UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE INGENIERÍA EN SISTEMAS, ELECTRÓNICA E INDUSTRIAL

POSGRADO

PROGRAMA DE MAESTRÍA EN MATEMÁTICA APLICADA

PROYECTO DE DESARROLLO

Trabajo de titulación previo a la obtención del grado académico de Magister en Matemática Aplicada

Tema:

"IMPLEMENTACIÓN DE UN SISTEMA INTELIGENTE PARA LA IDENTIFICACIÓN VEHICULAR"

Autor: Ing. Paúl Alejandro Cáceres Mayorga

Directora: Ing. CRISTINA ISABEL REINOSO ASTUDILLO, PhD.

Ambato – Ecuador

2021

APROBACIÓN DEL TRABAJO DE TITULACIÓN

A la Unidad de Titulación/ Unidad Académica de Titulación de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial. El Tribunal receptor de la Defensa del Trabajo de Titulación presidido por la Ing. Elsa Pilar Urrutia, Mg. e integrado por los señores: Ing. Clara Augusta Sánchez Benítez Mg. e Ing. Víctor Santiago Manzano Villafuerte Mg. Designados por la Unidad Académica de Titulación de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial de la Universidad Técnica de Ambato, para receptar el trabajo de Titulación con el tema: "Implementación de un sistema inteligente para la identificación vehicular", elaborado y presentado por el Ing. Paúl Alejandro Cáceres Mayorga, para optar el Grado Académico de Magister en Matemática Aplicada; una vez escuchada la defensa oral del Trabajo de Titulación el Tribunal aprueba y remite el trabajo para uso y custodia en las bibliotecas de la Universidad Técnica de Ambato.

Ing. Elsa Pilar Urrutia Urrutia, Mg.

Presidente y Miembro del Tribunal de Defensa

Ing. Clara Augusta Sánchez Benítez Mg.

Miembro del Tribunal de Defensa

Ing. Víctor Santiago Manzano Villafuerte Mg.

Miembro del Tribunal de Defensa

AUTORÍA DEL TRABAJO DE TITULACIÓN

La responsabilidad de las opiniones, comentarios y críticas emitidas en el trabajo de titulación presentado con el tema: “Implementación de un sistema inteligente para la identificación vehicular”, le corresponde exclusivamente a: Ing. Paúl Alejandro Cáceres Mayorga, autor bajo la dirección de: Ing. Cristina Isabel Reinoso Astudillo, PhD, directora del trabajo de investigación; y el patrimonio intelectual pertenece a la Universidad Técnica de Ambato.

PAUL ALEJANDRO CACERES MAYORGA

Firmado digitalmente por PAUL ALEJANDRO CACERES MAYORGA
Fecha: 2021.08.13 10:01:16 -05'00'

Ing. Paúl Alejandro Cáceres Mayorga

AUTOR

Firmado electrónicamente por:
CRISTINA ISABEL REINOSO ASTUDILLO

Ing. Cristina Isabel Reinoso Astudillo, PhD

DIRECTORA

DERECHOS DE AUTOR

Autorizo a la Universidad Técnica de Ambato, para que el trabajo de titulación, sirva como un documento disponible para su lectura, consulta y procesos de investigación, según las normas de la Institución.

Cedo mis derechos de mi trabajo de titulación, con fines de difusión pública, además apruebo la reproducción de este, dentro de las regulaciones de la Universidad Técnica de Ambato.

PAUL ALEJANDRO CACERES MAYORGA

Firmado digitalmente por PAUL ALEJANDRO CACERES MAYORGA
Fecha: 2021.08.13 10:01:29 -05'00'

Paúl Alejandro Cáceres Mayorga

C.I.: 180296674-5

ÍNDICE GENERAL

Contenido Pág.

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

AGRADECIMIENTO

No serán suficientes las palabras que escribo para expresar mi agradecimiento a todos mis familiares, mis padres Vinicio y Carmen, mi hermana Vivian, mis tíos Myriam y Hugo Oswaldo, mi mujer Pamela y mis Hijos Dorian y Agnes.

DEDICATORIA

El trabajo realizado es fruto del esfuerzo de toda mi familia, que a lo largo de este tiempo me ha apoyado e inspirado, a mi papi Manuel Vinicio que con su empeño y consejos me ha alentado a seguir adelante, a mi mami Carmen Elizabeth que con su esfuerzo y dedicación me inspiro a salir a delante, a mi hermana Vivian Anahí que con su ayuda y predisposición hizo esto posible, a mi Tía Myriam Yolanda que con su preocupación y atenciones me alentó a continuar con el proceso.

A mi mujer Ángela Pamela que con su ayuda incondicional y sabiduría me alentó y comprendió en momentos difíciles al desarrollar este proyecto, a mis hijos Dorian Stephano y Agnes Sybil por ser la inspiración para mi superación personal.

UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE SISTEMAS, ELECTRÓNICA E INDUSTRIAL

CENTRO DE POSGRADOS

MAESTRÍA EN MATEMÁTICA APLICADA

TEMA:

IMPLEMENTACIÓN DE UN SISTEMA INTELIGENTE PARA LA IDENTIFICACIÓN VEHICULAR

AUTOR: Ingeniero Paúl Alejandro Cáceres Mayorga

DIRECTOR: Ing. Cristina Isabel Reinoso Astudillo, Mg, PhD.

LÍNEA DE INVESTIGACIÓN: Tecnología de la Información y Sistemas de Control

FECHA: 07 de julio del 2021

RESUMEN EJECUTIVO

El presente trabajo de investigación tuvo como objetivo principal implementar un sistema inteligente capaz de clasificar placas vehiculares y automotores, así como, autocorregir errores de reconocimiento, para lo cual se partió del diseño de un algoritmo capaz de detectar y clasificar a los vehículos, implementando inteligencia artificial. Una vez identificado el proceso a seguir se implementó un código fuente para la detección de los tipos de placas vehiculares utilizando la red neuronal convolucional WPOD en la cual se especificaron los datos del borde, ancho y alto de la placa para que proporcione solamente la foto de la placa vehicular. Para el proceso de binarización utilizado en la investigación se empleó el algoritmo Otsu, que permitió hacer la conversión de las imágenes en las escalas de gris, blur, binaria y dilatación, aplicando filtros que permitan obtener los segmentos de ubicación de las letras y números. Finalmente se obtuvo un sistema efectivo, con capacidad de detección aceptable, pues se establecieron parámetros de diseño de la arquitectura de cada tipo de red, lo cual permitió una solución satisfactoria al problema de identificación, clasificación y validación de caracteres de las placas vehiculares ecuatorianas.

PALABRAS CLAVES: Reconocimiento, Placa vehicular, Inteligencia artificial, Redes neuronales, Binarización.

UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE SISTEMAS, ELECTRÓNICA E INDUSTRIAL

CENTRO DE POSGRADOS

MAESTRÍA EN MATEMÁTICA APLICADA

TEMA:

IMPLEMENTACIÓN DE UN SISTEMA INTELIGENTE PARA LA IDENTIFICACIÓN VEHICULAR

AUTOR: Ingeniero Paúl Alejandro Cáceres Mayorga

DIRECTOR: Ing. Cristina Isabel Reinoso Astudillo, Mg, PhD.

LÍNEA DE INVESTIGACIÓN: Tecnología de la Información y Sistemas de Control

FECHA: 07 de julio del 2021

ABSTRACT

The main objective of this research work was to implement an intelligent system capable of classifying vehicle and automotive license plates, as well as self-correcting recognition errors, for which it was based on the design of an algorithm capable of detecting and classifying vehicles, implementing artificial intelligence. Once the process to be followed was identified, a source code was implemented for the detection of the types of license plates using the convolutional neural network WPOD in which the data of the edge, width and height of the plate were specified so that it only provides the photo of license plate. For the binarization process used in the research, the Otsu algorithm was used, which converts the images into the gray, blur, binary and dilation scales, applying filters that can obtain the location segments of the letters and numbers. Finally, an effective system was obtained, with acceptable detection capacity, since design parameters of the architecture of each type of red were established, which achieved a satisfactory solution to the problem of identification, classification and validation of the characters of Ecuadorian license plates.

KEYWORDS: Recognition, License plate, Artificial intelligence, Neural networks, Binarization.

CAPÍTULO I

EL PROBLEMA DE INVESTIGACIÓN

1. Introducción

Durante las últimas décadas, el aumento de la población en todo el mundo ha provocado un incremento similar en la complejidad de los medios de transporte, en particular vehículos como automóviles y camiones. De esta forma, la gran cantidad de vehículos en carreteras y autopistas se está convirtiendo en un problema muy difícil de resolver, por lo cual se requiere automatizar una serie de tareas, en particular la identificación de vehículos. Es por ello que con el fin de lograr un control eficaz se están utilizando los sistemas automáticos de identificación de placas de vehículos [**1**].

El marco de reconocimiento de placas vehiculares ha surgido como una metodología central para garantizar las aplicaciones de tráfico y la seguridad que van desde el monitoreo de acceso al estacionamiento hasta la vigilancia de vehículos, cobro automático de peajes, monitoreo de tráfico vial, aplicación de la ley vehicular, cálculo del volumen de tráfico, actividad de vehículos análisis, seguimiento por seguridad y persecuciones delictivas [**2**].

Sin embargo, el sistema de reconocimiento de placas vehiculares se lo realiza mediante procesos manuales, esto genera largos tiempos de espera para obtener los datos del propietario del automotor, lo que conlleva pérdida de tiempo en ocasiones de premura, como lo son, accidentes, secuestros, robos y demás escenarios de seguridad en los que el tiempo de respuesta juega un papel primordial, que puede decidir la vida o la muerte de una o varias personas. La importancia de mejorar el sistema de reconocimiento de placas vehiculares radica en automatizar el método de búsqueda y respuesta para obtener la información.

El trabajo de investigación se basó en el análisis de información para diseñar un

algoritmo capaz de detectar y clasificar a los vehículos, implementando inteligencia artificial para posteriormente validar y corregir los caracteres usando el método matemático de binarización OTSU y de esa forma poder evaluar el desempeño del sistema inteligente utilizando la métrica exactitud (accuracy) enfocada a la evaluación por caracteres o por placas completas**.**

La investigación se realizó en cuatro etapas, la primera se enfocó en el planteamiento del problema, la justificación y los objetivos del trabajo, en la segunda se realizó una búsqueda bibliográfica en textos, artículos y otras fuentes confiables acerca de los antecedentes investigativos, en la tercera se planteó el marco metodológico de la investigación, en la cuarta etapa se expusieron los principales resultados obtenidos y la discusión y finalmente se plantearon las conclusiones y recomendaciones.

Una de las principales limitaciones del estudio fue la escasa información acerca de la metodología empleada el diseño del sistema inteligente capaz de clasificar placas vehiculares y automotores, así como, autocorregir errores de reconocimiento.

1.1. Justificación

El desarrollo de la presente investigación tiene un enfoque novedoso debido a que pretende resolver el problema de Reconocimiento Óptico de Caracteres utilizando sistemas inteligentes en el contexto de la solución del problema de clasificación de placas vehiculares y automotores, así como la autocorrección de errores de reconocimiento.

La investigación tiene gran impacto social debido a que actualmente no se cuenta con sistemas de reconocimiento de placas vehiculares en el Ecuador que permitan aumentar la tasa de detección y recuperación de vehículos por medio de aplicaciones inteligentes, basadas en redes neuronales [**3**], constituyéndose en una de las principales

necesidades existentes para el control de los vehículos de manera rápida y eficaz, motivo por el cual la automatización de este proceso con facilidad de configuración y programación atraen la atención de investigadores en el área de procesamiento y análisis digital de imágenes.

De igual manera, con este trabajo se pretende mejorar la funcionalidad del sistema de reconocimiento de placas vehiculares para disminuir radicalmente el tiempo para obtener la información del vehículo (nombres del propietario, cédula de identidad, número de teléfono, lugar de residencia, sexo), evitando el conflicto de intereses al reportar automáticamente los datos del infractor en el caso de multas o delitos, además de las molestias a los usuarios del sistema.

Al mejorar el sistema de reconocimiento de placas vehiculares se beneficiará al público en general permitiendo la inserción del sistema en la era tecnológica para el futuro desarrollo de aplicaciones en entes de regulación y control familiarizados con la seguridad vehicular, disminuyendo el tiempo de acceso a los datos registrados del automotor, además la liberación de la dependencia manual del reporte de datos.

El proyecto de investigación presenta gran originalidad ya que al aplicar Machine Learning (Redes Neuronales) permite el reconocimiento de imágenes, convirtiendo la imagen a caracteres, los cuales mediante un interfaz ingresan automáticamente a una base de datos, la cual retorna la información en un corto periodo de tiempo.

1.2. Objetivos

1.2.1. General

Implementar un sistema inteligente capaz de clasificar placas vehiculares y automotores, así como, autocorregir errores de reconocimiento.

1.2.2. Específicos

a. Diseñar un algoritmo capaz de detectar y clasificar a los vehículos, implementando inteligencia artificial.

b. Implementar un código fuente para la detección de los tipos de placas vehiculares.

c. Validar y corregir los caracteres usando el método matemático de binarización OTSU.

d. Evaluar el desempeño del sistema inteligente utilizando la métrica exactitud (accuracy) enfocada a la evaluación por caracteres o por placas completas.

CAPITULO II

ANTECEDENTES INVESTIGATIVOS

En las últimas dos décadas, se han desarrollado sistemas de transporte inteligentes para mejorar la seguridad y la movilidad del transporte público mediante la integración de múltiples tecnologías avanzadas. La identificación automática de vehículos se ha vuelto cada vez más importante en muchas aplicaciones; por ejemplo, tarifas de estacionamiento y pago de peajes, vigilancia del tráfico, emisión de boletos, control de acceso, entre otras. Una placa de matrícula es una característica única por la cual identificar cada vehículo individual. El reconocimiento automático de matrículas de vehículos, como área de investigación ya se ha estudiado ampliamente durante algunas décadas [**4**].

Es por ello que aunque se ha logrado un progreso significativo en las técnicas de reconocimiento de placas en la última década, todavía es una tarea desafiante reconocer las placas de matrícula a partir de imágenes complejas, pues un sistema robusto debería funcionar eficazmente en una variedad de condiciones, como día soleado, noche o con diferentes colores y fondos complejos [**5**].

Para la detección de placa, se han generado varias propuestas a cerca de esta temática, tales como el sistema desarrollado por Dueñas y Stemmer [**6**] en el año 2017, el cual presentó una propuesta para la detección y reconocimiento automático de placas vehiculares, considerando factores como cambios de iluminación de imágenes, ruido, sombras, y diversas perspectivas de adquisición. Para la detección de la placa se aplicó un filtro basado en operaciones morfológicas matemáticas, seguido de operaciones morfológicas-geométricas y un clasificador SVM-HOG y posteriormente se segmentaron los caracteres por medio de un algoritmo de clasificación de calidad de imágenes de placas y finalmente se utilizó dos calificadores SVM-HOG para reconocimiento de caracteres. Los resultados obtenidos mostraron un acierto global de 95,65% en la etapa de detección de la placa y el 93% en la etapa de reconocimiento de caracteres, por lo cual se determinó la

viabilidad para ser implementado en diversas aplicaciones de sistemas inteligentes de transporte.

En la investigación efectuada por Elbamby y Elsayed [**7**] se presentó un algoritmo novedoso para la detección automática de placas de matrícula de varios estilos en tiempo real en videos. El algoritmo propuesto pudo detectar en tiempo real múltiples placas de licencia con varios tamaños en un entorno desconocido y complejo. En este sistema, las regiones de placa se extraen utilizando una función de preprocesamiento para aumentar la precisión mientras se reduce el tiempo de cálculo. Luego, se usó un árbol de clasificadores en cascada basados en LBP para clasificar las regiones de placa candidatas en uno de los estilos aprendidos. El enfoque propuesto se aplicó a placas de matrícula egipcias con cuatro estilos de placa diferentes. El enfoque propuesto logró una tasa de éxito del 94% a 25 cuadros / seg utilizando una computadora portátil moderada.

El trabajo desarrollado por Barcia [**8**] se enfoca en la aplicación de las técnicas de visión artificial, adaptando el reconocimiento de matrículas de los vehículos del Ecuador para detectar el tipo de vehículo, provincia en la que se generó la placa y el tipo de servicio al que pertenece, teniendo en consideración la ubicación de los caracteres de la placa de acuerdo a las normativas de tránsito vigente. El algoritmo propuesto se adapta a la cantidad y forma de caracteres utilizando el software Matlab y procesos de visión artificial reconociendo las imágenes a través de la media estadística de correlación. La investigación se delimitó a las placas de matrículas de Ecuador u otro país que tenga 6 o 7 caracteres, a pesar que el algoritmo puede ser modificado para otros números de caracteres alfanuméricos. De igual forma se consideró que el color de la placa es blanco, con letras y números en negro con tamaño de 40.4 cm de alto por 15.4 cm en base a la ley y reglamento de tránsito.

Al abordar diversos estudios relacionados al reconocimiento de placas de vehículos utilizando técnicas de inteligencia artificial, especialmente con los sistemas de visión

por computadora, se ha apreciado que estas requieren una cantidad considerable de tiempo para procesar imágenes de alta resolución. Para dar solución a este problema, Giannoukos et al., [**9**] planteó un método de escaneo novedoso denominado "*Operator Context Scanning*" (OCS), que utiliza operadores de píxeles en forma de ventana deslizante, asociando un píxel y su vecino a la posibilidad de pertenecer al objeto que el método está buscando, con lo cual se pudo determinar que este algoritmo acelera el proceso de escaneo de imágenes y escala bien cuando se trata de imágenes de mayor resolución.

Con ello se añade que el análisis de ventana deslizante es una técnica de localización de patrones aplicada en el campo del procesamiento de señales. Varias técnicas exitosas de reconocimiento/ detección de objetos se basan en algoritmos de ventana deslizante, como detección de bordes, segmentación y binarización de texto, detección de rostros, detección humana, autenticación de imágenes, textura segmentación e indexación de vídeo. Sin embargo, estos enfoques aumentan el costo computacional, ya que el operador de reconocimiento debe aplicarse de manera exhaustiva en un gran espacio de píxeles de búsqueda de una imagen de entrada [**9**], así lo demuestra Hendry & Chen [**10**], cuyo detector de placa vehiculares funcionó mediante la ventana deslizante para todas las clases en cada imagen para evitar problemas de detección de los objetos pequeños. Los experimentos demostraron que la pérdida promedio del modelo es 0,40, con una velocidad promedio de detección y reconocimiento de 825,81 ms. El sistema no aplicó segmentación de caracteres ni reducción de ruido y constó de un proceso de detección y reconocimiento de una sola fase, alcanzando un 98,22% de precisión en la detección de matrículas y un 78% de precisión en el reconocimiento de matrículas.

Otro de los métodos eficaces para mejorar la calidad y velocidad de reconocimiento de placas vehiculares es la detección por regiones de interés, que consiste en evaluar inicialmente las características específicas de la zona que se desea extraer, para posteriormente seleccionar el método de segmentación pertinente. El área de interés seleccionado se debe separar del resto de la imagen, lo cual representa que se debe

identificar regiones de propiedades constantes y discontinuadas a través de la segmentación, considerando que en una imagen segmentada el elemento principal no es el pixel, sino la unión de éstos.

De esta forma Perdomo et al., [**11**] mencionan que esta técnica permite realizar mediciones de las diferentes regiones y obtener relaciones entre las adyacentes, por lo cual es muy utilizada en la interpretación cuantitativa de los datos de una imagen. Además, dentro del proceso de segmentación, donde se extrae de la imagen mejorada sólo las áreas que son de interés para analizarla, se realiza la extracción de características esenciales de los caracteres a recocer, los mismos que posteriormente son procesados por el algoritmo de Inteligencia Artificial [**12**].

De igual manera Chun, et al., [**13**] indican que la binarización de imágenes, que es un proceso para convertir una imagen en blanco y negro, es un paso importante en el reconocimiento de matrículas. Entre los métodos de binarización propuestos, el método Otsu es el más famoso y comúnmente utilizado en un sistema de reconocimiento de matrículas, ya que es el más rápido y puede alcanzar una precisión de reconocimiento comparable. La principal desventaja del método Otsu es que es sensible al efecto de luminancia y al ruido, y esta propiedad no es práctica ya que la mayoría de las imágenes de vehículos se capturan en un entorno abierto. De acuerdo a ello, en la investigación se propuso un sistema para mejorar el rendimiento de la reorganización automática de matrículas en el entorno abierto. Este sistema utiliza un método de binarización inspirado en los principios de simetría. Los resultados experimentales mostraron que cuando el método tiene una complejidad temporal similar a la de Otsu, el método aplicado puede mejorar la tasa de reconocimiento hasta 1,30 veces mejor que Otsu.

Así también Hidayah et al., [**14**] en su investigación determinaron la implementación y precisión del Método Otsu para el reconocimiento de matrículas. El método utilizado en la investigación fue el método Otsu para extraer las características y la imagen de

la placa en una imagen binaria y KNN como método de clasificación de reconocimiento de cada carácter. El desarrollo del programa de reconocimiento de matrículas mediante el método Otsu y la clasificación de KNN siguió los pasos del reconocimiento de patrones, como entrada y detección, preprocesamiento, función de extracción, método Otsu binario, segmentación, método de clasificación KNN y posprocesamiento por calculando el nivel de precisión. El estudio mostró que este programa puede reconocer en un 82% a partir de 100 placas de prueba con un 93,75% de precisión en el reconocimiento de números y un 91,92% de precisión en el reconocimiento de letras.

Además, Lin et al., [**15**] indicaron que los sistemas de reconocimiento de matrículas generalmente incluyen preprocesamiento de imágenes, ubicación y corrección de matrículas, segmentación de caracteres y reconocimiento de caracteres, por lo cual diseñaron un sistema para mejorar algunos de los tres ítems anteriores. En primer lugar, la binarización de preprocesamiento de imágenes de la matrícula se llevó a cabo utilizando el algoritmo Otsu más adaptativo. En segundo lugar, según la morfología matemática, utilizaron el astuto operador de detección de bordes para realizar la ubicación de la matrícula. El algoritmo de transformación de perspectiva se utilizó para corregir el problema de inclinación de la placa de matrícula después del posicionamiento, y el método de proyección vertical se utilizó para segmentar los caracteres de la imagen de matrícula corregidos. A diferencia del algoritmo tradicional de coincidencia de plantillas para hacer coincidir los píxeles uno por uno, el algoritmo de distancia de Hausdorff se utilizó para hacer coincidir los caracteres de la placa de matrícula segmentada con los caracteres de la biblioteca de plantillas. Finalmente, los resultados de la simulación del software MATLAB mostraron que el sistema puede reconocer los caracteres de la matrícula de la imagen de la matrícula existente con éxito.

Yoon, et al. [**16**] indican que un marco de binarización multimétodo es un enfoque poderoso para imágenes de entrada ruidosas; sin embargo, la selección de métodos y parámetros de binarización en el marco de reconocimiento de placas no se ha analizado

previamente en profundidad, por tal motivo en su artículo revelaron una combinación de métodos y parámetros de binarización, a través de un análisis en profundidad del esquema de binarización de múltiples métodos para una mejor segmentación de caracteres, para lo cual llevaron a cabo una extensa evaluación cuantitativa que muestra una mejora significativa con respecto a los métodos de binarización de métodos convencionales. Para encontrar las mejores combinaciones, primero se compara seis métodos de binarización populares y sus parámetros correspondientes, luego se evaluaron las combinaciones de los métodos de binarización.

Selmi, et al. [**17**] en el año 2019 indicaron que la detección y el reconocimiento automático de matrículas (ALPR) se utiliza para hacer que los procesos de detección y reconocimiento sean más robustos y eficientes en entornos muy complicados, lo cual hace necesario que se realicen investigaciones más profundas debido a algunas limitaciones tales como: integridad de los sistemas de numeración de los países, diferentes colores, varios idiomas, varios tamaños y fuentes variadas. Para ello, los autores presentaron un marco automático para la detección y el reconocimiento de matrículas (LP) de escenas complejas basado en redes neuronales convolucionales de regiones de máscara utilizadas para la detección, segmentación y reconocimiento de las matrículas. De acuerdo a varios experimentos, la investigación mostró la robustez y eficiencia del sistema sugerido pues logra una tasa de precisión del 99,3% en AOLP y del 98,9% en el conjunto de datos de Caltech.

De igual manera en el trabajo desarrollado por Soghadi et al. [**18**] se propuso un sistema de red neuronal convolucional profunda de extremo a extremo para el reconocimiento de matrículas que no se limita a una región o país específico. Aplicaron una versión modificada de YOLO v2 para reconocer primero el vehículo y luego ubicar la matrícula. Además, a través de los procedimientos convolucionales, se mejoró una red de reconocimiento óptico de caracteres (OCR-Net) para reconocer los números y letras de las matrículas. El trabajo estuvo compuesto de tres partes principales, a) detección de vehículos, b) detección de matrículas y c) reconocimiento óptico de caracteres. El primer paso es detectar vehículos en una imagen de entrada y

luego, en cada región de detección, utilizaron la Red de Detección de Objetos Planar Deformados (WPOD-NET) como una caja semi negra para la localización de matrículas que transforma las matrículas inclinadas y las rectifica a una forma rectangular como vistas frontales o traseras. Estas placas de matrícula mejoradas detectadas se introducen en una red OCR para la tarea de reconocimiento de caracteres.

En la investigación mencionada se eligieron tres conjuntos de datos para la evaluación y prueba, de los cuales dos de ellos están disponibles en línea, en detalle Open-ALPR (BR, EU, US) consta de 115 imágenes brasileñas, 108 europeas y 222 norteamericanas, que cubren muchas situaciones diferentes, UFPR-ALPR que incluye 4500 imágenes de 150 vehículos en movimiento, 1200 de imágenes son de sedán y autobuses con diferentes colores de fondo de matrícula (gris y rojo) y 300 de ellos son de motos. También evaluaron y probaron un conjunto de datos privado CENPARMI (Centro de Reconocimiento de Patrones e Inteligencia de Máquina) que incluye 440 imágenes generalmente vistas desde atrás y la mezcla de distancias oblicuas, ruidosas y diferentes de China, EE. Quebec. El sistema superó las imágenes de matrículas inclinadas y distorsionadas y funcionó adecuadamente en diversas condiciones de iluminación y fondos ruidosos, pues los resultados experimentales en 4,837 imágenes de vehículos estacionarios y en movimiento (automóviles, autobuses, motocicletas y camiones) de diferentes países muestran que el sistema propuesto logró tasas de reconocimiento entre 88.5% y 98.04%, superando el desempeño comercial de última generación [**18**].

De acuerdo investigaciones realizadas en Ecuador, Ortega y Torres [**19**] en el año 2020 desarrolló el trabajo titulado: “Reconocimiento automático de la placa de un vehículo de Ecuador”, para el procesamiento de las imágenes se realizó el redimensionamiento y binarización OTSU en las imágenes originales, con la finalidad de reducir la cantidad de información a procesar y agilitar el proceso de segmentación. Para la segmentación de imágenes se utilizaron detección de bordes, el algoritmo de componentes conectados y un algoritmo que permita identificar la ubicación de las letras y los números de la placa. Dicho reconocimiento se realizó a través de dos redes neuronales

sobre las secciones definidas como caracteres (una para la clasificación de letras y otra para clasificar los números) considerando las particularidades de las placas ecuatorianas. Las redes neuronales se configuraron usando apps correspondientes de MATLAB, y un código escrito en C#. Para la evaluación de la eficiencia de la clasificación se utilizaron 150 fotografías tomadas a una distancia aproximada de 2 a 3 metros del vehículo durante el día para tener iluminación natural, bajo estas condiciones se obtuvo una tasa de efectividad del 97%. Sin embargo, el modelo debería ser mejorado para responder de manera robusta a cambios como: tipos especiales de placa (cuerpo diplomático, consular, motos, etc.), imágenes tomadas a ángulos oblicuos con respecto a la superficie de la placa, iluminación variable (autos que ingresan a diferentes horas del día o en la noche), entre otros.

Para Wang et al., [**20**] la binarización de la imagen de la placa es una de las principales tareas del sistema de reconocimiento de matrículas (LPR). Muchos investigadores han establecido varios tipos de teorías para reducir los efectos de degradación en la placa de un automóvil. Sin embargo, la mayoría de estas teorías no son eficientes en la práctica debido a su elevado coste computacional. Por tal motivo el estudio se centró en la tarea de binarización de la iluminación deficiente en un sistema práctico de matrículas de automóviles comerciales. En el método propuesto utilizó un filtro de morfología matemática como filtro de paso bajo para mejorar la iluminación de la placa de matrícula. Para eliminar el impacto de la iluminación, se eligieron el filtro gaussiano y el filtro mediano para llenar la imagen de la placa, respectivamente. Por último, convierte la imagen en escala de grises a imagen binaria por el umbral local obtenido de la imagen convolucionada con coeficiente de compensación de iluminación.

Azam & Islam [**21**] en su investigación propusieron un nuevo método de detección automática de matrículas (ALPD) que detecta eficazmente el área LP a partir de una imagen en condiciones peligrosas. Para la eliminación de la lluvia, aplicaron un método novedoso que utiliza una máscara de dominio de frecuencia para filtrar las rayas de lluvia de una imagen. En el ALPD propuesto se introdujo un nuevo método de mejora del contraste con un enfoque de binarización estadística para manejar imágenes de bajos contrastes en interiores, nocturnos, borrosos y con niebla. Para

corregir LP inclinado, aplicaron por primera vez el método de corrección de inclinación basado en transformada de radón. Para filtrar regiones que no son LP, se usó una nueva condición que se basa en la entropía de la imagen. Dicho método se probó en 850 imágenes de automóviles que tienen diferentes condiciones peligrosas y lograron resultados satisfactorios en la detección de la placa.

En otra investigación se propuso una solución para la detección y el reconocimiento de matrículas de Bangladesh. En primer lugar, se determina la posición del vehículo considerando que las placas de matrícula comerciales tienen el color único (verde) propio. Es por eso que se seleccionó las porciones de color verde con la intensidad RGB correspondiente de la placa. Se propuso un algoritmo de contorno, área y relación de aspecto basados en límites para rastrear la placa de matrícula en la región del vehículo. Las filas de matrículas que contienen información de registro se separaron mediante proyección horizontal con umbral. Los caracteres y los dígitos de las filas se segmentaron mediante proyección vertical con valor umbral. Por último, se utilizó la coincidencia de plantillas para reconocer los caracteres y los dígitos de la matrícula en bengalí. Se realizaron las pruebas de los algoritmos para más de 180 imágenes fijas capturadas desde la carretera, con lo cual se obtuvo un 93% de éxito en la detección de matrículas, un 98,1% de éxito en la segmentación y una tasa de éxito del 88,8% en el reconocimiento de matrículas de Bangladesh [**22**].

Así también, Mundaca [**23**] desarrolló un sistema para la identificación de caracteres en las placas de automóviles sin importar la perspectiva con la que la cámara capture la imagen, implementado con algoritmos de visión artificial en Matlab; el sistema detecta la placa a través de la variación de gradientes, filtro Sobel y transformada de Bottom-Hat, corrige la perspectiva de captura de la placa localizada utilizando la transformada de Hough y transformada proyectiva bidimensional, y por último se realiza el reconocimiento de los caracteres haciendo uso del método de correlación de Pearson. El sistema implementado tuvo un costo computacional de 2,69 segundos, permitió localizar el 100% de la posición de la placa del automóvil, sin embargo, los autores mencionaron que los resultados del reconocimiento de los caracteres dependen

principalmente del estado en el que se encuentran las placas, y se encuentran sujetos a que la evaluación de la placa se efectúe en escenas estáticas y a una distancia considerable.

Barbecho y Zhindón [**3**] en el año 2020 realizaron una investigación con el tema: "Diseño de un algoritmo de reconocimiento de placas vehiculares ecuatorianas usando redes neuronales convolucionales" en la cual propusieron un modelo para detectar placas de automóviles en movimiento con la utilización de redes neuronales convolucionales, para lo cual, primero realizaron el etiquetado manual sobre las imágenes de los vehículos, identificando la ubicación de la placa y caracteres dentro de la imagen. Esta información se introdujo en una arquitectura de redes neuronales convolucionales para entrenamiento y pruebas. La arquitectura de la red neuronal se desarrolló en Google, COCO Inception V2, y se utilizó como base de entrenamiento, de donde se obtuvo un modelo propio entrenado para placas ecuatorianas. Una vez realizadas las pruebas con las 1000 fotos obtenidas desde el servidor lector de placas del Sistema Integrado de Seguridad ECU 911 Ecuador se muestra una precisión de reconocimiento favorable de 85.1% en términos del conjunto de datos de fotos de entrenamiento de placas ecuatorianas. Cabe mencionar que las placas con más fallas de detección son aquellas que son de color tomate, es decir de transporte publico ecuatoriano. Además, el modelo no responde de la mejor manera cuando se localiza dentro de la imagen más de una placa.

En referencia a lo indicado, la presente investigación se centrará en la implementación de un algoritmo basado en inteligencia artificial para el reconocimiento de placas vehiculares, haciendo énfasis en la etapa de corrección para mejorar el reconocimiento de caracteres. Como punto de partido se plantea definir la base de datos con la que se va a trabajar el proyecto, teniendo como alternativas (i) grande (ej. 10000 imágenes – ECU911) para entrenar una red neuronal convolucional - CNN, (ii) mediana (ej. 3000-5000 imágenes) para hacer sintonización de los pesos de una CNN y, (iii) pequeña (ej. 200-400 imágenes) para implementar una CNN ya entrenada. En cualquiera de los tres casos, la base de datos necesita ser etiquetada (imágenes etiquetadas), por lo que su

tamaño es necesario a tomar en cuenta.

Las etapas de las que consta el trabajo de investigación forman parte de un sistema completo de detección y reconocimiento de placas vehiculares, específicamente, enfocadas a placas ecuatorianas. En primera instancia se encontró la etapa de detección y clasificación de tipo de vehículo, que consistió en distinguir entre tipos de vehículos, por ejemplo: motocicleta, automóvil, camión o bus. Esta etapa se pudo realizar mediante la implementación de una de las Redes Neuronal Convolucionales (CNN) más utilizadas para la detección de objetos denominada YOLO. La segunda etapa fue la de detección de la placa, para lo cual se propone la utilización de la CNN WPOD-NET pre-entrenada que tiene un buen desempeño y robustez por su estructura y con ayuda de funciones de apoyo, puede detectar placas en escenarios complejos. Sin embargo, si se cuenta con una base de datos lo suficientemente grande, se podría implementar una CNN, para esto sería necesario el etiquetado (manual) de las imágenes y las zonas de interés (placas). La tercera etapa propuesta es la de segmentación para la cual se plantea la utilización de diversos algoritmos: conversión a escala de grises, filtrado (ej. Blur, median), umbralización (ej. OTSU) y morfología (ej. dilatación), sintonizando los parámetros de estos. Además, se propone utilizar algoritmos de detección de contornos o regiones de interés y seleccionar los caracteres con alguna relación (ej. Alto/ancho de los caracteres) y/o bounding boxes.

Debido a que se trabaja con una base de datos de imágenes, se plantea la utilización de algoritmos y métodos enfocados a estos, es decir filtros y morfología en 2D, así como redes neuronales convolucionales (CNN). Es por esto que el lenguaje de programación Python será considerado como la principal opción, ya que existen muchas librerías desarrolladas y ampliamente utilizadas por la comunidad del Machine Learning, Deep Learning y Computer Vision, aunque otra opción es utilizar la herramienta Matlab, que si bien puede ser bastante útil y su programación se encuentra en un nivel de dificultad moderado, puede presentar potenciales dificultades, como por ejemplo, la utilización de funciones privadas (derechos de autor, ya que no es libre como es el caso de python).

CAPITULO III

MARCO METODOLÓGICO

3.1. Ubicación

El proyecto se realizó en la ciudad de Ambato, la cual se encuentra ubicada en el centro del país en la provincia de Tungurahua, colinda hacia el norte con las provincias de Cotopaxi y Napo; al sur con las provincias de Chimborazo y Morona Santiago, al este con Pastaza y al Oeste con la provincia de Bolívar. Ambato cuenta con una superficie territorial de 1016,454 Km^2, que equivale al 29,94% de la extensión de la provincia de Tungurahua.

De acuerdo al censo poblacional del INEC (2010), el cantón Ambato cuenta con una población de 329,856 habitantes, de los cuales el 50,1% pertenecen al área urbana y el 49,9 a la población rural [**24**].

3.2. Equipos y materiales

Entre los materiales principales que se utilizaron en el desarrollo del proyecto se encuentran los siguientes:

- Computadores para el desarrollo del proyecto
- Software de reconocimiento de caracteres
- Cámara IP
- Cableado de alimentación y transmisión de datos

3.3. Tipo de investigación

El presente trabajo tuvo una modalidad cual-cuantitativa. Cualitativa debido a que se caracterizó el problema de investigación para de esa forma encontrar la mejor solución de implementación del sistema de reconocimiento de placas y así alcanzar el resultado esperado. De igual forma fue cuantitativa por cuanto se utilizaron los conocimientos técnicos para desarrollar una solución totalmente factible, empleando análisis, cálculos y pruebas adecuadas que ayudaron al diseño del sistema.

3.3.1. Investigación experimental

Se aplicó la investigación experimental debido a que se manipuló, evaluó y organizó las variables que intervienen en el estudio para definir las relaciones causa efecto entre ella. Además, se utilizó este método debido a que se realizaron las pruebas de funcionamiento respectivas para verificar su funcionamiento, efectuar los ajustes respectivos y corregir alguna falla en caso de existirla.

3.3.2. Investigación Descriptiva

De igual forma se aplicó la investigación descriptiva para definir las características relevantes de las variables de estudio, sin alterar ninguna de estas, en base a los conocimientos adquiridos y su puesta en práctica.

3.4. Prueba de Hipótesis - pregunta científica – idea a defender

La implementación de un algoritmo basado en inteligencia artificial para validar y corregir los caracteres usando el método matemático de binarización OTSU en las imágenes originales permitirá autocorregir errores de reconocimiento y de esa forma clasificar las placas vehiculares y automotores en un corto periodo de tiempo.

3.5. **Población o muestra**:

No aplica

3.6. **Recolección de información:**

La información fue recolectada de fuentes como libros, revistas, a artículos científicos, páginas web, entre otras, que contenían relación con a los sistemas de reconocimiento de placas vehiculares y acerca de la inteligencia artificial, además de la aplicación de métodos para mejor la velocidad de reconocimiento como la binarización OTSU para ampliar los conocimientos sobre el tema de estudio y de esta forma tener una idea clara acerca de los parámetros que debe cumplir el sistema que se requiere implementar para su óptima utilización a beneficio de la población en general.

3.7. **Procesamiento de la información y análisis estadístico:**

Para el procesamiento y análisis de datos se realizaron las siguientes actividades:

- Recolección de información de fuentes como los libros, artículos, tesis, papers, entre otras.
- Análisis crítico de la información para determinar el método adecuado de solución.
- Interpretación de resultados.

Con la información obtenida, el proyecto de investigación se desarrolló de la siguiente manera:

- Investigar y recolectar información acerca de los sistemas inteligentes para la identificación vehicular.
- Recopilar información acerca del método matemático de binarización OTSU.
- Explorar los parámetros técnicos del software de reconocimiento e

identificación de placas vehiculares.

- Definir los requerimientos tanto de software como de hardware que se necesita para el diseño de la fase de entrenamiento y ejecución del sistema inteligente.
- Seleccionar el algoritmo adecuado para la validación y corrección de los caracteres usando el método matemático de binarización OTSU.
- Seleccionar los componentes que van a formar parte del sistema de reconocimiento de placas.
- Implementar el sistema inteligente para la identificación vehicular
- Evaluar el desempeño del sistema inteligente de reconocimiento de placas vehiculares utilizando la métrica exactitud (accuracy).
- Corregir los errores detectados para evitar problemas mayores del sistema
- Pruebas finales de funcionamiento del sistema.
- Documentación del proceso de desarrollo del sistema.

3.8. Variables respuesta o resultados alcanzados

Con el desarrollo de la investigación se obtuvo una aplicación que permita el reconocimiento de caracteres de una placa vehicular con la utilización de inteligencia artificial con binarización Otsu para el almacenamiento y obtención de datos del propietario de los vehículos, además de la categorización de vehículos en: auto, van, sub, camión, moto, trailer, entre otros.

CAPITULO IV

RESULTADOS Y DISCUSIÓN

4.1. Resultados

El desarrollo del presente proyecto se lo realizó en lenguaje de programación Python, utilizando la plataforma gratuita de Google Colaboratory – Colab. Cada etapa detallada en el proyecto, se trató por separado.

La entrada del sistema inteligente fue una imagen (que contenga algún vehículo). Debido a que el sistema se encuentra en la nube, se necesitó una cuenta de Google para la comunicación y ejecución. La salida del sistema inteligente fue la placa identificada (número de placa) y, si la placa se encuentra en la base de datos proporcionada se presentó adicionalmente información básica del vehículo.

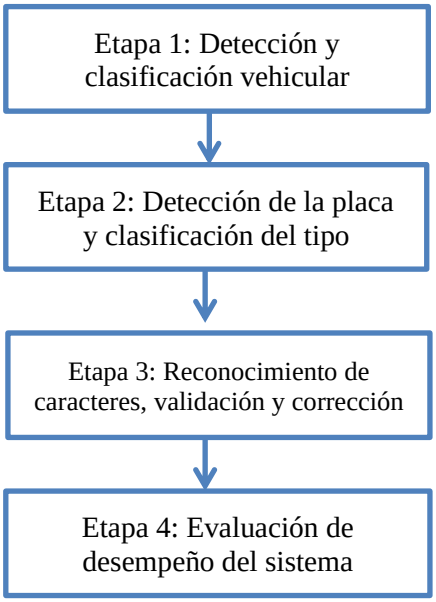

Figura 4-1: Etapas del sistema de reconocimiento de placas vehiculares

Todo el procesamiento que se encuentra conformado por 4 etapas como se muestra en la figura 1 se realizó utilizando los recursos que la plataforma Colab pone a disposición gratuitamente. De igual manera, como interfaz gráfica se utilizó la misma plataforma mediante los denominados “notebooks” interactivos de Jupyter.

4.4.1. Etapa 1. Detección y clasificación vehicular

Mediante una imagen se propuso realizar la detección de un vehículo (el que se encuentre en el plano principal) y posterior reconocimiento/clasificación de acuerdo con su tipo (automóvil o camión/bus). Para esto, se planteó la utilización de la red neuronal convolucional YOLO que permite mediante imágenes la detección de objetos con una alta efectividad. YOLO se encarga de codificar implícitamente la información contextual, modelar el tamaño y la forma de los objetos y su apariencia.

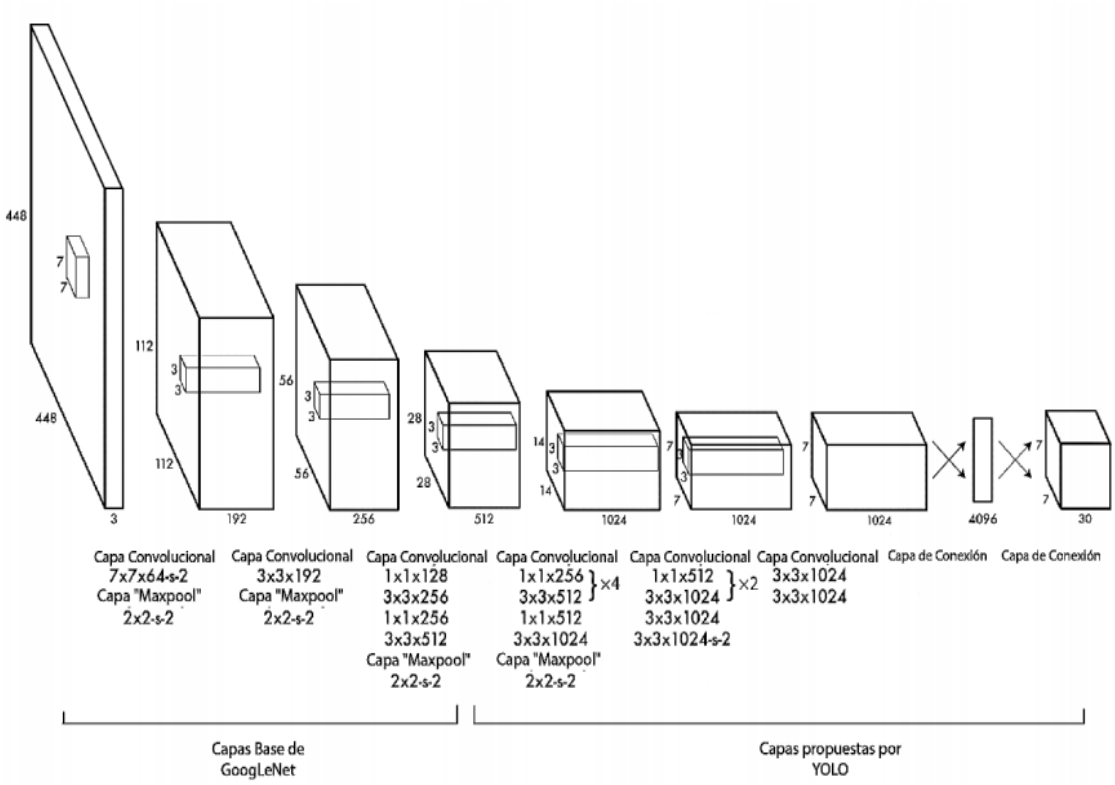

Figura 4-2: Arquitectura de la red neuronal YOLO **[**25**].**

En referencia a la arquitectura de la red neuronal que emplea YOLO se encuentra conformada por 24 capas convolucionales seguidas por dos capas completamente conectadas. Algunas de las capas convolucionales son de 1x1 con la finalidad de reducir la profundidad [**25**].

El tiempo aproximado de desarrollo de esta etapa de detección y clasificación vehicular fue de 4 semanas.

Para iniciar con el desarrollo de la primera etapa del sistema se requiere conectar el

entorno de colab.notebook al entorno drive, en donde se encuentran los archivos necesarios para la ejecución del sistema, como lo son imágenes, base de datos, y los cuadernos de apuntes.

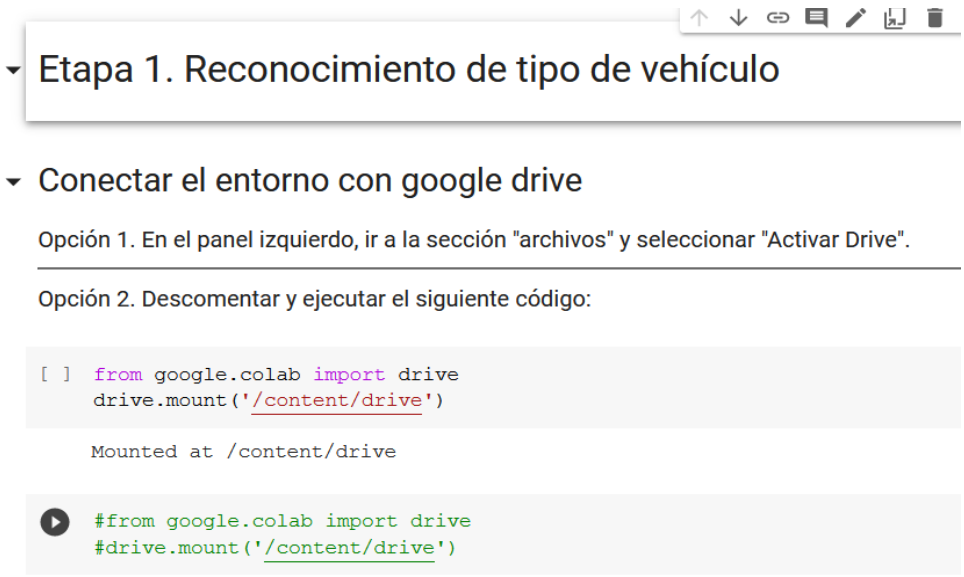

Figura 4-3. Reconocimiento de tipo de vehículo

Una vez realizada la conexión se debe importar desde Keras, que es la más grande librería en relación a machine learnning, la red neuronal que se va a utilizar para reconocer el tipo de vehículos.

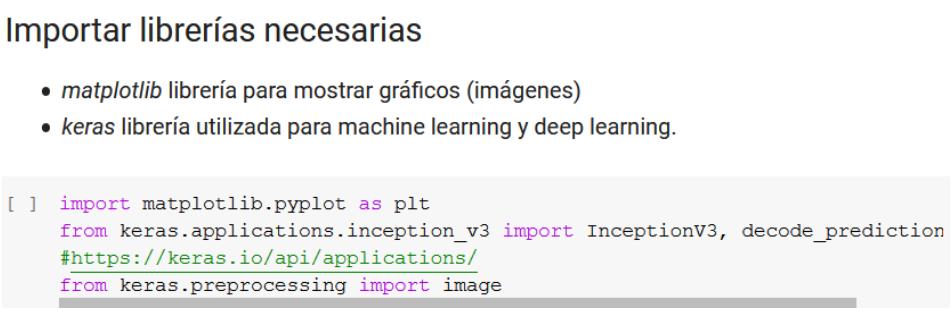

Figura 4-4: Importar librerías necesarias

keras.applications tiene disponible varias opciones para redes neuronales convolucionales:

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
Xception	88 MB	0.790	0.945	22,910,480	126
VGG16	528 MB	0.713	0.901	138,357,544	23
VGG19	549 MB	0.713	0.900	143,667,240	26
ResNet50	98 MB	0.749	0.921	25,636,712	-
ResNet101	171 MB	0.764	0.928	44,707,176	-
ResNet152	232 MB	0.766	0.931	60,419,944	-
ResNet50V2	98 MB	0.760	0.930	25,613,800	-
ResNet101V2	171 MB	0.772	0.938	44,675,560	-
ResNet152V2	232 MB	0.780	0.942	60,380,648	-
InceptionV3	92 MB	0.779	0.937	23,851,784	159
InceptionResNetV2	215 MB	0.803	0.953	55,873,736	572
MobileNet	16 MB	0.704	0.895	4,253,864	88
MobileNetV2	14 MB	0.713	0.901	3,538,984	88
DenseNet121	33 MB	0.750	0.923	8,062,504	121
DenseNet169	57 MB	0.762	0.932	14,307,880	169
DenseNet201	80 MB	0.773	0.936	20,242,984	201
NASNetMobile	23 MB	0.744	0.919	5,326,716	-
NASNetLarge	343 MB	0.825	0.960	88,949,818	-

Figura 4-5: Librerías de Keras

Posterior a ello se realizaron pruebas con los diferentes modelos de redes neuronales para determinar la más óptima para cumplir con los requerimientos del trabajo de investigación.

Figura 4-6: Ejecución utilizando InceptionV3

En esta etapa se realiza la lectura de la imagen y se utiliza el pre-procesamiento necesario para el ingreso a la red neuronal. Además se realiza la evaluación de la imagen y se predice la clase.

```
# Main Path
dbpath = "/content/drive/MyDrive/LPDR/DB/"

# Image name
img_name = "Camioneta1.jpeg" #PCR-9699

# Loading image (PIL format)
img_org = image.load_img(dbpath+img_name, target_size=(299,299))  # this i
print("Image size: "+str(img_org.size))

x = image.img_to_array(img_org)
plt.imshow(img_org)

# Mapping (new range) 0 - 255 => -1 - 1
x /= 255
x -= 0.5
x *= 2

# Adding a dimension (input format to CNN)
x = x.reshape([1, x.shape[0], x.shape[1], x.shape[2]])
print(x.shape)

# Method to evaluate the image (predict)
y = iv3.predict(x)
print(y.shape)  # CNN Output: 1000 classes
```

Figura 4-7: Lectura de imagen

Luego se realiza el procesamiento de la imagen del vehículo con la que se va a trabajar,

esto a través del mapeo de la imagen y el reescalamiento dependiendo de los parámetros de la Red Neuronal que se utiliza.

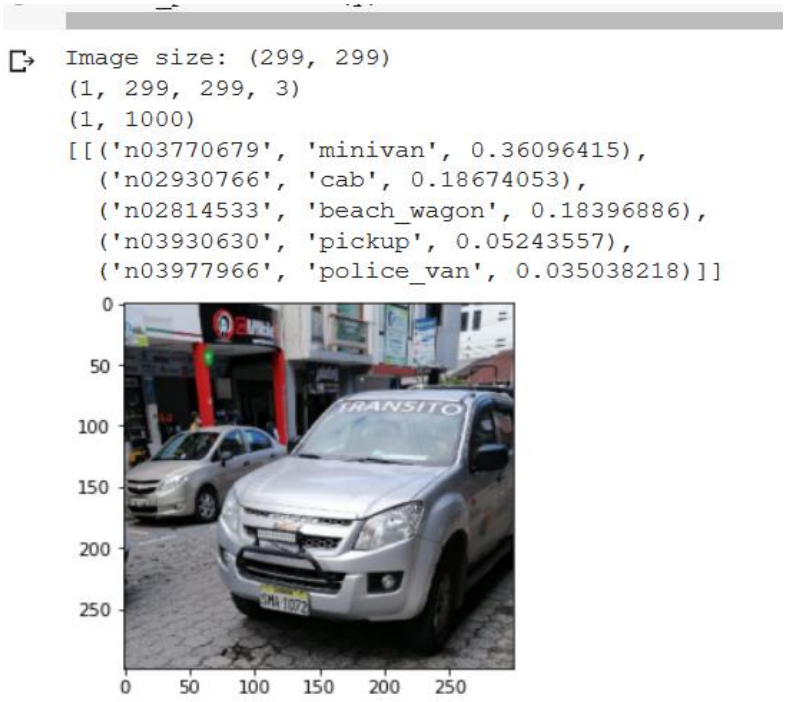

Figura 4-8: Procesamiento de la imagen del vehículo

Una vez procesada la imagen, la red neuronal la reconoce y proporciona los diferentes resultados. Después de realizar varias pruebas se determinó que la red neuronal NASNetLarge es la más adecuada para el sistema de reconocimiento vehicular, debido a su precisión en el reconocimiento de los objetos inmersos en ellas, además de ser ligera, lo cual ayuda al tiempo de procesamiento.

Ejecución utilizando NASNetLarge

```
[ ]  from keras.applications.nasnet import NASNetLarge, decode_predictions
     nnl = NASNetLarge();

     Downloading data from https://storage.googleapis.com/tensorflow/keras-appli
     359751680/359748576 [==============================] - 2s 0us/step

[ ]  # Loading image (PIL format)
     img_org = image.load_img(dbpath+img_name, target_size=(331,331))  # this i
     print("Image size: "+str(img_org.size))

     x = image.img_to_array(img_org)
     plt.imshow(img_org)

     # Mapping (new range) 0 - 255 => -1 - 1
     x /= 255
     x -= 0.5
     x *= 2

     # Adding a dimension (input format to CNN)
     x = x.reshape([1, x.shape[0], x.shape[1], x.shape[2]])
     print(x.shape)

     # Method to evaluate the image (predict)
     y = nnl.predict(x);

     # Method to show prediction information
```

Figura 4-9: Ejecución utilizando la RN NASNetLarge

Después de que el código se ha desarrollado se obtienen los posibles resultados del vehículo reconocido en la imagen.

```
Image size: (331, 331)
(1, 331, 331, 3)
[[('n03977966', 'police_van', 0.6471628), ('n03770679', 'minivan', 0.135377
```

Figura 4-10: Resultados del vehículo reconocido en la imagen

La RN NASNetLarge posee 1000 objetos en su base de datos que pueden ser reconocidos, gracias al preentrenamiento realizado previamente. Para llegar al objetivo

de reconocer el tipo de vehículo se debe realizar un re-etiquetado, en donde se disminuye el abanico de posibilidades a automóvil, camioneta/jeep, minibús/buseta, bus, motocicleta (si se desea aumentar los tipos de vehículos simplemente se requiere procesar esta información en esta etapa de re-etiquetado), es decir cuando la RN reconozca al vehículo como:

- 'cab', 'minivan', 'beach_wagon', 'grille', 'police_van', 'convertible', 'car_wheel', 'sports_car', 'limousine', 'racer', 'maze' ----- lo etiquetará como 'automóvil'
- 'pickup', 'jeep', 'golfcart', 'harvester', 'forklift'----- lo etiquetará como 'camioneta/jeep'
- 'minibus'---- lo etiquetará como 'minibus/buseta'
- 'moving_van', 'recreational_vehicle'----- lo etiquetará como 'bus'
- 'motor_scooter'----- lo etiquetará como 'motocileta'

```
import numpy as np

# Classes from ImageNET
all_lists = [['cab', 'minivan', 'beach_wagon', 'grille', 'police_van', 'cc
              ['pickup', 'jeep', 'golfcart', 'harvester', 'forklift'],
              ['minibus'],
              ['moving_van', 'recreational_vehicle'],
              ['motor_scooter']]

# My classes
all_classes = ['automóvil',
               'camioneta/jeep',
               'minibus/buseta',
               'bus',
               'motocileta']

# ismember function
def ismember(A, B):
  return np.sum([ np.sum(a == B) for a in A ])

# Re-labeling
result = cnn_output[0][0][1]

onehot = [];
for l in all_lists:
  onehot.append(ismember(l, result))
```

Figura 4-11: Re-etiquetado tipo de vehículo

En el caso de procesar una imagen que no sea un vehículo la etiqueta que aparece en el sistema será “No definido”.

```
final_class = ""
for i in range(len(onehot)):
  final_class = final_class + onehot[i]*all_classes[i]

if final_class == "":
  final_class = "No definido"

print("Se reconoció el vehículo: "+ final_class)
```

```
Se reconoció el vehículo: No definido
```

Figura 4-12: Etiqueta no definido

4.1.2. Etapa 2. Detección de la placa y clasificación del tipo

Esta etapa se podría considerar como la etapa de mayor interés, ya que no existen bases de datos para el reconocimiento de placas ecuatorianas (caracteres o fuente de las placas) e incluso se considera como uno de los principales aportes del proyecto.

Mediante una imagen se realizó la detección de la placa vehicular (objeto) y posterior clasificación de acuerdo con su tipo (vehículo particular, comercial o gubernamental). Para la detección se plantea la utilización de la red neuronal convolucional WPOD pre-entrenada, la cual se enlaza desde la librería Keras, y para la clasificación se plantea un análisis de color sobre la placa.

```
from os.path import splitext#, basename
from keras.models import model_from_json

def load_model(path):
    try:
        path = splitext(path)[0]
        with open('%s.json' % path, 'r') as json_file:
            model_json = json_file.read()
        model = model_from_json(model_json, custom_objects={})
        model.load_weights('%s.h5' % path)
        print("Loading model successfully...")
        return model
    except Exception as e:
        print(e)

path = '/content/drive/MyDrive/Colab Notebooks/'
wpod_net_path = path+"wpod-net"
wpod_net = load_model(wpod_net_path)
```

```
Loading model successfully...
```

Figura 4-13: Carga del modelo WPOD

Para realizar la lectura de la imagen y que la Red Neuronal pueda detectar la ubicación de la placa (si existe), es necesario preparar la imagen, utilizando reescalación.

```
[ ] import cv2
    import numpy as np
    import matplotlib.pyplot as plt
    from keras.preprocessing.image import load_img, img_to_array
    import matplotlib.gridspec as gridspec

    from importlib.machinery import SourceFileLoader
    utilsWPOD = SourceFileLoader("utilsWPOD", "/content/drive/MyDrive/Colab

    # forward image through model and return plate's image and coordinates
    # if error "No Licensese plate is founded!" pop up, try to adjust Dmin
    def get_plate(image_path, Dmax=608, Dmin=256):
        vehicle = preprocess_image(image_path)
        ratio = float(max(vehicle.shape[:2])) / min(vehicle.shape[:2])
        side = int(ratio * Dmin)
        bound_dim = min(side, Dmax)

        _ , LpImg, _, cor = utilsWPOD.detect_lp(wpod_net, vehicle, bound_din
        return vehicle, LpImg, cor

    def preprocess_image(image_path):
        img = cv2.imread(image_path) #BGR
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB) #RGB
        img = img / 255
```

Figura 4-14: Evaluación del modelo WOD

La Red Neuronal se la configura con datos del borde, ancho y alto de la placa, para que realice el recorte de la imagen y entregue únicamente la foto de la placa vehicular.

```
vehicle, LpImg, cor = get_plate(dbpath+img_name)

# Obtain plate image and its coordinates from an image
print("Coordinate of plate(s) in image: \n", cor)

fig = plt.figure(figsize=(12,6))
grid = gridspec.GridSpec(ncols=2,nrows=1,figure=fig)
fig.add_subplot(grid[0])
plt.axis(False)
plt.imshow(vehicle)
grid = gridspec.GridSpec(ncols=2,nrows=1,figure=fig)
fig.add_subplot(grid[1])
plt.axis(False)
plt.imshow(LpImg[0])
```

```
WARNING:tensorflow:5 out of the last 7 calls to <function Model.make_pre
Coordinate of plate(s) in image:
 [array([[240.6505708 , 335.88442892, 333.47955321, 238.24569508],
       [449.0196824 , 466.65659488, 511.44861089, 493.8116984 ],
       [  1.        ,   1.        ,   1.        ,   1.        ]])]
<matplotlib.image.AxesImage at 0x7f75c980bed0>
```

Figura 4-15: Configuración de la RN con datos del borde ancho y alto de la placa

Opción dos: Todo lo que se encuentra en la librería de apoyo.

Para determinar el tipo de vehículo ("comercial", "gubernamental", "gad", "particular") se debe analizar el color de la placa, para lo cual se adecua la imagen para trabajar en formato HSV.

- Hue – Este canal codifica la tonalidad del color: rojo, azul, amarillo, verde, entre otros. En OpenCV el valor del Hue está entre 0-179.
- Saturation – Este canal codifica la intensidad o pureza del color: cuanta menos saturación más grisácea será el color. En OpenCV el valor de Saturation oscila entre 0-255.
- Value – Este canal codifica la luminosidad: cuanto menor sea, más negro/oscuro será el color.

Figura 4-16: Análisis de color

Además, se configura la red con los colores naranja, amarillo, verde y blanco, al igual que en la etapa uno si se desea aumentar el tipo de vehículos se requiere configurar y añadir colores y su configuración de la paleta en esta etapa.

```
color_list = ["comercial", "gubernamental", "gad", "particular"]#, "dipl
color_ranges = {}
color_ranges[color_list[0]+"Low"] = np.array([0, 170, 20], np.uint8)
color_ranges[color_list[0]+"High"] = np.array([15, 255, 255], np.uint8)
color_ranges[color_list[1]+"Low"] = np.array([21, 170, 20], np.uint8)
color_ranges[color_list[1]+"High"] = np.array([33, 255, 255], np.uint8)
color_ranges[color_list[2]+"Low"] = np.array([40, 170, 20], np.uint8)
color_ranges[color_list[2]+"High"] = np.array([70, 255, 255], np.uint8)
color_ranges[color_list[3]+"Low"] = np.array([0, 0, 150], np.uint8)
color_ranges[color_list[3]+"High"] = np.array([178, 75, 255], np.uint8)
#color_ranges[color_list[4]+"Low"] = np.array([110, 170, 20], np.uint8)
#color_ranges[color_list[4]+"High"] = np.array([130, 255, 255], np.uint8

from google.colab.patches import cv2_imshow
paletColor = cv2.imread(dbpath+"HSV_2.png")
paletColor = cv2.cvtColor(paletColor, cv2.COLOR_BGR2HSV)

color = color_list[1]
mask = cv2.inRange(paletColor, color_ranges[color+"Low"], color_ranges[c
masked = cv2.bitwise_and(paletColor, paletColor, mask = mask)
cv2_imshow(cv2.cvtColor(masked, cv2.COLOR_HSV2BGR))
```

Figura 4-17: Sintonización de los rangos de colores

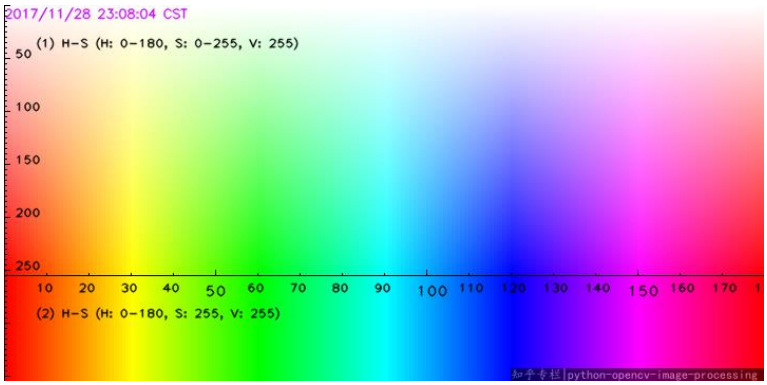

Figura 4-18: Rango de colores

Una vez configurada la sintonización de colores en la Red Neuonal se procede a configurar la imagen obtenida en el paso anterior para trabajar en el formato HSV, y así se devuelva a que tipo de vehículo pertenece el color de placa ingresado.

Figura 4-19: Obtención del tipo de vehículo al que pertenece el color de placa

Esta codificación se realiza siguiendo la configuración que se muestra en la siguiente tabla:

Tabla 4-1: Tipo de vehículo por color

Tipo	Color
Comercial	Naranja
Gubernamental	Amarillo
Gad	Verde
Particular	Blanco

4.1.3. Etapa 3. Reconocimiento de caracteres, validación y corrección

A partir de la segmentación realizada en la etapa anterior (Etapa 2) se llevó a cabo otra segmentación y el reconocimiento de caracteres. Posteriormente, este resultado se validó y corrigió debido a los errores que contenía. La etapa de validación y corrección de caracteres se propone en función de la estructura de las placas de Ecuador (3 letras y, 3 o 4 números), con el propósito de rectificar el posible reconocimiento incorrecto con caracteres similares entre sí, por ejemplo ‘0’ con ‘O’, ‘6’ con ‘G’, entre otros.

Para el preprocesamiento se plantea el uso de diversos algoritmos (por ejemplo: conversión a escala de grises, filtrado Blur o median, umbralización con Otsu y morfología. Para la segmentación se plantea el uso algoritmos de detección de contornos o regiones de interés, y para la detección de los caracteres segmentados se propone el uso de la red neuronal MobileNet.

Para la utilización del método OTSU en la segmentación de los caracteres se utilizó la siguiente línea de código:

binary=cv2.treshold(blur,180,255,
cv2.TRESHOLD_BINARY_INV+cv2.TRESHOLD_OTSU)

Traduciendo matemáticamente dicho código, se parte de una imagen de gris con N pixeles y L posibles niveles diferentes. De esta forma la probabilidad de ocurrencia del nivel de gris y en la imagen se representa como:

$$p_i = \frac{f_i}{N} \tag{1}$$

Donde

f_i= Frecuencia de repetición del nivel de gris-i-ésimo con i=1, 2,…, L.

En el caso particular de umbralización en dos niveles (binarización), los pixeles se dividen en dos clases: C_1 y C_2, con niveles de gris [1,2,…,t] y [t+1,t+2,…L] de manera respectiva, donde las distribuciones de probabilidad de las dos clases son:

$$C_1 : \frac{P_1}{\omega_1(t)}, \dots, \frac{P_t}{\omega_1(t)} \tag{2}$$

$$C_1 : \frac{P_{t+1}}{\omega_2(t)}, \frac{P_{t+2}}{\omega_2(t)} \dots, \frac{P_L}{\omega_2(t)} \tag{3}$$

Donde

$$\omega_1(t) = \sum_{i=1}^{t} p_i$$

$$\omega_2(t) = \sum_{i=t+1}^{L} p_i$$

Las medidas para cada una de las clases son definidas como:

$$\mu_1 = \sum_{i=1}^{t} \frac{i \cdot p_i}{\omega_1(t)} \tag{4}$$

$$\mu_2 = \sum_{i=t+1}^{L} \frac{i \cdot p_i}{\omega_2(t)} \tag{5}$$

La intensidad media total de la imagen se define, siendo fácil demostrar así mismo:

$$\omega_1 \cdot \mu_1 + \omega_2 \cdot \mu_2 = \mu_T \tag{6}$$

$$\omega_1 + \omega_2 = 1 \tag{7}$$

Utilizando un análisis discriminante, Otsu definió la varianza entre las clases de una imagen umbralizada como:

$$\sigma_B^2 = \omega_1 \cdot (\mu_1 - \mu_T)^2 + \omega_2 \cdot (\mu_2 - \mu_T)^2 \tag{8}$$

Luego se debe encontrar el umbral t, que maximice la varianza (Otsu demostró que este era el umbral óptimo):

$$t^* = Max\{\sigma_B^2(t)\} \tag{9}$$

t

Dónde:

$$1 \leq t \leq L$$

Para iniciar con la codificación de esta etapa se debe cargar la imagen del automóvil con el que se está trabajando para obtener el extracto de la placa vehicular.

```
[12] img_name = "Test.jpeg"; #SMA-1072 GSD-3674 GSR-3885
     vehicle, LpImg, cor = get_plate(dbpath+img_name);
     plate_image = LpImg[0];
```

Figura 4-20: Cargar la imagen del automóvil

Luego se procesa la imagen y se la convierte a 8 bits para poder trabajar con esta, las funciones que se utilizó en esta parte del código se encuentran en openCV que es una librería de código abierto creada por INTEL. Uno de los filtros más importantes para obtener la imagen convertida es el filtro OTSU.

Se realizó la conversión de las imágenes en las escalas de gris, blur, binaria y dilatación, para así poder aplicar filtros que permitan obtener los segmentos de ubicación de las letras y números.

```
[14] def morph(LpImg):
       # Scales, calculates absolute values, and converts the result to 8-bit.
       plate_image = cv2.convertScaleAbs(LpImg, alpha=(255.0))

       # convert to grayscale and blur the image
       gray = cv2.cvtColor(plate_image, cv2.COLOR_BGR2GRAY) #COLOR_BGR2GRAY
       blur = cv2.GaussianBlur(gray,(7,7),0)

       # Applied inversed thresh_binary
       binary = cv2.threshold(blur, 180, 255,
                              cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)[1]

       kernel3 = cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3)) #3, 5
       thre_mor = cv2.morphologyEx(binary, cv2.MORPH_DILATE, kernel3)
       return plate_image, gray, blur, binary, thre_mor

    pl_image, gray, blur, binary, thre_mor = morph(plate_image)

    # visualize results
    fig = plt.figure(figsize=(12,7))
    plt.rcParams.update({"font.size":18})
    grid = gridspec.GridSpec(ncols=2,nrows=3,figure = fig)
    plot_image = [pl_image, gray, blur, binary, thre_mor]
    plot_name = ["plate_image","gray","blur","binary","dilation"]

    for i in range(len(plot_image)):
        fig.add_subplot(grid[i])
        plt.axis(False)
        plt.title(plot_name[i])
        if i ==0:
            plt.imshow(plot_image[i])
        else:
            plt.imshow(plot_image[i],cmap="gray")
```

Figura 4-21: Conversión de las imágenes en las escalas de gris, blur, binaria y dilatación

Las imágenes pueden ser tomadas en el día o la noche para lo cual los filtros que se aplicarán en cada ocasión deben variar.

En el día se puede aplicar:

Figura 4-22: Filtros que se aplican en el día

Y en la noche se aplica los siguientes:

Figura 4-23: Filtros que se aplican en la noche

Como se puede observar en el día y la noche el nivel de iluminación varía, por lo cual en las fotos en la noche se aplicará un filtro adaptativo para variar la iluminación, lo que hace este filtro es invertir las tonalidades blanco a negro y negro a blanco.

```
# Applied inversed thresh_binary
if day:
  binary = cv2.threshold(blur, 180, 255, cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)[1]
else:
  binary = cv2.adaptiveThreshold(blur,255,cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY_I

kernel3 = cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3)) #3, 5
thre_mor = cv2.morphologyEx(binary, cv2.MORPH_DILATE, kernel3)
return plate_image, gray, blur, binary, thre_mor
```

Figura 4-24: Aplicación de un filtro adaptativo

La segmentación binaria es la adecuada para trabajar debido a que definen claramente

los componentes de la placa, en esta imagen se procede a remarcar su contorno para que la inteligencia artificial no confunda excesos de imagen.

```
[15] def remove_edges(binary):
       mask = np.zeros(binary.shape, dtype=np.uint8)
       cnts = cv2.findContours(binary, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
       cnts = cnts[0] if len(cnts) == 2 else cnts[1]
       for c in cnts:
           area = cv2.contourArea(c)
           if area < 10000:
               cv2.drawContours(mask, [c], -1, (255,255,255), -1)
       result = cv2.bitwise_and(binary,binary,mask=mask)
       return result

     fig = plt.figure()
     fig.add_subplot(121)
     plt.imshow(binary, cmap='gray')
     #plt.axis(False)
     fig.add_subplot(122)
     plt.imshow(remove_edges(binary), cmap='gray')
     #plt.axis(False)
```

```
<matplotlib.image.AxesImage at 0x7f4d79245850>
```

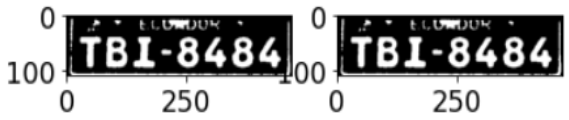

Figura 4-25: Segmentación binaria

Se crea una función auxiliar denominada CAJAS que permitirán reconocer y diferenciar los componentes de la imagen, configurando su alto y ancho, se debe asegurar que solamente bordee a las letras y números de la placa. Excluyendo el guion y letras más pequeñas en este caso las que conforman la palabra “ECUADOR”.

```
# Create sort_contours() function to grab the contour of each digit from left to right
def sort_contours(cnts,reverse = False):
    i = 0
    boundingBoxes = [cv2.boundingRect(c) for c in cnts]
    (cnts, boundingBoxes) = zip(*sorted(zip(cnts, boundingBoxes),
                                        key=lambda b: b[1][i], reverse=reverse))
    return cnts

def characters(binary, thre_mor, plate_image):
  cont, _  = cv2.findContours(binary, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE) #binary

  # creat a copy version "test_roi" of plat_image to draw bounding box
  test_roi = plate_image.copy()

  # Initialize a list which will be used to append charater image
  crop_characters = []

  # define standard width and height of character
  digit_w, digit_h = 30, 60 # it may help to improve ***

  for c in sort_contours(cont):
      (x, y, w, h) = cv2.boundingRect(c)
      ratio = h/w
      #print(ratio)
      if 0.82<=ratio<=5: # Only select contour with defined ratio(change numbers to modify boxes)
          if 0.35<=h/plate_image.shape[0]<0.8 and w/plate_image.shape[1]<1/6: #H&W modifier
              #print(ratio)
              # Draw bounding box arroung digit number
              cv2.rectangle(test_roi, (x, y), (x + w, y + h), (0, 255,0), 2) #2

              # Sperate number and gibe prediction
              curr_num = thre_mor[y:y+h,x:x+w]
              curr_num = cv2.resize(curr_num, dsize=(digit_w, digit_h))
              _, curr_num = cv2.threshold(curr_num, 220, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
              crop_characters.append(curr_num)

  return test_roi, crop_characters

test_roi, crop_characters = characters(binary, thre_mor, pl_image) # binary cropped_bin

print("Detect {} letters...".format(len(crop_characters)))
fig = plt.figure(figsize=(10,6))
#plt.axis(False)
plt.grid('both')
plt.imshow(test_roi)
```

Figura 4-26: Función auxiliar denominada CAJAS

```
Detect 7 letters...
<matplotlib.image.AxesImage at 0x7f4d70bad1d0>
```

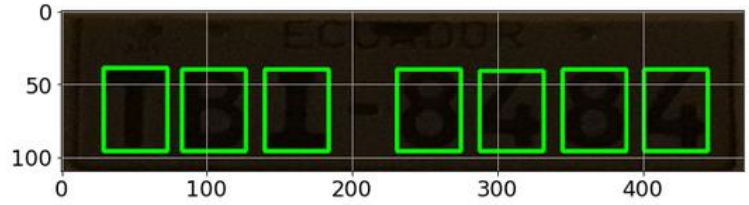

Figura 4-27: Componentes de la imagen, configurando su alto y ancho

Cada una de las cajas que reconoció un carácter se imprime una a continuación de la otra para obtener los dígitos y letras totales de la placa en escala de grises.

```
29] fig = plt.figure(figsize=(14,4))
    grid = gridspec.GridSpec(ncols=len(crop_characters),nrows=1,figure=fig)

    for i in range(len(crop_characters)):
        fig.add_subplot(grid[i])
        plt.axis(False)
        plt.imshow(crop_characters[i],cmap="gray")
```

Figura 4-28: Dígitos y letras totales de la placa en escala de grises.

Se ingresa esos datos de las cajas a la Red Neuronal "MobileNets_character_recognition.json" una red neuronal de arquitectura completa para el reconocimiento de imágenes pre-entrenada con 10000 imágenes.

```
[30] from sklearn.preprocessing import LabelEncoder

    # Load model architecture, weight and labels
    json_file = open(path+'MobileNets_character_recognition.json', 'r')
    loaded_model_json = json_file.read()
    json_file.close()
    model_MNCR = model_from_json(loaded_model_json)
    model_MNCR.load_weights(path+"License_character_recognition_weight.h5")

    labels = LabelEncoder()
    labels.classes_ = np.load(path+'license_character_classes.npy', allow_pickle = True)
    #load(file, mmap_mode, allow_pickle, fix_imports, encoding)

    # pre-processing input images and pedict with model
    def predict_from_model(image,model,labels):
        image = cv2.resize(image, (80,80))
        image = np.stack((image,)*3, axis=-1)
        prediction = labels.inverse_transform([np.argmax(model.predict(image[np.newaxis,:]))])
        return prediction
```

```
[31] fig = plt.figure(figsize=(15,3))
    cols = len(crop_characters)
    grid = gridspec.GridSpec(ncols=cols,nrows=1,figure=fig)

    final_string = ''
    for i,character in enumerate(crop_characters):
        fig.add_subplot(grid[i])
        title = np.array2string(predict_from_model(character,model_MNCR,labels))
        plt.title('{}'.format(title.strip("'[]"),fontsize=20))
        final_string+=title.strip("'[]")
        plt.axis(False)
        plt.imshow(character,cmap='gray')

    print(final_string)
```

Figura 4-29: Datos de las cajas a la Red Neuronal

Se aplica una pequeña línea de código que imprime arriba de cada imagen obtenida por las cajas la letra o número reconocido.

```
fig = plt.figure(figsize=(15,3))
cols = len(crop_characters)
grid = gridspec.GridSpec(ncols=cols,nrows=1,figure=fig)

final_string = ''
for i,character in enumerate(crop_characters):
    fig.add_subplot(grid[i])
    title = np.array2string(predict_from_model(character,model_MNCR,labels))
    plt.title('{}'.format(title.strip("'[]"),fontsize=20))
    final_string+=title.strip("'[]")
    plt.axis(False)
    plt.imshow(character,cmap='gray')

print(final_string)
```

```
TB18484
```

Figura 4-30: Impresión de la letra o número reconocido

Finalmente se aplicó una etapa de rectificación automática, la cual permite corregir los datos de la placa gracias a la configuración de las placas del país que consta de 3 letras al inicio y (3 o 4) números al final.

Rectification

```
[35] char_rect = {'O':'0', 'D':'0', 'Q':'0', 'I':'1', 'J':'1', 'L':'1', 'B':'3', 'A':'4', 'H':'4', 'G':'6', 'T':'7', 'B':'8',
                  '0':'O', '1':'I', '3':'B', '4':'A', '6':'G', '7':'T', '8':'B'}
```

```
[36] def string_rectifier(full_string):
       if 6 <= len(full_string) <=7:
         full_string_r = ''
         for i in range(len(full_string)):
           if (full_string[i].isnumeric() and i < 3) or (full_string[i].isalpha() and i > 2):
             c = char_rect[full_string[i]]
           else:
             c = full_string[i]
           full_string_r += c;
       else:
         full_string_r = full_string;
       return full_string_r

     rect_final_str = string_rectifier(final_string);
     print(rect_final_str)
```

```
TBI8484
```

Figura 4-31: Etapa de rectificación automática

Se puede aplicar un algoritmo que transforme cualquier número ubicado en las tres primeras posiciones a letras, o cualquier letra en las cuatro últimas posiciones a

números para así obtener el dato correcto de la placa como se observa en la figura:

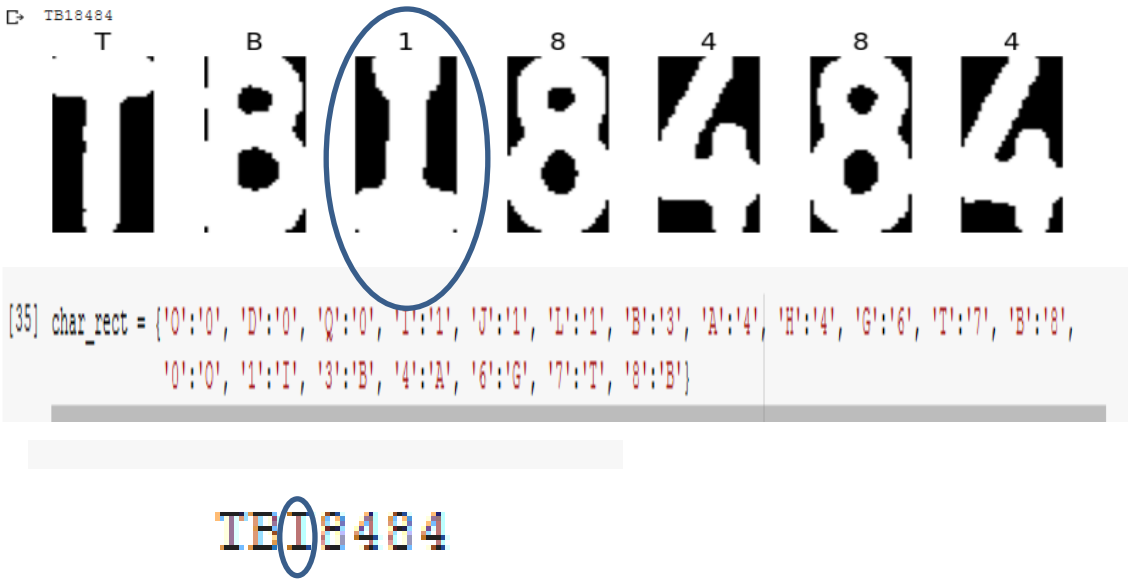

Figura 4-32: Dato correcto de la placa

En definitiva, se aplicó el método matemático Otsu en la segmentación de imágenes creando los recuadros verdes para diferenciar las letras y números que componen la placa.

Figura 4-33. Método matemático Otsu en la segmentación de imágenes

4.1.4. Etapa 4: Evaluación de desempeño del sistema

Para la evaluación de desempeño se consideró la validez por carácter individual o por placa vehicular completa. La base de datos proporcionada consta con alrededor de 50 imágenes de vehículos particulares e información básica (por ejemplo: placa, marca, modelo, año, color). En esta etapa se propone la sección de evaluación de desempeño, utilizando la métrica exactitud (accuracy) enfocada a la evaluación por caracteres o por placas completas.

Se evalúa el sistema determinando cuántos caracteres (porcentaje) son correctamente reconocidos, en función del valor real de cada caracter en cada placa (base de datos). Se "comparan" los caracteres obtenidos por el sistema con los caracteres reales de las placas. Esta última etapa permitió saber cuan bien trabaja el sistema, para ello se construyó una base de datos en formato “xlsx” denominada Placas.xlxs de los vehículos a analizar (puede ser modificada a lo largo del tiempo), y aplica el algoritmo de reconocimiento imagen por imagen, carácter por carácter, y la compara con la celda “PLACA” que contiene los datos de la placa y arroja un resultado porcentual de dicha comparación. Es decir, para determinar si el sistema trabaja bien se compara la base de datos “placas reconocidas” y las imágenes que tiene la misma base de datos.

```
# ========================================================================== #
# Stage4. Evaluation
def stage4(ToPrint):
  df = pd.read_excel (r"/content/drive/MyDrive/LPDR/DB/Placas.xlsx")
  print("Evaluating over {} images...".format(len(df.Placa)))
  T_c_rec = []
  T_total_c = []
  T_rate_c = []
  for p in df.Placa:
    new_img = p+".jpg"
    [_, vehic_plate, _] = stage2(dbPath, new_img)
    [_, _, _, final_str] = stage3(vehic_plate, bool(df[df.Placa==p].Dia.item()))
    c_rec, total_c, rate_c = stringc(p, final_str)
    T_c_rec.append(c_rec)
    T_total_c.append(total_c)
    T_rate_c.append(rate_c)
    if ToPrint == "FinalResult":
      clear_output()
      print("Evaluating over {} images...".format(len(df.Placa)))
  acc = sum(T_c_rec)/sum(T_total_c)
  print("====================================")
  print("Accuracy: {}".format(acc))
  print("====================================")
  return acc

def stringc(s1, s2):
  count=0
  for s in s1:
    for t in s2:
      if s==t:
        count+=1
  count = min(count, len(s1))
  return count, len(s1), count/len(s1)
```

Figura 4-34: Evaluación del desempeño del sistema

Si dicha base de datos contiene un número mayor de datos que las imágenes solamente procesarán la cantidad de imágenes que se encuentran en la carpeta de imágenes vehiculares, por lo que si en dicha carpeta se encuentra otra imagen que no corresponda a una placa vehicular no será tomada en cuenta.

```
# ================================================================= #
# Printing
def print_plate(vehicle, LpImg, thre_mor, test_roi, final_string):
  fig = plt.figure(figsize=(12,6))
  grid = gridspec.GridSpec(ncols=2,nrows=2,figure=fig)
  fig.add_subplot(grid[0])
  plt.axis(False)
  plt.imshow(vehicle)
  fig.add_subplot(grid[1])
  plt.axis(False)
  plt.imshow(LpImg)
  fig.add_subplot(grid[2])
  plt.axis(False)
  plt.imshow(thre_mor, cmap='gray')
  fig.add_subplot(grid[3])
  plt.axis(False)
  plt.imshow(test_roi)
  plt.title(final_string)
  return

def is_printed(vehicle1, labeled_vehicle, vehicle2, vehicle_plate, thre_mor, test_roi, final_string, img_print):
  if img_print:
    if labeled_vehicle == "No definido":
      plt.imshow(vehicle1)
      plt.axis(False)
    else:
      print_plate(vehicle2, vehicle_plate, thre_mor, test_roi, final_string)
  return
```

Figura 4-35: Impresión

La búsqueda de datos se la realiza conectándose a la base de datos antes mencionada e imprime las celdas que corresponden a dicha placa, esta etapa al ser de búsqueda cuenta con un apartado de seguridad manual, el cual permite ingresar los datos digito por digito para evitar cualquier error del sistema.

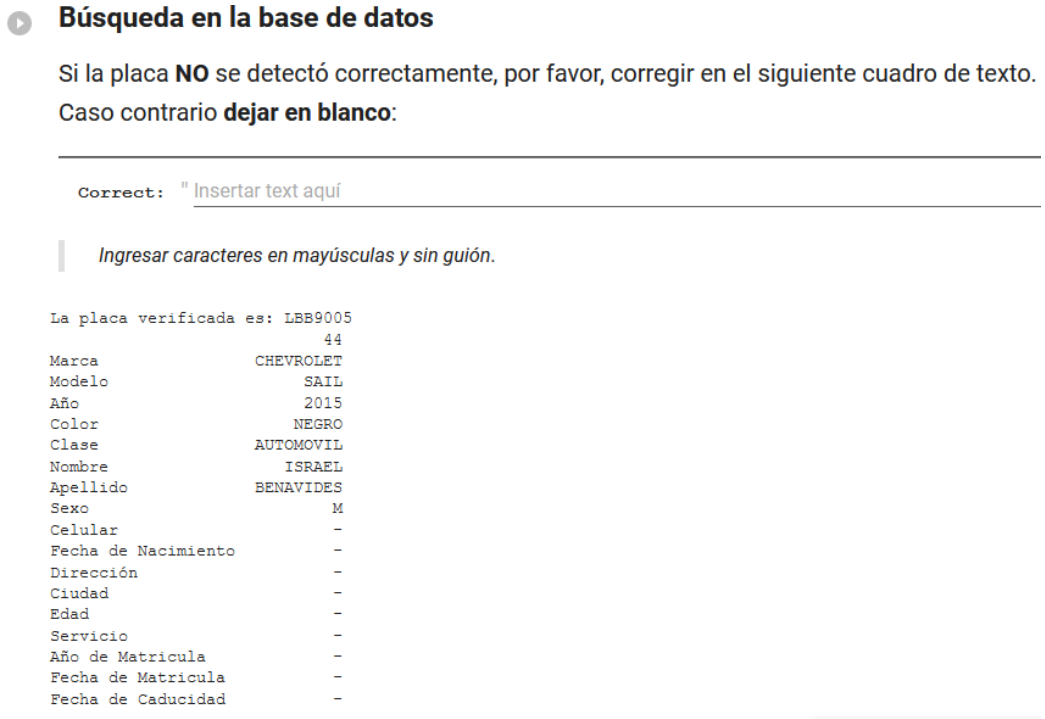

Figura 4-36: Búsqueda de base de datos

4.1.5. Interfaz gráfica del sistema

Como interfaz gráfica se propuso usar los "notebooks" interactivos de Jupyter (soportados por Google Colab). Se entregaron dos notebooks: uno con todo el código separado por bloques para la presentación de cada etapa individual y otro con código estrictamente necesario para la ejecución, el que pudo ser usado por el usuario final (sin necesidad que sepa la programación de fondo, es decir, la GUI). La interfaz gráfica también conocida como interfaz de usuario, en este sistema inteligente es muy amigable con el usuario para su fácil uso.

Para el desarrollo de la interfaz gráfica primero se debe copiar los archivos (archivo .py, imágenes, y base de datos) a nuestra unidad de nube Drive.

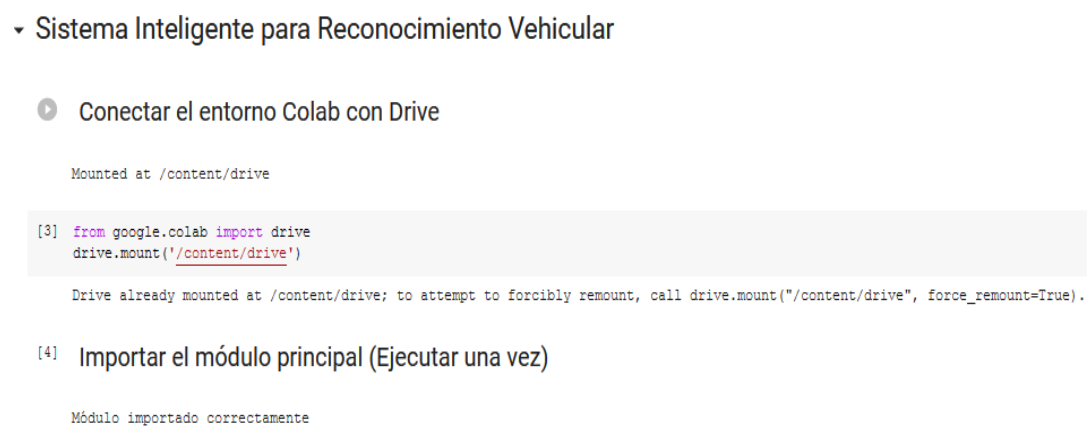

Figura 4-37: Conectar el entorno Colab con Drive

Se selecciona el PATH si se desea subir la imagen desde el drive o subirla directamente de la cámara fotográfica del dispositivo, se elige el nombre de la imagen, se coloca un check en la opción PRINTIMAGE si se requiere que se imprima la imagen, de igual manera si la imagen fue tomada en el día clic en DAY y si fue en la noche se desactiva esa opción.

El programa arroja como resultado directamente los datos programados como lo son:

- Tipo de vehículo
- Sector al que pertenece,
- caracteres de la placa vehicular.

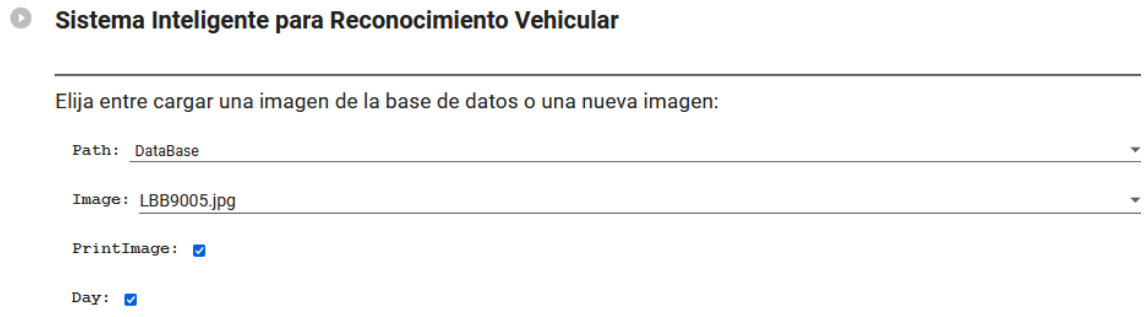

Figura 4-38: Sistema inteligente para reconocimiento vehicular

Figura 4-39: Reconocimiento de la placa

Con los datos de los caracteres se conecta a la base de datos de vehículos registrados y da los datos del automotor. La evaluación del sistema se lo hace en base a las imágenes y datos de la base.

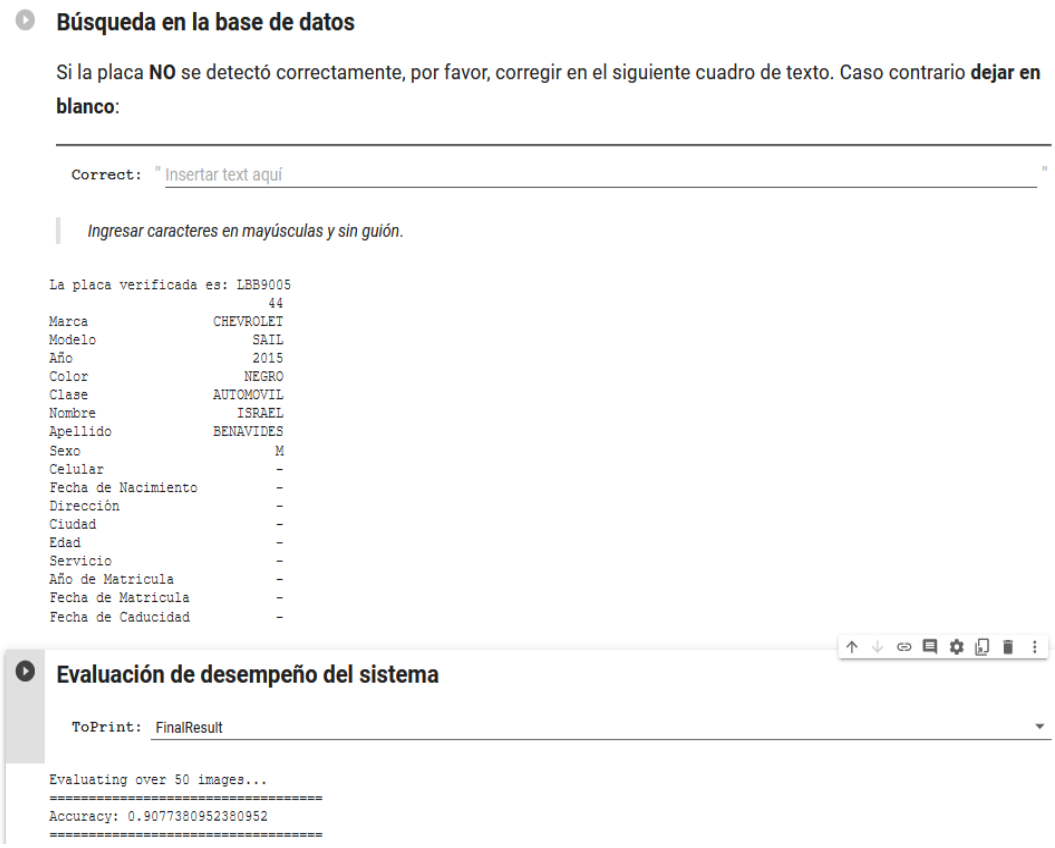

Figura 4-40: Búsqueda de base de datos

Finalmente se puede indicar que el conjunto de datos utilizado para el entrenamiento

de la red neuronal no es tan extenso, sin embargo, sirvió para realizar las pruebas necesarias y comprobar el adecuado funcionamiento del sistema. El sistema implementado en la presente investigación servirá como punto de partida para trabajos futuros, al realizar adecuaciones y mejoras en sus herramientas que permitan utilizar el sistema en un ambiente real, con vehículos en movimiento y a velocidades considerables.

CAPÍTULO V

CONCLUSIONES, RECOMENDACIONES, BIBLIOGRAFÍA Y ANEXOS

5.1. Conclusiones

El sistema inteligente desarrollado obtuvo un 90.77% de eficacia en el reconocimiento de placas ecuatorianas al momento de realizar pruebas aleatorias y la manipulación del sistema es de libre uso con Google Drive en el entorno ColabNotebook, adicionando un entorno de corrección manual para subsanar posibles errores.

La detección de un vehículo, reconocimiento, y posterior clasificación de acuerdo a su tipo, ya sea automóvil o camión/bus se realiza a través de la captura de una imagen por visión artificial; para este proceso se utilizó la red neuronal convolucional YOLO, la misma que por medio de una imágenes permitió la detección de objetos con una alta efectividad, además esta red neuronal se encargó de codificar implícitamente la información contextual, modelar el tamaño y la forma de los objetos y su apariencia.

Para detección del tipo de placas vehiculares se optó la utilización de la red neuronal convolucional WPOD, la cual fue configurada con datos específicos del borde, ancho y alto de la placa, para que pueda realizar el recorte de la imagen y proporcione solamente la foto de la placa vehicular, mientras que para determinar el tipo de vehículo se analizó el color de la placa, adecuando la imagen para trabajar en formato HSV.

Para el proceso de binarización utilizado en la investigación se empleó el algoritmo OTSU, el mismo que eligió de manera dinámica el valor umbral, recorriendo toda la imagen buscando el valor de intensidad más alto, así como también el valor más bajo para la generación de manera estadística de dicho valor. De esta forma se realiza la conversión de las imágenes en las escalas de gris, blur, binaria y dilatación, aplicando filtros que permitan obtener los segmentos de ubicación de las letras y números.

Para la evaluación del desempeño del sistema se utilizó la métrica exactitud (accuracy) enfocada a la evaluación por caracteres o por placas completas, la cual brindó una idea de la forma en la que trabaja el sistema, esta evaluación se efectuó aplicado el algoritmo de reconocimiento imagen por imagen, carácter por carácter, y la comparándolo con la celda “PLACA” donde se encuentran los datos de la placa para posteriormente proporcionar un resultado porcentual de dicha comparación.

5.2. Recomendaciones

Es importante que la cámara con la cual se realiza la captura de la imagen tenga una buena resolución, para de esta forma poder distinguirla con mejor calidad y no se dañe los pixeles de la imagen al realizar el binarizado, pues con ello se logrará que el reconocimiento de caracteres sea más exacto.

Es necesario ubicar la cámara en un lugar estratégico que facilite la adecuada captura de la imagen, ya que el ángulo donde se ubica puede generar problemas para identificar la placa vehicular, debido a que se encuentra muy cerca o muy lejos del vehículo, además este lugar debe tener buenas condiciones de iluminación.

Para que exista efectividad en el reconocimiento de las placas vehiculares es importante que estas se rijan al sistema de reglamentación propuesto por la Ley de Tránsito, en donde se exige mantener en perfecto estado la legibilidad de la placa vehicular.

El presente proyecto puede ser utilizado como punto de partida para futuras investigaciones, utilizando otros recursos o procedimientos tecnológicos que les permita alcanzar mejores resultados.

5.3. BIBLIOGRAFÍA

[1] Serkan Ozbay and Ergun Ercelebi, "Automatic Vehicle Identification by Plate Recognition," *World Academy of Science, Engineering and Technology*, pp. 222-225, 2005.

[2] Yeasir Arafat, Anis Mohd, Uswah Khairuddin, and Raveendran Paramesran, "Systematic review on vehicular licence plate recognition framework in intelligent transport systems," *IET Intelligent Transport Systems*, vol. 13, no. 5, pp. 745 – 755, 2019.

[3] Eduardo Barbecho and Martín Zhindón, "Diseño de un algoritmo de reconocimiento de placas vehiculares ecuatorianas usando redes neuronales convolucionales," *Journal of Science and Research*, vol. 5, no. 4, pp. 76-86, 2020, https://revistas.utb.edu.ec/index.php/sr/article/view/865/673.

[4] Di Zang, Zhenliang Chai, Junqi Zhang, Dongdong Zhang, and Jiujun Cheng, "Vehicle license plate recognition using visual attention model and deep learning," *Journal of Electronic Imaging*, vol. 24, no. 3, mayo 2015, https://www.spiedigitallibrary.org/journals/journal-of-electronic-imaging/volume-24/issue-3/033001/Vehicle-license-plate-recognition-using-visual-attention-model-and-deep/10.1117/1.JEI.24.3.033001.full.

[5] C Gou, K Wang, Y Yao, and Z Li, "Vehicle License Plate Recognition Based on Extremal Regions and Restricted Boltzmann Machines.," *IEEE Transactions on intelligent transportation system*, vol. 17, no. 4, pp. 1096-1107, 2016, https://sci-hub.st/10.1109/TITS.2015.2496545.

[6] Alexander Dueñas and Marcelo Stemmer, "Sistema automático para el reconocimiento de placas vehculares en SVM y HOG," *XIII Simpósio Brasileiro de Automacao Inteligente*, vol. 1, pp. 319-325, octubre 2017, https://www.ufrgs.br/sbai17/papers/paper_111.pdf.

[7] Asmaa Elbamby and Elsayed Hemayed, "Real-Time Automatie Multi-Style Lieense Plate Deteetion in Videos," *12th International Computer Engineering Conference (ICENCO)*, 2016.

[8] Orlando Barcia, "Reconocimiento automático a través de visión artificial, correlación estadística y Matlab aplicado a las matrículas de vehículos," *Primer Congreso Salesiano de Ciencia, Tecnología e Innovación para la Sociedad*, pp.

181-197, 2014.

[9] Ioannis Giannoukos, Christos Anagnostopoulos, Vassili Loumos, and Eleftherios Kayafas, "Operator context scanning to support high segmentation rates for real time license plate recognition," *Pattern Recognition*, vol. 43, no. 11, pp. 3866-3878, 2010, https://sci-hub.st/https://doi.org/10.1016/j.patcog.2010.06.008.

[10] Hendry and Rung Chen, "Automatic License Plate Recognition via sliding-window darknet-YOLO deep learning," *Image and Vision Computing*, vol. 87, pp. 47-56, 2019.

[11] Elias Perdomo, Orlando Amaral, David Monsanto, and Javier Pérez, "Algoritmo para detección e identificación de matrículas Cubanas sobre Raspberry Pi," *VI Simposio Internacional de Electrónica: Diseño, Aplicaciones, Técnicas Avanzadas y Retos Actuales*, pp. 1-10, Marzo 2018, http://www.informaticahabana.cu/sites/default/files/ponencias2018/ELE11.pdf.

[12] Luis Alanís, Moisés Márquez, Viridiana Hernández, and Octavio Sánchez, "Sistema de reconocimiento de placas vehiculares haciendo uso de modelos asociativos," *Research in Computing Science*, vol. 147, no. 12, pp. 127–136, 2018, https://www.cicling.org/micai/rcs-local/2018_147_12/Sistema%20de%20reconocimiento%20de%20placas%20vehiculares%20haciendo%20uso%20de%20modelos%20asociativos.pdf.

[13] Peng Chun, Tsai Cheng, Chang Ting, and Yeh Jen, "A Fast and Noise Tolerable Binarization Method for Automatic License Plate Recognition in the Open Environment in Taiwan," *Symmetry*, vol. 12, no. 1374, pp. 1-20, 2020, https://www.researchgate.net/publication/343731176_A_Fast_and_Noise_Tolerable_Binarization_Method_for_Automatic_License_Plate_Recognition_in_the_Open_Environment_in_Taiwan.

[14] Maulidia Hidayah, Isa Akhlis, and Endang Sugiharti, "Recognition Number of The Vehicle Plate Using Otsu Method and K-Nearest Neighbour Classification," *Scientific Journal of Informatics*, vol. 4, no. 1, pp. 66-75, may 2017, https://core.ac.uk/download/pdf/205903815.pdf.

[15] Guocong Lin, Boyang Xu, Binqiang Xue, and Chuanguang Chen, "License

plate recognition based on mathematical morphology and template matching," *Chinese Automation Congress*, pp. 405-410, 2019.

[16] Youngwoo Yoon, Kyu-Dae Ban, Hosub Yoon, Jaeyeon Lee, and Jaehong Kim, "Best Combination of Binarization Methods for License Plate Character Segmentation," *ETRI Journal*, vol. 35, no. 3, pp. 491-500, junio 2013.

[17] Z Selmi, M Halima, U Pal, and M Alimi, "DELP-DAR System for License Plate Detection and Recognition," *Journal pre-proof*, 2019, https://sci-hub.st/https://doi.org/10.1016/j.patrec.2019.11.007.

[18] Z Soghadi and C Suen, "License Plate Detection and Recognition by Convolutional Neural Networks.," *International Conference on Pattern Recognition and Artificial Intelligence*, pp. 380-393, 2020.

[19] Holger Ortega and Milton Torres, "Reconocimiento automático de la placa de un vehículo de Ecuador," *Universidad Politécnica Salesiana*, pp. 1-13, 2020.

[20] Y Wang, B Fang, L Lan, H Lou, and Y Tang, "Adaptative binarization: A new approach to license plate characters segmentation," *Proceedings of the 2012 International Conference on Wavelet Analysis and Pattern Recognition*, pp. 91-99, 2012.

[21] Samiul Azam and Monirul Islam, "Automatic license plate detection in hazardous condition," *Journal of Visual Communication and Image Representation*, vol. 36, pp. 172-186, 2016, https://www.sciencedirect.com/science/article/abs/pii/S1047320316000249.

[22] Animesh Roy, Muhammad Hossen, and Debashis Nag, "License plate detection and character recognition system for commercial vehicles based on morphological approach and template matching," *3rd International Conference on Electrical Engineering and Information Communication Technology*, pp. 1-6, 2016.

[23] George Mundaca, *Detección de caracteres de placas de automóviles mediante técnicas de visión artificial*. Piura, Perú: Universidad de Piura, 2016.

[24] INEC, *Resultados del Censo 2010 de población y vivienda en el Ecuador.*, 2010.

[25] Manlio Massiris, Claudio Delrieux, and Álvaro Fernández, "Detección de equipos de protección personal mediant red neuronal convolucional," *Actas de*

las XXXIX Jornadas de Automática, pp. 1022-1029, septiembre 2018, https://ruc.udc.es/dspace/bitstream/handle/2183/24891/2018_Massiris_Manlio_Detecci%c3%b3n-equipos-protecci%c3%b3n-personal-red-neuronal-convolucional-YOLO.pdf?sequence=3&isAllowed=y.

5.4. ANEXOS

Anexo 1. Códigos del sistema de reconocimiento

Código etapa 1

```
# Libraries
import matplotlib.pyplot as plt
from keras.applications.nasnet import NASNetLarge, decode_predictions
from keras.preprocessing import image
import numpy as np

# Stage1. Vehicle type recognition
def stage1(path, img_name, img_print):
  cnn_output = vehicle_type(path, img_name, img_print)
  relab = relabeling(cnn_output[0])
  print("Se reconoció el vehículo: "+ relab)
  return relab

# NASnet object
nnl = NASNetLarge();

def vehicle_type(path, img_name, img_print):
  #Loading image (PIL format)
  mpath = "" if path == "Root" else "/content/drive/MyDrive/LPDR/DB/";
  #if path == "Root":
  #  mpath = "";
  #elif path == "DataBase":
  #  mpath = "/content/drive/MyDrive/LPDR/DB/";

  img_org = image.load_img(mpath+img_name, target_size=(331,331));

  x = image.img_to_array(img_org);
  if img_print:
    plt.imshow(img_org)
    plt.xticks([])
    plt.yticks([])


  #Mapping (new range) 0 - 255 => -1 - 1
  x /= 255;
  x -= 0.5;
  x *= 2;

  # Adding a dimension (input format to CNN)
  x = x.reshape([1, x.shape[0], x.shape[1], x.shape[2]]);

  # Method to evaluate the image (predict)
  y = nnl.predict(x);

  # Method to show prediction information
```

```
  classes_predicted = decode_predictions(y);
  # Show top 3 predictions
  classes_predicted = [classes_predicted[0][0][1], classes_predicted[0][1][1],
classes_predicted[0][2][1]];
  return classes_predicted

# Re-labeling
# Classes from ImageNET
all_lists = [['cab', 'minivan', 'beach_wagon', 'grille', 'police_van', 'convertible',
'car_wheel', 'sports_car', 'limousine', 'racer', 'maze'],
            ['pickup', 'jeep', 'golfcart', 'harvester', 'forklift'],
            ['minibus'],
            ['moving_van', 'recreational_vehicle', 'trailer_truck', 'garbage_truck',
'tow_truck', 'crane'],
            ['motor_scooter', 'moped', 'crash_helmet']]

# My classes
all_classes = ['automóvil',
              'camioneta/jeep',
              'minibus/buseta',
              'bus/camión',
              'motocileta']

# ismember function
def ismember(A, B):
  return np.sum([ np.sum(a == B) for a in A ])

def relabeling(result):
  onehot = [];
  for l in all_lists:
    onehot.append(ismember(l, result))

  final_class = ""
  for i in range(len(onehot)):
    final_class = final_class + onehot[i]*all_classes[i]

  if final_class == "":
    final_class = "No definido"

  return final_class
```

Código etapa 2

```
#
================================================================
======== #
# Stage2. License Plate Detection
def stage2(path, img_name):
  vehicle, LpImg, cor = get_plate(path+img_name);
  plate_image = LpImg[0];
  plate = (255*plate_image).astype(np.uint8);
  plate = cv2.cvtColor(plate, cv2.COLOR_BGR2HSV);
  p_type = plate_type(plate, color_ranges, color_list);
  print("Cuya placa corresponde al tipo: "+ p_type)
  return vehicle, plate_image, p_type

def load_model(path):
  try:
    path = splitext(path)[0]
    with open('%s.json' % path, 'r') as json_file:
      model_json = json_file.read()
    model = model_from_json(model_json, custom_objects={})
    model.load_weights('%s.h5' % path)
    #print("Loading model successfully...")
    return model
  except Exception as e:
    print(e)

# WPOD
wpod_net_path = notebooksPath+"wpod-net";
wpod_net = load_model(wpod_net_path);
#
utilsWPOD = SourceFileLoader("utilsWPOD",
notebooksPath+"utilsWPOD.py").load_module();

# forward image through model and return plate's image and coordinates
# if error "No Licensese plate is founded!" pop up, try to adjust Dmin
def get_plate(image_path, Dmax=608, Dmin=350): #256
  vehicle = preprocess_image(image_path)
  ratio = float(max(vehicle.shape[:2])) / min(vehicle.shape[:2])
  side = int(ratio * Dmin)
  bound_dim = min(side, Dmax)
  _ , LpImg, _, cor = utilsWPOD.detect_lp(wpod_net, vehicle, bound_dim,
lp_threshold=0.5)
  return vehicle, LpImg, cor

def preprocess_image(image_path):
  img = cv2.imread(image_path)
  img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
```

```
  img = img / 255
  return img

color_list = ["comercial", "gubernamental", "gad", "particular"]
color_ranges = {}
color_ranges[color_list[0]+"Low"] = np.array([0, 170, 20], np.uint8)
color_ranges[color_list[0]+"High"] = np.array([15, 255, 255], np.uint8)
color_ranges[color_list[1]+"Low"] = np.array([21, 170, 20], np.uint8)
color_ranges[color_list[1]+"High"] = np.array([33, 255, 255], np.uint8)
color_ranges[color_list[2]+"Low"] = np.array([40, 170, 20], np.uint8)
color_ranges[color_list[2]+"High"] = np.array([70, 255, 255], np.uint8)
color_ranges[color_list[3]+"Low"] = np.array([0, 0, 150], np.uint8)
color_ranges[color_list[3]+"High"] = np.array([178, 75, 255], np.uint8)

def plate_type(plate, types, color_list):
  """
  plate: HSV format & 0-255 range
  """
  portion_plate = plate[int(plate.shape[0]*0.1)-10:int(plate.shape[0]*0.1)+30,
plate.shape[1]//2-50:plate.shape[1]//2+50, :]
  #cv2_imshow(cv2.cvtColor(portion_plate, cv2.COLOR_HSV2BGR))
  mask = []
  for c in color_list:
    m = cv2.inRange(portion_plate, types[c+"Low"], types[c+"High"])
    mask.append(np.sum(m/255))
  return color_list[mask.index(max(mask))]

#
============================================================
======== #
# Printing
def print_plate(vehicle, LpImg):
  fig = plt.figure(figsize=(12,6))
  grid = gridspec.GridSpec(ncols=2,nrows=1,figure=fig)
  fig.add_subplot(grid[0])
  plt.axis(False)
  plt.imshow(vehicle)
  grid = gridspec.GridSpec(ncols=2,nrows=1,figure=fig)
  fig.add_subplot(grid[1])
  plt.axis(False)
  plt.imshow(LpImg)
  return

def is_printed(vehicle1, labeled_vehicle, vehicle2, vehicle_plate, img_print):
  if img_print:
    if labeled_vehicle == "No definido":
      plt.imshow(vehicle1)
      plt.axis(False)
    else:
      print_plate(vehicle2, vehicle_plate)
```

```
  return
```

Código etapa 3

```
# ================================================================
======== #
# Stage3. License Plate Recognition
def stage3(plate_image, day):
  pl_image, gray, blur, binary, thre_mor = morph(plate_image, day)
  test_roi, crop_characters = characters(binary, thre_mor, pl_image)
  if len(crop_characters) == 0:
    test_roi, crop_characters = characters(remove_edges(binary), thre_mor, pl_image)
  print("{} caracteres detectados.".format(len(crop_characters)))
  final_string = ''
  for i, character in enumerate(crop_characters):
      c = np.array2string(predict_from_model(character,model_MNCR,labels))
      final_string+=c.strip("'[]")
  final_string = string_rectifier(final_string);
  print("La placa reconocida es: "+ final_string)
  return thre_mor, test_roi, crop_characters, final_string

# Load model architecture, weight and labels
json_file = open(notebooksPath+'MobileNets_character_recognition.json', 'r')
loaded_model_json = json_file.read()
json_file.close()
model_MNCR = model_from_json(loaded_model_json)
model_MNCR.load_weights(notebooksPath+"License_character_recognition_weigh
t.h5")
labels = LabelEncoder()
labels.classes_ = np.load(notebooksPath+'license_character_classes.npy')

def morph(LpImg, day):
  # Scales, calculates absolute values, and converts the result to 8-bit.
  plate_image = cv2.convertScaleAbs(LpImg, alpha=(255.0))

  # convert to grayscale and blur the image
  gray = cv2.cvtColor(plate_image, cv2.COLOR_BGR2GRAY) #COLOR_BGR2GR
AY
  blur = cv2.GaussianBlur(gray,(7,7),0)

  # Applied inversed thresh_binary
  if day:
    binary = cv2.threshold(blur, 180, 255, cv2.THRESH_BINARY_INV + cv2.THRE
SH_OTSU)[1]
  else:
    binary = cv2.adaptiveThreshold(blur,255,cv2.ADAPTIVE_THRESH_MEAN_C,
cv2.THRESH_BINARY_INV,11,2)

  kernel3 = cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3)) #3, 5
  thre_mor = cv2.morphologyEx(binary, cv2.MORPH_DILATE, kernel3)
```

```
  return plate_image, gray, blur, binary, thre_mor

# Create sort_contours() function to grab the contour of each digit from left to right
def sort_contours(cnts,reverse = False):
    i = 0
    boundingBoxes = [cv2.boundingRect(c) for c in cnts]
    (cnts, boundingBoxes) = zip(*sorted(zip(cnts, boundingBoxes),
                                        key=lambda b: b[1][i], reverse=reverse))
    return cnts

def characters(binary, thre_mor, plate_image):
  cont, _  = cv2.findContours(binary, cv2.RETR_EXTERNAL, cv2.CHAIN_APPRO
X_SIMPLE) #binary

  # creat a copy version "test_roi" of plat_image to draw bounding box
  test_roi = plate_image.copy()

  # Initialize a list which will be used to append charater image
  crop_characters = []

  # define standard width and height of character
  digit_w, digit_h = 30, 60 # it may help to improve ***

  for c in sort_contours(cont):
      (x, y, w, h) = cv2.boundingRect(c)
      ratio = h/w
      #print(ratio)
      if 0.82<=ratio<=5: # Only select contour with defined ratio
          if 0.35<=h/plate_image.shape[0]<0.8 and w/plate_image.shape[1]<1/6:
              #print(ratio)
              # Draw bounding box arroung digit number
              cv2.rectangle(test_roi, (x, y), (x + w, y + h), (0, 255,0), 2) #2

              # Sperate number and gibe prediction
              curr_num = thre_mor[y:y+h,x:x+w]
              curr_num = cv2.resize(curr_num, dsize=(digit_w, digit_h))
              _, curr_num = cv2.threshold(curr_num, 220, 255, cv2.THRESH_BINARY
+ cv2.THRESH_OTSU)
              crop_characters.append(curr_num)

  return test_roi, crop_characters

def remove_edges(binary):
  mask = np.zeros(binary.shape, dtype=np.uint8)
  cnts = cv2.findContours(binary, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPL
E)
  cnts = cnts[0] if len(cnts) == 2 else cnts[1]
  for c in cnts:
      area = cv2.contourArea(c)
      if area < 10000:
```

```
        cv2.drawContours(mask, [c], -1, (255,255,255), -1)
  result = cv2.bitwise_and(binary,binary,mask=mask)
  return result

# pre-processing input images and pedict with model
def predict_from_model(image, model, labels):
    image = cv2.resize(image, (80,80))
    image = np.stack((image,)*3, axis=-1)
    prediction = labels.inverse_transform([np.argmax(model.predict(image[np.newaxi
s,:]))])
    return prediction

char_rect = {'O':'0', 'D':'0', 'Q':'0', 'I':'1', 'F': '1', 'J':'1', 'L':'1', 'B':'3', 'A':'4', 'H':'4', 'G':'6',
 'T':'7', 'B':'8',
           '0':'O', '1':'I', '2':'2', '3':'B', '4':'A', '6':'G', '7':'T', '8':'B'}

def string_rectifier(full_string):
  if 6 <= len(full_string) <=7:
    full_string_r = ''
    for i in range(len(full_string)):
      if (full_string[i].isnumeric() and i < 3) or (full_string[i].isalpha() and i > 2):
        try:
          c = char_rect[full_string[i]]
        except:
          c = full_string[i]
      else:
        c = full_string[i]
      full_string_r += c;
  else:
    full_string_r = full_string;
  return full_string_r
```

Código etapa 4

```
# ============================================================
========= #
# Stage4. Evaluation
def stage4(ToPrint):
  df = pd.read_excel (r"/content/drive/MyDrive/LPDR/DB/Placas.xlsx")
  print("Evaluating over {} images...".format(len(df.Placa)))
  T_c_rec = []
  T_total_c = []
  T_rate_c = []
  for p in df.Placa:
    new_img = p+".jpg"
    [_, vehic_plate, _] = stage2(dbPath, new_img)
    [_, _, _, final_str] = stage3(vehic_plate, bool(df[df.Placa==p].Dia.item()))
    c_rec, total_c, rate_c = stringc(p, final_str)
    T_c_rec.append(c_rec)
    T_total_c.append(total_c)
    T_rate_c.append(rate_c)
    if ToPrint == "FinalResult":
      clear_output()
      print("Evaluating over {} images...".format(len(df.Placa)))
  acc = sum(T_c_rec)/sum(T_total_c)
  print("====================================")
  print("Accuracy: {}".format(acc))
  print("====================================")
  return acc

def stringc(s1, s2):
  count=0
  for s in s1:
    for t in s2:
      if s==t:
        count+=1
  count = min(count, len(s1))
  return count, len(s1), count/len(s1)
# ============================================================
========= #
# Printing
def print_plate(vehicle, LpImg, thre_mor, test_roi, final_string):
  fig = plt.figure(figsize=(12,6))
  grid = gridspec.GridSpec(ncols=2,nrows=2,figure=fig)
  fig.add_subplot(grid[0])
  plt.axis(False)
  plt.imshow(vehicle)
  fig.add_subplot(grid[1])
  plt.axis(False)
  plt.imshow(LpImg)
```

```
  fig.add_subplot(grid[2])
  plt.axis(False)
  plt.imshow(thre_mor, cmap='gray')
  fig.add_subplot(grid[3])
  plt.axis(False)
  plt.imshow(test_roi)
  plt.title(final_string)
  return

def is_printed(vehicle1, labeled_vehicle, vehicle2, vehicle_plate, thre_mor, test_roi, f
inal_string, img_print):
  if img_print:
    if labeled_vehicle == "No definido":
      plt.imshow(vehicle1)
      plt.axis(False)
    else:
      print_plate(vehicle2, vehicle_plate, thre_mor, test_roi, final_string)
  return
```