



Universidad Nacional Mayor de San Marcos

Universidad del Perú. Decana de América

Facultad de Ingeniería Electrónica y Eléctrica

Escuela Profesional de Ingeniería Electrónica

Desarrollo de un sistema automático para el reconocimiento de placas de rodaje peruanas en el departamento de Lima en el año 2023

TESIS

Para optar el Título Profesional de Ingeniero Electrónico

AUTOR

Luis Angel BASTIDAS CERAZO

ASESOR

Dr. Carlos Alberto SOTELO LÓPEZ

Lima, Perú

2025



Reconocimiento - No Comercial - Compartir Igual - Sin restricciones adicionales

<https://creativecommons.org/licenses/by-nc-sa/4.0/>

Usted puede distribuir, remezclar, retocar, y crear a partir del documento original de modo no comercial, siempre y cuando se dé crédito al autor del documento y se licencien las nuevas creaciones bajo las mismas condiciones. No se permite aplicar términos legales o medidas tecnológicas que restrinjan legalmente a otros a hacer cualquier cosa que permita esta licencia.

Referencia bibliográfica

Bastidas, L. (2025). *Desarrollo de un sistema automático para el reconocimiento de placas de rodaje peruanas en el departamento de Lima en el año 2023*. [Tesis de pregrado, Universidad Nacional Mayor de San Marcos, Facultad de Ingeniería Electrónica y Eléctrica, Escuela Profesional de Ingeniería Electrónica]. Repositorio institucional Cybertesis UNMSM.

Metadatos complementarios

Datos de autor	
Nombres y apellidos	Luis Angel Bastidas Cerazo
Tipo de documento de identidad	DNI
Número de documento de identidad	48156130
URL de ORCID	https://orcid.org/0000-0001-9613-7068
Datos de asesor	
Nombres y apellidos	Carlos Alberto Solelo López
Tipo de documento de identidad	DNI
Número de documento de identidad	07017259
URL de ORCID	https://orcid.org/0000-0001-8960-6567
Datos del jurado	
Presidente del jurado	
Nombres y apellidos	Celso Ysidro Gerónimo Huamán
Tipo de documento	DNI
Número de documento de identidad	09191500
Miembro del jurado 1	
Nombres y apellidos	Juan Francisco Tisza Contreras
Tipo de documento	DNI
Número de documento de identidad	08596442
Miembro del jurado 2	
Nombres y apellidos	Jean Carlos Malca Fernández
Tipo de documento	DNI
Número de documento de identidad	45535181

Datos de investigación	
Línea de investigación	C.0.3.16. Procesamiento de imágenes y visión por computadora
Grupo de investigación	NO APLICA
Agencia de financiamiento	SIN FINANCIAMIENTO
Ubicación geográfica de la investigación	País: Perú Departamento: Lima Provincia: Lima Distrito: Lima Latitud: -12.05514 Longitud: -77.08689
Año o rango de años en que se realizó la investigación	2023
URL de disciplinas OCDE	Ingeniería eléctrica, Ingeniería electrónica https://purl.org/pe-repo/ocde/ford#2.02.01



UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS
Universidad del Perú. Decana de América
FACULTAD DE INGENIERÍA ELECTRÓNICA Y ELÉCTRICA
ESCUELA PROFESIONAL DE INGENIERÍA ELECTRÓNICA

ACTA DE SUSTENTACIÓN N° 003-EPIE-FIEE/2025

Los suscritos miembros de Jurado, docentes permanentes de las Escuelas Profesionales de Facultad Ingeniería Electrónica y Eléctrica, reunidos en la fecha 5 de setiembre del 2025, como presidente de jurado el **Dr. CELSO YSIDRO GERÓNIMO HUAMÁN**, integrado por el miembro de jurado el **Mg. JUAN FRANCISCO TISZA CONTRERAS**, el miembro de jurado **Mg. JEAN CARLOS MALCA FERNÁNDEZ** y miembro asesor el **Dr. CARLOS ALBERTO SOTELO LOPEZ**.

Después de escuchar la Sustentación de Tesis del Bachiller **LUIS ANGEL BASTIDAS CERAZO** con código N° 10190258 para optar el Título Profesional de Ingeniero Electrónico sustentó la tesis titulada: **"DESARROLLO DE UN SISTEMA AUTOMÁTICO PARA EL RECONOCIMIENTO DE PLACAS DE RODAJE PERUANAS EN EL DEPARTAMENTO DE LIMA EN EL AÑO 2023"**.

El jurado examinador procedió a formular las preguntas reglamentarias y luego de una deliberación en privado, decidió otorgarle el calificativo de 16 (Satisfacción)

Ciudad Universitaria, 5 de setiembre del 2025

Dr. CELSO YSIDRO GERÓNIMO HUAMÁN
Presidente Jurado

Mg. JUAN FRANCISCO TISZA CONTRERAS
Miembro Jurado

Mg. JEAN CARLOS MALCA FERNÁNDEZ
Miembro Jurado

Dr. CARLOS ALBERTO SOTELO LOPEZ
Miembro Asesor

Anexo 1
CERTIFICADO DE SIMILITUD

Yo, CARLOS ALBERTO SOTELO LÓPEZ en mi condición de asesor acreditado con el Dictamen de la UPG o de la EP N° DICTAMEN N°002-EPIE-2025-FIEE/UNMSM de la tesis/monografía/ informe de investigación/trabajo académico, cuyo título es DESARROLLO DE UN SISTEMA AUTOMÁTICO PARA EL RECONOCIMIENTO DE PLACAS DE RODAJE PERUANAS EN EL DEPARTAMENTO DE LIMA EN EL AÑO 2023, presentado por el bachiller/ magíster/ egresado/ licenciado/ estudiante LUIS ANGEL BASTIDAS CERAZO para optar el grado* de/ título*/ especialidad en TITULO PROFESIONAL DE INGENIERO ELECTRÓNICO,

CERTIFICO que se ha cumplido con lo establecido en la Directiva de Originalidad y de Similitud de Trabajos Académicos, de Investigación y Producción Intelectual. Según la revisión, análisis y evaluación mediante la plataforma PIDS, el documento evaluado cuenta con el 6...% de similitud, nivel PERMITIDO para continuar con los trámites correspondientes y para su **publicación en el repositorio institucional**.

Se emite el presente certificado en cumplimiento de lo establecido en las normas vigentes, como uno de los requisitos para la obtención del grado*/ título*/ especialidad* TITULO PROFESIONAL DE INGENIERO ELECTRÓNICO



Firmado digitalmente por SOTELO
LOPEZ Carlos Alberto FAU
20148092282 soft
Motivo: Soy el autor del documento
Fecha: 13.10.2025 12:26:10 -05:00

Lima, 13 de OCTUBRE de 2025

Firma del Asesor _____

Nombres y Apellidos del asesor: CARLOS ALBERTO SOTELO LÓPEZ

DNI: 07017259

* Según corresponda: bachiller, magíster, doctor; título profesional o título de segunda especialidad

La vigencia del presente certificado corresponde a un período de noventa (90) días desde la fecha de su emisión; siempre que no haya observaciones en el texto durante el proceso de revisión por el jurado informante o surja alguna contingencia.

Dedicatoria

A mis queridos padres, Áurea y Lucio, por todo el esfuerzo que empeñaron en mi desarrollo académico y personal. Son el ejemplo de perseverancia que me motiva a seguir creciendo y mejorando cada día más.

Agradecimiento

Al Dr. Carlos Sotelo López, por los consejos, la guía y el acompañamiento metodológico brindados durante la elaboración del presente trabajo de tesis.

Índice General

Dedicatoria	ii
Agradecimiento	iii
Índice General	iv
Índice de Cuadros	vii
Índice de Figuras	x
Resumen	xiii
Abstract	xiv
I. INTRODUCCIÓN	1
1.1. Introducción	1
1.2. Planteamiento del problema	2
1.2.1. Determinación del problema	2
1.2.2. Formulación del problema	3
1.3. Objetivos	4
1.3.1. Objetivo principal	4
1.3.2. Objetivos específicos	4
1.4. Importancia y alcance de la investigación	5
1.4.1. Importancia de la investigación	5
1.4.2. Alcance de la investigación	6
1.5. Limitaciones de la investigación	6
II. REVISIÓN DE LITERATURA	7
2.1. Marco teórico	7
2.2. Antecedentes del estudio	19
2.2.1. Antecedentes internacionales	19
2.2.2. Antecedentes nacionales	21
2.3. Bases teóricas y técnicas	22

2.4. Definición de términos	35
3.2. Variables	38
III. HIPÓTESIS Y VARIABLES	38
3.1. Hipótesis	38
3.1.1. Hipótesis Principal	38
3.1.2. Hipótesis Específicas:	39
3.3. Operacionalización de las variables	40
3.4. Matriz de consistencia	42
IV. MATERIALES Y MÉTODOS	45
4.1. Área de estudio	45
4.2. Diseño de investigación	45
4.3. Población y muestra	46
4.4. Procedimientos, técnicas e instrumentos de recolección de información	48
4.5. Análisis estadístico	51
4.5.1. Análisis estadístico de las imágenes	51
4.5.2. Análisis estadístico de las placas de rodaje etiquetadas	53
4.5.3. Resumen del análisis estadístico	68
V. RESULTADOS	70
5.1. Presentación y análisis de resultados	70
5.1.1. Consideraciones previas con relación al entrenamiento de los modelos de detección	70
5.1.2. Resultados del entrenamiento y evaluación de los modelos de detección	73
5.1.3. Resultados del reconocimiento de caracteres	111
5.1.4. Implementación del prototipo	114
5.1.5. Resultados del test del sistema prototipo	118
5.1.6. Evaluación práctica del prototipo	122

5.2. Resultados obtenidos en función a los indicadores de evaluación de las variables de la investigación	127
VI. DISCUSIÓN	128
VII. CONCLUSIONES Y RECOMENDACIONES	131
7.1. Conclusiones	131
7.2. Recomendaciones	132
REFERENCIAS BIBLIOGRÁFICAS	133
ANEXOS	142
ANEXO 1: SCRIPT EN PYTHON EMPLEADO EN EL PROTOTIPO PARA TEST DEL MODELO COMPLETO: pplatedetector_test.py	143
ANEXO 2: SCRIPT EN PYTHON EMPLEADO EN EL PROTOTIPO PARA TEST DEL MODELO LIGERO: pplatedetector_test_tflite.py	145
ANEXO 3: FICHA PARA RECOLECCIÓN DE IMÁGENES	147

Índice de Cuadros

Cuadro 1. Taxonomía de la Inteligencia Artificial propuesta por Watch IA ...	10
Cuadro 2. Reglas para los números de las placas de rodaje en Perú	25
Cuadro 3. Modelos pre-entrenados para la detección de objetos proporcionados por la Object Detection API de Tensorflow	34
Cuadro 4. Operacionalización de las Variables de la Investigación	41
Cuadro 5. Matriz de consistencia.....	43
Cuadro 6. Características generales de las imágenes del dataset	52
Cuadro 7. Número de Placas por Imagen.....	52
Cuadro 8. Placas de rodaje según la cantidad de veces que se repite en el dataset	54
Cuadro 9. Parámetros estadísticos de la relación de aspecto de las placas del dataset	57
Cuadro 10. Inclinação de placas en el dataset	59
Cuadro 11. Parámetros estadísticos de la relación de área de las placas del dataset	60
Cuadro 12. Contraste de las placas en el dataset	62
Cuadro 13. Parámetros estadísticos del enfoque de las placas del dataset	63
Cuadro 14. Parámetros estadísticos del enfoque de las placas del primer intervalo	66
Cuadro 15. Borrosidad de las placas en el dataset.....	68
Cuadro 16. Evaluación de la generalidad del dataset según las características analizadas.....	69
Cuadro 17. Modelos pre entrenados seleccionados para el análisis comparativo	71
Cuadro 18. Resultados del proceso de entrenamiento por ajuste fino del modelo “peruvian_efficientdet_d0_512x512”	74
Cuadro 19. Resultados de la evaluación del modelo “peruvian_efficientdet_d0_512x512”	76
Cuadro 20. Resultados de la evaluación de la inferencia del modelo “peruvian_efficientdet_d0_512x512” sobre la totalidad del dataset	77

Cuadro 21. Resultados del proceso de entrenamiento por ajuste fino del modelo “peruvian_efficientdet_d1_640x640”	79
Cuadro 22. Resultados de la evaluación del modelo “peruvian_efficientdet_d1_640x640”	82
Cuadro 23. Resultados de la evaluación de la inferencia del modelo “peruvian_efficientdet_d1_640x640” sobre la totalidad del dataset	83
Cuadro 24. Resultados del proceso de entrenamiento por ajuste fino del modelo “peruvian_ssd_mobilenet_v2_320x320”	85
Cuadro 25. Resultados de la evaluación del modelo “peruvian_ssd_mobilenet_v2_320x320”	87
Cuadro 26. Resultados del proceso de entrenamiento por ajuste fino del modelo “peruvian_ssd_mobilenet__fpn_640x640”	88
Cuadro 27. Resultados de la evaluación del modelo “peruvian_ssd_mobilenet__fpn_640x640”	90
Cuadro 28. Resultados de la evaluación de la inferencia del modelo “peruvian_ssd_mobilenet__fpn_640x640” sobre la totalidad del dataset	91
Cuadro 29. Resultados del proceso de entrenamiento por ajuste fino del modelo “peruvian_mobilenet_fpnlite_320x320”	94
Cuadro 30. Resultados de la evaluación del modelo “peruvian_mobilenet_fpnlite_320x320”	96
Cuadro 31. Resultados de la evaluación de la inferencia del modelo “peruvian_mobilenet_fpnlite_320x320” sobre la totalidad del dataset	97
Cuadro 32. Resultados del proceso de entrenamiento por ajuste fino del modelo “peruvian_mobilenet_fpnlite_640x640”	100
Cuadro 33. Resultados de la evaluación del modelo “peruvian_mobilenet_fpnlite_640x640”.	102
Cuadro 34. Resultados de la evaluación de la inferencia del modelo “peruvian_mobilenet_fpnlite_640x640” sobre la totalidad del dataset	103
Cuadro 35. Modelos entrenados y sus etiquetas para fines de comparación	106
Cuadro 36. Comparación de los modelos entrenados con relación el proceso de entrenamiento.	107
Cuadro 37. Comparación de los modelos entrenados con relación el proceso de evaluación de desempeño.	108

Cuadro 38. Comparación de los modelos entrenados con relación a la inferencia.	109
Cuadro 39. Evaluación integral de los modelos entrenados.	110
Cuadro 40. Detalles del prototipo implementado (Indicadores I2 e I3).	116
Cuadro 41. Resultados de los tiempos empleados por el modelo en la SBC Raspberry Pi 4 B+.	119
Cuadro 42. Resultados de los tiempos empleados por el modelo simplificado en la SBC Raspberry Pi 4 B+.	121
Cuadro 43. Comparación de resultados entre el modelo completo y el modelo simplificado.	122
Cuadro 44. Resultados de la evaluación de indicadores definidos en el numeral 3.3: Operacionalización de las variables.	127

Índice de Figuras

Figura 1. Modelo de Neurona propuesto por McCulloch y Pits en 1943.	13
Figura 2. Arquitectura General de un Perceptrón Multicapa.	15
Figura 3. Diseños de las de placas de rodaje ordinarias de Perú.	23
Figura 4. Diseños de las de placas de rodaje especiales de Perú.	23
Figura 5. Concepto del parámetro IoU.	31
Figura 6. Ejemplo de Etiquetado.	50
Figura 7. Distribución de Número de Placas por Imagen.	53
Figura 8. Distribución de número de veces que se repite una placa en el dataset.	54
Figura 9. Imágenes correspondientes a la placa que se repite 12 veces	55
Figura 10. Imágenes correspondientes a la placa que se repite 7 veces	56
Figura 11. Distribución de la relación de aspecto de las placas de rodaje etiquetadas.	57
Figura 12. Distribución de la relación de aspecto de las placas de rodaje etiquetadas.	58
Figura 13. Distribución de la relación de área de las placas de rodaje respecto al área de la imagen.	60
Figura 14. Placas con bajo contraste en el dataset	62
Figura 15. Distribución del enfoque de las placas de rodaje.	64
Figura 16. Placas vehiculares correspondientes a los intervalos con menores niveles de enfoque.	65
Figura 17. Muestreo de placas vehiculares en el intervalo con menor enfoque.	67
Figura 18. Curva de la tasa de entrenamiento considerado decaimiento por coseno.	73
Figura 19. Curvas de los errores durante el entrenamiento del modelo “peruvian_efficientdet_d0_512x512”	75
Figura 20. Falso positivo encontrado durante la evaluación de los tiempos de inferencia del modelo “peruvian_efficientdet_d0_512x512”.	78
Figura 21. Distribución de los tiempos de inferencia calculados para el modelo “peruvian_efficientdet_d0_512x512”	78

Figura 22. Curvas de los errores durante el entrenamiento del modelo “peruvian_efficientdet_d1_640x640”.....	81
Figura 23. Falsos positivos encontrados durante la evaluación de los tiempos de inferencia del modelo “peruvian_efficientdet_d1_640x640”	83
Figura 24. Distribución de los tiempos de inferencia calculados para el modelo “peruvian_efficientdet_d1_640x640”.....	84
Figura 25. Curvas de los errores durante el entrenamiento del modelo “peruvian_ssd_mobilenet_v2_320x320”	86
Figura 26. Curvas de los errores durante el entrenamiento del modelo “peruvian_ssd_mobilenet__fpn_640x640”.....	89
Figura 27. Falsos positivos encontrados durante la evaluación de los tiempos de inferencia del modelo “peruvian_ssd_mobilenet__fpn_640x640”.	92
Figura 28. Distribución de los tiempos de inferencia calculados para el modelo “peruvian_ssd_mobilenet__fpn_640x640”.....	93
Figura 29. Curvas de los errores durante el entrenamiento del modelo “peruvian_mobilenet_fpnlite_320x320”.....	95
Figura 30. Falsos positivos encontrados durante la evaluación de los tiempos de inferencia del modelo “peruvian_mobilenet_fpnlite_320x320”.	98
Figura 31. Distribución de los tiempos de inferencia calculados para el modelo “peruvian_mobilenet_fpnlite_320x320”	99
Figura 32. Curvas de los errores durante el entrenamiento del modelo “peruvian_mobilenet_fpnlite_640x640”.....	101
Figura 33. Falsos positivos encontrados durante la evaluación de los tiempos de inferencia del modelo “peruvian_mobilenet_fpnlite_640x640”	104
Figura 34. Distribución de los tiempos de inferencia calculados para el modelo “peruvian_mobilenet_fpnlite_640x640”	105
Figura 35. Imágenes generadas por el modelo detector en las que el OCR no pudo reconocer el número de placa.....	113
Figura 36. Imágenes en los que el número de placa reconocido tiene errores.	114
Figura 37. Prototipo implementado.....	117
Figura 38. Captura de pantalla indicando el resultado de la aplicación de prueba con el modelo completo.....	118

Figura 39. Captura de pantalla indicando el resultado de la aplicación de prueba con el modelo simplificado.....	120
Figura 40. Evaluación en interiores con smartphone.....	124
Figura 41. Evaluación en exteriores de día.....	125
Figura 42. Evaluación en exteriores de noche con vehículos.	126

Resumen

La identificación de vehículos en las calzadas de Perú es una tarea de gran relevancia. En atención a ello, el presente trabajo de tesis abordó la implementación de un sistema para el reconocimiento automático de placas de rodaje peruanas.

Se realizó un análisis de la problemática referida a la identificación de vehículos en las vías peruanas. Este análisis permitió identificar la necesidad de disposición de sistemas que puedan contribuir con la automatización de dicha tarea.

En el desarrollo de la investigación, se abordó un marco teórico que abarca los principales conceptos referidos a la inteligencia artificial y el aprendizaje automático y profundo. Asimismo, se revisaron antecedentes relacionados a nivel nacional e internacional.

Luego de ello, se definieron las variables con las cuales se realizó la evaluación de las hipótesis planteadas. Estas variables se definieron con la finalidad de establecer criterios objetivos y estandarizados, y cuya medición permitió realizar comparaciones.

Durante la ejecución de la investigación, se realizó el entrenamiento de modelos, para luego definir el modelo con las mejores prestaciones, para lo cual se emplearon fotografías de placas de rodaje obtenidas de entornos no controlados, con lo cual se pudo incrementar la robustez del sistema, obteniéndose una precisión de hasta el 96.9% sobre más de 350 fotografías.

El modelo final se integró con una solución de hardware a nivel de prototipo, quedando demostrado que es posible desarrollar un sistema automático para el reconocimiento de placas de rodaje peruanas empleando inteligencia artificial.

Palabras Clave: Inteligencia artificial, reconocimiento de placas de rodaje, entrenamiento, modelos, aprendizaje profundo.

Abstract

The identification of vehicles on roadways in Peru is a task of considerable importance, given its implications for traffic management, law enforcement, and public safety. In response to this, the present thesis examines the implementation of a system designed for the automatic recognition of Peruvian license plates.

An in-depth analysis of the challenges associated with vehicle identification on Peruvian roadways was conducted. This analysis underscored the pressing need for systems capable of automating the task, which would significantly enhance efficiency and accuracy in vehicle tracking and monitoring processes.

The research incorporates a comprehensive theoretical framework that explores the fundamental concepts of artificial intelligence, machine learning, and deep learning. It also includes a review of related studies at both national and international levels, providing context and highlighting advancements in the field.

Following this theoretical groundwork, specific variables were defined to evaluate the hypotheses formulated in the study. These variables were carefully selected to establish objective and standardized criteria, allowing for precise measurement and enabling meaningful comparisons.

The research methodology involved training multiple models to identify the one with the highest performance, for which is was used a dataset of license plate photographs captured in uncontrolled environments, simulating real-world conditions. This approach enhanced the system's robustness, resulting in an accuracy rate of up to 96.9% based on the analysis of more than 350 images.

The final model was integrated with a hardware solution at a prototype level, demonstrating the feasibility of developing an automated system for the recognition of Peruvian license plates using artificial intelligence.

Keywords: Artificial intelligence, license plate recognition, training, models, deep learning.

I. INTRODUCCIÓN

1.1. Introducción

El desarrollo de modelos basados en inteligencia artificial ha experimentado un crecimiento vertiginoso en los últimos años, permitiendo su aplicación en diversas áreas y campos de la vida humana.

La visión artificial, la cual se puede definir como la aplicación de técnicas computacionales para emular la visión humana con la finalidad de realizar tareas sobre información de tipo visual, es una de las áreas en las que este desarrollo ha sido notable. Por ejemplo, se tienen investigaciones que abordan el uso de la visión artificial en sistemas de transporte inteligentes o autónomos (Dilek y Dener, 2023).

De las diversas tareas que pueden abordarse mediante herramientas de visión artificial, el presente trabajo tesis se enfocó en dos: la detección de objetos y el reconocimiento de caracteres. Se eligieron estas tareas con la finalidad de desarrollar un sistema que pueda reconocer de manera automática el número de placa de rodaje de un vehículo peruano en un archivo de información visual (ya sea en formato de fotografía o video), en el marco de una problemática asociada con la identificación de vehículos en las vías públicas terrestres del Perú, la cual se sustenta principalmente en la necesidad del Estado Peruano de contar con herramientas que faciliten el monitoreo y control de los vehículos en las calzadas, el monitoreo y control de infracciones y robo de vehículos, entre otros.

En ese sentido, en la presente tesis se abordó el entrenamiento de modelos basados en inteligencia artificial para la ejecución de dichas tareas. Se realizó el análisis de rendimiento de estos modelos con el objetivo de determinar el más idóneo para la finalidad indicada. Finalmente, con el propósito de determinar su viabilidad práctica, se aplicó el modelo con mejor performance en una implementación a manera de prototipo, empleando para ello la tarjeta de desarrollo Raspberry Pi.

1.2. Planteamiento del problema

1.2.1. Determinación del problema

En el Perú, de acuerdo la Ley General de Transporte y Tránsito Terrestre, “todo vehículo de transporte automotor que circule por vías públicas está obligado a exhibir la placa única nacional de rodaje” (Congreso de la República del Perú, 1999, Ley N. ° 27181, art. 32).

Asimismo, la placa única nacional de rodaje es definida como “el elemento de identificación de los vehículos durante la circulación de éstos por las vías públicas terrestres” (Ministerio de Transportes y Comunicaciones del Perú [MTC], 2008, D.S. N.° 017-2008-MTC, art. 4).

Por otro lado, mediante el D.S. N.° 006-2010-MTC, el cual modifica al Reglamento de Placa Única Nacional de Rodaje, se crea el Sistema Integral de Identificación Vehicular. Este sistema se define como un conjunto de elementos tecnológicos que permite la identificación inmediata e integral a todos los vehículos que circulen por las vías públicas terrestres del Perú (MTC, 2010, D.S. N.° 006-2010-MTC, art. 12).

La incorporación del Sistema Integral de Identificación Vehicular dentro del marco jurídico peruano refleja la necesidad de la administración pública peruana de contar con mecanismos que permitan identificar vehículos que circulen en las vías públicas terrestres, de manera inmediata e integral.

La identificación de vehículos presenta una problemática bastante amplia. Por ejemplo, en el portal del Sistema Nacional de Información Ambiental del Perú [SINIA] (s. f.), referido al Número de vehículos por cada mil habitantes al año 2016, se enuncian limitaciones para identificar las movilizaciones de los vehículos en ámbitos distintos adonde se registraron.

El robo de vehículos, por otro lado, es otra de las problemáticas relacionadas con la identificación vehicular. De acuerdo con estadísticas del Instituto Nacional de Estadística e Informática [INEI], (s. f.-a), en el año 2021 se registraron un total de 19,991 denuncias de robos vehiculares en el Perú, de las cuales un 68% corresponden al departamento de Lima. El mismo reporte señalar, que para ese mismo año, el número de vehículos recuperados fue de

12,108, de lo cual se obtiene que un total de 7,783 vehículos no pudieron ser recuperados. El disponer de sistemas que permitan identificar vehículos robados en las vías públicas terrestres permitiría a la Policía Nacional del Perú incrementar el número de vehículos recuperados.

Del mismo modo, la identificación de los vehículos permite a la administración pública identificar vehículos que han cometido infracciones de tránsito. Al respecto, de acuerdo con otro reporte de estadísticas del INEI (s. f.-b), al año 2021, se registraron 763,944 infracciones en todo el Perú. La identificación de estos vehículos permite que la administración pública lleve a cabo su rol fiscalizador y sancionador. Además, la información que se puede generar a partir de la identificación de vehículos infractores puede contribuir con la definición de políticas orientadas a la reducción de estas infracciones.

Asimismo, es importante señalar que el parque automotor peruano se ha mantenido en constante crecimiento durante una década, sumando al año 2019 más de 700,000 vehículos nuevos (Superintendencia Nacional de Registros Públicos del Perú, 2020). Solamente en el departamento de Lima, al año 2020, se estimó que el parque automotor superó los 2 millones de unidades vehiculares terrestres (Lima Cómo Vamos, 2022, p. 12).

De este modo, la gran cantidad de vehículos circulando por las vías terrestres peruanas, considerando la problemática asociada con ello, resalta la pertinencia de que el proceso de identificación vehicular disponga de mecanismos tecnológicos capaces de automatizar dicha tarea, toda vez que sin dichos mecanismos la cantidad de logística, así como los recursos económicos y humanos requeridos para dicho fin harían insostenible su realización, afectando de este modo la capacidad de administración pública del Estado Peruano.

1.2.2. Formulación del problema

El problema que se abordó en esta investigación, a partir de los puntos descritos en el apartado 1.2.1, que sustentan la necesidad de implementar sistemas automáticos que permitan la identificación de placas de rodaje peruanas, se formuló del modo siguiente:

- ¿Cómo desarrollar un sistema automático que permita la identificación de placas de rodaje peruanas en el departamento de Lima, en el año 2023?

Adicionalmente, sobre este problema principal se plantearon los siguientes problemas específicos:

- a. ¿Es posible realizar el entrenamiento de un sistema automático que permita el reconocimiento de placas de rodaje peruanas en el departamento de Lima, en el año 2023?
- b. ¿Es posible diseñar un subsistema de software para implementar un sistema automático que permita el reconocimiento de placas de rodaje peruanas en el departamento de Lima, en el año 2023?
- c. ¿Es posible integrar un subsistema de hardware para implementar un sistema automático que permita el reconocimiento de placas de rodaje peruanas en el departamento de Lima, en el año 2023?

1.3. Objetivos

1.3.1. Objetivo principal

Para abordar el problema principal formulado en el apartado previo, se estableció el siguiente objetivo principal:

- Desarrollar un sistema automático para el reconocimiento de placas de rodaje peruanas en el departamento de Lima, en el año 2023.

1.3.2. Objetivos específicos

Adicionalmente, los objetivos específicos que se plantearon para el logro del objetivo principal son los siguientes:

- a. Realizar el entrenamiento de modelos basados en inteligencia artificial para realizar el reconocimiento de placas de rodaje peruanas en el departamento de Lima en el año 2023.

- b. Diseñar un subsistema de software para la implementación de un sistema automático que permita el reconocimiento de placas de rodaje peruanas en el departamento de Lima, en el año 2023.
- c. Integrar un subsistema de hardware para la implementación de un sistema automático que permita el reconocimiento de placas de rodaje peruanas en el departamento de Lima, en el año 2023.

1.4. Importancia y alcance de la investigación

1.4.1. Importancia de la investigación

El desarrollo de mecanismos para la identificación automática y eficiente de vehículos en vías públicas, a través del reconocimiento de sus placas de rodaje a partir de fotografías o videos obtenidos en campo, permitirá que el Estado Peruano lleve a cabo su rol de administrador con mayor eficiencia.

A continuación, se detalla la relevancia específica de la presente investigación desde tres enfoques principales:

- a. Importancia metodológica: La presente tesis desarrolla una metodología basada en la implementación de modelos de inteligencia artificial para abordar el reconocimiento de placas de rodaje.
- b. Importancia teórica: La presente investigación sirve como punto de partida para futuras investigaciones o desarrollos relacionados con el reconocimiento de placas de rodaje en el Perú. Asimismo, se proporciona el modelo detector de placas de rodaje peruanas para su uso en desarrollos posteriores.
- c. Importancia práctica: La presente investigación aporta un sistema basado en inteligencia artificial para el reconocimiento de placas de rodaje en el Perú de manera automática que permita que los procesos de la administración pública relacionados se beneficien en términos de cobertura, tiempo de respuesta, entre otros aspectos.

1.4.2. Alcance de la investigación

La presente investigación se circunscribe al desarrollo de un sistema basado en inteligencia artificial para la identificación de placas de rodaje en el Perú, empleando un muestreo de placas de Lima Metropolitana.

1.5. Limitaciones de la investigación

Para el desarrollo de la presente tesis, se identificaron las siguientes limitaciones:

- a. Limitación en el número y variedad de muestras fotográficas de placas de rodaje con etiquetas para entrenamiento y validación puesto que, al no disponerse de una base de datos libre con este tipo de información, fue necesaria su recolección como parte de la investigación. Por este motivo, la cantidad y variedad de muestras se vio limitada por el tiempo y recursos disponibles para dicha tarea.
- b. Limitación de la potencia computacional para el proceso de entrenamiento, la cual define los parámetros de entrenamiento de manera que su realización sea viable en términos de tiempo y costo.

II. REVISIÓN DE LITERATURA

2.1. Marco teórico

Habiéndose planteado el problema de investigación en el capítulo anterior, se abordó la sustentación teórica del estudio, proceso que es denominado desarrollo de la perspectiva teórica (Hernández et al., 2014, p. 60). En ese sentido, en el presente apartado se desarrolla el marco teoría en el que se soportó la presente investigación.

Definición de inteligencia artificial

La definición de la inteligencia artificial depende de muchos aspectos tales como el contexto histórico, los fines (académicos, investigación, comerciales, industriales, legales), entre otros.

Con relación al término en sí, su primer uso en un contexto de investigación es acuñado al taller propuesto en el año 1955 por John McCarthy, Marvin Minsky, Nathaniel Rochester y Claude Shannon, denominado “Dartmouth Summer Research Project on Artificial Intelligence (DSRPAI) [”Proyecto de verano en inteligencia artificial de Dartmouth”], el cual definía los siguientes problemas a ser abordados por la inteligencia artificial: computadoras automáticas, ¿cómo una computadora puede programar para usar un lenguaje?, redes de neuronas, la teoría del tamaño de un cálculo, automejora, abstracciones y la aleatoriedad y creatividad (McCarthy et al., 1955). No obstante, desde el año 1950, tras la publicación del artículo “Computing machinery and intelligence [Maquinaria de computación e inteligencia]”, en el que se propone por primera vez la pregunta: ¿pueden las máquinas pensar? (Turing, 1950), los problemas, métodos y aplicaciones de la inteligencia artificial se han ido actualizando constantemente, siguiendo una evolución marcada por diversas tendencias en la investigación y el desarrollo hasta el surgimiento de los conceptos modernos de aprendizaje automático y aprendizaje profundo, los cuales han permitido muchas de las aplicaciones tecnológicas que se conocen en la actualidad.

Esta diversidad de aplicaciones y tendencias no permite una definición moderna y estándar de inteligencia artificial, por lo cual existen muchas definiciones que dependen de los contextos en los que fueron enunciadas. Por ejemplo, Deng (2018) define a la inteligencia artificial como una rama de las ciencias de la computación y una tecnología cuyo objetivo es desarrollar las teorías, métodos, algoritmos y aplicaciones para simular y extender la inteligencia humana.

Asimismo, en un reporte AI Watch¹ se propone una definición sencilla para inteligencia artificial, concebida como la inteligencia de las máquinas, o la inteligencia demostrada por las máquinas, diferenciándola así de la inteligencia humana (Delipetrev et al., 2020, p. 5).

No obstante, una definición más completa de inteligencia artificial es propuesta en otro reporte de AI Watch, en el cual se indica que los sistemas de la inteligencia artificial (AI) son sistemas de software (y posiblemente hardware) diseñados por humanos que, dado un objetivo complejo, actúa en las dimensiones físicas o digitales por medio de la percepción de su entorno a través de la adquisición de datos, la interpretación de datos recolectados de forma estructura o no estructurada, el razonamiento sobre el conocimiento (o el procesamiento de información) derivado de esta data, de manera que decida la mejor o mejores acciones a seguir para el logro del objetivo dado (Samoili et al., 2020, p. 14).

Esta última definición permite concluir que un sistema basado en inteligencia artificial debe contar con los mecanismos para obtener datos de su entorno, procesarlos y a partir de ello, realizar acciones orientadas al cumplimiento de objetivos específicos. De este modo, esta definición es la que se tomó como marco de referencia para el desarrollo de la presente investigación.

Dominios de la inteligencia artificial

Los dominios de la inteligencia artificial hacen referencia a los ámbitos y especializaciones en los cuales se investiga, desarrolla o hace uso de la inteligencia artificial en la actualidad. En ese sentido, estos dominios son

¹ AI Watch es una iniciativa de la Comisión Europea encargada de monitorear el desarrollo, uso e impacto de la inteligencia artificial en Europa, iniciada en el año 2018.

dinámicos, por lo cual su definición debe actualizarse atendiendo a cualquier avance disruptivo.

Al respecto, una propuesta de AI Watch, denominada taxonomía de la inteligencia artificial plantea la existencia de dos tipos de dominio: unos principales directamente relacionados con los objetivos más actuales de la inteligencia artificial, y unos transversales que están indirectamente relacionados con la inteligencia artificial. En el Cuadro 1 se pueden apreciar los detalles de esta propuesta (Samoili et al., 2020, p. 11).

Cabe precisar que los dominios y subdominios no son excluyentes. Como se señaló en la definición de referencia, un sistema basado en inteligencia artificial debe tener la capacidad de obtener de datos de su entorno, procesarlos y, además, realizar acciones para el logro de objetivos específicos y complejos. En atención a ello y considerando los objetivos de la presente investigación, se resaltan los dominios principales de aprendizaje (uso de aprendizaje automático para el entrenamiento orientado a reconocer placas de rodaje) y percepción (el objetivo de visión artificial es reconocer placas de rodaje a partir de datos de imagen o video), y los dominios transversales de integración e interacción (integrar el modelo desarrollado en un sistema de múltiples agentes).

Cuadro 1

Taxonomía de la Inteligencia Artificial propuesta por Watch IA

Tipo de dominio y dominio de IA	Subdominio de IA
Dominios principales	
Razonamiento	Representación del conocimiento
	Razonamiento automático
	Sentido común
Planeamiento	Planificación y programación
	Búsqueda
	Optimización
Aprendizaje	Aprendizaje automático
Comunicación	Procesamiento natural del lenguaje (NPL)
Percepción	Visión artificial o por computador
	Procesamiento de audio
Dominios transversales	
Integración e interacción	Sistemas multi agentes
	Robótica y automatización
	Vehículos conectados y autónomos
Servicios	Servicios de la IA
Ética y filosofía	Ética de la IA
	Filosofía de la IA

Fuente. Adaptado de AI Watch. Defining Artificial Intelligence. Towards an operational definition and taxonomy of artificial intelligence de Samoili et al., 2020, p. 11.

Aprendizaje automático o machine learning

El aprendizaje automático, término proveniente de la expresión en inglés machine learning, de forma similar que el concepto mismo de inteligencia artificial tiene múltiples definiciones.

De este modo, en 1994, un sistema que aplica aprendizaje automático se definió como el sistema que se especializa en resolver problemas a partir de la experiencia acumulada adquirida de casos o problemas resueltos previamente (Moreno et al., 1994, p. 7).

En ese mismo año, Michie et al. (1994/2009, p. 2) señalaron que el aprendizaje automático comprendía procedimientos computacionales automáticos basados en operaciones lógicas o binarias, que aprenden a realizar una tarea a partir de una serie de ejemplos.

Una definición más reciente del aprendizaje automático la propone Zhou, (2016/2021), quien la concibió como la técnica que permite la mejora en el desempeño de un sistema aprendiendo de la experiencia a través de métodos computacionales.

La comparación de estas definiciones permite identificar que el aprendizaje automático es un proceso cuya finalidad es lograr que un sistema realice una tarea (mejorando su desempeño en su ejecución) a partir de un proceso de aprendizaje centrado en la experiencia (ejemplos).

De este modo, es posible definir al aprendizaje automático como el subdominio de la inteligencia artificial que consiste en el aprendizaje de la experiencia para la mejora en el desempeño en la realización de una o varias tareas.

Este componente de experiencia hace referencia a datos que contengan información sobre la tarea o tareas cuya ejecución se busca mejorar. En ese contexto surge el concepto de dataset (que puede traducirse como set o conjunto de datos), el cual se puede definir como una estructuración de datos que aportan información sobre una tarea específica, para su posterior uso en un proceso de aprendizaje automático.

Con relación al aprendizaje, este puede concebirse como el procedimiento mediante el cual se construye un modelo a partir de los datos disponibles. Este proceso de aprendizaje, que es llevado a cabo mediante algoritmos, se puede denominar entrenamiento.

A la fecha, existe una amplia variedad de algoritmos de aprendizaje automático, cada uno con características acordes al tipo de problema que se busca solucionar. No obstante, pueden clasificarse del modo siguiente:

- a. Algoritmos de aprendizaje supervisado: cuando las salidas del sistema se van comparando, empleando alguna métrica preestablecida, con las salidas esperadas para los datos aplicados en función a una estrategia de medición de proximidad, es decir, qué tan lejos o cerca se encuentra la salida del sistema respecto a la esperada. A las salidas esperadas empleadas por estos algoritmos se denomina etiquetas. Una aplicación de estos algoritmos es el reconocimiento del habla (Schuster, 1999).
- b. Algoritmos de aprendizaje no supervisado: cuando el entrenamiento se orienta a realizar un ordenamiento de los datos en función a características propias de los mismos. Este tipo de algoritmos de entrenamiento no requiere que los datos estén etiquetados. Una aplicación de estos algoritmos es el minado de textos para investigaciones (Janasik et al., 2009).
- c. Algoritmos de aprendizaje mixto: que emplean en su entrenamiento datos con y sin etiquetas, por lo cual también se denominan algoritmos de aprendizaje semi supervisado.
- d. Algoritmos de aprendizaje por reforzamiento: cuando el algoritmo está planteado de tal forma que al recibir un feedback del exterior, actualiza el modelo inicial generado a partir de los datos, de manera que ante una entrada ya conocida tienda a producir una salida que sea calificada de una mejor manera, en función de alguna métrica preestablecida. En cuanto a sus aplicaciones, Nian et al., (2020) hacen una revisión de las aplicaciones de estos algoritmos en los procesos de control industrial.

Redes neuronales artificiales

Uno de los principales modelos de aprendizaje automático son las redes neuronales artificiales. Estas, como su nombre lo indican, se basan en el funcionamiento de las neuronas en el cerebro humano

El primer intento por realizar un modelamiento matemático de la función neuronal (y, por lo tanto, pasible de implementarse de manera computacional) fue llevado a cabo en el año 1943, y proponía una neurona básica cuya salida era del tipo todo o nada, además de introducir muchos conceptos que se siguen usando hasta la actualidad (McCulloch y Pitts, 1943). En la Figura 1 se pudo apreciar este modelo pionero de neurona, que pasó a denominarse, en honor a sus creadores, como modelo de McCulloch-Pits.

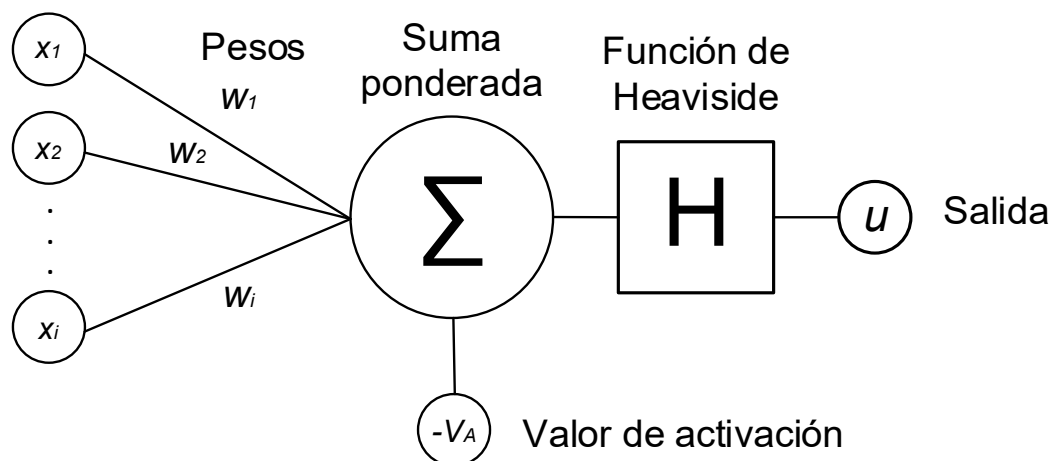


Figura 1. Modelo de Neurona propuesto por McCulloch y Pitts en 1943.

Fuente: Elaborado a partir de McCulloch y Pitts, 1943.

El modelo de McCulloch-Pits, en ese sentido, señala que su salida (variable “u”), depende de la suma ponderada de los valores en sus entradas (variable “x”) con unos pesos (variable “w”), la cual luego es comparada con un valor de activación θ . La salida “u” en este caso viene dada por una función de activación del tipo salto o de Heaviside (todo o nada). Esto puede presentarse de manera matemática con la Ecuación 1.

$$u = H\left(\sum_i w_i \times x_i - \theta\right) \quad (1)$$

Más adelante, en 1957, a partir del modelo de McCulloch-Pits se creó el perceptrón (Rosenblatt, 1957), el cual es considerado el primer modelo de neurona artificial capaz de entrenarse en función de valores esperados en su salida. Por otro lado, en 1960, se propuso un modelo de neurona denominado Adaline (Adaptive Linear [Adaptativo Lineal]), el cual considera un método alternativo de entrenamiento de una neurona artificial (Widrow y Hoff, 1960).

No obstante, uno de los principales avances llegaría 02 años después. En 1962 se presenta por primera vez una red de neuronas artificiales con la capacidad de entrenarse en su conjunto. Estas neuronas estaban basadas en los perceptrones, por lo cual a la red se le denominó perceptrón multicapa (Rosenblatt, 1962), con lo cual se sentarían las bases de lo que más adelante se denominaría aprendizaje profundo.

El perceptrón multicapa, a nivel de arquitectura, está conformado por una capa de entrada, que recibe los datos del entorno, capas intermedias, denominadas capas ocultas, y las salidas. Cada capa está conformada por perceptrones, cada uno con una función de activación determinada, y con sus valores de activación que pasan a denominarse bias. En la Figura 2, se presenta la topología de un perceptrón multicapa, el cual también es denominado como red neuronal de retroalimentación.

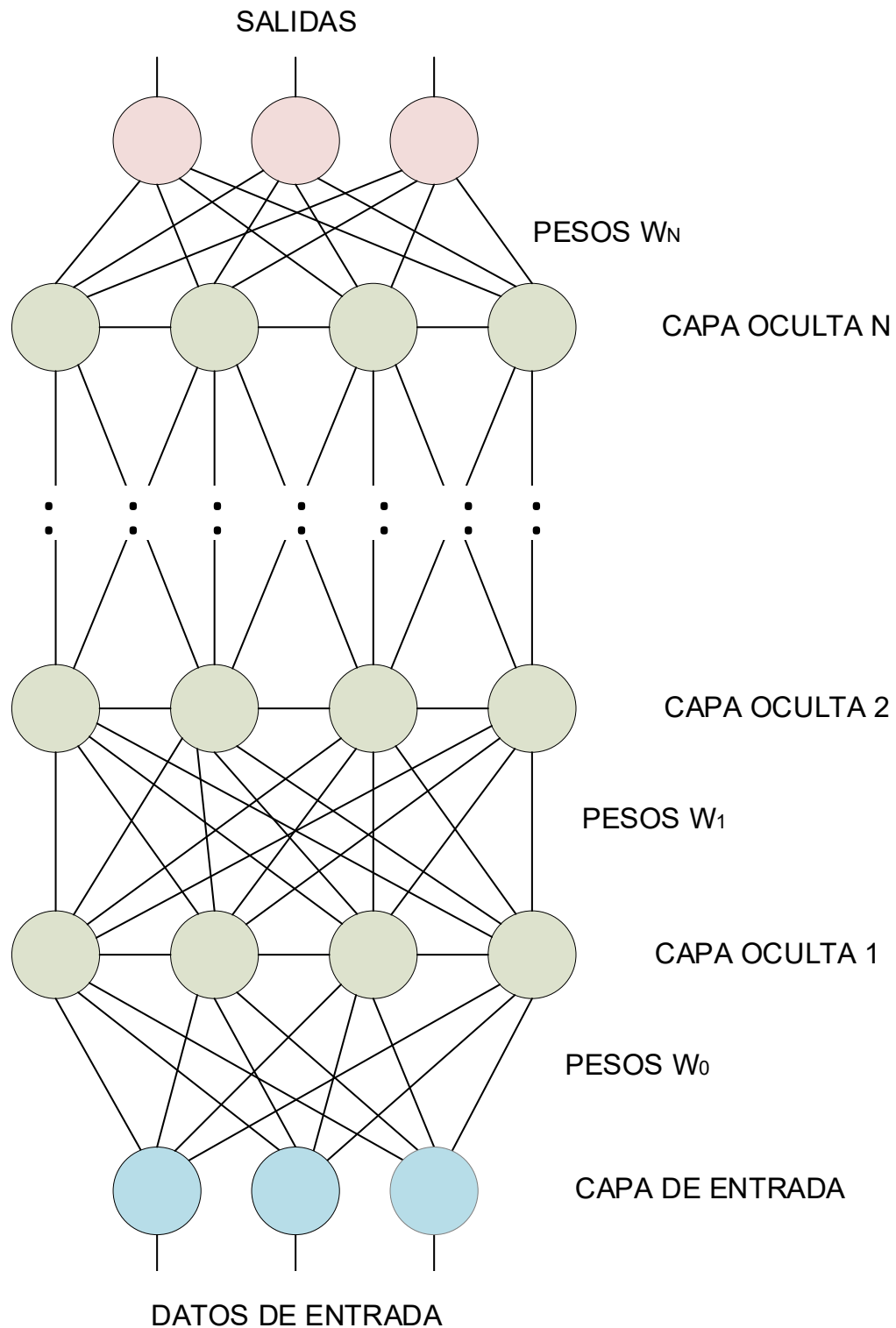


Figura 2. Arquitectura General de un Perceptrón Multicapa.

Fuente: Elaboración propia a partir de la definición de Rosenblatt (1962).

El entrenamiento de estas redes consiste principalmente en ajustar los valores de los pesos que conectan las neuronas entre capas, de manera que las

salidas de la red se optimicen para cumplir un objetivo específico, como clasificar una imagen dentro de una categoría determinada.

En cuanto a los algoritmos desarrollados para entrenar estas redes, uno de los más empleados es el denominado back-propagation (propagación hacia atrás). Como su nombre sugiere, este algoritmo está diseñado para actualizar los pesos que conectan las neuronas entre capas, comparando las salidas con respecto a un patrón de referencia. Para realizar esta optimización de la salida, se emplea una métrica de error basada en una función que se denomina función de pérdida (loss function), como la función de error cuadrático promedio. Este error se emplea para calcular gradientes, los cuales permiten actualizar los pesos entre las capas, de manera que se minimice el error de la salida de la red con respecto a un patrón o referencia (target). Este proceso se caracteriza por realizar los cálculos de actualización de los pesos desde la última capa hacia la capa de entrada. Por este motivo, este método se denomina propagación hacia atrás. El algoritmo culmina cuando el valor del error converge en un mínimo, que puede ser local o absoluto.

Aprendizaje profundo o deep learning

El aprendizaje profundo es el subcomponente del aprendizaje automático que se encarga de modelar características de alto nivel de datos empleando múltiples capas de neuronas artificiales, relacionadas con operaciones matemáticas complejas o no lineales. Se emplea el término de aprendizaje profundo en la medida de que si se desea realizar una representación gráfica de las capas que conforman la red, el gráfico resultante es extenso o profundo (Goodfellow et al., 2016, p. 2).

En ese sentido, como se señaló en el numeral previo, los perceptrones multicapa se pueden considerar como los primeros sistemas que aplicaron el concepto de aprendizaje profundo.

Cada capa de una red que emplea aprendizaje profundo codifica la información de sus entradas en un nivel de abstracción superior al de la capa previa. Estas redes se emplean para la realización de tareas complejas de reconocimiento de patrones en imágenes, la clasificación, detección de

objetos y la segmentación de objetos en imágenes; reconocimiento de voz, procesamiento del lenguaje natural, entre otras.

Uno de los algoritmos de entrenamiento de redes basado en aprendizaje profundo es el back-propagation, empleando el algoritmo de descenso de gradiente como método para minimización del error (Shrestha y Mahmood, 2019). En cuanto a las principales arquitecturas de redes basadas en aprendizaje profundo, las más conocidas son las redes neuronales convolucionales, los autoencoders, las máquinas restringidas de Boltzmann (RBM) y las redes de Memoria de Largo y Corto Plazo o LSTM. Sin embargo, estos últimos vienen siendo superados en rendimiento por modelos denominados Transformadores (Nadendla, 2020).

Redes neuronales convolucionales (CNN o ConvNets)

Uno de los modelos más importantes basados en aprendizaje profundo, para los fines de la presente investigación, es el denominado modelo de red neuronal convolucional (CNN) o *ConvNet*. Estas redes neuronales son diseñadas para para procesar datos de entrada en la forma de arreglos o matrices (LeCun et al., 2015, p. 439), realizando operaciones matriciales internamente en sus capas ocultas. Por ello, este tipo de red neuronal es ampliamente usado para tareas que involucran imágenes (como clasificación, detección de objetos, entre otras) debido a que su configuración permite codificar características de las imágenes en su propia arquitectura (O'Shea y Nash, 2015).

En lo que respecta a su implementación experimental, Li et al., (2022) propusieron un conjunto de recomendaciones orientadas a la implementación de CNN:

- a. En lo que respecta a las funciones de activación de las neuronas que conforman una CNN, recomiendan emplear funciones sigmoideas para problemas de clasificación binarios y funciones del tipo softmax (función exponencial normalizada) para problemas de clasificación múltiple. Para casos generales, la función de activación recomendada es la denominada ReLU (Rectified Linear Unit).

- b. En lo que respecta a la función de pérdida, si bien se señala que el uso específico de algún tipo particular de función de pérdida depende de la tarea específica a abordar, para las tareas de regresión las funciones del tipo L1 (basado en el error absoluto promedio) y L2 (basado en el error cuadrático promedio) son más empleadas, así como las funciones de entropía cruzada para redes que empleen funciones de activación en su última capa del tipo softmax.
- c. Para la reducción del error, existen diversos optimizadores, cuya aplicación depende del tipo de datos disponibles para el entrenamiento, así como de las tareas a realizar.
- d. Para el ajuste de hiperparámetros², algunas estrategias consideran: iniciar con una tasa de aprendizaje o learning rate cuyo valor vaya disminuyendo conforme se realizan las iteraciones del entrenamiento; seleccionar un refresco o epoch lo suficientemente grande considerando los recursos de hardware disponibles y teniendo en cuenta que un valor demasiado grande podría dar a lugar a un problema de sobreajuste u overfitting³. Con relación al tamaño de lotes de muestra o batch size, se recomienda elegir el tamaño más grande que se permite considerando la memoria disponible, en la medida de que, a mayor tamaño, se requiere que más información sea trasladada a la memoria. No obstante, un tamaño de lote grande permite que al algoritmo alcance un punto óptimo de una manera más sencilla.

En cuanto a arquitecturas o modelos de redes convolucionales, una de las más importantes para los fines de la presente investigación es la denominada MobileNets (Howard et al., 2017). Este modelo se caracteriza por ser de tamaño ligero y converger con mayor rapidez que otros modelos. Este modelo,

² Los hiperparámetros de un modelo son las variables que determinan el modo en el que una red es entrenada, así como parámetros de su arquitectura. Incluyen, entre otros: la tasa de aprendizaje, tamaño de lotes de muestras o batch size, refrescos o epochs, entre otros.

³ El sobreajuste u overfitting se da cuando una red se ajusta en demasía a los valores esperados por los datos con los que es entrenado, lo cual hace que tenga muchas dificultades para generalizar (generar salidas coherentes ante datos nuevos).

a su vez, permite abordar tareas de reconocimiento de objetos aplicando procesos de ajuste preciso o fine-tuning.

2.2. Antecedentes del estudio

El reconocimiento automático de placas consiste en identificar, dada una imagen, la ubicación de la placa vehicular y una vez realizado ello, reconocer los caracteres que conforman el código o número de la placa. Para dicho objetivo, a lo largo del tiempo se han ido proponiendo y desarrollando diversas técnicas.

En el presente apartado de la tesis se aborda la revisión de los antecedentes del estudio, tanto internacionales como nacionales, de tal forma que dicho análisis permita contar con un diagnóstico del estado del arte actual referido a la presente investigación.

2.2.1. Antecedentes internacionales

Shashirangana et al., (2021) realizó un estudio del estado del arte referente a la detección y el reconocimiento de placas vehiculares, proponiendo además una clasificación de las mismos: por un lado se tienen las técnicas basadas en visión por computador tradicional y procesamiento de imágenes (detección de bordes, análisis de color, texturas, etc.) y, por el otro lado, técnicas basadas en inteligencia artificial empleando clasificación estadística como las máquinas de soporte vectorial (SVM), y clasificación por aprendizaje profundo empleando para ello modelos CNN, modelos pre entrenados para la detección de objetos, entre otros.

Raghunandan et al., (2018) abordó el problema de imágenes de placas vehiculares de baja calidad, las cuales implicaron dificultades en su reconocimiento, empleando para ellos filtros basados en cálculo fraccional. La detección y reconocimiento de caracteres es abordada mediante una adaptación del detector MSER (Donoser y Bischof, 2006), obteniendo una precisión de hasta 84.6% para la detección de las placas, y hasta 76.3% en la detección y reconocimiento de los caracteres de la placa.

Por su parte, Zou et al. (2018) abordaron la tarea de detección de placas en imágenes en entornos complejos mediante el uso de CNN, especializando un modelo basado en AlexNet (Krizhevsky et al., 2017). Las salidas de este modelo, candidatos a placas vehiculares, luego son procesadas para, a partir de una cuantificación de los bordes existentes, determinar si el candidato corresponde a una placa vehicular o no, logrando una precisión del 98.1% en la detección de placas.

De manera similar, Montazzolli y Rosito (2018) abordaron la detección y reconocimiento de placas vehiculares en entornos no controlados, empleando para ello un nivel conformado por una red basada en YOLOv2, denominada WPOD-NET, implementada a partir de la integración de los modelos YOLO, SSD y STN (Jaderberg et al., 2015), cuya finalidad fue validar las salidas del modelo. La capa de reconocimiento de caracteres se desarrolló a partir de un modelo pre entrenado de YOLO, al cual se le hizo un entrenamiento adicional con un conjunto de datos más amplio. Los autores reportan una eficacia de su sistema del 89.33%, evaluado sobre 5 datasets distintos.

Asimismo, Kessentini et al., (2019) propusieron un método de dos etapas para el reconocimiento de placas vehiculares de distintos países. La primera etapa consistió en un detector basado en YOLOv2. La segunda etapa, en la que se realiza la detección de los caracteres que conforman el número de placa, se desarrolló a partir de dos modelos basados en YOLOv2 y en el uso de una CNN recurrente. Una de las características de esta investigación es que considera imágenes de vehículos obtenidas en entornos no controlados. Este método es similar al seguido por Laroca et al., (2019), quienes abordaron la tarea en 3 etapas: ubicación del vehículo, detección de la placa en el vehículo y reconocimiento de los caracteres de la placa. En estas 3 etapas, los investigadores emplean modelos basados en YOLO.

Por otro lado, Han et al., (2020) realizaron el entrenamiento de un modelo basado en YOLOv2, con la particularidad de que el dataset empleado durante el entrenamiento estuvo conformado por imágenes sintéticas de placas, las cuales fueron generadas por un modelo basado en inteligencia artificial. De este modo, a partir de 159 imágenes de placas, pudieron implementar un

dataset con miles de muestras que se emplearon durante el entrenamiento, logrando una eficacia del 98.72%.

Como se puede colegir de los antecedentes descritos, los principales enfoques para abordar la tarea de reconocer placas vehiculares se basan en redes que emplean aprendizaje profundo, esto principalmente porque sus resultados superan a los resultados obtenidos mediante técnicas de procesamiento de imágenes tradicionales. Asimismo, puede encontrarse que en su mayoría se empleó el modelo base YOLOv2. No obstante, si bien este modelo es de inferencia rápida, su precisión suele ser menor que otros modelos como el SSD (Sebastian, 2022).

2.2.2. Antecedentes nacionales

El reconocimiento de placas vehiculares en Perú ha sido abordado, principalmente, mediante una combinación de métodos basados en procesamiento de imágenes tradicionales e inteligencia artificial.

Por ejemplo, Diaz (2006) desarrolló un método que emplea la detección de bordes, análisis estadístico vertical y horizontal, y transformaciones morfológicas para la localización de la placa dentro de la imagen; y una red neuronal para el reconocimiento de los caracteres del número de la placa detectada.

Un método similar fue seguido por Espinoza (2014), quien empleó una técnica de detección de bordes, binarización, operaciones morfológicas y detección de contornos para la localización de la placa vehicular.

Chu (2019) propuso un método de reconocimiento de placas para el registro de vehículos de un hospedaje empleando un método de segmentación basado en el algoritmo de Haar-Cascade (Viola y Jones, 2001) y reconocimiento de caracteres con la librería Tesseract, la cual al momento de elaboración de la presente investigación se encuentra en situación de código libre.

En cuanto a la aplicación de técnicas basadas en aprendizaje profundo, Ramírez y Tito (2020) realizaron el diseño de una CNN para el reconocimiento

de placas, obteniendo una efectividad de hasta 95% sobre un total de 20 placas consideradas durante el entrenamiento.

Un enfoque alternativo fue propuesto por Aguilar (2022), quien no desarrolló un algoritmo específico para la detección de placas vehiculares, sino más bien empleó una solución comercial denominada Plate Recognizer⁴, la cual proporciona una API para su integración en otros sistemas. Al respecto, se señala que esta solución implica costos por el uso de la API (no es software libre).

Es importante señalar que los antecedentes nacionales revisados se han centrado, en su mayoría, en el análisis de imágenes de placas vehiculares obtenidas en entornos controlados, donde se han estandarizado variables como la iluminación, el ángulo de captura y la inclinación, entre otras. En este sentido, no se cuenta con evidencia suficiente que respalde el desempeño de estos enfoques en condiciones no controladas. Esta brecha constituye una limitación significativa para el presente estudio, ya que las imágenes de las placas de rodaje analizadas provienen de fuentes de campo capturadas en entornos no controlados, lo que introduce una mayor variabilidad en los datos y plantea desafíos adicionales para su procesamiento.

2.3. Bases teóricas y técnicas

En el presente apartado de la tesis se abordan las bases teóricas y técnicas que se emplearon para el desarrollo de la investigación.

Descripción de las placas de rodaje peruanas

De acuerdo con la normativa peruana vigente a la elaboración de la presente investigación, en el Perú se tienen 2 tipos de placas de rodaje. El primer tipo de placas, denominado placa ordinaria, aplica a vehículos menores (motos, mototaxis, etc.), vehículos livianos (por ejemplo, autos) y vehículos pesados (por ejemplo, camiones). El segundo tipo de placa se denomina placa especial, y aplica a vehículos que cumplen actividades especiales (vehículos policiales, de emergencias, etc.), así como a vehículos que no pueden llevar

⁴ La página web de la solución de Plate Recognizer se encuentra disponible, al 09 de abril de 2023, en el siguiente enlace: <https://platerecognizer.com/>

una placa ordinaria (vehículos en exhibición, vehículos que solo se encuentran de paso por el Perú, etc.) (MTC, 2008, D.S. N.° 017-2008-MTC, Título I). El diseño de estas placas se presenta en las figuras 3 y 4.



Figura 3. Diseños de las de placas de rodaje ordinarias de Perú.

Fuente. Adaptado de Anexo II: Cuadro de distribución de colores y representación gráfica de la placa única nacional de rodaje, por (MTC, 2008, D.S. N.° 017-2008-MTC), (<https://aap.org.pe/placas/tipos/ordinarias/base-legal/anexos/>).



Figura 4. Diseños de las de placas de rodaje especiales de Perú

Fuente. Adaptado de Anexo II: Cuadro de distribución de colores y representación gráfica de la placa única nacional de rodaje, por (MTC, 2008, D.S. N.° 017-2008-MTC), (<https://aap.org.pe/placas/tipos/ordinarias/base-legal/anexos/>).

En cuanto a los aspectos geométricos, las placas de rodaje peruanas deben tener dimensiones de 110 mm x 190 mm +/- 2 mm para vehículos menores y 150 mm x 300 mm +/- 2 mm para vehículos livianos y pesados (MTC, 2008, D.S. N.° 017-2008-MTC, Anexo IV). De este modo, comparando el ancho con

la altura de las placas se encuentra que la relación de aspecto es de 1.73 para vehículos menores y 1.5 para vehículos livianos y pesados. Cabe señalar que esta relación de aspecto se aplica a la placa vista de frente. Si la imagen de la placa se toma desde un plano no paralelo a la placa y con inclinación, esta relación de aspecto podría ser menor o mayor.

La misma normativa define además el tamaño de los caracteres que conforman los dígitos y letras del número de la placa, el cual debe ser de 20 mm de ancho x 50 mm de alto x 5 mm de espesor para vehículos menores y 35 mm de ancho x 80 mm de alto x 10 mm de espesor para vehículos livianos y pesados. Asimismo, las letras que conforman la palabra “PERU” deben tener 13.5 mm de ancho x 20 mm de alto x 3 mm de espesor para vehículos menores y 20 mm de ancho x 25 mm de alto x 4 mm de espesor para vehículos livianos y pesados.

En lo que se refiere a los números de las placas de rodaje, que son únicos para vehículo, la normativa vigente a la fecha de la elaboración de la presente investigación establece un conjunto de reglas, las cuales se pueden resumir en el Cuadro 2. Cabe precisar que el número de caracteres que conforma el número de placas es de 6 caracteres, los cuales son agrupados en 2, 3 o 4 según el tipo de placa, separados por un guion (MTC, 2008, (MTC, 2008, D.S. N.º 017-2008-MTC, Título II).

Cuadro 2

Reglas para los números de las placas de rodaje en Perú

Tipo de placa	Agrupación	Reglas de formación	Ejemplos
Placa ordinaria vehículo menor	4 y 2	- Primeros 4 caracteres son números. - Quinto carácter es alfanumérico (36 combinaciones posibles). - Sexto carácter es letra del abecedario según la zona registral de inscripción del vehículo (25 combinaciones, no considera la letra "E").	1234-AA 1234-0B
Placa ordinaria vehículo liviano o pesado	3 y 3	- Primer carácter es letra del abecedario según la zona registral de inscripción del vehículo (25 combinaciones, no considera la letra "E"). - Segundo y tercer carácter son alfanuméricos (36 combinaciones posibles). - Últimos 3 caracteres son números.	B12-123 CCC-123 AC0-123 A1A-123
Placa especial	3 y 3	- Primer carácter es letra "E", la cual debe tener menores dimensiones que el resto de los caracteres. - Segundo y tercer carácter son alfanuméricos según lo indicado en el Anexo III del Reglamento de la Placa única nacional de rodaje. - Últimos 3 caracteres son números.	EPA-123 EUA-123 EEX-123 ER2-123 EZA-123

Fuente. Elaborado a partir del Título II: Sistema de codificación de la placa única nacional de rodaje, del Reglamento de la Placa Única Nacional de Rodaje, aprobada mediante D.S. N.º 017-2008-MTC, modificado mediante Decreto Supremo N.º 043 -2009-MTC y Decreto Supremo N.º 016-2012-MTC.

Estas reglas que establecen la conformación de los números de placa de rodaje son de importancia para poder identificar errores durante la etapa de reconocimiento de caracteres.

Detección y reconocimiento de placas vehiculares

De la revisión de los antecedentes referidos a los sistemas para el reconocimiento automático de placas vehiculares, se pudo concluir que en general, dicha tarea implica dos procesos: la localización o detección de la

placa vehicular y el reconocimiento de los caracteres que conforman el número de placa.

La detección de objetos es una tarea de la visión por computador que tiene como finalidad determinar la localización de un objeto dentro de una imagen. Para ello, como se pudo apreciar de la revisión de los antecedentes, existen muchas técnicas. No obstante, el empleo de redes neuronales permite obtener resultados para imágenes provenientes de entornos no controlados, siempre y cuando se dispongan de las imágenes suficientes y adecuadas para su entrenamiento.

Para la realización de la tarea de detección de objetos se han desarrollado muchas librerías de código. De todas, en la presente investigación se empleó la librería Tensorflow (Abadi et al., 2016) y, en particular, una API denominada Tensorflow Object Detection (TFOD) API. Esta API implementa algoritmos para el entrenamiento y validación de modelos basados en aprendizaje profundo para la detección de objetos.

En lo que respecta al reconocimiento de caracteres, se emplearon librerías de código abierto, en lenguaje Python. En particular, se implementó la integración con la solución desarrollada de la plataforma PaddlePaddle, denominada PaddleOCR (Du et al., 2020), en la medida que dicha solución es ligera y puede incorporarse en sistemas electrónicos como la tarjeta Raspberry Pi. Cabe precisar que computadoras modulares como la Raspberry Pi son denominadas SBC, siglas provenientes de la expresión en inglés Single Board Computer [Computadora de una sola tarjeta].

En lo que respecta a las imágenes empleadas para el entrenamiento, se conformó un dataset propio con etiquetado en el formato de PASCAL-VOC (Pascal Visual Object Classes), el cual es un estándar de etiquetado para tareas de reconocimiento de objetos por clases. El proceso de etiquetado se realizó empleando la herramienta de código libre denominada labellmg2 (Chinakook, 2018).

Procedimiento de entrenamiento y ajuste fino de un modelo

El procedimiento de entrenamiento y ajuste fino de un modelo se realiza a través de algoritmos de aprendizaje, los cuales tienen por objetivo optimizar las salidas del modelo de acuerdo con alguna métrica determinada.

En los modelos que emplean un entrenamiento supervisado, esta métrica consiste en la desviación o error de la salida con respecto del valor indicado en la referencia.

En el caso de la presente investigación, se implementó un modelo para la detección de las placas vehiculares en imágenes. Esta tarea corresponde a al problema de la de detección de objetos, en el cual el modelo recibe como entrada una imagen y proporciona como salida las coordenadas de ubicación de la región con mayor probabilidad de contener una placa de rodaje. Cabe señalar que en la imagen de entrada pueden encontrarse, adicionalmente a las placas de rodaje, muchos objetos más. Por lo tanto, el modelo debe ser capaz de ubicar y clasificar objetos adecuadamente, discriminando aquellos que no son los de interés, pero que presentan características similares.

Tomando como referencia lo descrito en el párrafo anterior, las principales métricas que se emplean durante la etapa de entrenamiento de un modelo para la detección de objetos son:

- a. El error de localización (localization loss): es el error que se genera cuando la región detectada por el modelo no corresponde con la región etiquetada o está desviada o desplazada con respecto de esta, por lo cual, el entrenamiento debe procurar minimizarlo.
- b. El error de clasificación (classification loss): es el error que se genera cuando el modelo clasifica incorrectamente un objeto detectado en la imagen de entrada, por lo cual el entrenamiento debe procurar minimizarlo.
- c. El error de regularización (regularization loss): se refiere al error adicional que se genera luego de que se aplique la función de regularización. El término regularización se refiere al procedimiento de prevenir el overfitting o el underfitting en el modelo entrenado. En ese sentido, lo

esperado es que este error disminuya con el entrenamiento, en la medida que ello implicaría que el modelo tiene una mejor capacidad de generalizar salidas.

- d. Error total: se refiere a la combinación de los 03 errores descritos previamente. Para el caso de la API de Tensorflow para la detección de objetos, el error total se calcula como la suma simple de los errores de clasificación, localización y regularización.

Estos errores se calculan en cada iteración del entrenamiento, y se retroalimentan en el modelo para actualizar los pesos de cada capa que conforme el modelo. En este procedimiento, la tasa de aprendizaje determina la intensidad con la que estos pesos son retroalimentados y considerados en la actualización de los pesos del modelo. Esta tasa suele tener un mayor valor al inicio, con la finalidad de acelerar el proceso de entrenamiento, y tender a 0 hacia el final del entrenamiento, para asegurar la convergencia.

El proceso de ajuste fino [fine-tuning] consiste en entrenar un modelo previamente implementado (pre entrenado), para realizar una actividad más específica que el modelo base. Los modelos pre entrenados son modelos preparados para realizar tareas de clasificación o detección sobre un gran número de clases u objetos, lo cual los hacen generales. No obstante, esta amplitud en la tarea hace que estos modelos sean tengan resultados óptimos para realizar tareas más específicas. En ese sentido, el ajuste fino permite que estos modelos actualicen sus pesos de tal manera que mejore su desempeño con relación a una tarea específica. Cabe señalar que las principales ventajas de realizar un ajuste fino con relación a realizar un entrenamiento desde cero son las siguientes:

- a. Disponibilidad de datos: un entrenamiento desde cero requiere mucha más información o datos de referencia que un ajuste fino.
- b. Tiempo de entrenamiento: realizar un entrenamiento desde cero, toma mucho más tiempo que realizar un ajuste fino, en la medida de que un modelo pre entrenado ya cuenta con información general codificada en sus pesos, que agiliza el proceso de entrenamiento. Esto a su vez

implica el requerimiento de recursos computacionales por mayor tiempo, lo cual a su vez implica un mayor costo.

Por consiguiente, para la presente investigación se realizó un procedimiento de ajuste fino partiendo de modelos pre entrenados para la detección de objetos.

Indicadores de desempeño de modelos para la detección de objetos

Los indicadores de desempeño del sistema permitirán evaluar si el modelo desarrollado cumple con la tarea de reconocer las placas vehiculares de manera adecuada.

El principal parámetro para medir el desempeño de un modelo para la detección de objetos es el parámetro AP (Average Precision o precisión promedio). Este parámetro evalúa el desempeño de un modelo a partir de la cuantificación de falsos y verdaderos positivos detectados (Hui, 2018). El concepto de presión promedio (AP) puede expresarse mediante la Ecuación 2.

$$AP = \frac{VP}{VP + FP} \quad (2)$$

Donde AP es Precisión promedio, VP son las detecciones consideradas como Verdaderos Positivos y FP son las detecciones consideradas como Falsos Positivos.

Para determinar si una detección es un verdadero positivo o un falso positivo, se emplea el concepto de IoU (Intersection over Union o Intersección sobre Unión). Básicamente, el parámetro IoU consiste en la relación entre el área de la intersección y el área de la unión de las regiones correspondientes al objeto etiquetado o real y al objeto predicho. De ese modo, una IoU igual a 1 implica que la región predicha por el modelo coincide con la región real del objeto (caso ideal), mientras que un valor igual a 0 implica que la región predicha está completamente fuera de lo esperado (no hay coincidencia).

A partir de este concepto, para definir un caso de verdadero positivo o falso positivo, se define un valor de IoU como referencia o umbral, de tal manera que los valores superiores a este son calificados como verdaderos positivos y

los valores menores a este, como falsos positivos. En la Figura 5 se presenta una visualización de lo anteriormente señalado, considerando como umbral de IoU el valor de 0.5. Se pudo apreciar lo siguiente:

- (a) Imagen original: El recuadro rojo señala la región que corresponde al objeto por detectar (placa de rodaje).
- (b). En amarillo se señala la región que corresponde a la predicción del modelo: puede apreciarse que la intersección (resaltada con sombreado en diagonal), si bien cubre la región del objeto (rojo), al ser comparada con el área de la intersección, resulta en AP menor de 0.5, por lo cual es un falso positivo. La zona de predicción es muy amplia.
- (c) En ese caso, la predicción (amarillo) está desplazada de la región del objeto (objeto), lo cual hace que el área de la intersección sea muy pequeña. Esto arroja un AP mucho menor que 0.5 y por lo tanto es un falso positivo.
- (d) La región de la predicción (amarillo) y la región del objeto (rojo) casi coinciden. En este caso, la intersección y la unión tienen áreas muy próximas, lo cual da un AP próximo a 1 (muy por encima de 0.5), por lo cual se evalúa la predicción como un verdadero positivo.



Figura 5. Concepto del parámetro IoU

Fuente. Elaborado a partir del dataset de fotografías compendiado en el marco de la presente investigación.

Por otro lado, en función de la cantidad de clases que un modelo haya sido entrenado para detectar, se define el parámetro mAP (precisión promedio media), el cual es el promedio de los valores AP del modelo para cada una de las clases para las cuales fue entrenado. En la Ecuación 3 se presenta la expresión para calcular dicho parámetro de evaluación.

$$mAP = \sum_i AP_i / N \quad (3)$$

Donde mAP es la media de la precisión promedio del modelo, AP_i es la precisión promedio del modelo para la clase i , y N es el número total de clases. En el caso de que el modelo haya sido entrenado para detectar una clase de objeto, el parámetro mAP es el mismo que el AP de dicha clase.

Otro parámetro que se suele usar, relacionado al anterior, es el de Recuperación, que queda definido por la Ecuación 4.

$$RE = \frac{VP}{VP + FN} \quad (4)$$

Donde RE es Recuperación, VP son las detecciones consideradas como verdaderos positivos y FN son las detecciones consideradas como falsos negativos. Cabe precisar que estos valores se calculan sobre IoU determinadas.

Planteado de este modo, el parámetro de recuperación permite medir la capacidad del modelo para detectar correctamente un objeto respecto al total de casos etiquetados. De este modo, este parámetro es diferente a la precisión, la cual mide la capacidad del modelo de detectar correctamente un objeto respecto al total de casos detectados. No obstante, estos dos parámetros en conjunto permiten medir la capacidad del modelo para detectar adecuadamente los objetos que se requieran.

Con relación al reconocimiento de caracteres, si bien es posible aplicar el concepto de AP para el componente de detección de las letras, para los fines de la presente investigación interesa más la capacidad de reconocimiento, por lo cual resulta más adecuado realizar una evaluación estadística del desempeño del sistema de reconocimiento óptico de caracteres empleado.

Modelos pre entrenados para la detección de objetos

Un modelo pre entrenado es un modelo que ha sido diseñado para realizar tareas específicas de modo general. A continuación, se desarrolla un análisis de los principales modelos existentes para la detección de objetos.

Estos modelos especializados en la detección de objetos se denominan detectores, y de acuerdo con su arquitectura se pueden categorizar como detectores de dos etapas o detectores de una sola etapa.

En los detectores de dos etapas, la primera etapa consiste en la ubicación y clasificación de los objetos a detectar, mientras que la segunda etapa tiene por objeto afinar la detección analizando los candidatos detectados a fin de eliminar falsos positivos. De esta manera, estos detectores logran una mayor precisión en sus resultados, no obstante, requieren de mayores recursos computacionales para su uso.

Entre los principales detectores de dos etapas se tiene: R-CNN (Girshick et al., 2014), Fast R-CNN (Girshick, 2015), Faster R-CNN (Ren et al., 2015), Mask R-CNN (He et al., 2017) y FPN (Lin et al., 2016).

Los detectores de una sola etapa, por otro lado, están diseñados de manera que todo el proceso de detección ocurra en un único paso. Si bien, estos modelos podrían tener menor precisión que los modelos de dos etapas, son mucho más rápidos en su ejecución, por lo cual pueden funcionar en tiempo real.

Entre los principales detectores de una sola etapa se encuentran: SSD(Liu et al., 2015), RetinaNet (Lin et al., 2017), CenterNet (Duan et al., 2019), EfficientDet (Tan et al., 2019) y YOLO (Redmon et al., 2015). Cabe señalar que, hasta la fecha de esta investigación, la última versión del modelo YOLO es la 7, denominado YOLOv7 (Wang et al., 2022). Este modelo presenta mejoras en términos de precisión y a tiempo de inferencia, logrando velocidades en el rango de 5 a 160 cuadros por segundo (fps). No obstante, en general, los modelos basados en YOLO tienden a tener una menor precisión que los modelos basados en SSD o en R-CNN.

Con relación a su implementación, Tensorflow pone a disposición un conjunto de modelos pre entrenados para su uso en inferencia o para fine-tuning en un repositorio de acceso libre (Tensorflow, s. f.). En la Cuadro 3 se presentan algunos modelos para la detección de objetos disponibles en dicho repositorio.

Cuadro 3

Modelos pre-entrenados para la detección de objetos proporcionados por la Object Detection API de Tensorflow

Nombre de modelo	Tiempo inferencia (ms)	mAP	Tamaño (MB)
EfficientDet D0 512 x 512	39	33.6	63.76
EfficientDet D1 640 x 640	54	38.4	91.19
EfficientDet D2 768 x 768	67	41.8	105.49
EfficientDet D3 896 x 896	95	45.4	141.58
EfficientDet D4 1024 x 1024	133	48.5	217.92
EfficientDet D5 1280 x 1280	222	49.7	325.78
EfficientDet D6 1280 x 1280	268	50.5	475.24
EfficientDet D7 1536 x 1536	325	51.2	475.24
SSD MobileNet v2 320 x 320	19	20.2	55.55
SSD MobileNet V1 FPN 640 x 640	48	29.1	99.79
SSD MobileNet V2 FPNLite 320 x 320	22	22.2	29.79
SSD MobileNet V2 FPNLite 640 x 640	39	28.2	29.79
SSD ResNet101 V1 FPN 640 x 640	57	35.6	411.91
SSD ResNet101 V1 FPN 1024 x 1024	104	39.5	411.91
SSD ResNet152 V1 FPN 640 x 640	80	35.4	537.01
SSD ResNet152 V1 FPN 1024 x 1024	111	39.6	537.01
Faster R-CNN ResNet50 V1 640 x 640	53	29.3	224.56
Faster R-CNN ResNet50 V1 1024 x 1024	65	31	224.54
Faster R-CNN ResNet50 V1 800 x 1333	65	31.6	230.06
Faster R-CNN ResNet101 V1 640 x 640	55	31.8	373.74
Faster R-CNN ResNet101 V1 1024 x 1024	72	37.1	373.71
Faster R-CNN ResNet101 V1 800 x 1333	77	36.6	379.23
Faster R-CNN ResNet152 V1 640 x 640	64	32.4	497.32
Faster R-CNN ResNet152 V1 1024 x 1024	85	37.6	497.30
Faster R-CNN ResNet152 V1 800 x 1333	101	37.4	502.82
Faster R-CNN Inception ResNet V2 640 x 640	206	37.7	472.10
Faster R-CNN Inception ResNet V2 1024 x 1024	236	38.7	476.23

Fuente: Adaptado de Tensorflow 2 Detection Model Zoo, por (Tensorflow, s. f.), GitHub.

(https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md).

Para la presente investigación, se tomaron como modelos base para la fase de entrenamiento los modelos basados en SSD MobileNet y EfficientDet D0, D1, en la medida de que son los modelos que tienen un tamaño de menos de 100 MB y son rápidos, además de tener una precisión inicial adecuada.

Este criterio se basa en el hecho de que el tamaño y el tiempo de inferencia son características de la arquitectura del modelo. En cambio, la precisión inicial (mAP), en la medida de que estos modelos se han pre entrenado sobre un dataset de 90 clases, brinda información que corresponde al efecto de dicho dataset. Al entrenarse sobre un dataset con una sola clase (placa vehicular), se espera que la precisión promedio (mAP) aumente.

2.4. Definición de términos

A continuación, se procede a realizar la definición de los principales términos empleados en el desarrollo de la presente tesis.

- **API:** De las siglas en inglés Application Programming Interface, es la interfaz de programación de una aplicación que permite que un módulo de código pueda interactuar con otro módulo de código, a fin de usar rutinas o variables definidas en esta última.
- **Aprendizaje automático:** Subdominio de la inteligencia artificial que tiene como objetivo lograr que un sistema realice o mejore la ejecución de una tarea específica a partir del aprendizaje de la experiencia.
- **Aprendizaje profundo:** Subcomponente del aprendizaje automático que aborda el diseño, implementación y uso de redes neuronales conformadas por varias capas. Cada capa codifica información de mayor nivel que la capa anterior, lo cual permite que se puedan extraer características particulares de los datos.
- **Código:** Es el algoritmo o conjunto de algoritmos o definiciones, elaboradas con uno o más lenguajes de programación, que permiten la resolución de uno o varios problemas.

- Código abierto: Hace referencia al código que puede ser revisado, utilizado o modificado, por cualquier persona de manera libre y generalmente gratuita.
- Computación en la nube o nube: Se define como la disponibilidad de recursos informáticos disponibles para su uso, ya sea a nivel de servicio, plataforma o hardware, a través de la Internet.
- Entrenamiento: Procedimiento mediante el cual se actualizan los pesos y parámetros de un modelo basado en inteligencia artificial, de manera que pueda realizar una tarea determinada.
- Hiperparámetro: Parámetro que permite configurar una determinada característica de un proceso de modelamiento y/o entrenamiento de un modelo basado en inteligencia artificial.
- Lenguaje de programación: Es el lenguaje que permite el registro de instrucciones que serán ejecutadas por una computadora o sistema informático.
- Librería: Conjunto de algoritmos o procedimientos elaborados para la solución de uno o más problemas específicos, y que están organizados de tal forma que permiten su uso en la programación de sistemas más amplios.
- Inteligencia artificial: Capacidad de un sistema (generalmente informático) para tomar decisiones para la realización de un objetivo a partir del razonamiento o procesamiento de datos obtenidos del entorno.
- Modelo: Para efectos de la presente tesis, el término modelo hace referencia al resultado del proceso de aprendizaje automático a partir de un dataset.
- Modelo pre entrenado: Como lo refiere el término, es un modelo que ha sido entrenado previamente, sobre un conjunto de datos muy amplio y diverso, para la solución de diversos problemas relacionados generales,

y que puede emplearse para solucionar problemas específicos luego de volverse a entrenar con datos más específicos.

- SBC: Siglas del término en inglés Single Board Computer (Computadora de una sola tarjeta). Se refiere a computadoras diseñadas para su uso en prototipado de aplicaciones, que dispone de todas sus interfaces y componentes en una sola tarjeta electrónica. Un ejemplo de SBC es la tarjeta Raspberry Pi.
- Tasa de aprendizaje: hiperparámetro de un modelo basado en redes neuronales, que determina la intensidad con la que los pesos son actualizados durante un entrenamiento. Una tasa de aprendizaje muy pequeña implica un mayor tiempo de entrenamiento, mientras que una tasa de aprendizaje muy grande podría evitar que la red logre converger en un punto óptimo de desempeño.

3.2. III. HIPÓTESIS Y VARIABLES

3.1. Hipótesis

De acuerdo con Hernández et al., (2014, p. 104), las hipótesis son respuestas provisionales a las preguntas de investigación y se derivan de la teoría existente, constituyéndose en la guía de la investigación.

Para la presente tesis, una vez realizado el análisis del problema se planteó la siguiente pregunta como problema principal de la investigación:

“¿Cómo desarrollar un sistema automático que permita la identificación automática de placas de rodaje peruanas en el departamento de Lima, en el año 2023?”

Asimismo, se plantearon los siguientes problemas específicos:

- a. ¿Es posible realizar el entrenamiento de un sistema automático que permita el reconocimiento de placas de rodaje peruanas en el departamento de Lima, en el año 2023?
- b. ¿Es posible diseñar un subsistema de software para implementar un sistema automático que permita el reconocimiento de placas de rodaje peruanas en el departamento de Lima, en el año 2023?
- c. ¿Es posible integrar un subsistema de hardware para implementar un sistema automático que permita el reconocimiento de placas de rodaje peruanas en el departamento de Lima, en el año 2023?

3.1.1. Hipótesis Principal

Tomando en cuenta el problema principal y habiendo realizado el análisis teórico y del estado del arte, así como de los antecedentes referentes de la investigación, se procedió a plantear la hipótesis como se indica a continuación:

“Es posible desarrollar un sistema automático para la identificación de placas de rodaje peruanas, en el departamento de Lima en el año 2023, empleando un modelo basado en inteligencia artificial adecuado”.

3.1.2. Hipótesis Específicas:

En función a los problemas específicos se definieron las siguientes hipótesis específicas:

- a. Es posible realizar el entrenamiento de un sistema automático que permita el reconocimiento de placas de rodaje peruanas en el departamento de Lima, en el año 2023.
- b. Es posible diseñar un subsistema de software para implementar un sistema automático que permita el reconocimiento de placas de rodaje peruanas en el departamento de Lima, en el año 2023.
- c. Es posible integrar un subsistema de hardware para implementar un sistema automático que permita el reconocimiento de placas de rodaje peruanas en el departamento de Lima, en el año 2023.

3.2. Variables

Las variables pueden definirse como constructos o conceptos que permiten el modelamiento de un determinado caso o fenómeno, que pueden medirse u observarse, y cuyo valor para una investigación se da cuando estas permiten analizar una hipótesis y su validez o una teoría (Espinoza, 2022, p. 46) (Hernández et al., 2014, p. 105).

Asimismo, considerando la función que la variable toma dentro de la investigación, se pueden distinguir dos tipos principales de variables: las independientes, cuyo control permite influir en el caso o fenómeno bajo estudio; y las dependientes, que permiten analizar el resultado o resultados de la investigación (Espinoza, 2022, p. 50-51).

Tomando en consideración lo antes mencionado, se plantearon las siguientes variables para la presente investigación:

- Variable independiente: Sistema Automático.
- Variable dependiente: Reconocimiento de placas de rodaje peruanas.

Con relación a estas variables, se procedió a desarrollar una definición dentro del alcance de la presente investigación, considerando el marco teórico analizado:

- Sistema Automático: Es un sistema que integra algoritmos para el modelamiento de una red neuronal, así como algoritmos para procesamiento de imágenes (software) y su aplicación en un dispositivo electrónico (hardware).
- Reconocimiento de placas de rodaje peruanas: Resultado esperado del sistema que integra software y hardware, según lo dispuesto en los objetivos de la presente investigación.

3.3. Operacionalización de las variables

La operacionalización de las variables consiste en el proceso mediante el cual se definen y estructuran las variables de forma concreta, permitiendo su medición y evaluación de manera objetiva y sistemática. En el Cuadro 4, tomando como referencia la matriz propuesta por Ávalos (2014), se presenta la operacionalización de las variables de la presente investigación.

Cuadro 4

Operacionalización de las Variables de la Investigación

Variable	Definición conceptual	Dimensiones	Indicadores	Escala	Índice
Variable independiente: Sistema Automático	Es un sistema que integra algoritmos para el modelamiento de una red neuronal, así como algoritmos para procesamiento de imágenes (software) y su aplicación en un dispositivo electrónico (hardware), según lo planteado en el numeral 3.2 de la presente investigación.	Entrenamiento	I1: Tiempo de entrenamiento	Segundos	$T_t = T_{tf} - T_{ti}$ T _t : Tiempo de entrenamiento T _{ti} : Tiempo de inicio de entrenamiento T _{tf} : Tiempo de finalización de entrenamiento
		Sistema de software	I2: Implementación	Unidad	Número de scripts o programas de software
		Sistema de hardware	I3: Implementación	Unidad	Número de integraciones con hardware
Variable dependiente: Reconocimiento de placas de rodaje peruanas	Resultado esperado del sistema que integra software y hardware, según lo dispuesto en los objetivos de la presente investigación, según lo planteado en el numeral 3.2 de la presente investigación.	Tiempo de ejecución	I4: Tiempo	Segundos	$T_e = T_{ef} - T_{ei}$ T _e : Tiempo de ejecución T _{ei} : Tiempo de inicio de ejecución T _{ef} : Tiempo de finalización de ejecución
		Detección de la placa de rodaje	I5: Precisión promedio (AP)	Relación	$AP = \frac{VP}{VP + FP}$ AP: precisión promedio. VP: verdaderos positivos FP: falsos positivos (*)
Variable dependiente: Reconocimiento de placas de rodaje peruanas	Resultado esperado del sistema que integra software y hardware, según lo dispuesto en los objetivos de la presente investigación, según lo planteado en el numeral 3.2 de la presente investigación.	Reconocimiento de los caracteres que conforman la placa de rodaje	I6: Exactitud de reconocimiento de caracteres	Porcentual	$E_R = \frac{C_C}{C_T} \times 100\%$ E _R : exactitud de reconocimiento. C _C : cantidad de números de placas reconocidos correctamente reconocidos. C _T : total de placas etiquetadas.

Fuente. Tabla adaptada de Ávalos (2014).

Nota. (*) Las herramientas para el entrenamiento, como la TFOD API, implementan las funciones para el cálculo de los indicadores AP y RE.

3.4. Matriz de consistencia

A continuación, en el Cuadro 5 se presenta la matriz de consistencia de la presente investigación, la cual permite verificar que los objetivos, hipótesis, así como las variables establecidas, se encuentren alineadas con los problemas propuestos por la presente investigación.

Cuadro 5

Matriz de consistencia

PROBLEMA	OBJETIVOS	HIPÓTESIS	VARIABLES E INDICADORES																				
Problema principal: ¿Cómo desarrollar un sistema automático que permita la identificación automática de placas de rodaje peruanas en el departamento de Lima, en el año 2023? Problemas específicos: a. ¿Es posible realizar el entrenamiento de un sistema automático que permita el reconocimiento de placas de rodaje peruanas en el departamento de Lima, en el año 2023? b. ¿Es posible diseñar un subsistema de software	Objetivo principal: Desarrollar un sistema automático para el reconocimiento de placas de rodaje peruanas en el departamento de Lima, en el año 2023. Objetivos específicos: a. Realizar el entrenamiento de modelos basados en inteligencia artificial para realizar el reconocimiento de placas de rodaje peruanas en el departamento de Lima en el año 2023. b. Diseñar un subsistema de software para la	Hipótesis principal: Es posible desarrollar un sistema automático para la identificación de placas de rodaje peruanas, en el departamento de Lima en el año 2023. Hipótesis específicas: a. Es posible realizar el entrenamiento de un sistema automático que permita el reconocimiento de placas de rodaje peruanas en el departamento de Lima, en el año 2023. b. Es posible diseñar un subsistema de software	Variable independiente: Sistema automático. <table><tr><th>Dimensión</th><th>Indicadores</th><th>Escala</th></tr><tr><td>Entrenamiento</td><td>I1: Tiempo de entrenamiento</td><td>Segundos</td></tr><tr><td>Sistema de software</td><td>I2: Implementación</td><td>Unidad</td></tr><tr><td>Sistema de hardware</td><td>I3: Implementación</td><td>Unidad</td></tr></table> Variable dependiente: Reconocimiento de placas de rodaje peruanas <table><tr><th>Dimensión</th><th>Indicadores</th><th>Escala</th></tr><tr><td>Tiempo de ejecución</td><td>I4: Tiempo</td><td>Segundos</td></tr></table>			Dimensión	Indicadores	Escala	Entrenamiento	I1: Tiempo de entrenamiento	Segundos	Sistema de software	I2: Implementación	Unidad	Sistema de hardware	I3: Implementación	Unidad	Dimensión	Indicadores	Escala	Tiempo de ejecución	I4: Tiempo	Segundos
Dimensión	Indicadores	Escala																					
Entrenamiento	I1: Tiempo de entrenamiento	Segundos																					
Sistema de software	I2: Implementación	Unidad																					
Sistema de hardware	I3: Implementación	Unidad																					
Dimensión	Indicadores	Escala																					
Tiempo de ejecución	I4: Tiempo	Segundos																					

PROBLEMA	OBJETIVOS	HIPÓTESIS	VARIABLES E INDICADORES		
para implementar un sistema automático que permita el reconocimiento de placas de rodaje peruanas en el departamento de Lima, en el año 2023?	implementación de un sistema automático que permita el reconocimiento de placas de rodaje peruanas en el departamento de Lima, en el año 2023.	para implementar un sistema automático que permita el reconocimiento de placas de rodaje peruanas en el departamento de Lima, en el año 2023.	Detección de la placa de rodaje	I5: Precisión promedio (AP)	Relación
			Reconocimiento de los caracteres que conforman la placa de rodaje	I6: Exactitud de reconocimiento de caracteres	Porcentual
c. ¿Es posible integrar un subsistema de hardware para implementar un sistema automático que permita el reconocimiento de placas de rodaje peruanas en el departamento de Lima, en el año 2023?	c. Integrar un subsistema de hardware para la implementación de un sistema automático que permita el reconocimiento de placas de rodaje peruanas en el departamento de Lima, en el año 2023.	c. Es posible integrar un subsistema de hardware para implementar un sistema automático que permita el reconocimiento de placas de rodaje peruanas en el departamento de Lima, en el año 2023.			

IV. MATERIALES Y MÉTODOS

4.1. Área de estudio

El área de estudio de la presente tesis corresponde a todo el territorio peruano, en la medida de que el sistema desarrollado se diseñó para identificar placas de rodaje vehiculares del Perú. No obstante, considerando que la reglamentación de diseño de las placas de rodaje es la misma para todo el territorio peruano, las muestras a recolectar para realizar el entrenamiento del sistema se tomaron de las ciudades de Lima Metropolitana y de la provincia de Cañete, en el departamento de Lima.

4.2. Diseño de investigación

El diseño de la investigación consiste en establecer la estrategia a seguir para evaluar las hipótesis definidas y poder dar respuesta a las preguntas planteadas como problemas de la presente investigación.

De acuerdo con Esteban (2018), la investigación aplicada o tecnológica se enfoca en encontrar soluciones a los problemas que surgen en las distintas etapas de producción, distribución, circulación y consumo de bienes y servicios relacionados con cualquier actividad humana. Asimismo, añade que este tipo de investigación tiene como objetivo principal la mejora, optimización o perfeccionamiento de los procesos productivos en la sociedad, por lo que su diseño se fundamenta en un enfoque que integra la observación, reflexión, diseño y praxis, combinados con un análisis y síntesis del objeto de estudio.

Dados los problemas y objetivos que la presente investigación aborda, y la definición descrita previamente, se puede concluir que la presente investigación es una investigación aplicada o tecnológica, puesto que tiene por objetivo desarrollar una solución para el problema de la automatización del proceso de reconocimiento de placas de rodaje peruanas, el cual es un servicio que el Estado Peruano tiene a su cargo a través del Sistema Integral de Identificación Vehicular.

Tomando en cuenta lo señalado, el diseño de la presente investigación siguió los siguientes procesos:

- Observación de la data: corresponde a realizar el análisis observacional de las imágenes de placas vehiculares, con el objeto de caracterizarlos para su posterior uso en el entrenamiento del sistema.
- Reflexión: la reflexión sobre las características de los datos de imágenes a emplear en el entramiento, así como las características de las placas de rodaje a detectar, permite determinar los modelos base con los cuales abordar el sistema a desarrollar.
- Diseño: se realiza el entrenamiento de los modelos, asimismo, se integra con el componente de reconocimiento de caracteres (OCR). El diseño del sistema, en este nivel, implica básicamente la estructuración de los algoritmos necesarios para el desarrollo del sistema.
- Praxis de la necesidad de análisis: se analizaron los indicadores de desempeño de los modelos entrenados con los datos recolectados, con la finalidad de determinar el más eficiente y/u óptimo.
- Síntesis del objeto de la investigación: una vez obtenido el modelo con el mejor desempeño, se procedió a integrar el sistema desarrollado en un prototipo práctico, con el cual se validen los indicadores obtenidos, así como su viabilidad práctica.

Para los procesos indicados, se empleó un enfoque de investigación teórico-experimental, basado en técnicas de entrenamiento de inteligencia artificial y mediciones de las variables del estudio.

4.3. Población y muestra

Tomando en cuenta que, en el Perú, tal y como se señaló en el numeral 2.3.1. de la presente tesis, existe una normativa de aplicación general que establece el tipo de placas de rodaje existentes y su diseño, se puede considerar que la población para la presente investigación está conformada por el total de vehículos existentes en el Perú.

Al respecto, de acuerdo con El Peruano (2019), se estimó que el parque automotor peruano es de aproximadamente 2.6 millones de vehículos al año 2019. Para determinar la muestra mínima a considerar en el análisis se aplicó la Ecuación 5.

$$n = \frac{N \times Z^2 \times p \times (1 - p)}{(N - 1) \times e^2 + Z^2 \times p \times (1 - p)} \quad (5)$$

Donde “n” es la cantidad de muestras a tomar, “N” es el tamaño de la población, “Z” es el coeficiente de confianza, “e” es el margen de error y “p” es la proporción de la población que se estima que cumplen la condición bajo estudio.

Para el presente estudio, se tomaron las siguientes premisas para determinar la cantidad de muestras a tomar:

N = 2 600 000 de vehículos con placas.

Z = 1.96 para un valor de confianza del 95%.

e = 5%

p = 50%

Aplicando ello se obtuvo la siguiente cantidad de muestras:

$$n = \frac{2600000 \times 1.96^2 \times 0.6 \times (1 - 0.6)}{(2600000 - 1) \times 0.05^2 + 1.96^2 \times 0.6 \times (1 - 0.6)} = 384 \quad (6)$$

De este modo, para la investigación se estima que se requiere de un total de 384 muestras como mínimo, dadas las premisas señaladas previamente.

Tomando como referencia esta muestra, se recolectó un total de 392 imágenes de placas de rodaje, en diferentes contextos, de manera que se pueda tener una amplia variedad para los fines de entrenamiento y evaluación del modelo a diseñar. Cabe señalar que en algunas imágenes se tiene más de una placa de rodaje, teniéndose de ese modo, un total de 406 placas de rodaje en el dataset.

4.4. Procedimientos, técnicas e instrumentos de recolección de información

Para la realización de la presente investigación, se recolectaron dos tipos de información:

- Información bibliográfica: basado en artículos, libros, publicaciones, cualquier documentación que aborde los procesos de inteligencia artificial, entrenamiento de modelos y afines.
- Imágenes de placas de rodaje: son las imágenes de las placas de rodaje empleadas en el entrenamiento de los modelos.

El procedimiento para la recolección de la información bibliográfica consistió en la revisión de repositorios públicos de información científica, tales como ArXiv, Google Scholar, repositorios del Institute of Electrical and Electronics Engineering (IEEE), entre otros. Toda la información se sistematizó a fin de elaborar con ella el marco teórico de la investigación, la exploración y análisis de antecedentes, así como la revisión de bases teóricas y técnicas para el desarrollo de la investigación.

Con relación a las imágenes de placas de rodaje, se empleó una cámara fotográfica y un celular smartphone para la captura de las imágenes. Estas imágenes se tomaron procurando tener un dataset lo suficientemente variado para los fines de entrenamiento del sistema a desarrollar como parte de la presente investigación (placas de vehículos en movimiento, placas con imperfecciones, placas de vehículos en la calle, placas de vehículos estacionados, placas inclinadas, etc.). Con relación a este procedimiento, se adjunta como anexo la “Ficha para recolección de imágenes”, en la cual establece los criterios para la recolección de imágenes para la presente investigación.

Posteriormente, las imágenes en formato digital fueron etiquetadas empleando el software LabelImg2 de código libre (Chinakook, 2018). El etiquetado consiste en señalar la región que corresponde a las placas de rodaje dentro de la imagen, generando como salida un archivo en formato XML, en el cual se describe, entre otra información, las coordenadas de los

vértices superior izquierdo e inferior derecho de la región que corresponde a cada placa. Asimismo, se registra el número de la placa de rodaje asociada. En la Figura 6 se presenta un ejemplo de etiquetado. Las coordenadas de los vértices de la región de la placa (rectángulo rojo), de acuerdo con el ejemplo, están definidas en el campo `<bndbox>`, el vértice superior izquierdo es: (*xmin*, *ymin*), y el vértice inferior derecho es: (*xmax*, *ymax*). En el campo `<name>` se indica el nombre de la clase: *licence* (placa de rodaje). En el campo `<extra>` se registra la información del número de la placa, en este ejemplo: F7E-078.



Figura 6. Ejemplo de Etiquetado

Fuente. Elaborado a partir del dataset de fotografías implementado en el marco de la presente investigación.

El etiquetado es un procedimiento fundamental, en la medida de que el modelo de detección de placas se entrenó de tal forma que pueda predecir la región en la que se encuentre la placa o placas en la imagen. Para ello, durante el entrenamiento, se actualizaron los pesos de las matrices convolucionales entre las capas del modelo de manera que el error entre la región predicha y la región etiquetada sea mínimo.

4.5. Análisis estadístico

En el presente apartado de la investigación se realiza el análisis estadístico de las imágenes que conforman el dataset empleado durante el entrenamiento y evaluación del modelo, así como de las placas de rodaje etiquetadas.

4.5.1. Análisis estadístico de las imágenes

El dataset, denominado “peruvian_licence_plates” está conformado por un total de 392 imágenes en formato .jpg, las cuales comprenden un total 409 placas de rodaje peruanas, obtenidas de vehículos bajo distintas condiciones (no controladas).

Las imágenes del dataset son a color, con dimensiones que varían en el ancho entre 825 y 1,600 píxeles, y en la altura entre 588 y 1,600 píxeles. Las resoluciones oscilan entre 0.50 y 1.92 megapíxeles. En relación con la distribución de los tamaños, la mayoría de las imágenes presentan un ancho de 1,024 píxeles (358 imágenes, que representan el 91.33% del total) y una altura de 768 píxeles (también 358 imágenes, equivalentes al 91.33%). En consecuencia, el 91.33% de las imágenes tienen una resolución de 0.79 megapíxeles, lo que refleja una predominancia de este tamaño dentro del dataset.

Por otro lado, respecto a la orientación de las imágenes, se define como horizontal aquella en la que el ancho es mayor que la altura. Siguiendo este criterio, se identificaron 379 imágenes con orientación horizontal y 13 con orientación vertical, lo que corresponde al 96.68% y al 3.32% del total, respectivamente. En el Cuadro 6 se detallan las principales características del dataset, precisándose que las siglas "px" hacen referencia a píxeles y "Mpx" a megapíxeles.

Cuadro 6

Características generales de las imágenes del dataset

Característica	Valor más frecuente	Mínimo valor	Máximo valor
Altura de imagen (px)	768	825	1 600
Ancho de imagen (px)	1 024	588	1 600
Resolución de imagen (Mpx)	0.79	0.50	1.92
Orientación de la imagen	Horizontal	Vertical	Horizontal

Fuente. Elaborado a partir del análisis estadístico propio de las fotografías del dataset.

Continuando con el análisis estadístico, como se señaló al inicio del presente apartado, las 392 imágenes que conforman el dataset tienen el registro de un total de 406 placas de rodaje. Analizando la cantidad de placas por imagen, se tiene que 379 imágenes tienen una sola placa de rodaje etiquetada, 12 imágenes tienen dos placas de rodaje etiquetadas y 1 una imagen tiene tres placas de rodaje etiquetadas. En el Cuadro 7 y en la Figura 7 se presenta esta información, en el cual los porcentajes indicados se calcularon respecto del total de imágenes (392 imágenes).

Cuadro 7

Número de Placas por Imagen

Número de placas por imagen	Frecuencia	Frecuencia porcentual
Imágenes con una sola placa etiquetada	379	96.68%
Imágenes con dos placas etiquetadas	12	3.06%
Imágenes con tres placas etiquetadas	1	0.26%
Total	392	100%

Fuente. Elaborado a partir del análisis estadístico propio de las fotografías del dataset.

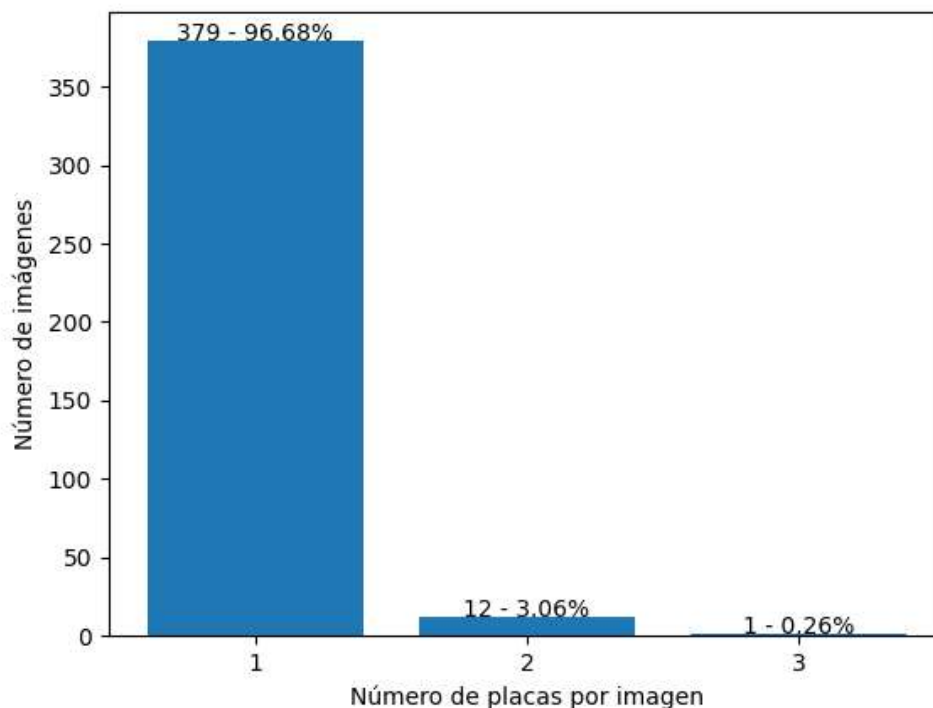


Figura 7. Distribución de Número de Placas por Imagen

4.5.2. Análisis estadístico de las placas de rodaje etiquetadas

El dataset implementado para la presente investigación, denominado “peruvian_licence_plates”, conformado por 392 imágenes, contiene las etiquetas de un total de 406 placas de rodaje. Tomando en consideración de que cada etiqueta, además de la ubicación de la región en la que se encuentra la placa de rodaje, indica también el número de la placa, es posible determinar la cantidad de códigos únicos contenidos en el dataset.

De este modo, en el Cuadro 8 se presenta la distribución de veces que se repite una determinada placa de rodaje en el dataset. Asimismo, en la Figura 8 se presenta su distribución, en donde los porcentajes indicados se calcularon respecto del total de placas de rodaje diferentes (312 placas de rodaje).

Cuadro 8

Placas de rodaje según la cantidad de veces que se repite en el dataset

Veces que se repite la placa de rodaje	Frecuencia	Frecuencia porcentual
1 vez	257	82.37%
2 veces	38	12.18%
3 veces	9	2.88%
4 veces	4	1.28%
De 5 a 7 veces	3	0.96%
12 veces	1	0.33%
Total de placas de rodaje diferentes	312	100%

Fuente. Elaborado a partir del análisis estadístico de las fotografías que conforman el dataset de la presente investigación.

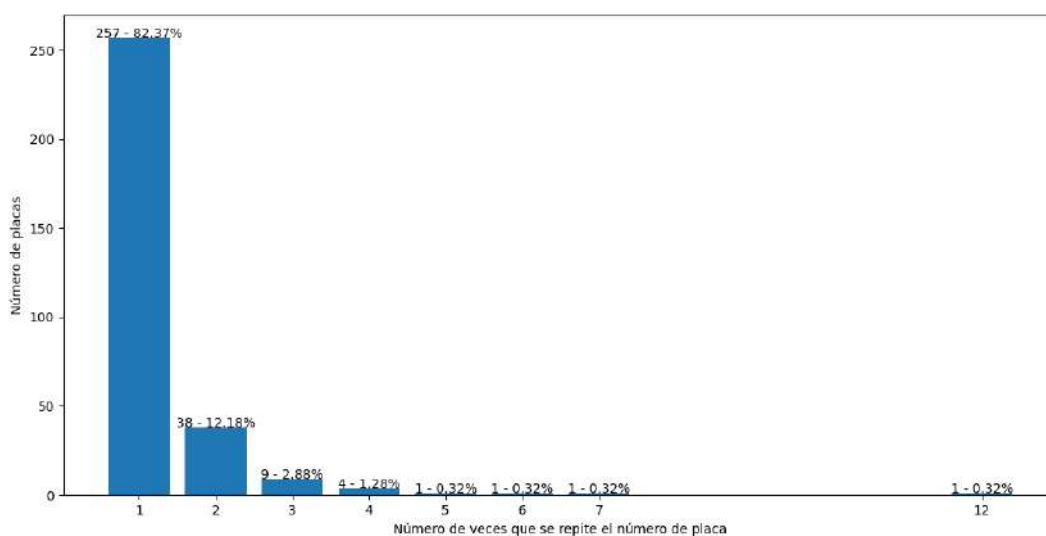


Figura 8. Distribución de número de veces que se repite una placa en el dataset

Fuente. Elaborado a partir del análisis estadístico de las fotografías que conforman el dataset de la presente investigación.

Del análisis previo se desprende que se tiene en total 312 placas de rodaje de distintos vehículos (que equivale al 76.85% del total de placas de rodaje etiquetadas), las cuales en su mayoría solo se repiten una sola vez. No

obstante, puede apreciarse que una placa de rodaje se repite hasta 12 veces. Este aspecto podría dar a lugar a un efecto de overfitting o sobreajuste, lo cual podría afectar la capacidad del modelo para generalizar a nuevos casos. En la Figura 9 se muestran las 12 imágenes que corresponden a la placa que se repite 12 veces en el dataset. Puede apreciarse que las mismas condiciones (inclinación, la iluminación, sombras, enfoque, ubicación de la placa en la imagen, entre otras) no se repiten entre las imágenes, lo cual permite reducir la probabilidad de overfitting [sobreajuste] durante el entrenamiento.

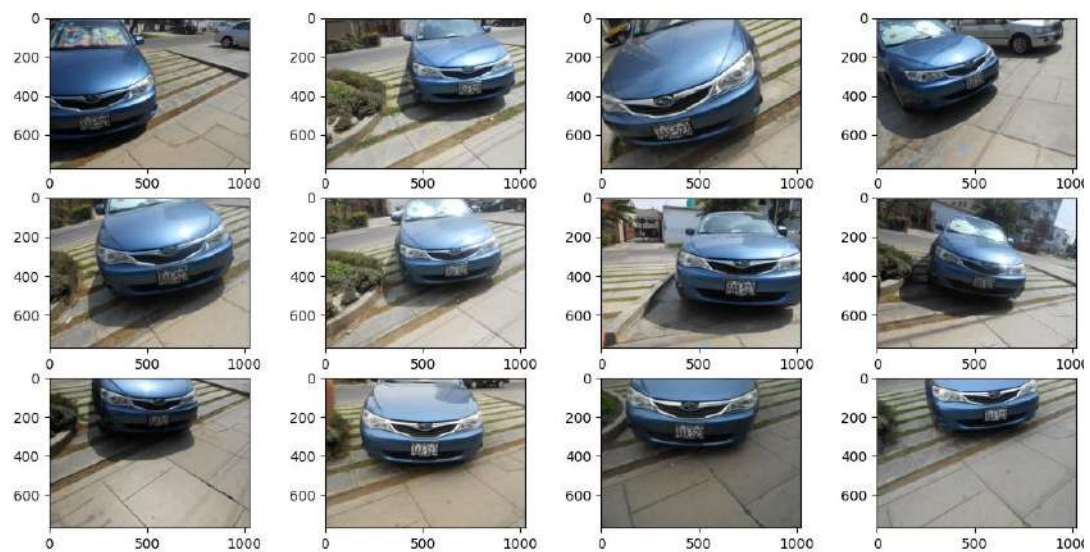


Figura 9. Imágenes correspondientes a la placa que se repite 12 veces
Fuente. Elaborado a partir del dataset de fotografías implementado en la presente investigación.

Se realiza este mismo análisis para el segundo caso con mayores repitencias, que corresponde a la placa que se repite 7 veces. En la Figura 10 se presentan las imágenes correspondientes a este caso. Puede apreciarse que cada imagen es única, lo cual permite reducir la probabilidad de overfitting durante el entrenamiento.



Figura 10. Imágenes correspondientes a la placa que se repite 7 veces
Fuente. Elaborado a partir del dataset de fotografías implementado en la presente investigación.

Otro aspecto relevante con relación a las placas de rodaje es conocer su relación de aspecto, la cual se puede definir como el cociente entre su anchura y su altura. Esta información permitió indicar, entre otros casos, si una placa está inclinada.

En la Figura 11 se presenta la distribución de la relación de aspecto de las placas etiquetadas en el dataset. Se precisa que para la determinación de los intervalos se aplicó la regla de Sturges. Los porcentajes mostrados fueron calculados con relación al total de 406 placas de rodaje etiquetadas.

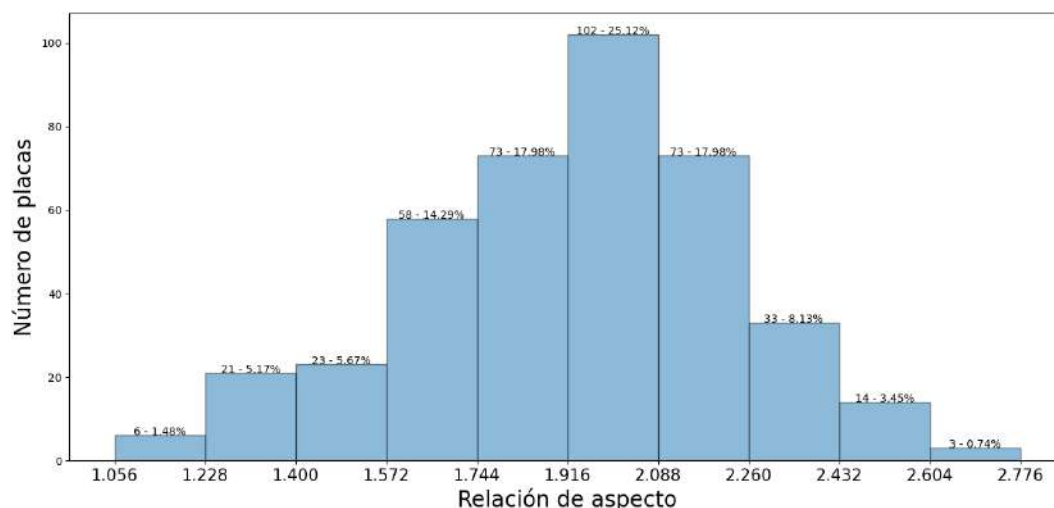


Figura 11. Distribución de la relación de aspecto de las placas de rodaje etiquetadas.

Fuente. Elaborado a partir del dataset de fotografías implementado en la presente investigación.

Se pudo apreciar de la distribución que esta tiende a ser normal. Los indicadores estadísticos de la distribución de la relación de aspecto de las placas de rodaje se presentan en el Cuadro 9.

Cuadro 9

Parámetros estadísticos de la relación de aspecto de las placas del dataset

Parámetro	Valor
Valor promedio	1.92
Desviación estándar	0.31
Valor mínimo	1.05
Valor máximo	2.77

Con base en la información obtenida sobre la relación de aspecto, se determinó un valor límite que permite identificar si una placa se encuentra inclinada. Para ello, se seleccionó y analizó una imagen representativa de cada intervalo de la distribución mostrada en la Figura 11, cuyos resultados

se presentan en la Figura 12. Cabe precisar que el término "RA" en las imágenes hace referencia al parámetro de relación de aspecto, calculado como el cociente entre el largo y el ancho de la imagen.



Figura 12. Distribución de la relación de aspecto de las placas de rodaje etiquetadas.

Fuente. Elaborado a partir del dataset implementado en la presente investigación.

Analizando las muestras de la Figura 12, se puede concluir que los primeros 5 intervalos presentan placas de rodaje con inclinación. En ese sentido, se puede definir al valor de 1.92, correspondiente al valor superior del intervalo 5, como límite para establecer la inclinación de la placa. De este modo, si la placa tiene una relación de aspecto inferior de 1.92 se considera que la placa está inclinada.

Tomando en cuenta este análisis, fue posible determinar cuántas placas de rodaje dentro del dataset se encuentran inclinadas y cuántas no. De este modo, se encontró que 184 placas de rodaje están inclinadas y 222 placas de

rodaje no están inclinadas, esto es, el 45.32% y el 54.68% del total de placas de rodaje respectivamente. De esta manera, se pudo concluir que el dataset incluye casi una misma proporción de placas de rodaje inclinadas y no inclinadas, con lo cual se cuenta con suficiente variedad, a diferencia de los trabajos evaluados en los antecedentes, en los cuales las placas analizadas se tomaban en condiciones controladas que permitían que la mayoría de estas se encuentren orientadas horizontalmente sin inclinación. Esto se resume en el Cuadro 10.

Cuadro 10.

Inclinación de placas en el dataset

Inclinación	Frecuencia	Frecuencia porcentual
Placas inclinadas (AR <1.92)	184	45.32%
Placas no inclinadas (AR > 1.92)	222	54.68%
Total	406	100%

La siguiente característica de las placas de rodaje analizado es el área de la región con relación al área total de la imagen. Este parámetro permitió tener una idea de la proporción de área que la placa de rodaje con respecto a la imagen total. En la Figura 13 se presenta esta distribución, en la cual los porcentajes mostrados fueron calculados con relación al total de 406 placas de rodaje etiquetadas. Puede apreciarse que la mayoría de las placas etiquetadas ocupan un área menos al 2.73% del área total de las imágenes (91.13% del total de las placas de rodaje), mientras que un 7.39% de las placas de rodaje ocupan entre 2.73% y 5.39% del total del área de la imagen. La información estadística de este parámetro se presenta en el Cuadro 11.

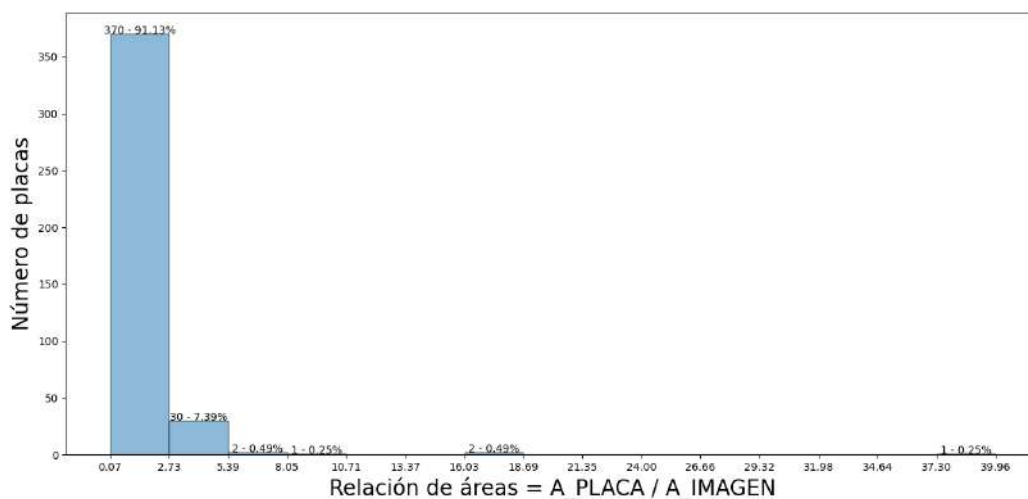


Figura 13. Distribución de la relación de área de las placas de rodaje respecto al área de la imagen.

Cuadro 11

Parámetros estadísticos de la relación de área de las placas del dataset

Parámetro	Valor
Valor promedio	1.50%
Desviación estándar	2.43%
Valor mínimo	0.07%
Valor máximo	39.96%

Observando los parámetros estadísticos, puede verse que la muestra tiene una desviación estándar que es más grande que la media. Esto implica que la muestra, en cuanto a la relación del área de la placa respecto del área de la imagen, es variada, lo cual es favorable para que el modelo pueda generalizar adecuadamente.

En cuanto a las características propias de la imagen, se analizó el contraste de las placas de rodaje. El contraste puede definirse como la diferencia en la iluminación en una imagen, que hace que se aprecie poca diferencia entre colores. Una imagen con bajo contraste puede ser o bien demasiado oscura o iluminada.

Para este análisis, se empleó la librería “scikit-image”⁵, la cual incluye el módulo “exposure”. En este módulo, se implementa la función “is_low_contrast()”, la cual admite como entradas la imagen a analizar y una fracción de disparo. Esta fracción de disparo, denominada como “fraction_threshold”, determina si una imagen tiene bajo contraste o no. Para ello, la función calcula el rango de brillos en la imagen y luego compara este rango contra el rango total de valores de la imagen. Si la fracción obtenida es menor a la fracción de disparo, se considera que la imagen tiene bajo contraste. Como ejemplo, una imagen en escala de grises tiene un rango total de 255, en la medida de que cada píxel puede tomar un valor entre 0 y 255. Si el rango de brillos en la imagen es de 50 (todos los píxeles son oscuros con valores entre 10 y 60), la fracción es de $50/255 = 0.19$. Si se considera que la fracción de disparo debe ser 0.36, entonces esta imagen tiene bajo contraste. El valor de 0.36 como fracción de disparo se elige de manera empírica a partir de varios experimentos realizados con las placas que conforman el dataset.

De este modo, se encuentra que en el dataset que 19 placas de rodaje tienen bajo contraste, equivalentes al 4.68% del total de placas de rodaje. En el Cuadro 12⁶ se presenta esta información. En la Figura 14 se muestran las 19 placas de rodaje que, de acuerdo con el análisis realizado, tienen bajo contraste.

⁵ La librería scikit-image es una librería para procesamiento de imágenes en Python, disponible en: <https://scikit-image.org/>

⁶ En la tabla, el término FT se refiere al parámetro fraction_threshold de la función is_low_contrast de la librería scikit-image.

Cuadro 12

Contraste de las placas en el dataset

Contraste	Frecuencia	Frecuencia porcentual
Placas con bajo contraste (FT < 0.36)	19	4.68%
Placas con contraste adecuado (FT > 0.36)	387	95.32%
Total	406	100%

**Figura 14. Placas con bajo contraste en el dataset**

Fuente. Elaborado a partir del dataset implementado para la presente investigación.

Estas placas de rodaje con bajo contraste se incluyeron en el proceso de entrenamiento con la finalidad de que el sistema sea capaz de generalizar y poder detectar placas en diversas condiciones.

Finalmente, se analizó la característica de las placas de rodaje denominada borrosidad, la cual se puede definir como la falta de definición en los bordes de una imagen.

Para determinar el nivel de borrosidad de una imagen, se tomó la métrica descrita por Pech-Pacheco et al., (2000), la cual consiste en calcular la

varianza de los valores absolutos de la resultante de aplicar un filtro Laplaciano sobre la imagen. En la medida que el nivel de borrosidad está planteado para determinar la falta de enfoque que tiene una imagen, se empleará este método para su análisis. De este modo, evaluando este parámetro y definiendo un valor como valor umbral, fue posible determinar si una imagen está borrosa o no.

Para ello, se procedió a calcular este indicador, al que se denominó enfoque, para cada una de las placas de rodaje que conforman el dataset. En el Cuadro 13 se presentan los resultados estadísticos del enfoque calculado para placa del dataset.

Cuadro 13

Parámetros estadísticos del enfoque de las placas del *dataset*

Parámetro	Valor
Valor promedio	1 784.42
Desviación estándar	1 937.35
Valor mínimo	31.24
Valor máximo	14 638.83

Para poder determinar el valor adecuado de enfoque con el cual determinar si una placa está borrosa o no, se procedió a analizar su distribución, la cual se presenta en la Figura 15, en la cual los porcentajes indicados fueron calculados con relación al total de 406 placas de rodaje etiquetadas.

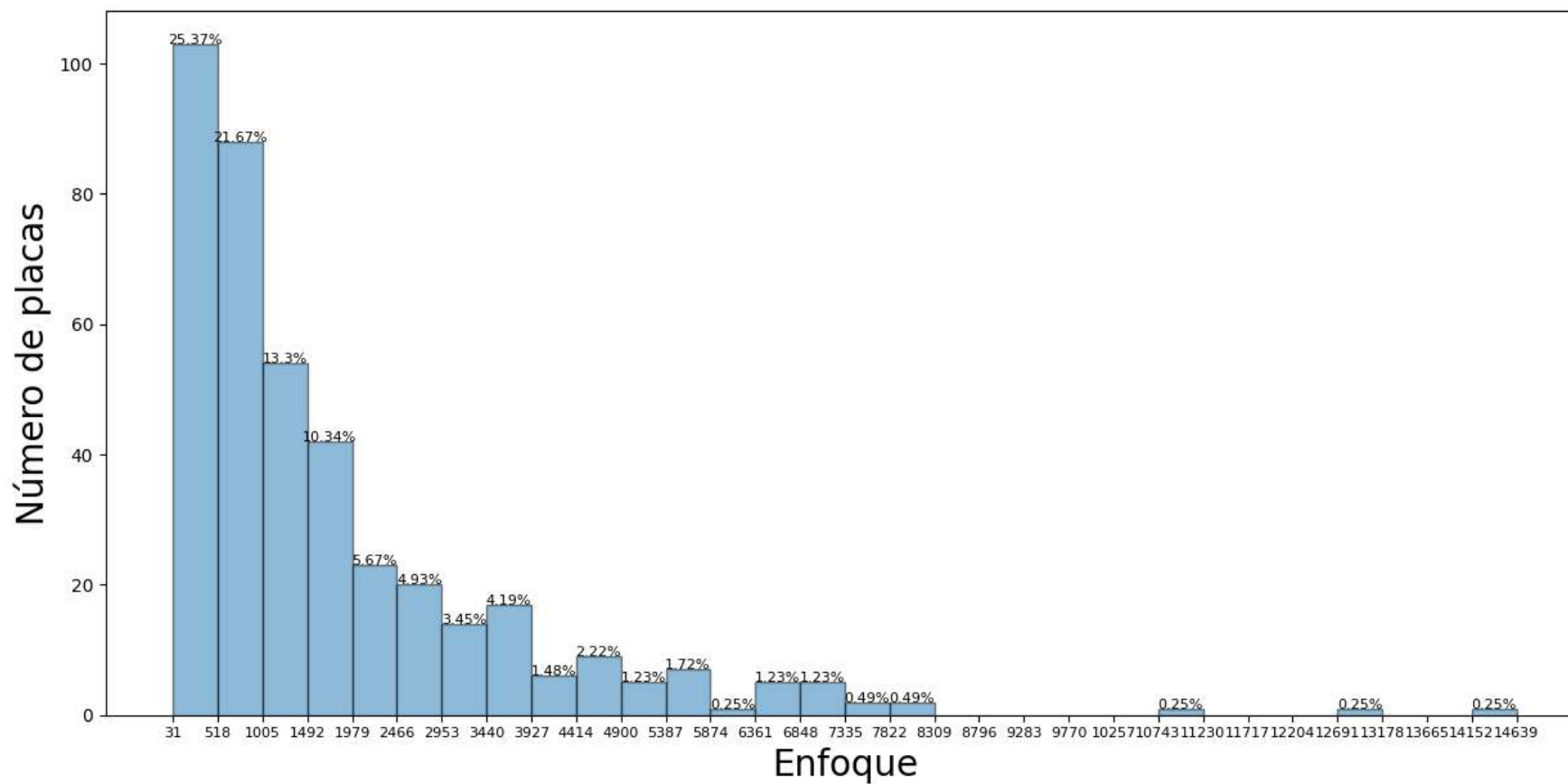


Figura 15. Distribución del enfoque de las placas de rodaje.

En particular, interesa analizar los intervalos para enfoques bajos, en la medida de que mientras menor valor tenga el enfoque, más borrosa es la imagen.

Para analizar el caso, se toman tres muestras aleatorias de los primeros tres intervalos, con la finalidad de determinar el nivel de borrosidad de las imágenes de cada de intervalo. Estas muestras se presentan en la Figura 16, en la que cada fila corresponde a placas de un solo intervalo. Se incorpora el valor F correspondiente al enfoque calculado para dicha placa de rodaje.



Figura 16. Placas vehiculares correspondientes a los intervalos con menores niveles de enfoque

Analizando las muestras, se pudo encontrar que las placas que corresponden al intervalo 1 son las más borrosas. Del intervalo 2 se tiene una placa algo borrosa, la correspondiente al enfoque de 530.91, no obstante, aún se pueden apreciar claramente los valores de las placas. Las placas del intervalo 3 en cambio tienen sus bordes más definidos, por lo cual la borrosidad de dicho intervalo, a partir de sus muestras, se puede considerar que no afecta la legibilidad de la placa.

Por lo tanto, se procedió a expandir la información de enfoque de las placas del primer intervalo, de manera que se tenga información para poder definir el valor de enfoque mínimo a partir del cual se puede afirmar que la placa de rodaje no está borrosa.

En el Cuadro 14 se presentan los valores de los principales parámetros estadísticos de las placas del intervalo 1 con relación al enfoque.

Cuadro 14

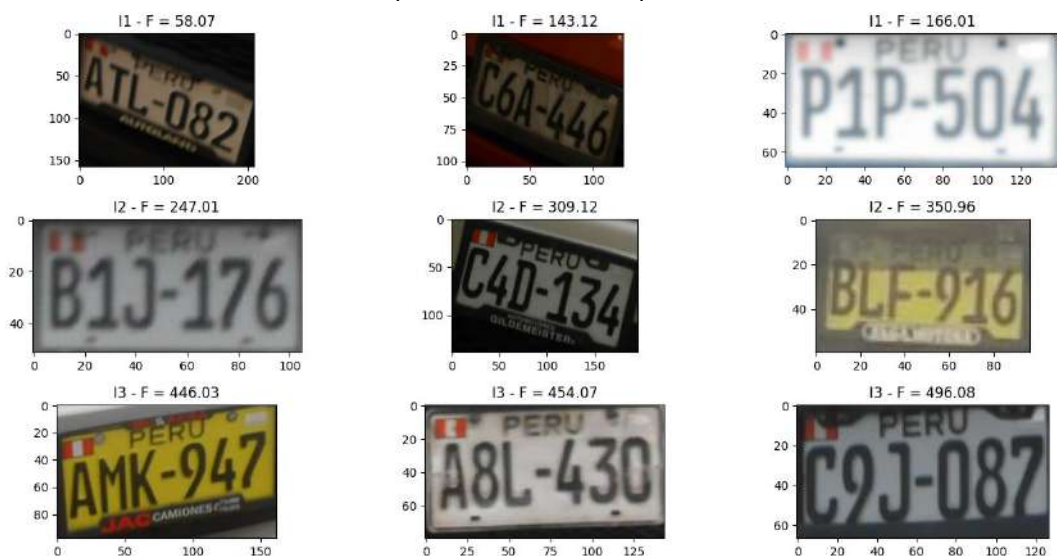
Parámetros estadísticos del enfoque de las placas del primer intervalo

Parámetro	Valor
Valor promedio	289.83
Desviación estándar	130.31
Valor mínimo	31.24
Valor máximo	517.94

Puede apreciarse que la desviación estándar del intervalo 1, en este caso es menor que el promedio, lo cual, tomando en cuenta el valor mínimo y el valor máximo, da a entender que para este intervalo el enfoque varía casi uniformemente.

En ese sentido, con la finalidad de determinar el valor de enfoque con el cual determinar si una placa de rodaje está borrosa o no con mejor precisión, se dividió el rango de este intervalo en 3 partes, y se plotean 3 muestras aleatorias de cada subintervalo dos veces, de manera que se pueda tener dos muestreos que permitan evaluar mejor la borrosidad. En la Figura 17 se presentan los resultados, en la cual cada fila corresponde a placas de un solo intervalo. Se incorpora el valor F correspondiente al enfoque calculado para dicha placa de rodaje.

(Primer muestreo)



(Segundo muestreo)



Figura 17. Muestreo de placas vehiculares en el intervalo con menor enfoque

De este experimento, al comparar los resultados, se pudieron identificar dos aspectos: (1) el indicador elegido tiende a dar un menor nivel de enfoque a las placas de rodaje inclinadas (este aspecto se debe tomar en cuenta al momento de realizar el preprocesamiento para fines de detección de caracteres) y (2) para niveles de enfoque inferiores a aproximadamente 400, la placa ya tiene un grado de borrosidad que podría dificultar su legibilidad. En ese sentido, para los fines de la presente investigación, se tomó como valor de disparo un enfoque de 400. Placas con un enfoque inferior a 400 fueron calificadas como borrosas.

Aplicando este criterio, se hace la evaluación estadística del dataset, cuyos resultados se presenta en el Cuadro 15, en el cual el parámetro F se refiere al parámetro de enfoque, calculado según el método de la segunda derivada descrito por Pech-Pacheco et al., (2000).

Cuadro 15

Borrosidad de las placas en el dataset

Contraste	Frecuencia	Frecuencia porcentual
Placas borrosas ($F < 400$)	77	18.97%
Placas no borrosas ($F > 400$)	329	81.03%
Total	406	100%

4.5.3. Resumen del análisis estadístico

El análisis estadístico del dataset permitió estudiar las principales características de las imágenes que lo conforman. Se evaluaron 3 características principales para las placas de rodaje: inclinación, contraste y borrosidad.

Las proporciones de las placas según estas 3 características se presentan en el Cuadro 16.

Cuadro 16

Evaluación de la generalidad del dataset según las características analizadas

Característica	Número de casos	Proporción respecto del total
Placa inclinada	184	45.32%
Placa con bajo contraste	19	4.68%
Placa borrosa	77	18.97%

Como puede apreciarse, el dataset implementado contiene casos para todas las características consideradas, por lo cual se puede concluir que cuenta con muestras que no han sido controladas, es decir, se tienen placas inclinadas, de bajo contraste y borrosas, lo cual, de cara al entrenamiento y evaluación, puede permitir obtener un modelo que generalice de manera adecuada.

V. RESULTADOS

5.1. Presentación y análisis de resultados

En el presente apartado se presentan los resultados de las etapas de entrenamiento y evaluación.

Para la implementación del modelo detector se utilizaron un total de 392 imágenes, de las cuales se usó el 80% del dataset para la fase de entrenamiento y el 20% para realizar las evaluaciones. Esto permitió medir el desempeño del modelo con mayor precisión y eficacia, en la medida que permite verificar la capacidad de detección del modelo ante imágenes que nunca les fueron presentadas durante el entrenamiento. La selección de qué imágenes se emplean en cada grupo se hizo de manera aleatoria.

5.1.1. Consideraciones previas con relación al entrenamiento de los modelos de detección

Con la finalidad de determinar el modelo con el mejor desempeño se tomaron previamente en consideración los siguientes aspectos:

- Velocidad de Inferencia (ms): tiempo que toma el modelo para generar una predicción. Este parámetro es importante, en la medida de que se busca un modelo base que tenga un tiempo de inferencia no muy elevado, pues se procura un funcionamiento en tiempo real.
- Tamaño del modelo (máximo 100 MB): tamaño total del modelo, que incluye el archivo con las matrices de pesos correspondientes, así como sus archivos de configuración. Se limita en 100 MB para que el modelo pueda embeberse en microcontroladores o tarjetas electrónicas pequeñas.

Considerando estos criterios, se seleccionaron los siguientes 6 modelos pre entrenados indicados en el Cuadro 17.

Cuadro 17

Modelos pre entrenados seleccionados para el análisis comparativo

Nombre de modelo	Tiempo inferencia (ms)	mAP (%)	Tamaño (MB)
EfficientDet D0 512x512	39	33.6	63.76
EfficientDet D1 640x640	54	38.4	91.19
SSD MobileNet v2 320x320	19	20.2	55.55
SSD MobileNet V1 FPN 640x640	48	29.1	99.79
SSD MobileNet V2 FPNLite 320x320	22	22.2	29.79
SSD MobileNet V2 FPNLite 640x640	39	28.2	29.79

Fuente. Adaptado de Tensorflow 2 Detection Model Zoo, por (Tensorflow, s. f.), GitHub.
https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md).

Se realizó el entrenamiento por ajuste fino de estos modelos con el dataset conformado por 392 imágenes (de las cuales el 80% se emplearon en el entrenamiento y el 20% en la evaluación). Este entrenamiento se realizó en la plataforma virtual de Google Colab⁷, para lo cual se implementó el script correspondiente en lenguaje Python.

De entre todos los hiperparámetros configurables para cada modelo, se ajustaron los siguientes:

- Tamaño de lote (batch size) de 4 muestras. Se eligió este número en la medida de que a mayor tamaño de lote se requiere mayor memoria de

⁷ Como parte de la investigación, se dispone del Script implementado en la plataforma Google Colab, disponible en la siguiente URL:
https://colab.research.google.com/drive/178hMJ4hOm9QHkOi7hubeZ2R_uvBA7XpV?usp=sharing

procesamiento, puesto que por cada iteración se tienen que cargar en la memoria la información de 4 imágenes.

- Número de pasos totales: 8,000. Esto implica que el entrenamiento considera 8,000 actualizaciones de los pesos del modelo, para cada lote (conformado por 4 imágenes), con una validación cada 1,000 pasos.
- Tamaño de lote para evaluación (batch size for evaluation): Se elige el valor de 1, en la medida de que para los fines de evaluación se desean calcular los parámetros de desempeño para cada imagen que conforme el subdataset de evaluación. Se reitera que, con la finalidad de obtener indicadores de desempeño consistentes, se recomienda realizar la evaluación del modelo con imágenes que no ha “visto” durante su entrenamiento. Esto permite evaluar la capacidad de generalización del modelo entrenado.
- Se empleó una tasa de aprendizaje dinámica, modelada por la función de decaimiento del coseno (cosine decay), la cual es la función que se emplea por defecto. Se considera tasa de aprendizaje base de 0.05 y tasa de aprendizaje de calentamiento de 0.001, con un calentamiento de 80 pasos. Bajo estos parámetros, la tasa de aprendizaje inició en 0.001, y se incrementó de tal forma que en la iteración 80 su valor fue de 0.05. Luego de alcanzar ese valor, la tasa de aprendizaje decae hasta ser 0 para el final del entrenamiento (iteración 8,000). Esto permitió que al inicio se realice un ajuste significativo de los pesos del modelo, en la medida de que en principio se asume que el error es grande. Conforme se avanzan las iteraciones, el error disminuye, por lo cual, para asegurar que el modelo converja y no sea inestable, la tasa de aprendizaje disminuye. En la Figura 18 se presenta la curva para la tasa de entrenamiento que se consideró durante los entrenamientos.

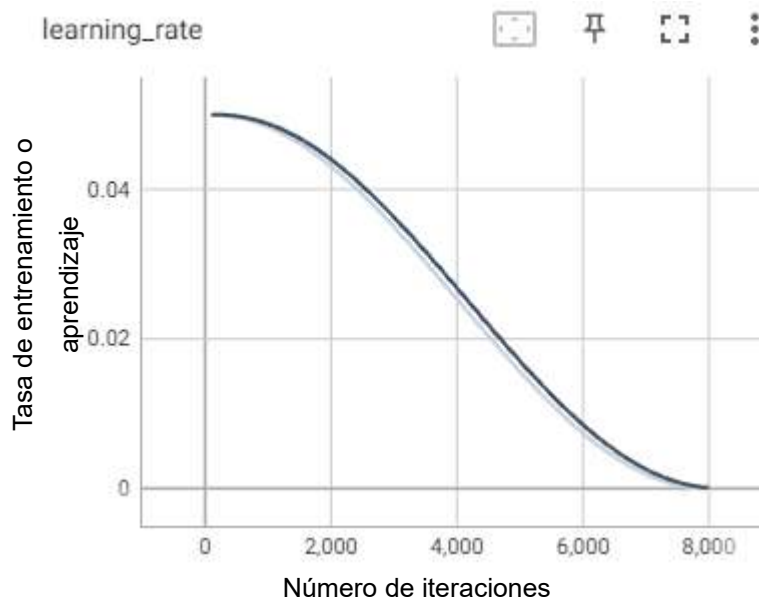


Figura 18. Curva de la tasa de entrenamiento considerado decaimiento por coseno.

Fuente. Obtenido por intermedio de la herramienta Tensorboard proporcionada por Tensorflow.

Para la medición del tiempo promedio de inferencia, se empleó una computadora portátil con procesador Intel Core I5, con 8GB de memoria RAM y sin GPU, en la medida de dicho hardware es el que se asemeja más a la SBC Raspberry Pi considerada para el prototipo.

5.1.2. Resultados del entrenamiento y evaluación de los modelos de detección

En el presente acápite se presentan los resultados del proceso de entrenamiento de ajuste fino de los modelos seleccionados, en función a los indicadores definidos en el Cuadro 4 y otros parámetros.

5.1.2.1. Resultados para el modelo base EfficientDet D0. Este proceso basado en el modelo EfficientDet D0 tuvo un tiempo de entrenamiento (T_t) de 2 181.80 segundos, con un tiempo de actualización de pesos de aproximadamente 0.25 segundos por iteración.

El modelo resultante se denominó “peruvian_efficientdet_d0_512x512”. Los resultados del proceso de entrenamiento, incluyendo los errores finales obtenidos se presentan en el Cuadro 18.

Cuadro 18

Resultados del proceso de entrenamiento por ajuste fino del modelo “peruvian_efficientdet_d0_512x512”

Ítem	Valor
Modelo base	EfficientDet D0
Fecha y hora de inicio de entrenamiento (T_{ti})	2023-04-03 02:32:35.26
Fecha y hora de fin de entrenamiento (T_{tf})	2023-04-03 03:08:57.06
I1: Tiempo de entrenamiento ($T_t = T_{tf} - T_{ti}$) (*)	2 181.80 segundos
Tiempo por iteración (actualización)	Aprox. 0.25 segundos
Error de clasificación al final del entrenamiento	0.08924257
Error de localización al final del entrenamiento	0.0012420485
Error de regularización al final del entrenamiento	0.032723896
Error total al final del entrenamiento	0.123208515

Fuente. Elaboración propia.

Nota. (*) De acuerdo con lo definido en el Cuadro 4. Operacionalización de variables de la presente investigación.

En la Figura 19 se presentan las curvas de los errores de entrenamiento en función de las iteraciones. En cuanto al error de clasificación, pudo observarse que este va disminuyendo conforme se avanza en las iteraciones, hasta alcanzar el valor de 0.089 en la iteración 8 000 (fin del entrenamiento). Este decaimiento es el esperado, indicando que el modelo converge hacia un mínimo de error. En lo que respecta al error de localización, se pudo observar que este también disminuye conforme se llega a la iteración 8 000, alcanzado el valor de 0.001 en dicha iteración. En cuanto al error de regularización, se pudo apreciar que este alcanzó su máximo valor aproximadamente en la iteración 3 000, luego de la cual disminuye para mantenerse casi constante, alcanzando el valor de 0.033 en la iteración 8 000. Observando el error total, el cual considera los 03 errores descritos previamente, se pudo apreciar que este disminuye conforme aumentan las iteraciones, hasta alcanzar el valor de 0.1232 al final del entrenamiento.

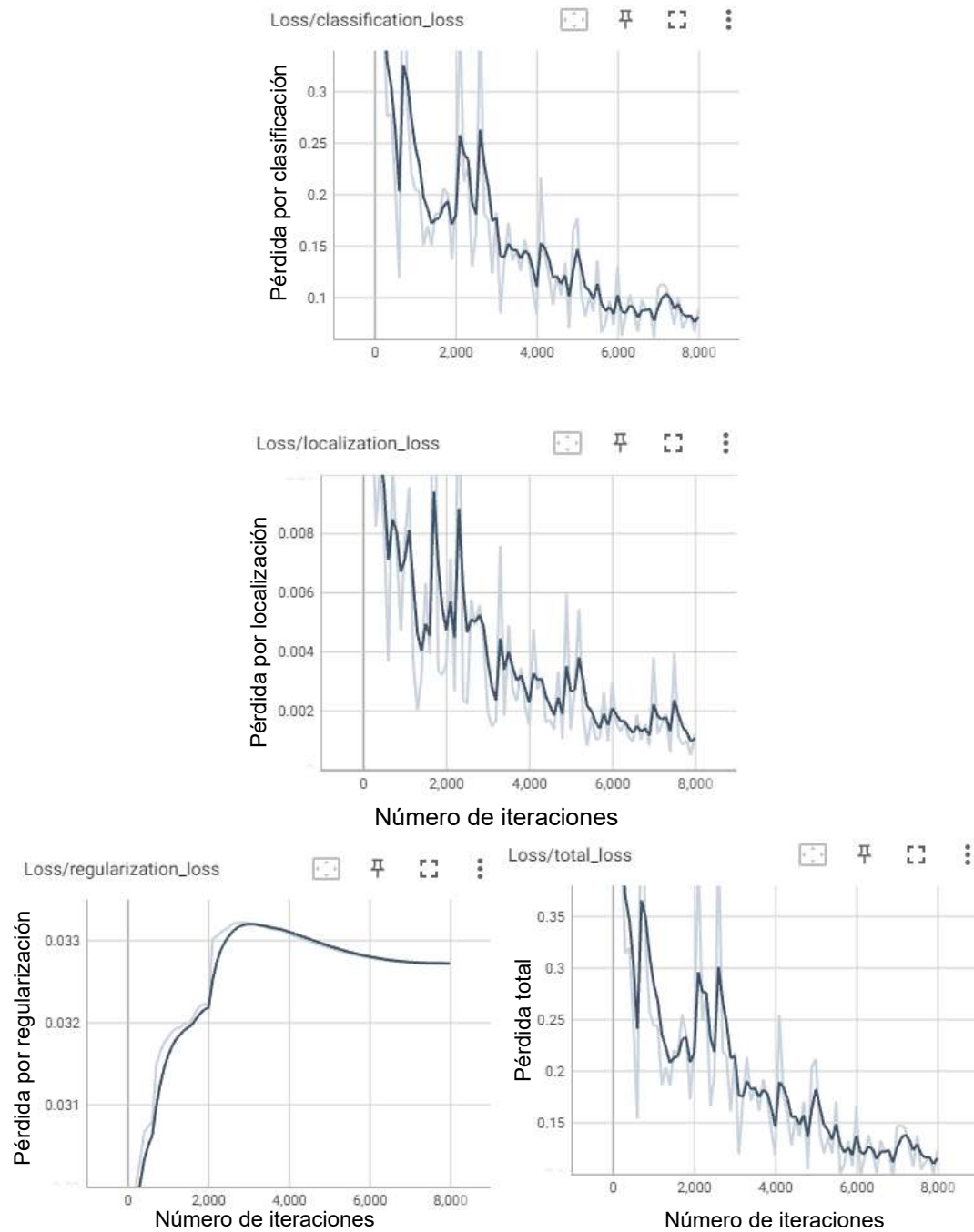


Figura 19. Curvas de los errores durante el entrenamiento del modelo “peruvian_efficientdet_d0_512x512”

Fuente. Obtenido por intermedio de la herramienta Tensorboard proporcionada por Tensorflow.

Con relación a la evaluación del desempeño del modelo, en el Cuadro 19 se presentan los resultados obtenidos. Esta evaluación se realizó tomando en cuenta el indicador I5: Precisión promedio, y de manera complementaria, con el indicador recuperación.

Cuadro 19

Resultados de la evaluación del modelo “peruvian_efficientdet_d0_512x512”

Indicador de desempeño	IoU	Máximo de detecciones por imagen	Valor
I5: Precisión promedio (AP)			
	IoU=0.50:0.95	100	0.752
	IoU=0.50	100	0.968
	IoU=0.75	100	0.949
Recuperación (RE)			
	IoU=0.50:0.95	1	0.773
	IoU=0.50:0.95	10	0.781
	IoU=0.50:0.95	100	0.786

Fuente. Elaborado a partir de los valores obtenidos calculados mediante la API de Tensorflow para la detección de objetos.

Tomando en cuenta que la precisión media promedio (mAP) del modelo base era de 0.336, se pudo apreciar que en todos los casos evaluados se logró una mejora de aproximadamente el 100% con respecto a este indicador.

En esta etapa, se calcularon los tiempos de inferencia para cada una de las imágenes que conforman el dataset, cuyo promedio se empleó para caracterizar al modelo en general. Los resultados de este análisis se presentan en el Cuadro 20, en el cual los tiempos fueron calculados considerando el caso de que todas las imágenes a evaluar no tienen las mismas dimensiones, lo cual tiende a generar faltas de optimización y reducciones en el rendimiento.

Cuadro 20

Resultados de la evaluación de la inferencia del modelo “peruvian_efficientdet_d0_512x512” sobre la totalidad del dataset

Ítem	Valor
Número de imágenes procesadas	392
Número de placas etiquetadas (N)	406
Número de placas predichas (P)	389
Número de placas detectadas (D)	388
Falsos positivos (P-D)	1
Ratio de detección (D/N)	0.9557
Mínimo tiempo de inferencia	0.52 segundos
Máximo tiempo de inferencia	35.31 segundos
Tiempo de inferencia promedio	2.07 segundos

Es relevante mencionar que durante la evaluación se encontró un caso de falso positivo. Este caso se muestra en la Figura 20, que corresponde a un error de clasificación, ya que el modelo clasificó como placa un objeto que no lo era. Esto podría mitigarse elevando el puntaje (score) a partir del cual se considera que un objeto es una placa, no obstante, esto podría afectar la detección de otros casos. Por ejemplo, si se eleva el puntaje a 0.90, la cantidad de placas detectadas correctamente disminuye en 10.

Caso 1
Score: 0.87285125

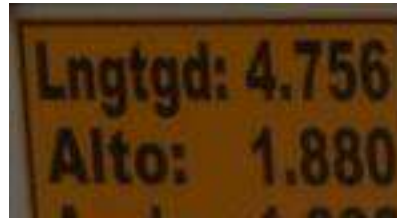


Figura 20. Falso positivo encontrado durante la evaluación de los tiempos de inferencia del modelo “peruvian_efficientdet_d0_512x512”.
Fuente. Elaborado a partir de la evaluación del modelo bajo análisis.

Finalmente, en la Figura 21 se presenta la distribución de los tiempos de inferencia para cada una de las imágenes, observándose que la gran mayoría se encuentra por debajo de los 2.5 segundos.

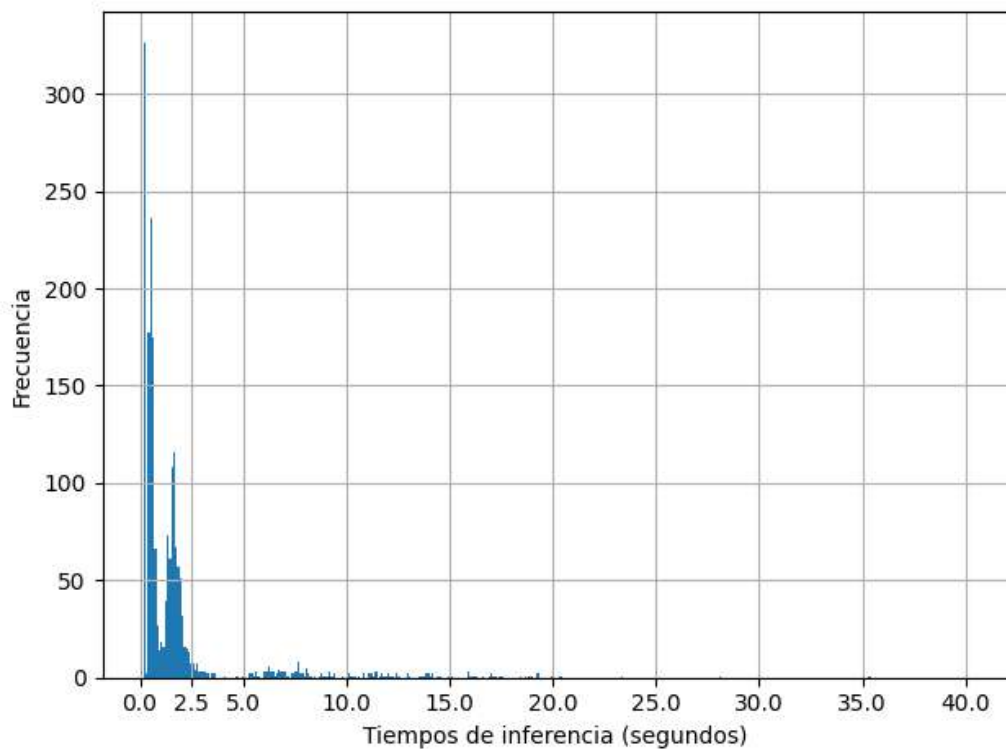


Figura 21. Distribución de los tiempos de inferencia calculados para el modelo “peruvian_efficientdet_d0_512x512”
Fuente. Elaborado a partir de los tiempos calculados empleando la función time de la librería time de Python. Se considera una placa detectada si la probabilidad de detección (score) es de mínimo 70%.

5.1.2.2. Resultados para el modelo base EfficientDet D1

El entrenamiento por ajuste fino realizado tomando como base el modelo EfficientDet D1 tuvo un tiempo de entrenamiento (T_t) que tomó 4,189.04 segundos en realizarse, con un tiempo de actualización de pesos de aproximadamente 0.5 segundos por iteración.

El modelo resultante se denominó “peruvian_efficientdet_d1_640x640”. Los resultados del proceso de entrenamiento, incluyendo los errores finales obtenidos se presentan en el Cuadro 21.

Cuadro 21

Resultados del proceso de entrenamiento por ajuste fino del modelo “peruvian_efficientdet_d1_640x640”

Ítem	Valor
Modelo base	EfficientDet D1
Fecha y hora de inicio de entrenamiento (T_{ti})	2023-04-03 03:15:08.86
Fecha y hora de fin de entrenamiento (T_{tf})	2023-04-03 04:24:57.90
I1: Tiempo de entrenamiento ($T_t = T_{tf} - T_{ti}$) (*)	4 189.04 segundos
Tiempo por iteración (actualización)	Aprox. 0.50 segundos
Error de clasificación al final del entrenamiento	0.076872915
Error de localización al final del entrenamiento	0.0011958205
Error de regularización al final del entrenamiento	0.034553345
Error total al final del entrenamiento	0.11262208

Fuente. Elaboración propia.

Nota. (*) De acuerdo con lo definido en el Cuadro 4. Operacionalización de variables de la presente investigación.

En la Figura 22 se presenta las curvas de los errores de entrenamiento en función de las iteraciones. En cuanto al error de clasificación, pudo observarse que este va disminuyendo conforme se avanzan en las iteraciones, hasta

alcanzar el valor de 0.077 en la iteración 8,000 (fin del entrenamiento). Este decaimiento es el esperado, indicando que el modelo converge hacia un mínimo de error, no obstante, puede apreciarse un segundo pico cerca de la iteración 2,300. En lo que respecta al error de localización, se pudo observar que este también disminuye conforme se llega a la iteración 8,000, alcanzado el valor de 0.001 en dicha iteración, con picos en los alrededores de las iteraciones 2,300 y 3,600. En cuanto al error de regularización, se pudo apreciar que este alcanzó su máximo valor aproximadamente en la iteración 3,000, luego de la cual disminuye para mantenerse casi constante, alcanzando el valor de 0.035 en la iteración 8,000. Observando el error total, el cual considera los 03 errores descritos previamente, se pudo apreciar que este disminuye conforme aumentan las iteraciones, hasta alcanzar el valor de 0.1232 al final del entrenamiento.



Figura 22. Curvas de los errores durante el entrenamiento del modelo “peruvian_efficientdet_d1_640x640”.

Fuente. Obtenido por intermedio de la herramienta Tensorboard proporcionada por Tensorflow.

Con relación a la evaluación del desempeño del modelo, en el Cuadro 22 se presentan los resultados obtenidos en función al indicado I5: Precisión promedio y de manera complementaria con el indicador de recuperación.

Cuadro 22

Resultados de la evaluación del modelo “peruvian_efficientdet_d1_640x640”

Indicador de desempeño	IoU	Máximo de detecciones por imagen	Valor
I5: Precisión promedio (AP)			
	IoU=0.50:0.95	100	0.766
	IoU=0.50	100	0.969
	IoU=0.75	100	0.929
Recuperación (RE)			
	IoU=0.50:0.95	1	0.793
	IoU=0.50:0.95	10	0.796
	IoU=0.50:0.95	100	0.796

Fuente. Elaborado a partir de los valores obtenidos calculados mediante la API de Tensorflow para la detección de objetos.

Tomando en cuenta que la precisión media promedio (mAP) del modelo base era de 0.384, se pudo apreciar que en todos los casos evaluados se logró una mejora de aproximadamente el 100% con respecto a este indicador.

En esta etapa, se calcularon los tiempos de inferencia para cada una de las imágenes que conforman el dataset, cuyo promedio se empleó para caracterizar al modelo en general. Los resultados de este análisis se presentan en el Cuadro 23, en el cual los tiempos fueron calculados considerando que todas las imágenes del dataset no tienen las mismas dimensiones, lo cual tiende a generar faltas de optimización y reducciones en el rendimiento.

Cuadro 23

Resultados de la evaluación de la inferencia del modelo “peruvian_efficientdet_d1_640x640” sobre la totalidad del dataset

Ítem	Valor
Número de imágenes procesadas	392
Número de placas etiquetadas (N)	406
Número de placas predichas (P)	399
Número de placas detectadas (D)	397
Falsos positivos (P-D)	2
Ratio de detección (D/N)	0.9778
Mínimo tiempo de inferencia	1.04 segundos
Máximo tiempo de inferencia	40.08 segundos
Tiempo de inferencia promedio	2.60 segundos

Se debe indicar que en la evaluación se encontraron casos de falsos positivos. Estos 2 casos se presentan en la Figura 23, los cuales corresponden a un error de localización, ya que, si bien el modelo pudo detectar la placa, segmentó el objeto en partes, generando así dos falsos positivos. Cabe resaltar que para esta imagen modelo si predijo, además de los falsos positivos, la placa completa.



Figura 23. Falsos positivos encontrados durante la evaluación de los tiempos de inferencia del modelo “peruvian_efficientdet_d1_640x640”
Fuente. Elaborado a partir de la evaluación del modelo bajo análisis.

En la Figura 24 se presenta la distribución de los tiempos de inferencia para cada una de las imágenes, observándose que la gran mayoría se encuentra por debajo de los 2.5 segundos.

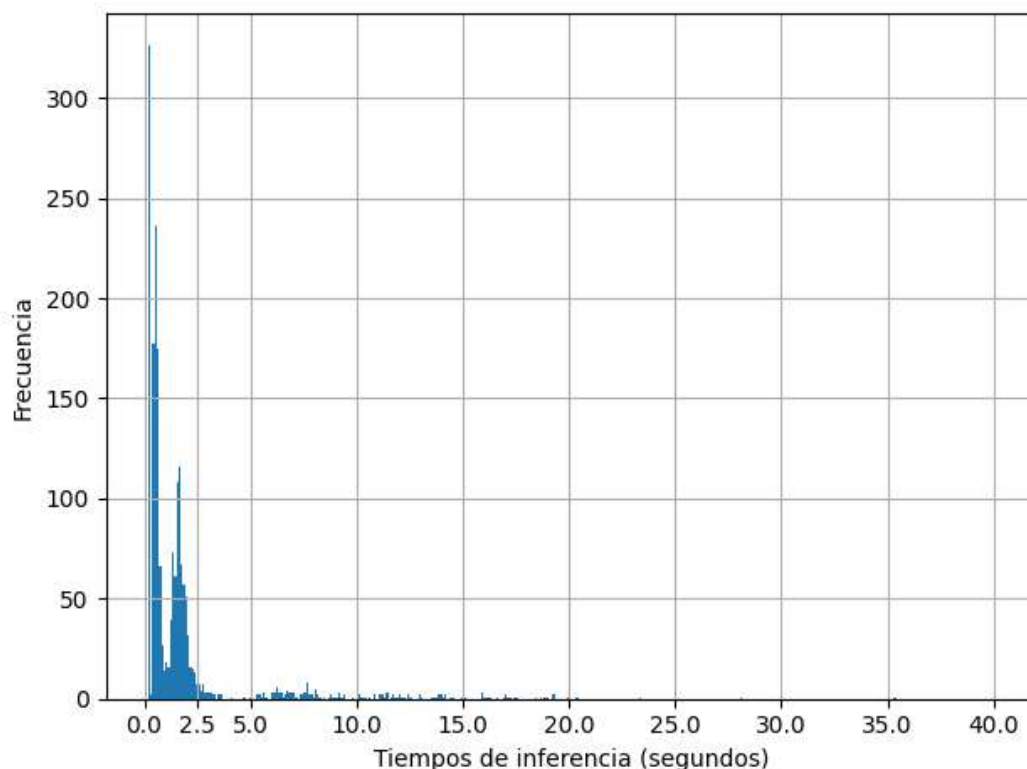


Figura 24. Distribución de los tiempos de inferencia calculados para el modelo “*peruvian_efficientdet_d1_640x640*”.

Nota. Elaborado a partir de los tiempos calculados empleando la función `time` de la librería `time` de Python. Se considera una placa detectada si la probabilidad de detección (`score`) es de mínimo 70%.

5.1.2.3. Resultados para el modelo base SSD MobileNet v2 320x320

El entrenamiento por ajuste fino realizado tomando como base el modelo SSD MobileNet v2 320x320 tuvo un tiempo de entrenamiento (T_t) que tomó 590.77 segundos en realizarse, con un tiempo de actualización de pesos de aproximadamente 0.05 segundos por iteración.

El modelo resultante se denominó “*peruvian_ssd_mobilenet_v2_320x320*”. Los resultados del proceso de entrenamiento, incluyendo los errores finales obtenidos se presentan en el Cuadro 24.

Cuadro 24

Resultados del proceso de entrenamiento por ajuste fino del modelo “peruvian_ssd_mobilenet_v2_320x320”

Ítem	Valor
Modelo base	SSD MobileNet v2 320x320
Fecha y hora de inicio de entrenamiento (T_{ti})	2023-04-03 01:37:54.32
Fecha y hora de fin de entrenamiento (T_{tf})	2023-04-03 01:47:45.09
I1: Tiempo de entrenamiento ($T_t = T_{tf} - T_{ti}$) (*)	590.77 segundos
Tiempo por iteración (actualización)	Aprox. 0.05 segundos
Error de clasificación al final del entrenamiento	1.438148
Error de localización al final del entrenamiento	0.5808509
Error de regularización al final del entrenamiento	1236.0234
Error total al final del entrenamiento	1238.0425

Nota. (*) De acuerdo con lo definido en el Cuadro 4. Operacionalización de variables de la presente investigación.

En la Figura 25 se presenta las curvas de los errores de entrenamiento en función de las iteraciones. En cuanto al error de clasificación, pudo observarse que este fluctúa sin seguir una tendencia creciente o decreciente conforme avanzan las iteraciones, hasta alcanzar el valor de 1.44 en la iteración 8,000 (fin del entrenamiento), lo cual indica que el modelo entrenado no tiene una buena capacidad de clasificación. En lo que respecta al error de localización, se pudo observar que este si sigue una tendencia decreciente, disminuyendo conforme se llega a la iteración 8,000, alcanzado el valor de 0.58 en dicha iteración, el cual, no obstante, es demasiado alto para los fines de la tarea de detectar placas vehiculares. En cuanto al error de regularización, se pudo apreciar que este alcanzó su máximo valor aproximadamente en la iteración 1,100, luego de la cual disminuye, alcanzando el valor de 1,238.04 en la iteración 8,000. Observando el error total, el cual considera los 3 errores

descritos previamente, se pudo apreciar que este sigue una tendencia oscilante hasta alcanzar el valor de 1,238.04 al final del entrenamiento.

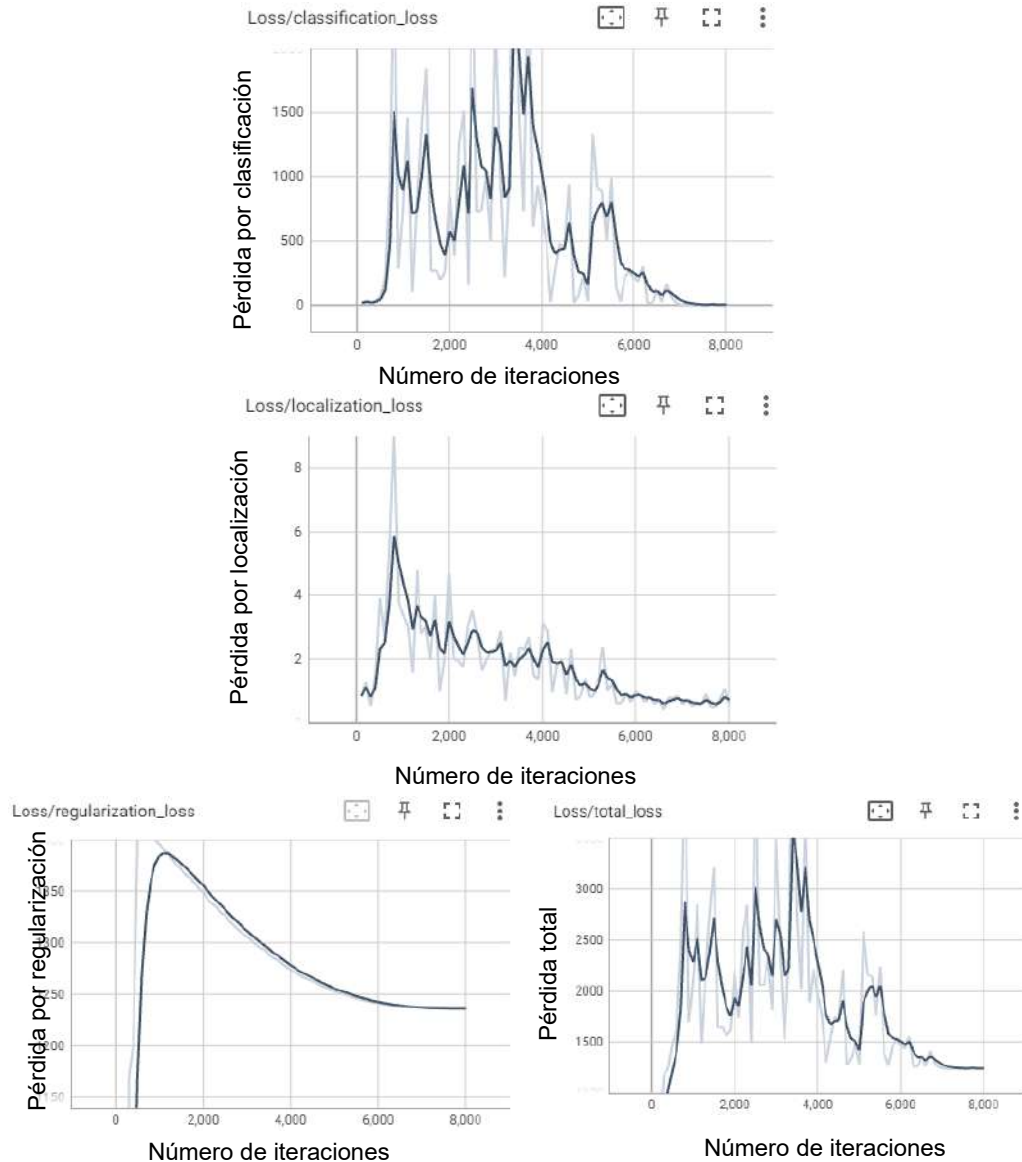


Figura 25. Curvas de los errores durante el entrenamiento del modelo “peruvian_ssd_mobilenet_v2_320x320”

Fuente. Obtenido por intermedio de la herramienta Tensorboard proporcionada por Tensorflow.

Considerando que los errores obtenidos al final del entrenamiento fueron demasiado altos, se consideró que el desempeño del modelo no fue el adecuado. Esto se puede verificar en el Cuadro 25, en el que se presentan los resultados obtenidos de la evaluación del modelo obtenido para el

indicador I5: Precisión media (AP) y de manera complementaría, el parámetro recuperación.

Cuadro 25

Resultados de la evaluación del modelo “peruvian_ssd_mobilenet_v2_320x320”

Indicador de desempeño	IoU	Máximo de detecciones por imagen	Valor
I5: Precisión media (AP)			
	IoU=0.50:0.95	100	0.000
	IoU=0.50	100	0.001
	IoU=0.75	100	0.000
Recuperación (RE)			
	IoU=0.50:0.95	1	0.005
	IoU=0.50:0.95	10	0.005
	IoU=0.50:0.95	100	0.047

Fuente. Elaborado a partir de los valores obtenidos calculados mediante la API de Tensorflow para la detección de objetos.

Como puede apreciarse en los resultados obtenidos, en ninguno de los casos se llega al 1% (0.01), lo cual indica que el modelo resultante no es el adecuado para la tarea de detección de placas vehiculares. En consecuencia, no se realizó la evaluación de los tiempos de inferencia para este modelo.

5.1.2.4. Resultados para el modelo base SSD MobileNet V1 FPN 640x640

El entrenamiento por ajuste fino realizado tomando como base el modelo SSD MobileNet V1 FPN 640x640 tuvo un tiempo de entrenamiento (T_t) que tomó 3 059.06 segundos en realizarse, con un tiempo de actualización de pesos de aproximadamente 0.375 segundos por iteración.

El modelo resultante se denominó “peruvian_ssd_mobilenet__fpn_640x640”. Los resultados del proceso de entrenamiento, incluyendo los errores finales obtenidos se presentan en el Cuadro 26.

Cuadro 26

Resultados del proceso de entrenamiento por ajuste fino del modelo “peruvian_ssd_mobilenet__fpn_640x640”

Ítem	Valor
Modelo base	SSD MobileNet V1 FPN 640x640
Fecha y hora de inicio de entrenamiento (T_{ti})	2023-04-03 00:35:34.96
Fecha y hora de fin de entrenamiento (T_{tf})	2023-04-03 01:26:34.02
I1: Tiempo de entrenamiento ($T_t = T_{tf} - T_{ti}$) (*)	3 059.06 segundos
Tiempo por iteración (actualización)	Aprox. 0.375 segundos
Error de clasificación al fin del entrenamiento	0.03677152
Error de localización al fin del entrenamiento	0.01316152
Error de regularización al fin del entrenamiento	0.8945658
Error total al fin del entrenamiento	0.9444988

Nota. (*) De acuerdo con lo definido en el Cuadro 4. Operacionalización de variables de la presente investigación.

En la Figura 26 se presentan las curvas de los errores de entrenamiento en función de las iteraciones. En cuanto a los errores de clasificación y localización, se observó que ambos parámetros siguen una tendencia decreciente, alcanzando los valores de 0.037 y 0.013 para la iteración 8,000. En cuanto al error de regularización, este alcanzó un pico en la iteración 1,000, y luego descendió hasta alcanzar el valor de 0.8945 al final del entrenamiento. Observando el error total, el cual considera los 3 errores descritos previamente, se pudo apreciar que este disminuye conforme aumentan las iteraciones, hasta alcanzar el valor de 0.9444 al final del entrenamiento.

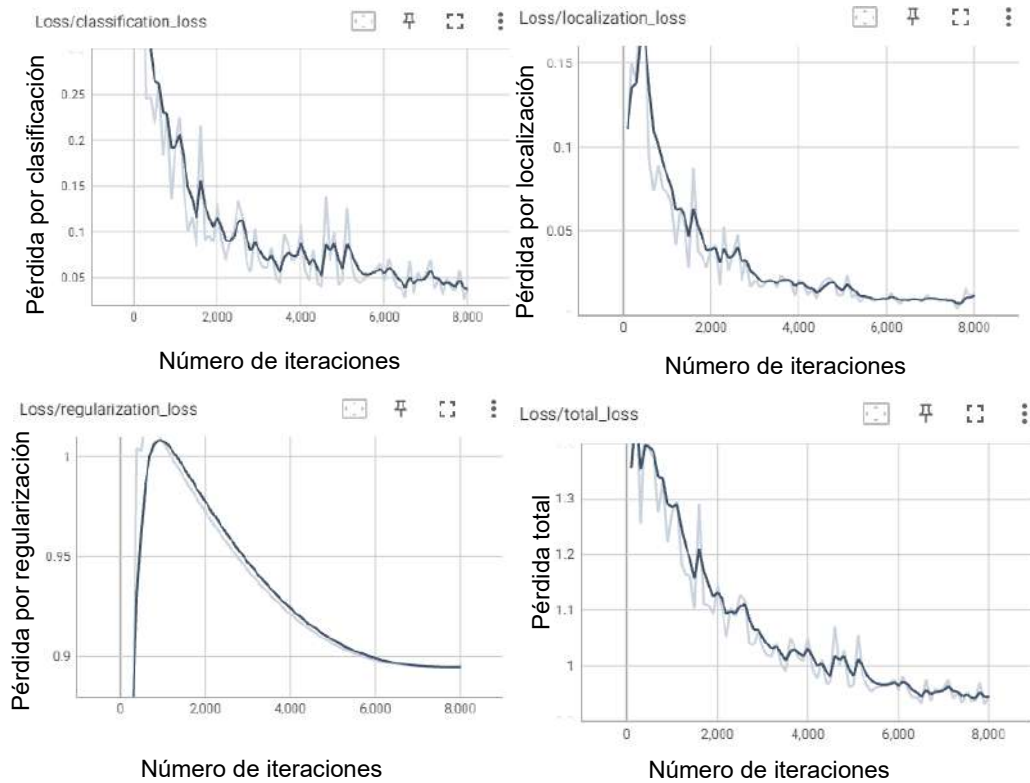


Figura 26. Curvas de los errores durante el entrenamiento del modelo “peruvian_ssd_mobilenet_fpn_640x640”.

Fuente. Obtenido por intermedio de la herramienta Tensorboard proporcionada por Tensorflow.

Con relación a la evaluación del desempeño del modelo, en el Cuadro 27 se presentan los resultados obtenidos para el indicador I5: Precisión promedio (AP) y de manera complementaria, el parámetro recuperación.

Cuadro 27

Resultados de la evaluación del modelo “peruvian_ssd_mobilenet__fpn_640x640”

Indicador de desempeño	IoU	Máximo de detecciones por imagen	Valor
I5: Precisión promedio (AP)			
	IoU=0.50:0.95	100	0.746
	IoU=0.50	100	0.961
	IoU=0.75	100	0.923
Recuperación (RE)			
	IoU=0.50:0.95	1	0.783
	IoU=0.50:0.95	10	0.796
	IoU=0.50:0.95	100	0.800

Fuente. Elaborado a partir de los valores obtenidos calculados mediante la API de Tensorflow para la detección de objetos.

Tomando en cuenta que la precisión media promedio (mAP) del modelo base era de 0.291, se pudo apreciar que en todos los casos evaluados se logró una mejora de aproximadamente el 200% con respecto a este indicador.

En esta etapa, se calcularon los tiempos de inferencia para cada una de las imágenes del dataset, cuyo promedio se empleó para caracterizar al modelo en general. Los resultados de este análisis se presentan en el Cuadro 28, en el cual los tiempos fueron calculados considerando que las imágenes del dataset tenían dimensiones variables, lo cual tendía a generar faltas de optimización y reducciones en el rendimiento.

Cuadro 28

Resultados de la evaluación de la inferencia del modelo “peruvian_ssd_mobilenet__fpn_640x640” sobre la totalidad del dataset

Ítem	Valor
Número de imágenes procesadas	392
Número de placas etiquetadas (N)	406
Número de placas predichas (P)	403
Número de placas detectadas (D)	399
Falsos positivos (P-D)	4
Ratio de detección (D/N)	0.9827
Mínimo tiempo de inferencia	1.29 segundos
Máximo tiempo de inferencia	10.84 segundos
Tiempo de inferencia promedio	2.45 segundos

Se debe indicar que en la evaluación se encontraron casos de falsos positivos. Estos 4 casos se presentan en la Figura 27. Los dos primeros casos corresponden a errores de clasificación, ya que el modelo clasificó como placas objetos que no lo eran. Esto podría mitigarse elevando el puntaje (score) a partir del cual se considera que un objeto es una placa, no obstante, esto podría afectar la detección de otros casos. Por ejemplo, si se eleva el puntaje a 0.90, la cantidad de placas detectadas correctamente disminuye en 3. Los casos 3 y 4 corresponden a un error de localización, ya que, si bien el modelo pudo detectar la placa, segmentó el objeto en dos partes, generando así dos falsos positivos. Cabe resaltar que para la imagen en la que se produjo este error, sí se detectó la placa correctamente en su totalidad (es decir, el modelo generó 3 predicción, siendo la primera, con puntaje 0.9782403, la que contiene a la placa completa).

Caso 1
Score: 0.8768213



Caso 2
Score: 0.7925038



Casos 3 y 4
Score: 0.88644767



Score: 0.7961956



Figura 27. Falsos positivos encontrados durante la evaluación de los tiempos de inferencia del modelo “peruvian_ssd_mobilenet_fpn_640x640”.

Fuente. Elaborado a partir de la evaluación del modelo bajo análisis.

En la Figura 28 se presenta la distribución de los tiempos de inferencia para cada una de las imágenes, observándose que la gran mayoría se encuentra por debajo de los 2.5 segundos.

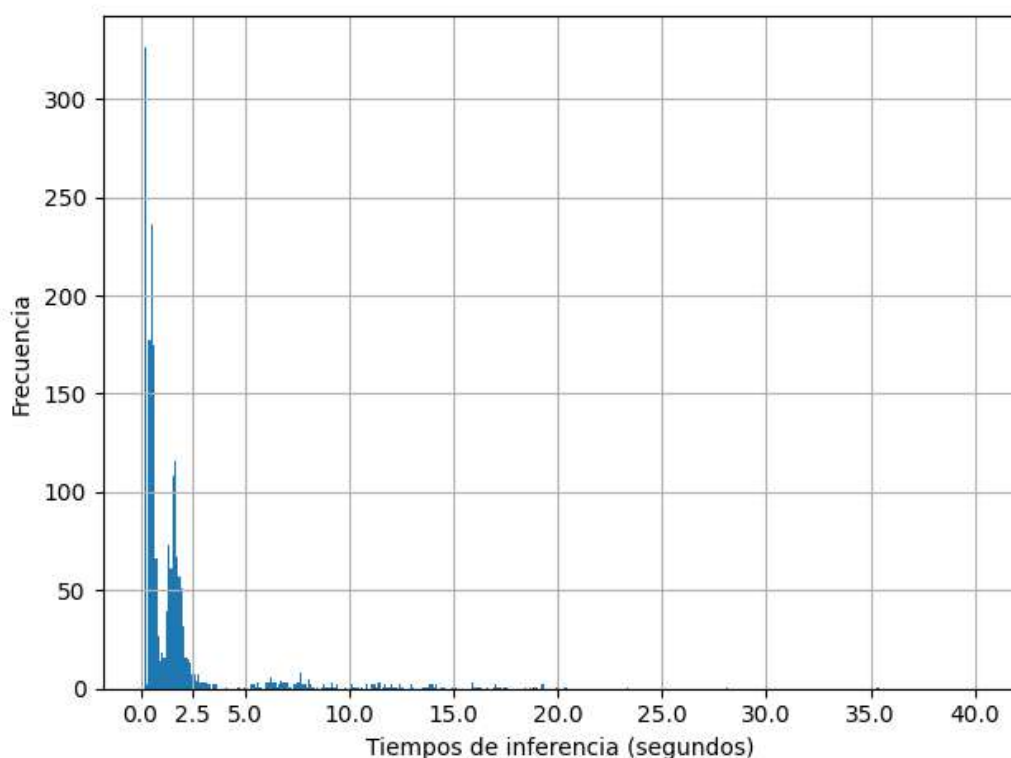


Figura 28. Distribución de los tiempos de inferencia calculados para el modelo “peruvian_ssd_mobilenet_fpn_640x640”.

Fuente. Elaborado a partir de los tiempos calculados empleando la función `time` de la librería `time` de Python. Se considera una placa detectada si la probabilidad de detección (score) es de mínimo 70%.

5.1.2.5. Resultados para el modelo base SSD MobileNet V1 FPNLite 320x320

El entrenamiento por ajuste fino realizado tomando como base el modelo SSD MobileNet V1 FPNLite 320x320 tuvo un tiempo de entrenamiento (T_t) que tomó 750.42 segundos en realizarse, con un tiempo de actualización de pesos de aproximadamente 0.09 segundos por iteración.

El modelo resultante se denominó “peruvian_mobilenet_fpnlite_320x320”. Los resultados del proceso de entrenamiento, incluyendo los errores finales obtenidos se presentan en el Cuadro 29.

Cuadro 29

Resultados del proceso de entrenamiento por ajuste fino del modelo “peruvian_mobilenet_fpnlite_320x320”

Ítem	Valor
Modelo base	SSD MobileNet V1 FPNLite 320x320
Fecha y hora de inicio de entrenamiento (T_{ti})	2023-04-02 01:25:11.57
Fecha y hora de fin de entrenamiento (T_{tf})	2023-04-02 01:37:42.00
I1: Tiempo de entrenamiento ($T_t = T_{tf} - T_{ti}$) (*)	750.42 segundos
Tiempo por iteración (actualización)	Aprox. 0.09 segundos
Error de clasificación al final del entrenamiento	0.023703322
Error de localización al final del entrenamiento	0.004606329
Error de regularización al final del entrenamiento	0.13654956
Error total al final del entrenamiento	0.1648592

Nota. (*) De acuerdo con lo definido en el Cuadro 4. Operacionalización de variables de la presente investigación.

En la Figura 29 se muestran las curvas de los errores de entrenamiento en función de las iteraciones. En cuanto a los errores de clasificación y localización, pudo observarse que ambos parámetros siguen una tendencia decreciente, alcanzando los valores de 0.024 y 0.005 para la iteración 8,000. En cuanto al error de regularización, este alcanzó un pico en la iteración 1,000, y luego descendió hasta tomar el valor de 0.1365 al final del entrenamiento. Observando el error total, el cual considera los 3 errores descritos previamente, se pudo apreciar que este disminuye conforme aumentan las iteraciones, hasta alcanzar el valor de 0.1648 al final del entrenamiento.

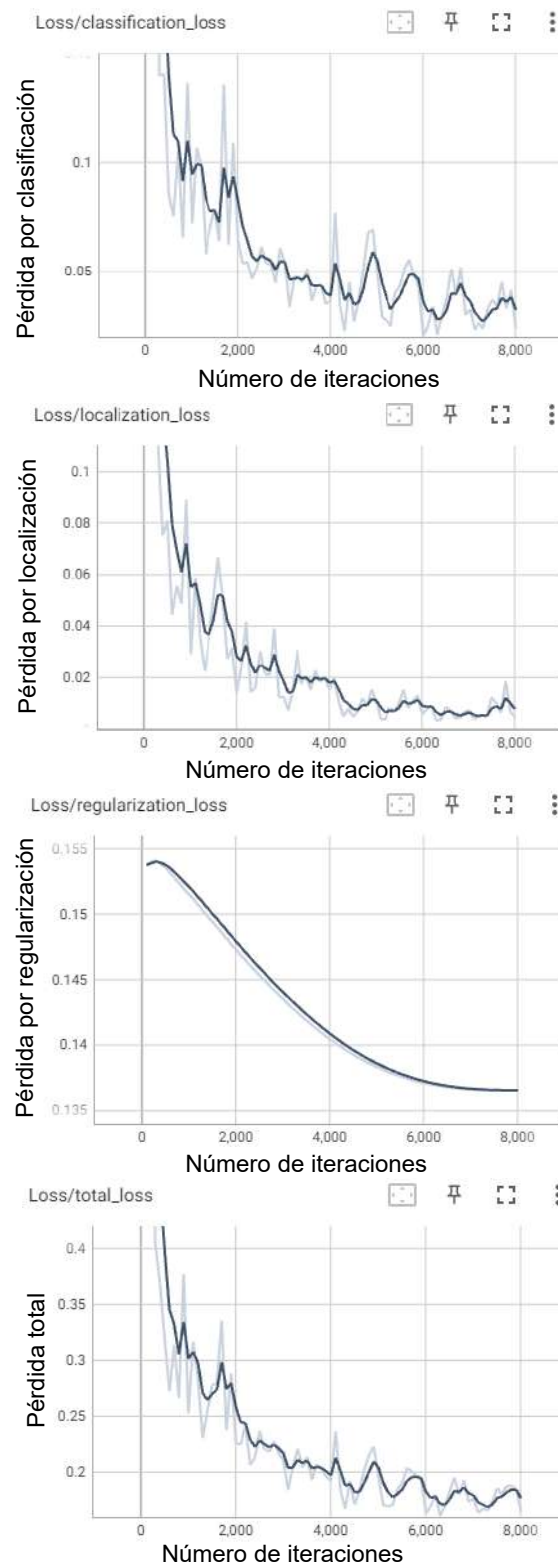


Figura 29. Curvas de los errores durante el entrenamiento del modelo “peruvian_mobilenet_fpnlite_320x320”.

Fuente. Obtenido por intermedio de la herramienta Tensorboard proporcionada por Tensorflow.

Con relación a la evaluación del desempeño del modelo, en el Cuadro 30 se presentan los resultados obtenidos para el indicador I5: Precisión promedio (AP) y de manera complementaria, el indicador recuperación (RE).

Cuadro 30

Resultados de la evaluación del modelo “peruvian_mobilenet_fpnlite_320x320”

Indicador de desempeño	IoU	Máximo de detecciones por imagen	Valor
I5: Precisión promedio (AP)			
	IoU=0.50:0.95	100	0.764
	IoU=0.50	100	0.951
	IoU=0.75	100	0.904
Recuperación (RE)			
	IoU=0.50:0.95	1	0.795
	IoU=0.50:0.95	10	0.799
	IoU=0.50:0.95	100	0.802

Fuente. Elaborado a partir de los valores obtenidos calculados mediante la API de Tensorflow para la detección de objetos.

Tomando en cuenta que la precisión media promedio (mAP) del modelo base era de 0.222, se pudo apreciar que en todos los casos evaluados se logró una mejora de aproximadamente el 300% con respecto a este indicador.

En esta etapa, se calcularon los tiempos de inferencia para cada una de las imágenes que conforman el dataset, cuyo promedio se empleó para caracterizar al modelo en general. Los resultados de este análisis se presentan en el Cuadro 31, en el cual los tiempos fueron calculados considerando que todas las imágenes del dataset no tienen las mismas dimensiones, lo cual tiende a generar faltas de optimización y reducciones en el rendimiento.

Cuadro 31

Resultados de la evaluación de la inferencia del modelo “peruvian_mobilenet_fpn-lite_320x320” sobre la totalidad del dataset

Ítem	Valor
Número de imágenes procesadas	392
Número de placas etiquetadas (N)	406
Número de placas predichas (P)	362
Número de placas detectadas (D)	355
Falsos positivos (P-D)	7
Ratio de detección (D/N)	0.8744
Mínimo tiempo de inferencia	0.1833 segundos
Máximo tiempo de inferencia	13.6185 segundos
Tiempo de inferencia promedio	0.6948 segundos

Se debe indicar que durante la evaluación se encontraron casos de falsos positivos. Estos 7 casos se presentan en la Figura 30. Los cuatro primeros casos corresponden a errores de clasificación, puesto que el modelo clasificó como placas objetos que no lo eran. Esto podría mitigarse elevando el puntaje [score] a partir del cual se considera que un objeto es una placa, no obstante, esto podría afectar la detección de otros casos. Por ejemplo, si se eleva el puntaje a 0.90, la cantidad de placas detectadas correctamente disminuye en 22. Los casos 4, 5 y 6 corresponden a errores de localización, ya que, si bien el modelo pudo detectar la placa, segmentó el objeto en partes, generando así dos falsos positivos. Es relevante resaltar que para las imágenes en las que se produjo este error (imágenes 00370 y 00372), el modelo si predijo, además de los falsos positivos, la placa completa.



Figura 30. Falsos positivos encontrados durante la evaluación de los tiempos de inferencia del modelo “peruvian_mobilenet_fpnlite_320x320”.

Fuente. Elaborado a partir de la evaluación del modelo bajo análisis.

En la Figura 31 se muestra la distribución de los tiempos de inferencia para cada una de las imágenes, observándose que la gran mayoría se encuentra por debajo de los 2.5 segundos.

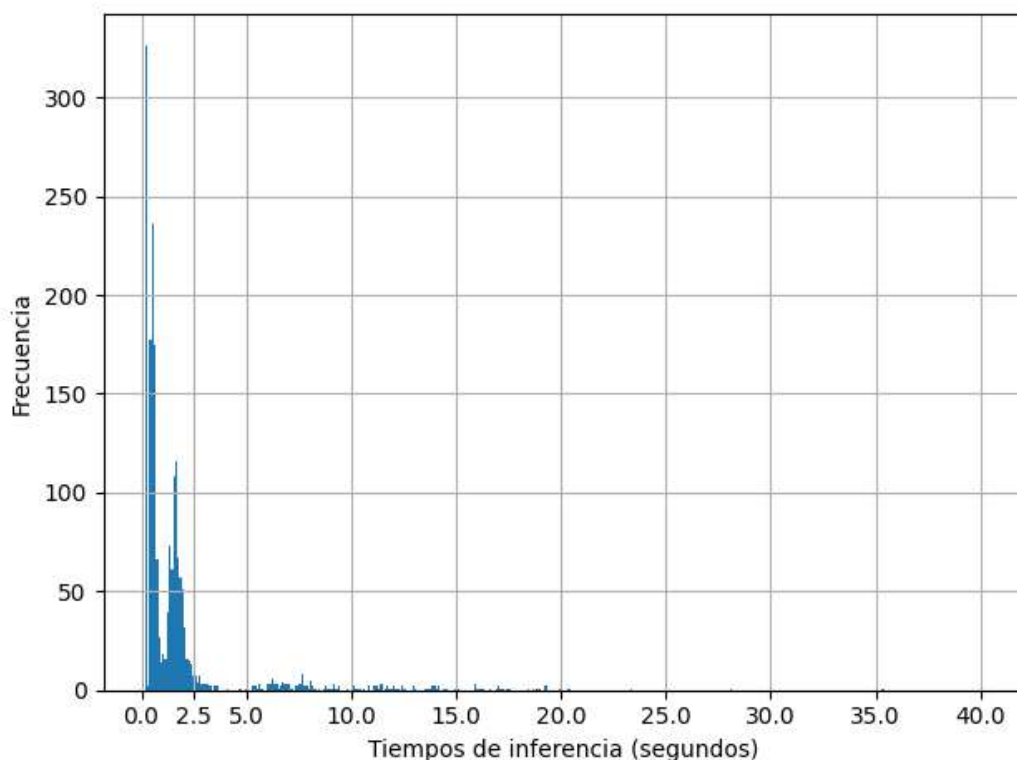


Figura 31. Distribución de los tiempos de inferencia calculados para el modelo “peruvian_mobilenet_fpnlite_320x320”

Fuente. Elaborador a partir de los tiempos calculados empleando la función `time` de la librería `time` de Python. Se considera una placa detectada si la probabilidad de detección (score) es de mínimo 70%.

5.1.2.6. Resultados para el modelo base SSD MobileNet V1 FPNLite 640x640

El entrenamiento por ajuste fino realizado tomando como base el modelo SSD MobileNet V1 FPNLite 640x640 tuvo un tiempo de entrenamiento (T_t) que tomó 1 592.29 segundos en realizarse, con un tiempo de actualización de pesos de aproximadamente 0.19 segundos por iteración.

El modelo resultante se denominó “peruvian_mobilenet_fpnlite_640x640”. Los resultados del proceso de entrenamiento, incluyendo los errores finales obtenidos se presentan en el Cuadro 32.

Cuadro 32

Resultados del proceso de entrenamiento por ajuste fino del *modelo* “peruvian_mobilenet_fpnlite_640x640”

Ítem	Valor
Modelo base	SSD MobileNet V1 FPNLite 640x640
Fecha y hora de inicio de entrenamiento (T_{ti})	2023-04-02 00:38:31.49
Fecha y hora de fin de entrenamiento (T_{tf})	2023-04-02 01:05:03.79
I1: Tiempo de entrenamiento ($T_t = T_{tf} - T_{ti}$) (*)	1 592.29 segundos
Tiempo por iteración (actualización)	Aprox. 0.19 segundos
Error de clasificación al final del entrenamiento	0.027417589
Error de localización al final del entrenamiento	0.0046948316
Error de regularización al final del entrenamiento	0.13537835
Error total al final del entrenamiento	0.16749077

Nota. (*) De acuerdo con lo definido en la Tabla 4. Operacionalización de variables de la presente investigación.

En la Figura 32 se muestran las curvas de los errores de entrenamiento en función de las iteraciones. En cuanto a los errores de clasificación y localización, pudo observarse que ambos parámetros siguen una tendencia decreciente, alcanzando los valores de 0.027 y 0.005 para la iteración 8 000. En cuanto al error de regularización, este alcanzó un pico en la iteración 1 000, y luego descendió hasta tomar el valor de 0.1354 al final del entrenamiento. Observando el error total, el cual considera los 3 errores descritos previamente, se pudo apreciar que este disminuye conforme aumentan las iteraciones, hasta alcanzar el valor de 0.1675 al final del entrenamiento.

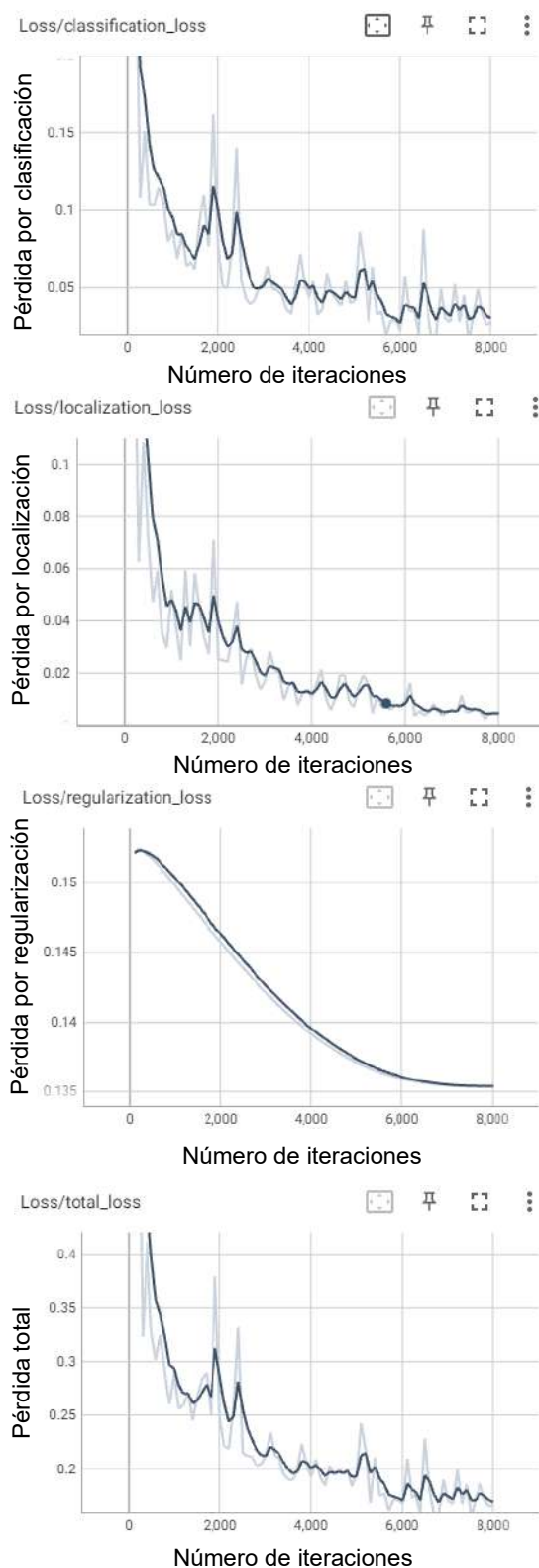


Figura 32. Curvas de los errores durante el entrenamiento del modelo “peruvian_mobilenet_fpnlite_640x640”.

Fuente. Obtenido por intermedio de la herramienta Tensorboard proporcionada por Tensorflow.

Con relación a la evaluación del desempeño del modelo, en el Cuadro 33 se presentan los resultados obtenidos para el indicador I5: Precisión promedio (AP) y, de manera complementaria, el parámetro recuperación.

Cuadro 33

Resultados de la evaluación del modelo “peruvian_mobilenet_fpnlite_640x640”.

Indicador de desempeño	IoU	Máximo de detecciones por imagen	Valor
I5: Precisión promedio (AP)			
	IoU=0.50:0.95	100	0.823
	IoU=0.50	100	0.969
	IoU=0.75	100	0.946
Recuperación (RE)			
	IoU=0.50:0.95	1	0.832
	IoU=0.50:0.95	10	0.853
	IoU=0.50:0.95	100	0.853

Fuente. Elaborado a partir de los valores obtenidos calculados mediante la API de Tensorflow para la detección de objetos.

Tomando en cuenta que la precisión media promedio (mAP) del modelo base era de 0.282, se pudo apreciar que en todos los casos evaluados se logró una mejora de aproximadamente el 400% con respecto a este indicador.

En esta etapa, se calcularon los tiempos de inferencia para cada una de las imágenes que conforman el dataset, cuyo promedio se empleó para caracterizar al modelo en general. Los resultados de este análisis se presentan en el Cuadro 34, en el cual los tiempos fueron calculados considerando que todas las imágenes del dataset no tienen las mismas dimensiones, lo cual tiende a generar faltas de optimización y reducciones en el rendimiento.

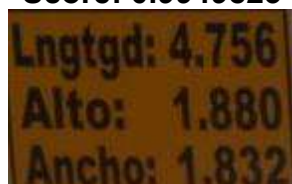
Cuadro 34

Resultados de la evaluación de la inferencia del modelo “peruvian_mobilenet_fpnlite_640x640” sobre la totalidad del dataset

Ítem	Valor
Número de imágenes procesadas	392
Número de placas etiquetadas (N)	406
Número de placas predichas (P)	406
Número de placas detectadas (D)	402
Falsos positivos (P-D)	4
Ratio de detección (D/N)	0.9901
Mínimo tiempo de inferencia	0.39 segundos
Máximo tiempo de inferencia	14.02 segundos
Tiempo de inferencia promedio	1.12 segundos

Se debe indicar que durante la evaluación se encontraron casos de falsos positivos. Estos 4 casos se muestran en la Figura 33. Los tres primeros casos corresponden a errores de clasificación, es decir, el modelo clasificó como placas objetos que no lo eran. Al respecto se precisa que, considerando el primer caso, el falso positivo tiene un puntaje de 0.9945, el cual es superior inclusive al puntaje calculado para placas detectadas correctamente. El último caso correspondió a un error de localización, ya que, si bien el modelo pudo detectar la placa, segmentó el objeto en partes, generando así un falso positivo. Cabe resaltar que para este caso el modelo si predijo, además del falso positivo, la placa completa.

Caso 1
Score: 0.9949823



Caso 2
Score: 0.7777428



Caso 3
Score: 0.8405739



Caso 4
Score: 0.70275116



Figura 33. Falsos positivos encontrados durante la evaluación de los tiempos de inferencia del modelo “peruvian_mobilenet_fpnlite_640x640”

Fuente. Elaborado a partir de la evaluación del modelo bajo análisis.

En la Figura 34 se presenta la distribución de los tiempos de inferencia para cada una de las imágenes, observándose que la gran mayoría se encuentra por debajo de los 2.5 segundos.

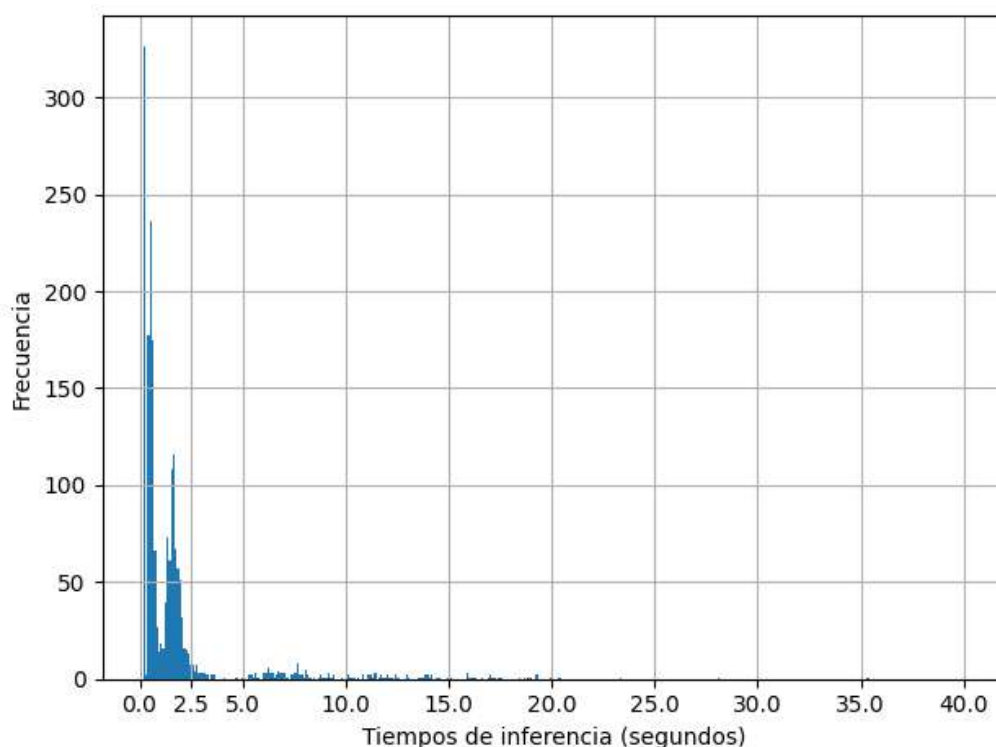


Figura 34. Distribución de los tiempos de inferencia calculados para el modelo “peruvian_mobilenet_fpn_lite_640x640”

Fuente. Elaborado a partir de los tiempos calculados empleando la función `time` de la librería `time` de Python. Se considera una placa detectada si la probabilidad de detección (score) es de mínimo 70%.

5.1.2.7. Análisis comparativo y selección del modelo para prototipo

En los numerales previos se presentaron los resultados obtenidos durante el entrenamiento y la evaluación de 6 modelos, con la observación que uno de ellos, el denominado “peruvian_ssd_mobilenet_v2_320x320” no presentó un desempeño adecuado. Para fines de simplificación del análisis comparativo, en Cuadro 35 se presentan los modelos con un alias simplificado, de acuerdo con el orden en el que fueron analizados previamente en el presente apartado.

Cuadro 35

Modelos entrenados y sus etiquetas para fines de comparación

Modelo entrenado	Etiqueta
peruvian_efficientdet_d0_512x512	M1
peruvian_efficientdet_d1_640x640	M2
peruvian_ssd_mobilenet__fpn_640x640	M3
peruvian_mobilenet_fpnlite_320x320	M4
peruvian_mobilenet_fpnlite_640x640	M5

Se inició el análisis comparativo por el proceso de entrenamiento, tomando como referencia de análisis los tiempos de entrenamiento (indicador I2) y los errores obtenidos en la última iteración (iteración 8,000). En el Cuadro 36 se presentan estos indicadores para todos los modelos entrenados.

En lo que respecta al tiempo de entrenamiento, el modelo que se entrenó con mayor rapidez fue el M4, seguido por el modelo M5.

En lo referente a los errores durante el entrenamiento, el modelo M4 presentó un menor error de clasificación, mientras que el modelo M2 presentó un menor error de localización y el modelo M1 presentó un menor error de regularización. Considerando los efectos de todos los errores de entrenamiento evaluados, el modelo M2 es el que presentó un menor error total, seguido por el modelo M1. Los modelos M3 y M4 presentaron errores totales similares, mientras que el modelo M3 presentó el error total más alto.

Cuadro 36

Comparación de los modelos entrenados con relación el proceso de entrenamiento.

Indicador	M1	M2	M3	M4	M5
I1: Tiempo de entrenamiento	2,181.80 s	4,189.04 s	3,059.06 s	750.42 s	1 592.29 s
Error de clasificación al final del entrenamiento	0.08924	0.07687	0.03677	0.02370	0.02742
Error de localización al final del entrenamiento	0.00124	0.00120	0.01316	0.00460	0.00469
Error de regularización al final del entrenamiento	0.03272	0.03455	0.89457	0.13655	0.13538
Error total al final del entrenamiento	0.12320	0.11262	0.94450	0.16486	0.16749

Fuente. Elaborado a partir de los datos obtenidos de la evaluación de cada modelo.

Nota. Se resalta por cada fila el modelo con el mejor indicador.

Como complemento del entrenamiento, se analizaron los indicadores de la evaluación de los modelos, tomando como referencia la precisión promedio AP (Indicador I5) y de manera complementaria la recuperación RE. En el Cuadro 37 se presentan estos indicadores para todos los modelos entrenados.

En lo que respecta al indicador de precisión promedio (Indicador I5), se pudo encontrar que el modelo M5 presente mejores resultados, sobre todo para un IoU en el intervalo de 0.50 a 0.95 (se promedia el indicador para distintos IoU en dicho intervalo). Solo para un IoU de 0.75 el modelo M1 presentó un mejor resultado, no obstante, es seguido por el modelo M5.

Por otro lado, en lo que se refiere al parámetro de recuperación, se pudo encontrar que el modelo M5 nuevamente es el que presente el mejor resultado.

Cuadro 37

Comparación de los modelos entrenados con relación el proceso de evaluación de desempeño.

Indicador	Consideración	M1	M2	M3	M4	M5
I5: Precisión promedio (AP)						
	IoU=0.50:0.95	0.752	0.766	0.746	0.764	0.823
	IoU=0.50	0.968	0.969	0.961	0.951	0.969
	IoU=0.75	0.949	0.929	0.923	0.904	0.946
I6: Recuperación (RE)						
	IoU=0.50:0.95	0.786	0.796	0.800	0.802	0.853

Fuente. Elaborado a partir de los datos obtenidos de la evaluación de cada modelo.

Nota. Se consideran los indicadores calculados para un máximo de 100 detecciones por imagen. Se resalta por cada fila el modelo con el mejor indicador.

Se prosiguió con el análisis de la capacidad de inferencia de los modelos y sus tiempos de inferencia, evaluados sobre el total de las 392 imágenes que conforman el dataset. En el Cuadro 38 se presentan estos indicadores para todos los modelos entrenados.

Pudo encontrarse que el modelo que detectó correctamente el mayor número de placas fue el modelo M5, seguido por el modelo M3. No obstante, el modelo que predijo un menor número de falsos positivos fue el modelo M1, seguido por el modelo M2, seguidos por los modelos M3 y M5, ambos con 4 falsos positivos. Considerando el número de placas detectadas correctamente respecto al total de placas etiquetadas en el dataset, el modelo M5 es el que presentó el mejor resultado, seguido por el modelo M2.

En lo que respecta al tiempo de inferencia, el modelo M4 es el que presentó un menor tiempo mínimo y promedio de inferencia, seguido por el modelo M5. En lo referente al tiempo máximo de inferencia, el M3 es el que presentó un mejor valor, seguido por los modelos M4 y M5.

Cuadro 38

Comparación de los modelos entrenados con relación a la inferencia.

Indicador	M1	M2	M3	M4	M5
Número de placas detectadas	388	397	399	355	402
Falsos positivos	1	2	4	7	4
Ratio de detección	0.9557	0.9778	0.9827	0.8744	0.9901
Mínimo tiempo de inferencia	0.52 s	1.04 s	1.29 s	0.18 s	0.39 s
Máximo tiempo de inferencia	35.31 s	40.08 s	10.84 s	13.62 s	14.02 s
Tiempo de inferencia promedio	2.07 s	2.60 s	2.45 s	0.69 s	1.12 s

Fuente. Elaborado a partir de los datos obtenidos de la evaluación de cada modelo.

Nota: Se resalta en cada fila el modelo que presente el mejor resultado.

Con la finalidad de evaluar el modelo a ser considerado para la implementación del prototipo, se procedió a elaborar una matriz de ranking por indicadores, considerándose un puntaje del 1 al 5 por cada atributo en el que modelo tenga mejores valores para sus indicadores. Este cálculo de presenta en el Cuadro 39.

Cuadro 39

Evaluación integral de los modelos entrenados.

Indicador	M1	M2	M3	M4	M5
I1: Tiempo de entrenamiento	3	1	2	5	4
Error de clasificación al final del entrenamiento	1	2	3	5	4
Error de localización al final del entrenamiento	4	5	1	3	2
Error de regularización al final del entrenamiento	5	4	1	2	3
I5: AP (IoU=0.50:0.95)	2	4	1	3	5
RE (IoU=0.50:0.95)	1	2	3	4	5
Falsos positivos	5	4	3	1	3
Ratio de detección	4	3	2	1	5
Mínimo tiempo de inferencia	3	2	1	5	4
Máximo tiempo de inferencia	2	1	5	4	3
Tiempo de inferencia promedio	3	1	2	5	4
Puntaje total	33	29	24	38	42

De los resultados calculados en el Cuadro 39 se pudo encontrar que el modelo M5, denominado “peruvian_mobilenet_fpnlite_640x640”, el cual está basado en el modelo pre entrenado MobileNet FPNLite 640x640, es el que presentó un mejor desempeño integral para la detección de placas de rodajes de Perú

y, por lo tanto, es el modelo que se consideró para la implementación del prototipo.

5.1.3. Resultados del reconocimiento de caracteres

Luego de que se detecta la placa y se haga la inferencia de su ubicación en la imagen, el siguiente proceso del sistema fue aplicar el reconocimiento de los caracteres con la finalidad de determinar el número de placa.

Para esta tarea, se empleó la librería PaddleOCR (Du et al., 2020), la cual está implementada con lenguaje Python. Esta librería dispone de la función `ocr`, la cual admite como entrada una imagen y devuelve como salidas las predicciones de los caracteres reconocidos, bajo un idioma de referencia. Para la presente aplicación, se eligió el idioma inglés, considerando que la librería no dispone del idioma español entre sus parámetros de configuración. No obstante, en la medida de que los números de placas están conformados por caracteres alfanuméricos, el proceso de reconocimiento no se vio afectado por el idioma configurado en la librería.

Se implementó un algoritmo en Python para esta tarea, la cual consistió en los siguientes pasos:

Paso 1: Cargar la matriz de la placa de rodaje detectada por el modelo basado en inteligencia artificial.

Paso 2: Aplicar el algoritmo ‘ocr’ sobre la placa detectada.

Paso 3: Si se obtiene un resultado con *score* o puntaje mayor que 0.90, y el formato del texto reconocido cumple con las reglas de formación del número de placa de rodaje peruana (según lo indicado en el Cuadro 2), se retorna la etiqueta con el *score* o puntaje que el OCR calculó para dicha predicción.

Paso 4: Si no se obtiene un resultado válido de acuerdo con lo señalado en el Paso 3, se aplica una máscara en los bordes, lo cual mejora la capacidad de reconocimiento.

Paso 5: Aplicar el algoritmo ocr sobre la placa procesada.

Paso 6: Si se obtiene un resultado válido de acuerdo con las reglas de formación del número de placa de rodaje peruana (según lo indicado en la

Tabla 2), se retorna la etiqueta con el *score* o puntaje que el OCR calculó para dicha predicción.

Paso 7: Retornar como salida la variable de texto: NO RECONOCIDO, con el *score* o puntaje de 0 (cero).

Para validar el algoritmo, este se aplicó sobre el total del dataset (392 imágenes con 406 placas de rodaje), tomando en cuenta que se buscó evaluar el desempeño del sistema final en su conjunto, más allá del modelo individualmente. De este modo, tras la evaluación del algoritmo aplicado sobre las placas detectadas por el modelo seleccionado en el apartado 5.1.2.7, se pudieron reconocer los caracteres de 397 placas de rodaje. De las 9 placas de rodaje en las que no se pudieron reconocer los caracteres, 4 correspondieron a falsos positivos detectados por el modelo, indicados en la Figura 33. En ese sentido, en 9 placas de rodaje, el OCR no pudo reconocer los números de placa. Cabe señalar que un caso correspondió a una imagen que no corresponde a una placa en absoluto. Estos 10 casos se presentan en la Figura 35, en la que se señala además una posible explicación de por qué el número de placa no pudo ser reconocido.



Figura 35. Imágenes generadas por el modelo detector en las que el OCR no pudo reconocer el número de placa.

Fuente. Elaborado a partir del resultado de la evaluación del sistema.

De los casos mostrados en la Figura 35, pudo encontrarse que el reconocimiento de caracteres no es eficaz si la placa detectada es muy pequeña.

Finalmente, se procedió a realizar la comparación de los números de placa reconocidos, con relación a los números de placas etiquetados. Tras esta evaluación, se pudo verificar que el OCR detectó correctamente los números de placa de 382 licencias, mientras que en 14 placas el número de placa reconocido no correspondió con el valor etiquetado. Considerando que el universo de placas etiquetadas fue de 406, el Indicador I6: Exactitud de reconocimiento de caracteres (definido en el Cuadro 4), tomó el valor de 94.09%. Este valor es satisfactorio para los fines de la presente investigación.

Con relación a los 14 números de placas que se reconocieron incorrectamente, en la Figura 36 se presentan estos casos. De los 14 casos en los que se reconoció incorrectamente el número de placa, solo en un caso el OCR el número de caracteres errados fue de 6. En el resto, solo hubo error

en el reconocimiento de un carácter. Para cada caso se presente el *score* o puntaje de la predicción del OCR, así como la descripción del error.

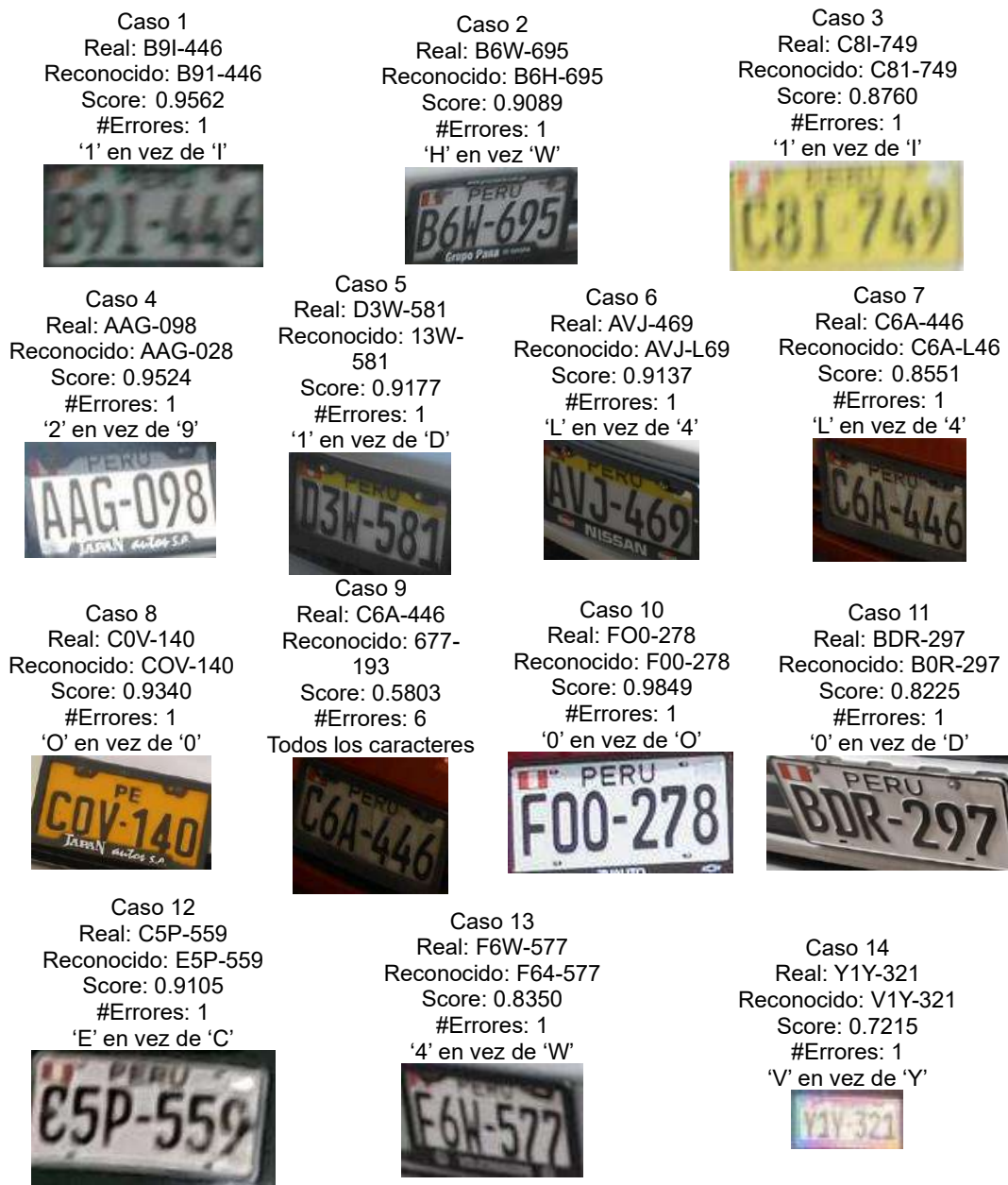


Figura 36. Imágenes en los que el número de placa reconocido tiene errores.

Fuente. Elaborado a partir del resultado de la evaluación del sistema.

5.1.4. Implementación del prototipo

Se implementó el prototipo empleando una SBC Raspberry Pi 4 B+ con 8GB de memoria RAM. Se exportó el modelo seleccionado para poder realizar la

inferencia en el prototipo. Se realizó la implementación empleando la versión total y la versión simplificada del modelo seleccionado. Si bien la versión simplificada del modelo podría tener un menor desempeño en comparación con el modelo íntegro (Alake, 2020)(Juras, 2023), la velocidad en la inferencia y el menor consumo de recursos computacionales es un aspecto relevante, sobre todo para aplicaciones en tiempo real y con sistemas electrónicos como la SBC Raspberry Pi 4 B+.

Se incorporó una cámara infrarroja como interfaz mediante la cual se capturaron imágenes de placas desde el entorno real.

Para la evaluación del prototipo, se elaboraron dos scripts en lenguaje Python. El primero, para obtener indicadores de desempeño del sistema, y el segundo, que integra a la cámara, para obtener inferencias del entorno inmediato del prototipo.

Cabe señalar que los indicadores I2 e I3: Implementación de las integraciones de software y hardware respectivamente, el cual señala el número de integraciones de los sistemas de software y hardware, toma el valor de 1, puesto solo se implementó un prototipo. Los detalles se presentan en el Cuadro 40. Asimismo, en la Figura 37 se presentan imágenes del prototipo implementado⁸.

⁸ La versión de SBC Raspberry Pi 4 B+ es la que incorpora 8 GB de memoria RAM. Se empleó a su vez una memoria SD de 64 GB como almacenamiento.

Cuadro 40

Detalles del prototipo implementado (Indicadores I2 e I3).

Elemento	Valor
Detalles de hardware	
Tarjeta base	Raspberry Pi 4 B+
Memoria RAM	8 GB
Almacenamiento	64 GB
Cámara	Infrarroja genérica
Detalles de software	
Sistema operativo	Raspberry OS “Bullseye” 64 bits
Lenguaje	Python versión 3.9
Backend para procesamiento de imágenes	OpenCV versión 4.6
Backend para detección	Tensorflow versión 2.10
Backend para OCR	PaddlePaddle versión 2.4.2

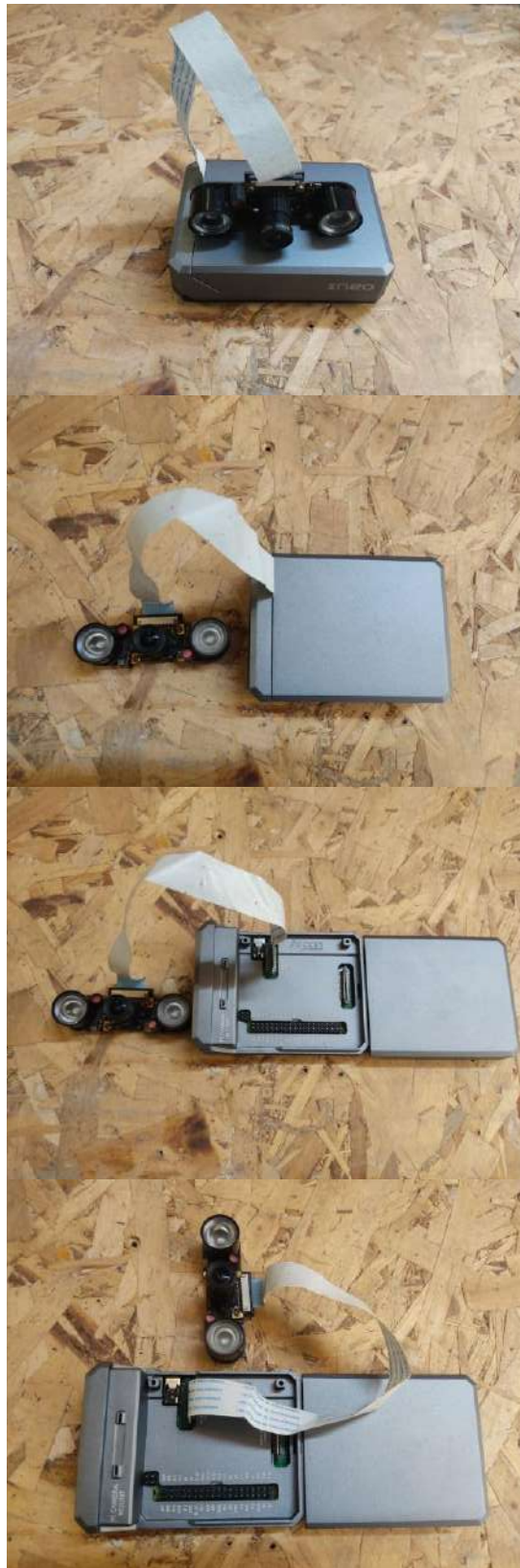


Figura 37. Prototipo implementado.

Fuente. Fotografías tomadas al prototipo implementado.

5.1.5. Resultados del test del sistema prototipo

En primer lugar, se realizó la evaluación del modelo íntegro implementado a nivel de prototipo con la tarjeta Raspberry Pi 4 B+. Esta evaluación se centró en analizar los tiempos de inicialización (carga de librerías y del modelo), preprocesamiento previo a su ingreso al modelo, detección y reconocimiento, a fin de determinar si los valores obtenidos fueron adecuados. Se precisa señalar que la suma de la duración de estos procesos es el tiempo de ejecución, definido como indicador I4 de acuerdo con el Cuadro 4. En la Figura 38 se presenta la captura de pantalla del sistema ejecutando la prueba. Para el monitoreo de la SBC Raspberry Pi 4 B+ se empleó la aplicación VNC Viewer.

```

2023/05/18 23:30:10 ppcr DEBUG: rec_res num : 1, elapsed : 0.3253939151763916
2023/05/18 23:30:11 ppcr DEBUG: dt_boxes num : 1, elapsed : 0.03625786541687012
2023/05/18 23:30:11 ppcr DEBUG: cls num : 1, elapsed : 0.02893495599622029
2023/05/18 23:30:11 ppcr DEBUG: rec_res num : 1, elapsed : 0.3141486644744873
2023/05/18 23:30:12 ppcr DEBUG: dt_boxes num : 1, elapsed : 0.036298513412475686
2023/05/18 23:30:12 ppcr DEBUG: cls num : 1, elapsed : 0.03195834159051074
2023/05/18 23:30:13 ppcr DEBUG: rec_res num : 1, elapsed : 0.3286668258668992
2023/05/18 23:30:14 ppcr DEBUG: dt_boxes num : 1, elapsed : 0.024951457977284922
2023/05/18 23:30:14 ppcr DEBUG: cls num : 1, elapsed : 0.029861641845703125
2023/05/18 23:30:14 ppcr DEBUG: rec_res num : 1, elapsed : 0.31864371480478516
2023/05/18 23:30:15 ppcr DEBUG: dt_boxes num : 0, elapsed : 0.020397186279296875
2023/05/18 23:30:15 ppcr DEBUG: cls num : 0, elapsed : 0
2023/05/18 23:30:15 ppcr DEBUG: rec_res num : 0, elapsed : 2.85102294921875e-06
2023/05/18 23:30:15 ppcr DEBUG: dt_boxes num : 1, elapsed : 0.03978109359741211
2023/05/18 23:30:15 ppcr DEBUG: cls num : 1, elapsed : 0.0321649809246582
2023/05/18 23:30:15 ppcr DEBUG: rec_res num : 1, elapsed : 0.316122161663513184
2023/05/18 23:30:16 ppcr DEBUG: dt_boxes num : 0, elapsed : 0.03396685412902832
2023/05/18 23:30:16 ppcr DEBUG: cls num : 0, elapsed : 0
2023/05/18 23:30:16 ppcr DEBUG: rec_res num : 0, elapsed : 3.5762788805234375e-06
2023/05/18 23:30:16 ppcr DEBUG: dt_boxes num : 1, elapsed : 0.04829835891723633
2023/05/18 23:30:16 ppcr DEBUG: cls num : 1, elapsed : 0.03481340498325195
2023/05/18 23:30:17 ppcr DEBUG: rec_res num : 1, elapsed : 0.3139336109101377
Test finished. Setting time was: 33.68448519706726 seconds.
Prediction time: 1.55 seconds.
  
```

Figura 38. Captura de pantalla indicando el resultado de la aplicación de prueba con el modelo completo.

Fuente. Captura de pantalla del Sistema en Raspberry Pi 4 B+.

Esta evaluación proporcionó los resultados que se muestran en el Cuadro 41, con tiempos expresados en segundos. Se resalta que la media de tiempo total para la realización de un reconocimiento completo de una placa de rodaje fue de 1.55 segundos.

Cuadro 41

Resultados de los tiempos empleados por el modelo en la SBC Raspberry Pi 4 B+.

Indicador	Promedio	Mínimo	Máximo
Tiempo de inicialización	33.68	-	-
Tiempo de preprocesamiento	0.03	0.02	0.09
Tiempo de detección	0.97	0.91	5.92
Tiempo de reconocimiento	0.55	0.00	4.63
Tiempo total (Indicador I4: tiempo de ejecución)	1.55	0.99	6.37

Fuente. Elaborado a partir de los resultados de la evaluación.

Con relación al indicador I6: “Exactitud de reconocimiento de caracteres”, obtenido en la SBC Raspberry Pi 4 B+, se encontró que el sistema pudo reconocer adecuadamente las placas de rodaje de 380 placas, de un total de 406, lo cual equivale a un 93.60%. Esto implica que el desempeño del modelo en la SBC Raspberry Pi 4 B+ se redujo en un 0.5% aproximadamente, no obstante, en cuando al tiempo de inferencia, se tiene un tiempo de 1.55 segundos.

A continuación, se realizó la misma evaluación, en esta oportunidad, empleando el modelo simplificado. En la Figura 39 se presenta la captura de pantalla del sistema ejecutando la prueba con el modelo simplificado. Para el monitoreo de la SBC Raspberry Pi 4 B+ se empleó la aplicación VNC Viewer.

```

1 import os
2 import re
3 import time
4 import cv2
5 import glob
6 import numpy as np
7 import pandas as pd
8 from tensorflow.keras.models import load_model
9 from paddleocr import PPOCR
10
11 #Paths definition
12 PATH_TO_PROJECT = "/home/rpi/projects/PeruvianPlateDetection"
13 PATH_TO_LITE_MODEL = "/home/rpi/projects/PeruvianPlateDetection/pplatedetector_test_tfLite.py"
14 PATH_TO_IMAGES = "/home/rpi/projects/PeruvianPlateDetection/images"
15
16 #Files
17 PATH_TO_CSV_TIME = "/home/rpi/projects/PeruvianPlateDetection/pplatedetector_test_tfLite.py"
18 PATH_TO_CSV_PRED = "/home/rpi/projects/PeruvianPlateDetection/pplatedetector_test_tfLite.py"
19
20 #Registering settings
21 tstart = time.time()
22
23 # Load the TensorFlow model
24 interpreter = tf.lite.Interpreter(model_path=PATH_TO_LITE_MODEL)
25 interpreter.allocate_tensors()
26
27 # Get model details
28 input_details = interpreter.get_input_details()
29 output_details = interpreter.get_output_details()
30 model_height = input_details[0]['shape'][1]
31 model_width = input_details[0]['shape'][2]
32
33 # Test finished. Setting time was: 3.199129104614258 seconds.
34
35 [2023/05/24 20:12:53] ppocr DEBUG: dt_boxes num : 1, elapse : 0.0363328456878562
36 [2023/05/24 20:12:53] ppocr DEBUG: cls num : 1, elapse : 0.029500961303710938
37 [2023/05/24 20:12:53] ppocr DEBUG: rec_res num : 1, elapse : 0.3188960947579801
38 [2023/05/24 20:12:54] ppocr DEBUG: dt_boxes num : 1, elapse : 0.0250973701477050
39 [2023/05/24 20:12:54] ppocr DEBUG: cls num : 1, elapse : 0.029464244842529297
40 [2023/05/24 20:12:54] ppocr DEBUG: rec_res num : 1, elapse : 0.3208241462707510
41 [2023/05/24 20:12:55] ppocr DEBUG: dt_boxes num : 0, elapse : 0.0194239016394042
42 [2023/05/24 20:12:55] ppocr DEBUG: cls num : 0, elapse : 0
43 [2023/05/24 20:12:55] ppocr DEBUG: rec_res num : 0, elapse : 3.5762786865234375
44 [2023/05/24 20:12:55] ppocr DEBUG: dt_boxes num : 1, elapse : 0.0273916721343994
45 [2023/05/24 20:12:55] ppocr DEBUG: cls num : 1, elapse : 0.03263214509991797
46 [2023/05/24 20:12:55] ppocr DEBUG: rec_res num : 1, elapse : 0.3229711055755615
47 [2023/05/24 20:12:56] ppocr DEBUG: dt_boxes num : 1, elapse : 0.0360395908355712
48 [2023/05/24 20:12:56] ppocr DEBUG: cls num : 1, elapse : 0.029619216918945312
49 [2023/05/24 20:12:56] ppocr DEBUG: rec_res num : 1, elapse : 0.3186781406402588
50
51 rpi@raspberrypi:~$

```

Figura 39. Captura de pantalla indicando el resultado de la aplicación de prueba con el modelo simplificado.

Fuente. Captura de pantalla del Sistema en Raspberry Pi 4 B+.

Los resultados con relación a los tiempos del sistema se presentan en el Cuadro 42, presentados en segundos.

Cuadro 42

Resultados de los tiempos empleados por el modelo simplificado en la SBC Raspberry Pi 4 B+.

Indicador	Promedio	Mínimo	Máximo
Tiempo de inicialización	3.20	-	-
Tiempo de preprocesamiento	0.03	0.02	0.10
Tiempo de detección	0.68	0.67	0.71
Tiempo de reconocimiento	0.53	0.00	2.43
Tiempo total (Indicador I4: tiempo de ejecución)	1.24	0.72	3.13

Fuente. Elaborado a partir de los resultados de la evaluación.

Con relación al indicador I6: exactitud de reconocimiento de caracteres, obtenido en la SBC Raspberry Pi 4 B+, se encontró que el sistema pudo reconocer adecuadamente las placas de rodaje de 374 placas, de un total de 406, lo cual equivale a un 92.12%. Esto implica que el desempeño del modelo en la SBC Raspberry Pi 4 B+ se redujo en un 1.48% con relación al desempeño del modelo completo en la misma SBC (93.60%).

Con la finalidad de realizar una evaluación comparativa, se presentan en el Cuadro 43 los resultados más relevantes para ambos modelos (completo y simplificado), con el objeto de poder determinar con ello el modelo a emplear para la implementación de operación con captura de imágenes del entorno en tiempo real.

Cuadro 43

Comparación de resultados entre el modelo completo y el modelo simplificado.

Indicador	Modelo completo			Modelo simplificado		
	Prom.	Mín.	Máx.	Prom.	Mín.	Máx.
Tiempo de inicialización	33.68	-	-	3.20	-	-
Tiempo de preprocesamiento	0.03	0.02	0.09	0.03	0.02	0.10
Tiempo de detección	0.97	0.91	5.92	0.68	0.67	0.71
Tiempo de reconocimiento	0.55	0.00	4.63	0.53	0.00	2.43
Indicador I4	1.55	0.99	6.37	1.24	0.72	3.13
Indicador I6	93.60%	-	-	92.12%	-	-

Fuente. Elaborado a partir de los resultados de la evaluación.

Nota. Los valores para los tiempos están presentados en segundos.

Pudo apreciarse que se logró una reducción de 0.31 segundos (una reducción de aproximadamente 20%) en el tiempo total de ejecución del sistema (Indicador I3), con una pérdida de tan solo 1.48% en la exactitud de reconocimiento de caracteres de las placas. Asimismo, pudo apreciarse que el tiempo de detección del modelo ligero es más uniforme, variando desde los 0.68 segundos hasta 0.71 segundos, en comparación con el modelo completo, en el cual el tiempo de detección osciló desde 0.91 segundos, llevando a tomar 5.92 segundos. Esto en particular resulta limitante para un posible desempeño en tiempo real. En ese sentido, se escogió el modelo ligero para la implementación del prototipo final empleando captura de imágenes del entorno a través de una cámara.

5.1.6. Evaluación práctica del prototipo

Se realizó una primera evaluación en un ambiente de interiores con poca luz. Se empleó un smartphone para mostrar la imagen de una placa vehicular peruana al prototipo, obteniéndose resultados satisfactorios: de 4 placas

mostradas, se obtuvo 4 detecciones y reconocimientos correctos. Estos resultados se presentan en la Figura 40.

Asimismo, se realizó una segunda evaluación, la cual consistió de pruebas en entornos exteriores, con placas de rodaje de vehículos reales expuestas a luz natural. Como resultado de esta evaluación se obtuvieron nuevamente resultados satisfactorios. Este resultado se presenta en la Figura 41.

Finalmente, se realizó una tercera evaluación, nuevamente en exteriores, de noche, con placas de vehículos reales. Como resultado de esta evaluación se obtuvieron nuevamente resultados satisfactorios. Este resultado se presenta en la Figura 42.

De este modo, se validó la funcionalidad del prototipo y la precisión del modelo y del sistema en su totalidad para detectar y reconocer adecuadamente placas de rodaje peruanas.

Evaluación 1: Placa detectada y reconocida correctamente (AJG-351)



Evaluación 2: Placa detectada y reconocida correctamente (F5U-597)



Evaluación 3: Placa detectada y reconocida correctamente (EEI-465)



Evaluación 3: Placa detectada y reconocida correctamente (D9F-620)

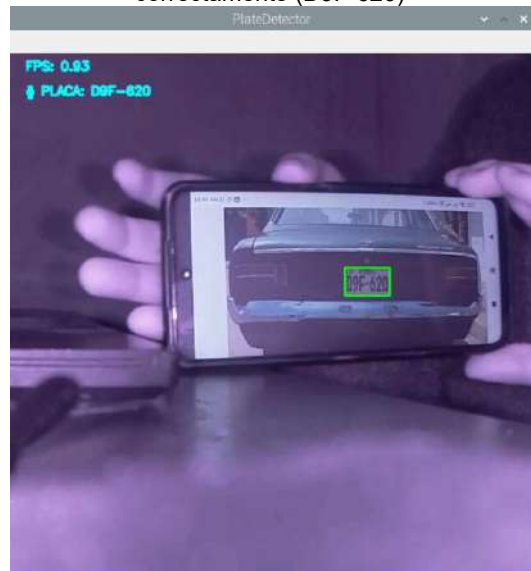


Figura 40. Evaluación en interiores con *smartphone*.

Fuente. Imágenes capturadas del stream en tiempo real, usando la aplicación VNC Viewer.

Evaluación de día: Configuración de la prueba en exteriores



Evaluación: Placa detectada y reconocida correctamente (A7Z-021)

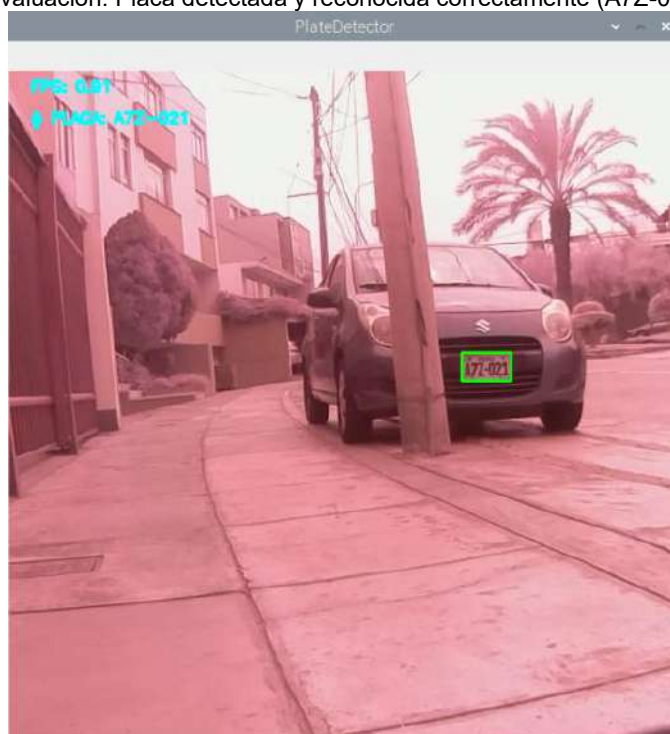


Figura 41. Evaluación en exteriores de día.

Fuente. Imágenes capturadas del stream en tiempo real, usando la aplicación VNC Viewer.

Evaluación 1: Placa detectada y reconocida correctamente (AZG-025)



Evaluación 2: Placa detectada y reconocida correctamente (2160-3Y)



Figura 42. Evaluación en exteriores de noche con vehículos.

Fuente. Imágenes capturadas del stream en tiempo real, usando la aplicación VNC Viewer.

5.2. Resultados obtenidos en función a los indicadores de evaluación de las variables de la investigación

Para finalizar la presentación de resultados, se presenta a continuación, en el Cuadro 44, los resultados obtenidos para los indicadores definidos en el numeral 3.3.

Cuadro 44

Resultados de la evaluación de indicadores definidos en el numeral 3.3: Operacionalización de las variables.

Indicador	Evaluación	Comentario
I1: Tiempo de entrenamiento	1 592.29 segundos	El tiempo de entrenamiento fue menor de 30 minutos, lo cual lo hace viable.
I2: Implementación del Sistema de Software	01 script	Se implementó un script en lenguaje Python que incorpora el modelo entrenado seleccionado y el módulo de reconocimiento de caracteres
I3: Implementación del Sistema de Hardware	01 prototipo	Se implementó un prototipo empleando la SBC Raspberry Pi 4 B+.
I4: Tiempo de ejecución	1.24 segundos en promedio	El tiempo de ejecución puede mejorarse empleando aceleradores de hardware.
I5: Precisión promedio (AP)	83.2% (sobre una IoU: 0.50-0.95)	Evaluación hecha sobre 79 imágenes de test distintas a las empleadas durante el entrenamiento.
I6: Exactitud de reconocimiento de caracteres	92.12%	Evaluación hecha sobre el total de 406 placas que conforman el conjunto de datos.

Fuente. Elaborado a partir de los resultados obtenidos en la evaluación del prototipo.

VI. DISCUSIÓN

Según los resultados obtenidos en la presente investigación, se pudo verificar que es técnicamente viable la implementación de un sistema basado en inteligencia artificial para la detección y el reconocimiento de las placas de rodaje de Perú.

De este modo, se validaron las hipótesis planteadas, las cuales responden a los problemas que establecen los objetivos de la presente tesis.

La hipótesis principal fue la siguiente: “Es posible desarrollar un sistema automático para la identificación de placas de rodaje peruanas, en el departamento de Lima en el año 2023, empleando un modelo basado en inteligencia artificial adecuado”.

A la luz de los resultados obtenidos, se pudo confirmar la validez de esta hipótesis, puesto que si fue factible el desarrollo del sistema automático para la identificación de placas de rodaje peruanas. El modelo basado en inteligencia artificial presentó un desempeño satisfactorio, obtenido gracias a la diversidad de muestras consideradas en el dataset que se empleó durante el entrenamiento y la evaluación, muestras que, de acuerdo con el análisis estadístico realizado, fueron amplias y generales (incluyendo placas inclinadas, placas borrosas, placas con bajo contraste, entre otros casos).

Asimismo, se pudo confirmar la validez de las hipótesis planteadas como respuesta a los problemas específicos que abordó la presente tesis.

El primer problema específico se definió del siguiente modo: “¿Es posible realizar el entrenamiento de un sistema automático que permita el reconocimiento de placas de rodaje peruanas en el departamento de Lima, en el año 2023?”. La hipótesis planteada para este problema específico fue: “Es posible realizar el entrenamiento de un sistema automático que permita el reconocimiento de placas de rodaje peruanas en el departamento de Lima, en el año 2023”.

Los resultados obtenidos permitieron confirmar la validez de la hipótesis planteada para el primer problema específico, puesto que fue posible entrenar no solo uno, sino hasta 5 modelos basados en inteligencia artificial con

resultados satisfactorios, empleando para ello un dataset que desde el punto de vista estadístico se puede considerar como amplio y general.

El segundo problema específico se definió del siguiente modo: “¿Es posible diseñar un subsistema de software para implementar un sistema automático que permita el reconocimiento de placas de rodaje peruanas en el departamento de Lima, en el año 2023?”. La hipótesis planteada para este problema específico fue: “Es posible diseñar un subsistema de software para implementar un sistema automático que permita el reconocimiento de placas de rodaje peruanas en el departamento de Lima, en el año 2023”.

Los resultados obtenidos permitieron confirmar la validez de la hipótesis planteada para el segundo problema específico, puesto que fue posible implementar un script en lenguaje Python que importa el modelo entrenado seleccionado y sobre él, fue capaz de realizar la inferencia sobre una imagen determinada para detectar la ubicación de la placa de rodaje y, posteriormente, aplicando un módulo de reconocimiento de caracteres, pudo determinar los caracteres que conforman el número de la placa. De acuerdo con los resultados obtenidos, todo este proceso se logró con una exactitud superior al 90%.

Finalmente, el tercer y último problema se definió del modo siguiente: “¿Es posible integrar un subsistema de hardware para implementar un sistema automático que permita el reconocimiento de placas de rodaje peruanas en el departamento de Lima, en el año 2023?”. La hipótesis planteada para dar respuesta a este problema fue: “Es posible integrar un subsistema de hardware para implementar un sistema automático que permita el reconocimiento de placas de rodaje peruanas en el departamento de Lima, en el año 2023”.

Nuevamente, los resultados obtenidos permitieron confirmar la validez de la hipótesis planteada, puesto que se verificó la viabilidad técnica de la integración del software con el hardware a través de un prototipo empleando una SBC Raspberry Pi, con la cual se evaluaron indicadores de desempeño, obteniéndose resultados satisfactorios.

Como puede desprenderse de la presente discusión de los resultados, la presente investigación logró desarrollar y dar respuesta válida a todos los problemas definidos, contribuyendo de este modo con en el desarrollo de sistemas automáticos que permitan abordar tareas de relevancia social y científica.

VII. CONCLUSIONES Y RECOMENDACIONES

7.1. Conclusiones

El trabajo realizado durante el desarrollo de la presente investigación permitió plantear las siguientes conclusiones:

1. Se implementó con éxito de un sistema automático para la detección de placas de rodaje de Perú, basado en un modelo de inteligencia artificial entrenado con un conjunto de datos fotográficos lo suficientemente variado. El sistema demostró una capacidad adecuada de generalización frente a condiciones variables, lo que permite afirmar que se cumplió el objetivo general de la investigación.
2. Respecto al entrenamiento de modelos de inteligencia artificial, se logró desarrollar y entrenar un modelo basado en SSD MobileNet V1 FPNLite 640 x 640 mediante ajuste fino, alcanzando una precisión media del 82.3 % y una recuperación del 83.2 % sobre datos de prueba. Esto demuestra un desempeño adecuado del modelo para la detección de placas en condiciones no controladas.
3. En cuanto al diseño del subsistema de software, se desarrolló una aplicación en lenguaje de Python para emplear una versión optimizada del modelo. Si bien este modelo simplificado presentó una ligera reducción en la exactitud (1.48%), mejoró significativamente el tiempo de inferencia (0.68 segundos), manteniendo una precisión superior al 90%. Además, el uso de herramientas de software de código abierto permitió reducir costos y facilitar el desarrollo.
4. Respecto a la integración del subsistema de hardware, se implementó un prototipo funcional en una Raspberry Pi 4 B+, dispositivo de recursos computacionales medios, que fue capaz de ejecutar correctamente el sistema de reconocimiento de placas. Esto valida la viabilidad práctica del sistema en entornos reales con recursos limitados.

7.2. Recomendaciones

El desarrollo de la presente investigación, así como los resultados obtenidos, permitieron que se planteen las siguientes recomendaciones:

1. Se recomienda realizar investigaciones posteriores para especializar el modelo, de manera que sea capaz de detectar placas según los tipos de placas de rodaje definidos en la normativa peruana vigente.
2. Se recomienda experimentar con otros modelos, además de los revisados en la presente investigación, en particular aquellos modelos que implican un mayor espacio en memoria que, debido a las limitaciones de la tarjeta considerada para su implementación práctica (SBC Raspberry Pi), no se abordaron. Al respecto, resulta de interés analizar si el entrenar un modelo computacionalmente más exigente permite obtener mejores indicadores que los obtenidos empleando modelos ligeros, considerando los costos asociados al incremento en la potencia computacional.
3. Se recomienda implementar el modelo con una tarjeta aceleradora de gráficos o de procesamiento, de manera que se logre reducir el tiempo de inferencia del modelo.
4. Finalmente, se recomienda al titular encargado de la emisión de placas de rodaje en el Perú que procure que los diseños de los caracteres de la letra 'O' y el número '0' sean lo suficientemente diferentes para evitar errores en la interpretación del número de placa. Lo mismo para el número '1' y la letra 'l'. Estas similitudes pueden generar errores en el reconocimiento, incluso para el ojo humano. Asimismo, en lo que respecta a la supervisión de las placas de rodaje, se recomienda evitar el uso de marcos gruesos o diseños que invadan el área donde se ubican los caracteres del número de placa, ya que esto podría dificultar el reconocimiento óptico de dichos caracteres.

REFERENCIAS BIBLIOGRÁFICAS

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., ... Google Brain. (2016). TensorFlow: A System for Large-Scale Machine Learning. *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI '16)*, 44.

Aguilar, J. (2022). *Prototipo de reconocimiento de placas vehiculares para detección de vehículos alertados en el complejo de control aduanero de Tomasiri, Tacna - 2022* [Universidad Privada de Tacna]. <https://repositorio.upt.edu.pe/bitstream/handle/20.500.12969/2429/Aguilar-Anaya-Juan.pdf?sequence=1&isAllowed=y>

Alake, R. (2020, febrero 5). *A Beginner's Introduction To TensorFlow Lite*. <https://towardsdatascience.com/a-beginners-introduction-to-tensorflow-lite-924320deed5>

Aphex34. (2015). *Typical CNN architecture*. https://commons.wikimedia.org/w/index.php?title=File:Typical_cnn.png&oldid=671717156

Ávalos, V. (2014). *Texto: el método científico aplicado en la elaboración de tesis para optar el título profesional de ingeniero químico. Informe Final. Resolución de aprobación N.º 136-2014-R*.

Chinakook. (2018). *LabelImg2*. Git code. <https://github.com/chinakook/labelimg2>

Chu, V. (2019). *Sistema de Reconocimiento de Placas Vehiculares para Mejorar el Registro de Vehículos en el Hospedaje Suites Recreo - 2019*. Universidad César Vallejo.

Congreso de la República del Perú. (1999). *Ley N.º 27181, Ley General de Transporte y Tránsito Terrestre*.

https://cdn.www.gob.pe/uploads/document/file/19071/1_0_3106.pdf?v=1607701977

Delipetrev, B., Tsinaraki, C. y Kostić, U. (2020). *Historical evolution of artificial intelligence* (Publications Office of the European Union, Ed.). <https://doi.org/10.2760/801580>

Deng, L. (2018). Artificial Intelligence in the Rising Wave of Deep Learning: The Historical Path and Future Outlook [Perspectives]. *IEEE Signal Processing Magazine*, 35(1), 177-180. <https://doi.org/10.1109/MSP.2017.2762725>

Diaz, K. (2006). *Localización y reconocimiento automático del número de la placa de un automóvil* [Pontificia Universidad Católica del Perú]. https://tesis.pucp.edu.pe/repositorio/bitstream/handle/20.500.12404/1085/DIAZ_ROJAS_KRISTIANS_RECONOCIMIENTO_PLACA_AUTOMOVIL.pdf?sequence=1&isAllowed=y

Dilek, E. y Dener, M. (2023). Computer Vision Applications in Intelligent Transportation Systems: A Survey. *Sensors*, 23(6), 2938. <https://doi.org/10.3390/s23062938>

Donoser, M. y Bischof, H. (2006). Efficient Maximally Stable Extremal Region (MSER) Tracking. *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 1 (CVPR'06)*, 553-560. <https://doi.org/10.1109/CVPR.2006.107>

Du, Y., Li, C., Guo, R., Yin, X., Liu, W., Zhou, J., Bai, Y., Yu, Z., Yang, Y., Dang, Q. y Wang, H. (2020). *PP-OCR: A Practical Ultra Lightweight OCR System*. <http://arxiv.org/abs/2009.09941>

Duan, K., Bai, S., Xie, L., Qi, H., Huang, Q. y Tian, Q. (2019). *CenterNet: Keypoint Triplets for Object Detection*. <http://arxiv.org/abs/1904.08189>

El Peruano. (2019, agosto 13). *Renovación del parque automotor*. Diario Oficial El Peruano. <https://elperuano.pe/noticia/82422-renovacion-del-parque->

automotor#:~:text=El%20parque%20automotor%20peruano%20es,y%20el%2015%25%20unidades%20pesadas.

Espinoza, E. (2022). El problema, el objetivo, la hipótesis y las variables de la investigación. *Portal de la Ciencia*, 1(2), 1-71. <https://doi.org/10.51247/pdlc.v1i2.320>

Espinoza, G. (2014). *Sistema de reconocimiento de patrones en placas vehiculares para el acceso automático de visitas a un edificio* [Pontificia Universidad Católica del Perú]. https://tesis.pucp.edu.pe/repositorio/bitstream/handle/20.500.12404/5753/ESPINOZA_GERARDO_PLACAS_VEHICULARES_AUTOMATICO_VISITAS_EDIFICIO.pdf?sequence=1&isAllowed=y

Esteban Nieto, T. N. (2018). *Tipos de investigación*. <https://core.ac.uk/download/pdf/250080756.pdf>

Girshick, R. (2015). *Fast R-CNN*. <http://arxiv.org/abs/1504.08083>

Girshick, R., Donahue, J., Darrell, T. y Malik, J. (2014). *Rich feature hierarchies for accurate object detection and semantic segmentation*. <http://www.cs.berkeley.edu/~rbg/rcnn>.

Goodfellow, I., Bengio, Y. y Courville, A. (2016). Introduction. En *Deep Learning* (p. 2). The MIT Press. <https://www.deeplearningbook.org/>

Han, B.-G., Lee, J. T., Lim, K.-T. y Choi, D.-H. (2020). License Plate Image Generation using Generative Adversarial Networks for End-To-End License Plate Character Recognition from a Small Set of Real Images. *Applied Sciences*, 10(8), 2780. <https://doi.org/10.3390/app10082780>

He, K., Gkioxari, G., Dollár, P. y Girshick, R. (2017). *Mask R-CNN*. <http://arxiv.org/abs/1703.06870>

Hernández, R., Fernández, C. y Baptista, P. (2014). Metodología de la investigación. En McGraw-Hill / Interamericana Editores S.A. de C.V. (Ed.), *Metodología de la investigación* (6.a ed.). McGraw-Hill.

- Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M. y Adam, H. (2017). *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. <http://arxiv.org/abs/1704.04861>
- Hui, J. (2018). *mAP (mean Average Precision) for Object Detection*. Medium. <https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173>
- Instituto Nacional de Estadística e Informática. (s. f.-a). *Robo de Vehículos*. Recuperado 15 de diciembre de 2022, de <https://m.inei.gob.pe/estadisticas/indice-tematico/vehicle-theft/>
- Instituto Nacional de Estadística e Informática. (s. f.-b). *Infracciones de tránsito*. Recuperado 15 de diciembre de 2022, de <https://m.inei.gob.pe/estadisticas/indice-tematico/traffic-violations/>
- Jaderberg, M., Simonyan, K., Zisserman, A. y Kavukcuoglu, K. (2015). *Spatial Transformer Networks*.
- Janasik, N., Honkela, T. y Bruun, H. (2009). Text Mining in Qualitative Research: Application of an Unsupervised Learning Method. *Organizational Research Methods*, 12(3), 436-460. <https://doi.org/10.1177/1094428108317202>
- Juras, E. (2023). *TensorFlow Lite Object Detection API in Colab*. GitHub. https://github.com/EdjeElectronics/TensorFlow-Lite-Object-Detection-on-Android-and-Raspberry-Pi/blob/master/Train_TFLite2_Object_Detection_Model.ipynb
- Kessentini, Y., Besbes, M. D., Ammar, S. y Chabbouh, A. (2019). A two-stage deep neural network for multi-norm license plate detection and recognition. *Expert Systems with Applications*, 136, 159-170. <https://doi.org/10.1016/j.eswa.2019.06.036>

- Krizhevsky, A., Sutskever, I. y Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6), 84-90. <https://doi.org/10.1145/3065386>
- Laroca, R., Zanlorensi, L. A., Gonçalves, G. R., Todt, E., Schwartz, W. R. y Menotti, D. (2019). *An Efficient and Layout-Independent Automatic License Plate Recognition System Based on the YOLO detector*. <https://doi.org/10.1049/itr2.12030>
- LeCun, Y., Bengio, Y. y Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436-444. <https://doi.org/10.1038/nature14539>
- Li, Z., Liu, F., Yang, W., Peng, S. y Zhou, J. (2022). A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects. *IEEE Transactions on Neural Networks and Learning Systems*, 33(12), 6999-7019. <https://doi.org/10.1109/TNNLS.2021.3084827>
- Lima Cómo Vamos. (2022). *¿Cómo vamos en Lima y Callao? Reporte urbano de indicadores de calidad de vida 2021*. <https://www.limacomovamos.org/wp-content/uploads/2022/11/ReporteIndicadoresLCV2021.pdf>
- Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B. y Belongie, S. (2016). *Feature Pyramid Networks for Object Detection*. <http://arxiv.org/abs/1612.03144>
- Lin, T.-Y., Goyal, P., Girshick, R., He, K. y Dollár, P. (2017). *Focal Loss for Dense Object Detection*. <http://arxiv.org/abs/1708.02002>
- Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y. y Berg, A. C. (2015). *SSD: Single Shot MultiBox Detector*. https://doi.org/10.1007/978-3-319-46448-0_2
- McCarthy, J., Minsky, M., Rochester, N. y Shannon, C. (1955). *A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence*. <http://jmc.stanford.edu/articles/dartmouth/dartmouth.pdf>

- McCulloch, W. y Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biology*, 5, 115-133.
- Michie, D., Spiegelhalter, D. J. y Taylor, C. C. (2009). Machine Learning. En *Machine Learning, Neural and Statistical Classification*. Overseas Press (Obra original publicada en 1994).
- Ministerio de Transportes y Comunicaciones del Perú. (2008). D.S. N.° 017-2008-MTC.
https://cdn.www.gob.pe/uploads/document/file/19225/1_0_1488.pdf?v=1530727394
- Ministerio de Transportes y Comunicaciones del Perú. (2010). D.S. N.° 006-2010-MTC.
https://cdn.www.gob.pe/uploads/document/file/404349/DS_006-2010-MTC.pdf?v=1614892197
- Montazzolli, S. y Rosito, C. (2018). License Plate Detection and Recognition in Unconstrained Scenarios. *15th European Conference on Computer Vision*.
https://openaccess.thecvf.com/content_ECCV_2018/papers/Sergio_Silva_License_Plate_Detection_ECCV_2018_paper.pdf
- Moreno, A., Armengol, E., Béjar, J., Belanche, L., Cortés, U., Gavalda, R., Gimeno, J. M., López, B., Martín, M. y Sánchez, M. (1994). Aprendizaje automático. En S. Edicions de la Universitat Politècnica de Catalunya (Ed.), *Aprendizaje Automático* (Primera edición, p. 7). Servei de Publicacions de la UPC y CPET (Centre de Publicacions del Campus Nord).
<https://upcommons.upc.edu/bitstream/handle/2099.3/36157/9788483019962.pdf?sequence=1&isAllowed=y>
- Nadendla, H. (2020, octubre 11). *Why are LSTMs struggling to matchup with Transformers?* Medium. <https://medium.com/analytics-vidhya/why-are-lstms-struggling-to-matchup-with-transformers-a1cc5b2557e3>

- Nian, R., Liu, J. y Huang, B. (2020). A review On reinforcement learning: Introduction and applications in industrial process control. *Computers & Chemical Engineering*, 139, 106886. <https://doi.org/10.1016/j.compchemeng.2020.106886>
- Oladipupo, T. (2010). Types of Machine Learning Algorithms. En Y. Zhang (Ed.), *New Advances in Machine Learning* (pp. 19-24). InTech.
- O'Shea, K. y Nash, R. (2015). An introduction to convolutional neural networks. *ArXiv*, *abs/1511.08458*.
- Pech-Pacheco, J. L., Cristobal, G., Chamorro-Martinez, J. y Fernandez-Valdivia, J. (2000). Diatom autofocusing in brightfield microscopy: a comparative study. *Proceedings 15th International Conference on Pattern Recognition*. *ICPR-2000*, 314-317. <https://doi.org/10.1109/ICPR.2000.903548>
- Raghunandan, K. S., Shivakumara, P., Jalab, H. A., Ibrahim, R. W., Kumar, G. H., Pal, U. y Lu, T. (2018). Riesz Fractional Based Model for Enhancing License Plate Detection and Recognition. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(9), 2276-2288. <https://doi.org/10.1109/TCSVT.2017.2713806>
- Ramirez, B. y Tito, M. (2020). *Reconocimiento automático de placas de rodaje utilizando una red neuronal convolucional para el ingreso de vehículos en la universidad Ricardo Palma* [Universidad Ricardo Palma]. <https://repositorio.urp.edu.pe/handle/20.500.14138/3646>
- Redmon, J., Divvala, S., Girshick, R. y Farhadi, A. (2015). *You Only Look Once: Unified, Real-Time Object Detection*. <http://arxiv.org/abs/1506.02640>
- Ren, S., He, K., Girshick, R. y Sun, J. (2015). *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. <http://arxiv.org/abs/1506.01497>

- Rosenblatt, F. (1957). *The perceptron: A perceiving and recognizing automaton*. <https://blogs.umass.edu/brain-wars/files/2016/03/rosenblatt-1957.pdf>
- Rosenblatt, F. (1962). *Principles of neurodynamics: perceptrons and the theory of brain mechanisms*. Spartan Books.
- Samoili, S., López Cobo, M., Gómez, E., De Prato, G., Martínez-Plumed, F. y Delipetrev, B. (2020). *AI Watch. Defining Artificial Intelligence. Towards an operational definition and taxonomy of artificial intelligence* (Publications Office of the European Union, Ed.). <https://doi.org/10.2760/382730>
- Schuster, M. (1999). *On supervised learning from sequential data with applications for speech recognition* [Nara Institute of Science and Technology].
<https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=ebb53aebcbc8edf6c52d94d5f75f16d1c8cf88f2>
- Sebastian. (2022, mayo 22). *Deep Learning Architectures for Object Detection: Yolo vs. SSD vs. RCNN*. Programmathically: A Blog on Building Machine Learning Solutions. <https://programmathically.com/deep-learning-architectures-for-object-detection-yolo-vs-ssd-vs-rcnn/>
- Shashirangana, J., Padmasiri, H., Meedeniya, D. y Perera, C. (2021). Automated License Plate Recognition: A Survey on Methods and Techniques. *IEEE Access*, 9, 11203-11225.
<https://doi.org/10.1109/ACCESS.2020.3047929>
- Shrestha, A. y Mahmood, A. (2019). Review of Deep Learning Algorithms and Architectures. *IEEE Access*, 7, 53040-53065.
<https://doi.org/10.1109/ACCESS.2019.2912200>
- Sistema Nacional de Información Ambiental del Perú. (s. f.). *Indicador: Vehículos por cada mil habitantes*. Recuperado 25 de febrero de 2023, de <https://sinia.minam.gob.pe/indicador/966>

- Superintendencia Nacional de Registros Públicos del Perú. (2020, enero 8). *Sunarp: número de autos que circulan en el país acumula una década de crecimiento continuo*.
<https://www.sunarp.gob.pe/PRENSA/inicio/post/2020/01/08/sunarp-numero-de-autos-que-circulan-en-el-pais-acumula-una-decada-de-crecimiento-continuo>
- Tan, M., Pang, R. y Le, Q. V. (2019). *EfficientDet: Scalable and Efficient Object Detection*. <http://arxiv.org/abs/1911.09070>
- Tensorflow. (s. f.). *Object Detection Model Zoo*. Github. Recuperado 15 de marzo de 2023, de https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md
- Turing, A. (1950). Compute machinery and intelligence. *Mind, New Series*, 59(236), 433-460. <http://www.jstor.org/stable/2251299?origin=JSTOR-pdf>
- Viola, P. y Jones, M. (2001). *Rapid Object Detection using a Boosted Cascade of Simple Features*.
- Wang, C.-Y., Bochkovskiy, A. y Liao, H.-Y. M. (2022). *YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors*. <http://arxiv.org/abs/2207.02696>
- Widrow, B. y Hoff, M. (1960). Adaptive Switching Circuits. *1960 IRE WESCON Convention Record, Part 4*, 96-104. <https://www-isl.stanford.edu/~widrow/papers/c1960adaptiveswitching.pdf>
- Zhou, Z.-H. (2021). Introduction (Shaowu, L., Trad). En *Machine Learning*. Springer Nature Singapore (Obra original publicada en 2016).
- Zou, L., Zhao, M., Gao, Z., Cao, M., Jia, H. y Pei, M. (2018). License Plate Detection with Shallow and Deep CNNs in Complex Environments. *Complexity*, 2018, 1-6. <https://doi.org/10.1155/2018/7984653>

ANEXOS

ANEXO 1: SCRIPT EN PYTHON EMPLEADO EN EL PROTOTIPO PARA TEST DEL MODELO COMPLETO: pplatedetector_test.py

```

1 import os
2 import tensorflow as tf
3 import cv2
4 import glob
5 import numpy as np
6 import time
7 import re
8 import pandas as pd
9 from paddleocr import PaddleOCR
10
11 #Paths definition
12 PATH_TO_PROJECT = '/home/rpi/projects/PeruvianPlateDetection'
13 PATH_TO_SAVED_MODEL = os.path.join(PATH_TO_PROJECT, 'model', 'saved_model')
14 PATH_TO_IMAGES = os.path.join(PATH_TO_PROJECT, 'dataset')
15
16 #Files
17 PATH_TO_CSV_TIMES = os.path.join(PATH_TO_PROJECT, 'model', 'times.csv')
18 PATH_TO_CSV_PREDICTIONS = os.path.join(PATH_TO_PROJECT, 'model', 'predictions.csv')
19
20 #Registering setting start time
21 tstart = time.time()
22
23 #Creating detection function from model
24 detect_fn = tf.saved_model.load(PATH_TO_SAVED_MODEL)
25
26 #Creating object to handle OCR operation
27 ocr = PaddleOCR(lang = 'en',
28                 use_gpu = False,
29                 use_angle_cls = True,
30                 rec_algorithm = 'CRNN')
31
32 #Patterns to match predicted string
33 pattern1 = "[A-Z,0-9]{4}[A-Z,0-9]{2}" #Current for minor vehicles
34 pattern2 = "[A-Z,0-9]{3}[A-Z,0-9]{3}" #Current for vehicles
35 pattern3 = "[A-Z,0-9]{2}[A-Z,0-9]{4}" #Old for vehicles
36
37 correctionsN2L = {
38     "0":"0",
39     "4":"A",
40     "8":"B"}
41 correctionsL2N = {
42     "0":"0",
43     "A":"4",
44     "B":"8"}
45
46 def padLicence(plate):
47     padding = 0.2
48     h, w, _ = plate.shape
49     padY = int(padding * h / 2)
50     padX = int(padding * w / 2)
51     padded = cv2.copyMakeBorder(plate, padY, padY, padX, padX, cv2.BORDER_CONSTANT, value = [0, 0, 0])
52     return padded
53
54 #Verifies if text detected fits licence patterns and returns type
55 def isLicence(licence):
56     if len(re.findall(pattern1, licence)) == 1:
57         return 1
58     elif len(re.findall(pattern2, licence)) == 1:
59         return 2
60     elif len(re.findall(pattern3, licence)) == 1:
61         return 3
62     else:
63         return 0
64
65 #Tries to correct text patterns
66 def verifyLicence(licence, typeOfLicence):
67     listLicence = list(licence)
68
69     if typeOfLicence == 1:
70         #Case corresponds to minor vehicle.
71         #Rules:
72         #First 4 characters are numbers
73         #'.' at the 5th position
74         #5th character is alphanumeric
75         #6th character is letter excluding "E"
76
77         #Verifying first 4 characters
78         if not licence[0:4].isnumeric():
79             for pos, letter in enumerate(licence[0:4]):
80                 if letter in correctionsL2N.keys():
81                     listLicence[pos] = correctionsL2N[letter]
82
83         #Updating group separator
84         listLicence[4] = '.'
85
86         #Verifying last character
87         if licence[6] in correctionsN2L.keys():
88             listLicence[6] = correctionsN2L[licence[6]]
89

```

```

90 elif typeOfLicence == 2:
91     #Case corresponds to minor vehicle.
92     #Rules:
93     #First character is letter
94     #Second and third letter are alphanumeric
95     #'-' at the 4th position
96     #Three lasts characters are numbers
97
98     #Verifying first character
99     if licence[0] in correctionsN2L.keys():
100         listlicence[0] = correctionsN2L[licence[0]]
101
102     #Updating group separator
103     listlicence[3] = '-'
104
105     #Verifying last 3 characters
106     if not licence[4:7].isnumeric():
107         for pos, letter in enumerate(licence[4:7]):
108             if letter in correctionsL2N.keys():
109                 listlicence[pos + 4] = correctionsL2N[letter]
110
111     licence = ''.join(listlicence)
112
113     return licence
114
115 #Function to process predictions and get if there is a plate
116 def getLicenceAndProb(img):
117     predictions = ocr.ocr(img, cls = True)
118
119     for prediction in predictions[0]:
120         lic, prob = prediction[1]
121         licType = isLicence(lic)
122         if (len(lic))>=7 and (licType != 0):
123             return [verifyLicence(lic, licType), prob]
124
125     return ['', 0]
126
127 #Reforancing images with plates
128 images = glob.glob(PATH_TO_IMAGES + '/*.jpg')
129
130 #Registering setting end time
131 tend = time.time()
132
133 setting_time = tend - tstart
134
135 #Loop to perform evaluation
136 predicted_licences = []
137 processing_times = []
138 for image in images:
139     image_name = os.path.basename(image)
140
141     #Preprocessing
142     tstart = time.time()
143     img_np = np.array(cv2.imread(image))
144     input_tensor = tf.convert_to_tensor(img_np)
145     input_tensor = input_tensor[tf.newaxis, ...]
146     tend = time.time()
147     preproc_time = tend - tstart
148
149     #Detection
150     tstart = time.time()
151     detections = detect_fn(input_tensor)
152     num_detections = int(detections.pop('num_detections'))
153     detections = {key: value[0, :num_detections].numpy() for key, value in detections.items()}
154     detections['num_detections'] = num_detections
155     detection_threshold = 0.7
156     scores = list(filter(lambda x: x> detection_threshold, detections['detection_scores']))
157     boxes = detections['detection_boxes'][:len(scores)]
158     tend = time.time()
159     detection_time = tend - tstart
160
161     #Recognition
162     tstart = time.time()
163     height, width, _ = img_np.shape
164     for box, score in zip(boxes, scores):
165         roi = box * [height, width, height, width]
166         roi = [int(i) for i in roi]
167         plate = img_np[roi[0] : roi[2], roi[1] : roi[3]]
168         prediction = getLicenceAndProb(plate)
169         if prediction[1] < 0.90:
170             plate_mod = padLicence(plate)
171             prediction = getLicenceAndProb(plate_mod)
172         predicted_licences.append([image_name] + prediction)
173     tend = time.time()
174     recognition_time = tend - tstart
175
176     processing_times.append([image_name, preproc_time, detection_time, recognition_time])
177
178 df_times = pd.DataFrame(processing_times, columns = ['image', 'preprocessing_time', 'detection_time', 'recognition_time'])
179 df_pred = pd.DataFrame(predicted_licences, columns = ['image', 'licence_number', 'pred_score'])
180
181 df_times.to_csv(PATH_TO_CSV_TIMES)
182 df_pred.to_csv(PATH_TO_CSV_PREDICTIONS)
183
184 print('Test finished. Setting time was: {} seconds.'.format(setting_time))

```

ANEXO 2: SCRIPT EN PYTHON EMPLEADO EN EL PROTOTIPO PARA TEST DEL MODELO LIGERO: pplatedetector_test_tflite.py

```

1  import os
2  import re
3  import time
4  import cv2
5  import glob
6  import numpy as np
7  import pandas as pd
8  from tensorflow.lite.python.interpreter import Interpreter
9  from paddleocr import PaddleOCR
10
11  #Paths definition
12  PATH_TO_PROJECT = '/home/rpi/projects/PeruvianPlateDetection'
13  PATH_TO_LITE_MODEL = os.path.join(PATH_TO_PROJECT, 'tflite_model', 'detect_quant.tflite')
14  PATH_TO_IMAGES = os.path.join(PATH_TO_PROJECT, 'dataset')
15
16  #Files
17  PATH_TO_CSV_TIMES = os.path.join(PATH_TO_PROJECT, 'tflite_model', 'times.csv')
18  PATH_TO_CSV_PREDICTIONS = os.path.join(PATH_TO_PROJECT, 'tflite_model', 'predictions.csv')
19
20  #Registering setting start time
21  tstart = time.time()
22
23  # Load the Tensorflow Lite model into memory
24  interpreter = Interpreter(model_path = PATH_TO_LITE_MODEL)
25  interpreter.allocate_tensors()
26
27  # Get model details
28  input_details = interpreter.get_input_details()
29  output_details = interpreter.get_output_details()
30  model_height = input_details[0]['shape'][1]
31  model_width = input_details[0]['shape'][2]
32
33  float_input = (input_details[0]['dtype'] == np.float32)
34
35  input_mean = 127.5
36  input_std = 127.5
37
38  #Creating object to handle OCR operation
39  ocr = PaddleOCR(lang = 'en',
40                  use_gpu = False,
41                  use_angle_cls = True,
42                  rec_algorithm = 'CRNN')
43
44  #Patterns to match predicted string
45  pattern1 = "[A-Z,0-9]{4}[A-Z,0-9]{2}" #Current for minor vehicles
46  pattern2 = "[A-Z,0-9]{3}[A-Z,0-9]{3}" #Current for vehicles
47  pattern3 = "[A-Z,0-9]{2}[A-Z,0-9]{4}" #Old for vehicles
48
49  correctionsN2L = {
50      "8": "0",
51      "4": "A",
52      "0": "B"
53  }
54  correctionsL2N = {
55      "0": "8",
56      "A": "4",
57      "B": "0"
58  }
59
60  def padlicence(plate):
61      padding = 0.2
62      h, w, _ = plate.shape
63      padY = int(padding * h / 2)
64      padV = int(padding * h / 2)
65      padX = int(padding * w / 2)
66      padded = cv2.copyMakeBorder(plate, padV, padV, padX, padX, cv2.BORDER_CONSTANT, value = [0, 0, 0])
67      return padded
68
69  #Verifies if text detected fits licence patterns and returns type
70  def islicence(licence):
71      if len(re.findall(pattern1, licence)) == 1:
72          return 1
73      elif len(re.findall(pattern2, licence)) == 1:
74          return 2
75      elif len(re.findall(pattern3, licence)) == 1:
76          return 3
77      else:
78          return 0
79
80  #Tries to correct text patterns
81  def verifylicence(licence, typeOflicence):
82      listlicence = list(licence)
83
84      if typeOflicence == 1:
85          #Case corresponds to minor vehicle.
86          #Rules:
87          #First 4 characters are numbers
88          #'-' at the 5th position
89          #5th character is alphanumeric
90          #6th character is letter excluding "E"
91
92          #Verifying first 4 characters
93          if not licence[0:4].isnumeric():
94              for pos, letter in enumerate(licence[0:4]):
95                  if letter in correctionsL2N.keys():
96                      listlicence[pos] = correctionsL2N[letter]
97
98          #Updating group separator
99          listlicence[4] = '-'
100
101          #Verifying last character
102          if licence[6] in correctionsN2L.keys():

```

```

106         listlicence[6] = correctionsN2L[licence[6]]
107
108     elif typeOfLicence == 2:
109         #Case corresponds to minor vehicle.
110         #Rules:
111         #First character is letter
112         #Second and third letter are alphanumeric
113         # '-' at the 4th position
114         #Three last characters are numbers
115
116         #Verifying first character
117         if licence[0] in correctionsN2L.keys():
118             listlicence[0] = correctionsN2L[licence[0]]
119
120         #Updating group separator
121         listlicence[3] = '-'
122
123         #Verifying last 3 characters
124         if not licence[4:7].isnumeric():
125             for pos, letter in enumerate(licence[4:7]):
126                 if letter in correctionsL2N.keys():
127                     listlicence[pos + 4] = correctionsL2N[letter]
128
129     licence = ''.join(listlicence)
130
131     return licence
132
133 #Function to process predictions and get if there is a plate
134 def getLicenceAndProb(img):
135     predictions = ocr.ocr(img, cls = True)
136
137     for prediction in predictions[0]:
138         lic, prob = prediction[1]
139         licType = isLicence(lic)
140         if (len(lic))>=7 and (licType != 0):
141             return [verifyLicence(lic, licType), prob]
142
143     return ['', 0]
144
145 #Referencing images with plates
146 images = glob.glob(PATH_TO_IMAGES + '/*.jpg')
147
148 #Registering setting end time
149 tend = time.time()
150
151 setting_time = tend - tstart
152
153 #Loop to perform evaluation
154 predicted_licences = []
155 processing_times = []
156 for image in images:
157     image_name = os.path.basename(image)
158     #Preprocessing
159     tstart = time.time()
160     #Load image and resize to expected shape [1xHxWx3]
161     image_bgr = cv2.imread(image)
162     image_rgb = cv2.cvtColor(image_bgr, cv2.COLOR_BGR2RGB)
163     imH, imW, _ = image_rgb.shape
164     image_resized = cv2.resize(image_rgb, (model_width, model_height))
165     input_data = np.expand_dims(image_resized, axis=0)
166
167     #Normalize pixel values if using a floating model (i.e. if model is non-quantized)
168     if float_input:
169         input_data = (np.float32(input_data) - input_mean) / input_std
170
171     tend = time.time()
172     preproc_time = tend - tstart
173
174     #Detection
175     tstart = time.time()
176     #Perform the actual detection by running the model with the image as input
177     interpreter.set_tensor(input_details[0]['index'], input_data)
178     interpreter.invoke()
179
180     # Retrieve detection results
181     boxes = interpreter.get_tensor(output_details[1]['index'])[0]
182     scores = interpreter.get_tensor(output_details[2]['index'])[0]
183     detection_threshold = 0.7
184     scores = list(filter(lambda x: x> detection_threshold, scores))
185     boxes = boxes[:len(scores)]
186     tend = time.time()
187     detection_time = tend - tstart
188
189     #Recognition
190     tstart = time.time()
191     height, width, _ = image_rgb.shape
192     for box, score in zip(boxes, scores):
193         roi = box * [height, width, height, width]
194         roi = [int(i) for i in roi]
195         plate = image_rgb[roi[0] : roi[2], roi[1] : roi[3]]
196         prediction = getLicenceAndProb(plate)
197         if prediction[1] < 0.90:
198             plate_mod = padLicence(plate)
199             prediction = getLicenceAndProb(plate_mod)
200         predicted_licences.append([image_name] + prediction)
201     tend = time.time()
202     recognition_time = tend - tstart
203
204     processing_times.append([image_name, preproc_time, detection_time, recognition_time])
205
206 df_times = pd.DataFrame(processing_times, columns = ['image', 'preprocessing_time', 'detection_time', 'recognition_time'])
207 df_pred = pd.DataFrame(predicted_licences, columns = ['image', 'licence_number', 'pred_score'])
208
209 df_times.to_csv(PATH_TO_CSV_TIMES)
210 df_pred.to_csv(PATH_TO_CSV_PREDICTIONS)
211
212 print('Test finished. Setting time was: {} seconds.'.format(setting_time))

```

ANEXO 3: FICHA PARA RECOLECCIÓN DE IMÁGENES

FICHA PARA RECOLECCIÓN DE IMÁGENES

FOTOGRAFÍAS PARA ENTRENAMIENTO Y TEST DE PLACAS VEHICULARES

- **Equipamiento a emplear:** Para la toma de fotografías se recomienda el empleo de los siguientes equipos:
 - Cámara digital de resolución mínima de 1MP.
 - Cámara de celular smartphone, con resolución mínima de 1MP.
- **Criterios para la recolección de información fotográfica:** Para el recojo de la de información fotográfica, se recomienda la aplicación de los siguientes criterios:
 1. Se tomará la fotografía de placas de vehículos ubicados en las vías públicas del departamento de Lima. No se puede tomar fotografías a placas de rodaje de vehículos cuyo acceso se encuentre dentro de propiedad privada.
 2. Por cada vehículo encontrado en las vías públicas, se considerará un mínimo de una (1) fotografía y máximo quince (15) fotografías.
 3. Cuando se tome más de una fotografía a una misma placa vehicular, deberá considerarse que cada fotografía incorpore información adicional, de tal manera que no se tenga la misma imagen repetida múltiples veces. Esto implica: cambiar la inclinación de la cámara empleada, ajustar el enfoque, cambiar la distancia desde la cámara hacia la placa, cambiar el ángulo de captura, cambiar la hora del día en el que se toman las fotografías, entre otros aspectos. Se debe evitar tener dos fotografías iguales, en la medida de que ello podría generar que se dé el fenómeno de *overfitting*.
 4. Las imágenes obtenidas deberán almacenarse en formato "jpg".
 5. La codificación de las imágenes obtenidas seguirá el patrón de 05 caracteres, que conforman números decimales que inician en 00001.
 6. Para la conformación del dataset, deberá verificarse que todas las fotografías tengan de lado como máximo 1024 píxeles. Si se tienen imágenes de mayor tamaño, deberá cambiarse el tamaño para que cumpla con lo señalado previamente.

Elaborado por: Bach. Luis Angel Bastidas Cerazo DNI 48156130	Validado por: Mg. Carlos Alberto Sotelo López DNI 07017259
	