué es LPR, cómo funciona y sus aplicaciones*. Consultado el 20 de marzo de 2024, desde <https://www.tecnoseguro.com/faqs/que-es-lpr-como-funciona-y-sus-aplicaciones/>
- TensorFlow. (2024). *tf.keras.models.load_model* [Accedido: 30 de diciembre de 2024]. https://www.tensorflow.org/api_docs/python/tf/keras/models/load_model
- Tourani, A., Shahbahrami, A., Soroori, S., Khazaee, S., & Suen, C. Y. (2020). A Robust Deep Learning Approach for Automatic Iranian Vehicle License Plate Detection and Recognition for Surveillance Systems. *IEEE Access*, 8(1), 201317-201330.
- viso.ai. (2024). *Object Tracking in Computer Vision (2024 Guide) Read more at: <https://viso.ai/deep-learning/object-tracking/>*. Consultado el 11 de abril de 2024, desde <https://viso.ai/deep-learning/object-tracking/>
- Wang, C. Y., Bochkovskiy, A., & Liao, H. Y. M. (2022). YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors. *arXiv preprint arXiv:2207.02696*. <https://doi.org/10.48550/arXiv.2207.02696>
- Wang, C.-Y., Bochkovskiy, A., & Liao, H.-Y. M. (2022). YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors [<https://arxiv.org/abs/2207.02696>]. *arXiv preprint arXiv:2207.02696*.
- Wang, F., Zhang, M., Wang, X., Ma, X., & Liu. (2020). Deep Learning for Edge Computing Applications: A State-of-the-Art Survey. *IEEE Access*, 1(8), 58322-58336.
- Wang, Y., & Mariano, V. (2024). A Multi Object Tracking Framework Based on YOLOv8s and ByteTrack Algorithm [<https://doi.org/10.1109/ACCESS.2024.3450370>]. *IEEE Access*, 12, 120711-120719. <https://doi.org/10.1109/ACCESS.2024.3450370>

- Yang, B., Tang, M., Chen, S., et al. (2020a). A vehicle tracking algorithm combining detector and tracker. *J Image Video Proc.*, 2020(17), 1-20.
- Yang, B., Tang, M., Chen, S., Wang, G., Tan, Y., & Li, B. (2020b). A vehicle tracking algorithm combining detector and tracker [<https://doi.org/10.1186/s13640-020-00505-7>]. *J Image Video Proc.* 2020, 1(17), 1-10. <https://doi.org/10.1186/s13640-020-00505-7>
- Younis, A., Shixin, L., Shelembi, J. N., & Hai, Z. (2020). Real-Time Object Detection Using Pre-Trained Deep Learning Models MobileNet-SSD. *Proceedings of the Conference*, 1-6. <https://doi.org/10.1145/3379247.3379264>
- Zhang, Y., Sun, P., Jiang, Y., Yu, D., Weng, F., Yuan, Z., Luo, P., Liu, W., & Wang, X. (2022). ByteTrack: Multi-Object Tracking by Associating Every Detection Box [https://doi.org/10.1007/978-3-031-19830-4_1]. *European Conference on Computer Vision*, 1(1), 1-21. <https://doi.org/10.1007>
- Zherzdev, S., & Gruzdev, A. (2018). LPRNet: License Plate Recognition via Deep Neural Networks. *arXiv preprint arXiv:1806.10447*. <https://doi.org/10.48550/arXiv.1806.10447>
- Zherzdev, S., & Gruzdev, A. (2019). Deep Learning for Vehicle License Plate Recognition: A Survey [<https://doi.org/10.1109/ICCSI62669.2024.10799401>]. *Pattern Recognition and Image Analysis*, 29(3), 606-617. <https://doi.org/10.1134/S1054661819030124>
- Zulko, Z. (2015). *MoviePy - Video editing with Python* [<https://github.com/Zulko/moviepy>].