



UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE QUITO

CARRERA DE INGENIERÍA DE SISTEMAS

**DESARROLLO DE UN SISTEMA PARA LA IDENTIFICACIÓN AUTOMÁTICA DE
MARCAS Y MODELOS DE AUTOS USANDO ALGORITMOS DE APRENDIZAJE
DE MÁQUINA Y VISIÓN POR COMPUTADOR.**

Trabajo de titulación previo a la obtención del
Título de Ingeniera de Sistemas

AUTORA: Daysi Carolina Pérez Viracocha

TUTOR: Holger Raúl Ortega Martínez

Quito – Ecuador

2022

CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN

Yo, Daysi Carolina Pérez Viracocha con documento de identificación N° 1719882217, manifiesto que:

Soy la autora y responsable del presente trabajo; y, autorizo a que sin fines de lucro la Universidad Politécnica Salesiana pueda usar, difundir, reproducir o publicar de manera total o parcial el presente trabajo de titulación.

Quito, 28 de julio del 2022

Atentamente,



Daysi Carolina Pérez Viracocha

1719882217


CERTIFICADO DE SESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE TITULACIÓN A LA UNIVERSIDAD POLITÉCNICA SALESIANA

Yo, Daysi Carolina Pérez Viracocha con documento de identificación N° 1719882217, expreso mi voluntad y por medio del presente cedo a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud que soy la autora del Artículo Académico: “Desarrollo de un Sistema para la Identificación Automática de Marcas y Modelos de Autos Usando Algoritmos de Aprendizaje de Máquina y Visión por Computador.”, el cual ha sido desarrollado para optar por el título de Ingeniera de Sistemas, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En concordancia con lo manifestado, suscribo este documento en el momento que hago la entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Quito, 28 de julio del 2022

Atentamente,



Daysi Carolina Pérez Viracocha

1719882217

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN.

Yo Holger Raúl Ortega Martínez con C.I.: 1708182728, docente de la Universidad Politécnica Salesiana, declaro que bajo mi dirección y asesoría fue desarrollado el Trabajo de Titulación: **DESARROLLO DE UN SISTEMA PARA LA IDENTIFICACIÓN AUTOMÁTICA DE MARCAS Y MODELOS DE AUTOS USANDO ALGORITMOS DE APRENDIZAJE DE MÁQUINA Y VISIÓN POR COMPUTADOR**, realizado por Daysi Carolina Pérez Viracocha, obteniendo como resultado final el trabajo de titulación bajo la opción de Artículo Académico con lo cual cumple con todos los requisitos determinados por la Universidad Politécnica Salesiana.

Quito, 28 de julio del 2022



Fís. Holger Raúl Ortega Martínez, MSc

1708182728

DESARROLLO DE UN SISTEMA PARA LA IDENTIFICACIÓN AUTOMÁTICA DE MARCAS Y MODELOS DE AUTOS USANDO ALGORITMOS DE APRENDIZAJE DE MÁQUINA Y VISIÓN POR COMPUTADOR

DEVELOPMENT OF A SYSTEM FOR AUTOMATIC IDENTIFICATION OF CAR MAKES AND MODELS USING MACHINE LEARNING AND COMPUTER VISION ALGORITHMS

Daysi Carolina Pérez Viracocha ¹, Holger Raúl Ortega Martínez ²

RESUMEN

El presente artículo académico detalla la creación de un sistema que emplea algoritmos de Machine Learning y Visión por computador para la detección de 10 modelos y marcas de autos con más demanda en nuestro país, incentivando la creación de soluciones que se puedan complementar con software desarrollados principalmente para la detección de placas vehiculares. Los objetivos principales conllevan a fases que van a formar parte de la estructuración de la metodología. Obteniendo como resultados un 14% de error de clasificación de los 10 modelos y marcas de vehículos mediante imágenes o fotografías. También se permitió evaluar los algoritmos mediante métricas que comprobaron su correcto funcionamiento y coherencia en sus resultados clasificados.

Palabras clave: Machine Learning, Visión por computador, métricas de evaluación.

ABSTRACT

This academic article details the creation of a system that uses Machine Learning and Computer Vision algorithms for the detection of 10 models and brands of cars with more demand in our country, encouraging the creation of solutions that can be complemented with software developed mainly for the detection of license plates. The main objectives lead to phases that will be part of the structuring of the methodology. Obtaining as results a 14% error of classification of the 10 models and brands of vehicles by means of images or photographs. It was also possible to evaluate the algorithms by means of metrics that verified their correct operation and coherence in their classified results.

Keywords: Machine Learning, Computer Vision, evaluation metrics.

¹ Estudiante de Ingeniería de Sistemas - Universidad Politécnica Salesiana, Egresada - Sede Quito - Correo institucional: dperezv1@est.ups.edu.ec

² Fís. Holger Raúl Ortega Martínez, Msc. Docente de la Carrera de Ingeniería de Sistemas y Computación - Universidad Politécnica Salesiana- Sede Quito – Correo institucional: hortega@ups.edu.ec

1. Introducción

Las aplicaciones Machine Learning solucionan ciertos tipos de inconvenientes en particular dirigidos al campo de la geología [15], logística [16], automotriz [17], en esto también se involucran las empresas que trabajan en la implementación de software dirigido a controles vehiculares, reconocimiento de placas, reconocimiento de personas. [18].

Esta aplicación de algoritmos de Machine Learning y Visión por Computador permite identificar automáticamente las marcas de autos de los 10 modelos más vendidos en el Ecuador.

1.1 Fase de preprocesamiento

La fase de preprocesamiento incluye técnicas mediante algoritmos de redimensionamiento, eliminación de ruido, conversión de píxeles, segmentación, realce y mejoramiento de la calidad de la imagen [5].

1.1.1 Redimensionamiento de imágenes

El redimensionamiento consiste en el cambio de tamaño de las imágenes predispuestas teniendo una buena interpretación y optimización para trabajar con ellas [1], tal como se presenta en la Figura 1.

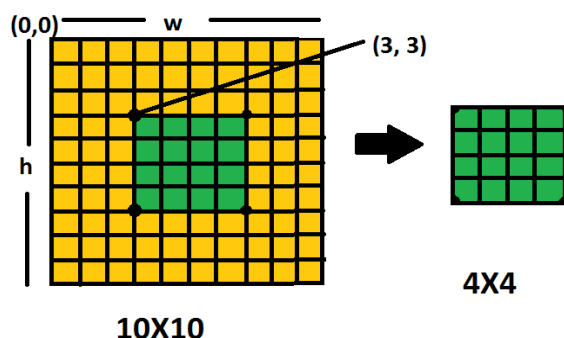


Figura 1. Representación del proceso de redimensionamiento de una imagen [1].

1.1.2 Escala de grises

La escala de grises consiste en convertir una imagen en tonos que escalan a los de blanco y negro para cada píxel [1]. Se puede

observar un ejemplo de conversión a escala de grises en la Figura 2.

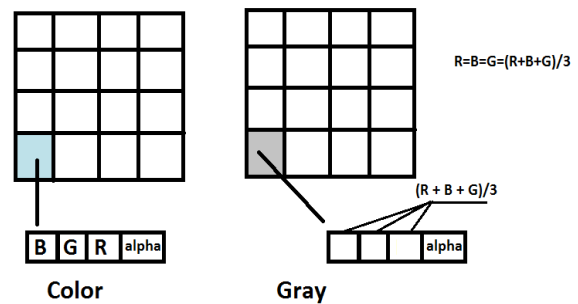


Figura 2. Representación del procedimiento de la escala de grises de una imagen [1].

1.1.3 Binarización

La binarización permite separar las distintas regiones, obteniendo las partes de interés que se desean analizar de la imagen [2]. Se puede observar un ejemplo de binarización en la Figura 3.



Figura 3. Resultado de una imagen binarizada.

1.1.4 Ecuación de histogramas

La ecuación de histogramas permite obtener un número de píxeles iguales para que exista una monotonía en la imagen, obteniendo un mejor contraste [3]. Se puede observar un ejemplo de ecualización en la Figura 4.

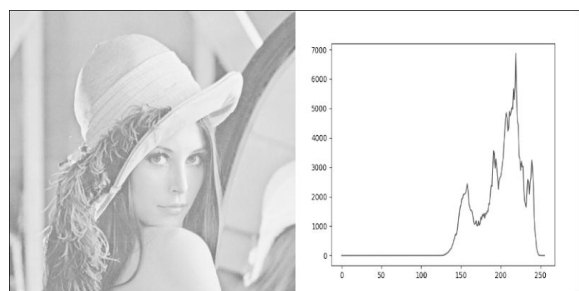


Figura 4. Representación del histograma en una imagen [3].

1.2 Fase de extracción de características

Esta fase permite extraer vectores de características que asocian a los distintos modelos de vehículos, enfocándose en sus

distintas características más relevantes [4]. Los algoritmos que se utilizaron, fueron los siguientes:

1.2.1 Hog (Histogram of Oriented Gradients)

Hog permite mediante su descriptor de características tomar la forma de un objeto y realizar la comparación con otra imagen parecida, permitiendo que su clasificación sea más eficaz y se reconozca mejor [6]. Se puede observar un ejemplo del resultado de Hog en la Figura 5.



Figura 5. Representación del funcionamiento del algoritmo Hog en una imagen.

1.2.2 Canny

Canny es un algoritmo que permite detectar los bordes de una imagen mediante un filtro gaussiano, el cual permite que los píxeles que constituyen a la imagen sean transformados a vectores de características [7]. Se puede observar un ejemplo del resultado de Canny en la Figura 6.



Figura 6. Representación del funcionamiento del algoritmo Canny en una imagen.

1.2.3 Sift (Scale-invariant feature transform)

Sift es un algoritmo que permite la extracción de características mediante puntos de interés, los cuales sirven para reconocer la imagen [8]. Se puede observar un ejemplo del resultado Sift en la Figura 7.



Figura 7. Representación del funcionamiento del algoritmo Sift en una imagen.

1.3 Fase de clasificación de los algoritmos

Para realizar la fase de clasificación es necesario haber “pulido y limpiado” las imágenes, esto quiere decir que después de las fases de preprocesamiento y extracción de características, las imágenes van hacer utilizadas para la clasificación de manera más eficiente, debido a que ciertas características que interpretan los algoritmos de clasificación, agrupan los distintos modelos para representar uno de ellos en específico. Los algoritmos utilizados se detallan a continuación:

1.3.1 K-NN (K-nearest neighbors)

El algoritmo K-NN permite buscar dentro de un conjunto de modelos que están más próximo o cercano al patrón inicial y los clasifica [9]. Se puede observar un ejemplo de K-NN en la Figura 8.

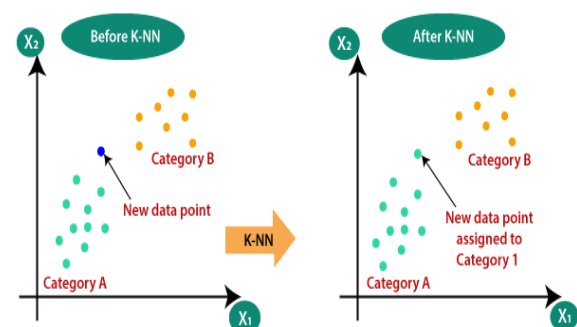


Figura 8. Ejemplo del funcionamiento del algoritmo K-NN [10].

1.3.2 K-Means

El algoritmo K-Means permite el agrupamiento de modelos que se basan en sus características y lo clasifica o agrupa minimizando la distancia que existe entre

cada elemento y centroide de su clase [11], como se observa en la Figura 9.

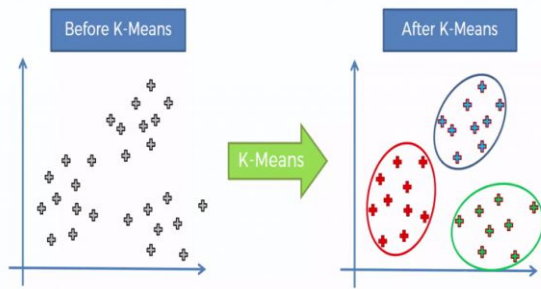


Figura 9. Ejemplo del funcionamiento del algoritmo K-Means [11].

1.3.3 SVM (Vector Support Machines)

El algoritmo SVM permite recolectar un número de datos grandes en un espacio de características, los cuales se van a categorizar permitiendo buscar y encontrar la forma más eficiente u óptima de clasificación entre distintas clases [13]. Se puede observar un ejemplo de SVM en la Figura 10.

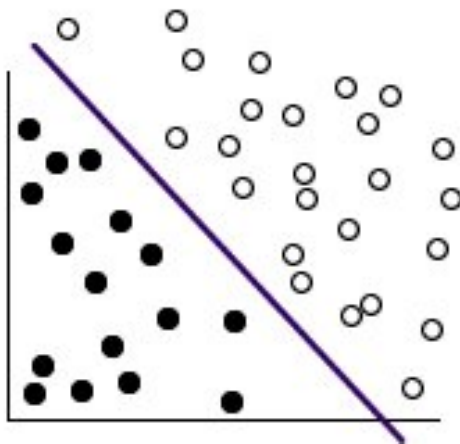


Figura 10. Representación en la gráfica de los datos transformados para la posterior predicción.

1.3.4 Redes Neuronales

Las redes neuronales son conocidas por emular ciertas funciones básicas del ser humano en cuanto a su forma de pensar y procesar el conocimiento. Están organizadas por capas que se dividen en tres partes; la capa de entrada que representa a los datos iniciales pasa por una o varias capas que se encuentran ocultas y se propagan por cada

red neuronal, mostrando sus resultados en la capa de salida [12].

Por último cabe señalar que usualmente estas redes no tienden a una interpretación sencilla, porque estas redes aprenden mediante la indagación de registros individuales los cuales predicen y realizan ajustes cuando predice erróneamente, hay que tener en cuenta que es un proceso bucle, que entre más veces se repita la red va a seguir mejorando en sus clasificaciones o predicciones [12]. Se puede observar un ejemplo de red neuronal en la Figura 11.

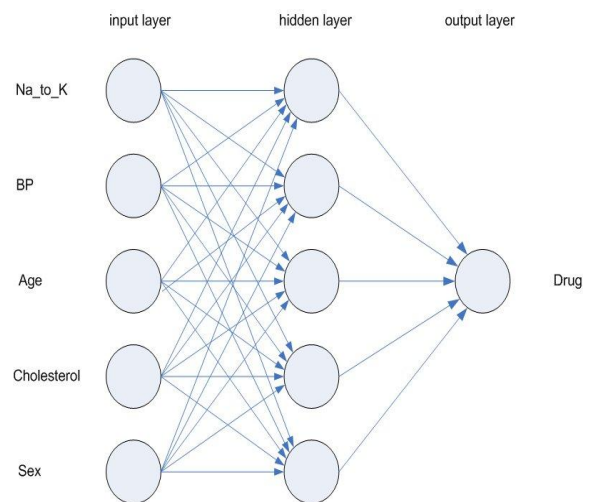


Figura 11. Estructura de la red neuronal [12].

1.5 Fase de evaluación de los algoritmos

En esta última fase se realiza la evaluación de los distintos algoritmos mediante métricas implementadas en las distintas librerías que Python dispone para trabajar de manera eficiente y precisa respecto a los resultados que estas métricas van a brindar, como los siguientes exponentes.

1.5.1 Matriz de confusión

La matriz de confusión permite medir el desempeño de los algoritmos [14]. Para esta métrica se utilizó una función de Python “**confusión_matrix**”.

1.5.2 Exactitud

La métrica de exactitud representa el número de objetos clasificados o predichos

correctamente [14]. Para esta métrica se utilizó una función de Python “accuracy_score”.

1.5.3 Precisión

La métrica de precisión representa las instancias más relevantes y cuáles de ellas se identificaron de manera afirmativa respecto a los demás [14]. Para esta métrica se utilizó una función de Python “precision_score”.

1.5.4 Sensibilidad

La métrica de sensibilidad representa la cantidad de aciertos positivos respecto a los negativos [14]. Para esta métrica se utilizó una función de Python “recall_score”.

1.5.5 Diagrama de estados

El diagrama de estados muestra el procedimiento de los pasos que realiza el sistema para clasificar una imagen de un vehículo en específico, como se observa en la Figura 12.

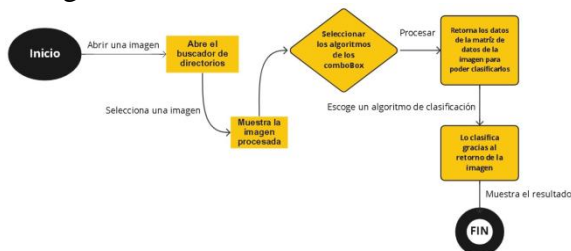


Figura 12. Diagrama de estados.

2. Métodos y materiales

2.1 Análisis del estado del arte

Este análisis facilitó la indagación de las distintas opciones que abarcaron los distintos libros de interés (referente al presente proyecto investigativo). Esta indagación permitió ayudar en la interpretación y aclaración para definir un alcance que va a dar un cierre formal al proyecto.

Se puede observar en la Tabla 1, los distintos artículos académicos consultados con sus respectivos objetivos, resultados y relevancia que proporcionaron al desarrollo del proyecto de titulación.

Tabla 1. Estado del arte.

No	Artículo (s)	Resultado (s)	Relevancia para el proyecto de titulación
1	[20]	Resultados positivos referente a la clasificación de los 100 modelos creados con alrededor de los 216 videos.	La relevancia es respecto a los distintos modelos de clasificación que están implementando.
2	[21]	La validación por precisión fue más rápida utilizando CNN.	Mediante esta comparación se puede obtener el mejor aprendizaje profundo.
3	[22]	Los caracteres OCR de reconocimientos se confunden con otros similares, debido a la mala calidad de las imágenes.	Esta investigación sirve para dar una guía al seguimiento de los procedimientos que se tienen que cumplir para los objetivos del proyecto académico.
4	[23]	Los tiempos de ejecución tardan demasiado y no permite ejecutarlo en tiempo real.	En esta investigación se va a permitir observar cómo se pueden implementar los algoritmos para la detección de imágenes usando un sistema de visión por computador.
5	[24]	Python mediante	Al hablar de la información sobre

		el uso de sus librerías puede ayudar en el conteo del tráfico y clasificarlo de manera eficaz.	el flujo del tráfico puede ayudar al proceso de saber cuántos modelos de vehículos pueden ser detectados al momento de atascamientos en las carreteras por tráfico o situación imprevista.
6	[25]	YOLOv3, tuvo mejores resultados debido a sus distintas virtudes de precisión y exactitud de objetos.	Al probar varios medios de transporte el algoritmo podrá reconocer a un porcentaje mayor la clasificación correcta de automóviles, logrando un pequeño porcentaje de error para que pueda identificar marcas y modelos específicamente de vehículos y no de otros transportes.
7	[26]	Sus niveles de error en los valores MSE, RMSE y MAE fueron bajos, teniendo un nivel de precisión alto.	Al tratarse de clasificadores puede aportar al proyecto, debido a su capacidad de predecir si la imagen es o no un vehículo.
8	[27]	El algoritmo para la detección de adelantamientos, obtuvo resultados favorables a través de pruebas en múltiples escenarios	El algoritmo de Lucas Kanade permite el reconocimiento del movimiento del vehículo; el cual, podría detectar los modelos y marcas de los vehículos en circulación.

		digitales.	
9	[28]	Las 4 redes neuronales obtuvieron porcentajes de acierto mayor al 80% llegando al 90% de fiabilidad.	Al ser un programa que detecta coches a partir de una imagen es conveniente para el proyecto, por el motivo de que el algoritmo pueda reconocer solo a los distintos tipos de autos y no al entorno que lo rodea.
10	[29]	Mediante el análisis de los distintos fragmentos de videos, se revelaron falsos positivos en el conteo de los distintos vehículos.	Al ser una investigación nos ofrece el mejor algoritmo de optimización respecto al tiempo de su ejecución para ser desarrollado en el proyecto.

2.2 Modelos y marcas de vehículos

Al tener una amplia gama de vehículos, se realizó una búsqueda para saber cuales fueron los modelos y marcas mas cotizados en el año 2021 para así optar por la selección de 10 modelos vehiculares, permitiendo la practicidad de trabajar con alrededor de 40 imágenes por cada uno de los siguientes automóviles presentes:

- Camioneta JAC HFC.
- Chevrolet Aveo Activo.
- Chevrolet Beat.
- Chevrolet Captiva.
- Chevrolet D Max.
- Great Wall Wingle.
- Kia Picanto.
- Kia Rio.
- Kia Sportage.
- Toyota Prius.

2.3 Desarrollo del sistema

2.3.1 Importación de librerías

Se importaron librerías que permitieron trabajar con los distintos algoritmos de Machine Learning, visión por computador y métricas de evaluación, como se puede observar en la Tabla 2.

Tabla 2. Importación de librerías en Python.

```
import cv2
import pandas
import numpy
import keras
```

También se procedió a realizar un diagrama de componentes para observar como funcionan las librerías utilizadas dentro del sistema, como se muestra en la Figura 13.

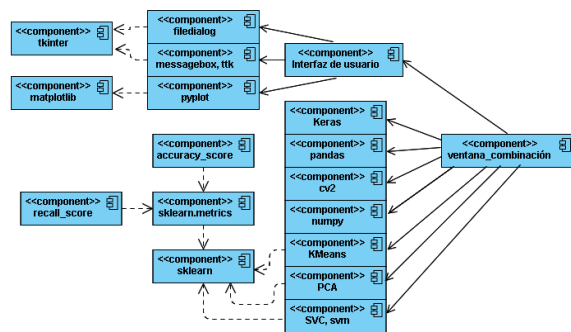


Figura 13. Diagrama de componentes.

2.3.2 Creación de directorios

La creación de directorios permitió la interacción con el conjunto de entrenamiento para que los diferentes algoritmos realicen el respectivo preprocesamiento, extracción de características y lo guarden en las distintas carpetas creadas, como se puede observar en la Tabla 3.

Tabla 3. Algoritmo para la creación de directorios.

```
global ruta_del_directorio
ruta_del_directorio =
'nombre_de_la_carpetas'
```

2.3.3 Preprocesamiento

Para el redimensionamiento de imágenes se necesitó crear un vector que permite cambiar el tamaño de todas las imágenes que se encontraron en la carpeta destino principal, guardándolo en una variable de un vector, como se puede observar en la Tabla 4.

Tabla 4. Algoritmo para el redimensionamiento de imágenes.

```
vector = cv2.resize(imagen,
posición=(500,270),
interpolación=cv2.INTER_CUBIC)
```

Después de haber realizado el redimensionamiento se aplicaron los distintos algoritmos de preprocesamiento. Comenzando por la escala de grises, que convirtió las imágenes en RGB o en escala que se extiende al blanco y negro, para después ser guardadas en distintas carpetas, como se puede observa en la Tabla 5.

Tabla 5. Algoritmo para la escala de grises.

```
vectores r,g,b =
im2.getpixel((i,j))
```

Para la binarización se creó un algoritmo que permitió crear una segmentación a las distintas imágenes, provocando datos de 0 a 255 bits y que al momento de procesarlos no existió una gran variedad de datos, lo cual hace que los distintos modelos de vehículos se parecieran entre sí, como se observa en la Tabla 6.

Tabla 6. Algoritmo de binarización.

```
vector_binario =
PIL.Image.new('L',(fila,columna))
```

Por último, se aplicó el algoritmo de ecualización de histogramas que partió en la creación de una función que especificó las distintas transformaciones de colores, específicamente grises para que exista una monocromía en las imágenes y puedan ser expresadas mediante un gráfico o histograma, como se observa en la Tabla 7.

Tabla 7. Algoritmo de ecualización de histogramas.

```
Vector de imagen =
cv2.cvtColor(imagen,cv2.COLOR_BGR2Y
UV)
```

```
Vector de imagen[:, :,0] =
cv2.equalizeHist(vector de
imagen[:, :,0])
```

Después que los algoritmos se implementaron, se procedió a trabajar sobre las imágenes, transformándolas y redimensionándolas en los distintos algoritmos de preprocesamiento. Se puede observar el algoritmo utilizado en la Tabla 8.

Tabla 8. Algoritmo de transformación de imágenes.

```
vector redimensión =
métodos.redimensionar(imágenes,
vector redimensión, archivo[n])
```

2.3.4 Extracción de características

El algoritmo de Hog tras implementarse permitió la conversión de un objeto en múltiples píxeles que representan a la imagen en un vector de características, formando un equivalente a 3968 bits, mediante la librería “**Hog**” de OpenCV.

El algoritmo de Sift realizó una extracción de puntos característicos de interés, el cual permitió mostrar sobre la imagen una distinción que se utilizó para identificar los distintos modelos de vehículos. Se utilizó la librería “**cv2.drawkeypoints**” de OpenCV.

Por último se implementó el algoritmo de Canny, el cual convirtió las imágenes en escalas de grises, reduciendo el ruido existente que procedió el cálculo de la gradiente, convirtiéndolo en coordenadas polares que se desplazó por cada píxel de la imagen y se transformó en un vector de característica. Se utilizó la librería “**cv2.cartToPolar**” de OpenCV.

Después de ser implementados los algoritmos de extracción, se realizaron las inicializaciones de las matrices que dieron paso para crear las combinaciones consecuentes, como se observa en la Tabla 9.

Tabla 9. Algoritmo para la inicialización de matrices.

```
global vector combinación
vector combinación =
numpy.empty(número de
características)
```

Cuando se iniciaron las matrices, se realizaron las combinaciones de algoritmos de preprocesamiento y de extracción de características, mediante el llamado de los métodos implementados con los distintos algoritmos, se procedió al almacenamiento de cada combinación en una variable. Y para que exista la coherencia de datos en las distintas combinaciones que se realizaron y poder crear la matriz de datos X, fue necesario realizar transformaciones de formato, redimensionamiento y transpuesta de las respectivas imágenes. Se utilizó la librería “**cv2.cvtColor**” y con las funciones “**numpy.reshape**”, “**numpy.transpose**”.

Cuando se realizaron y terminaron los pasos anteriores, se procedió a crear las matrices X de datos para cada combinación, el cual permitió agregar nuevos datos en una nueva columna dentro de la matriz, de acuerdo a la selección de algoritmos que se implementó. Se puede observar el algoritmo utilizado en la Tabla 10.

Tabla 10. Algoritmo para crear matrices X de datos.

```
matriz X de combinaciones =
numpy.c_[matriz X de
combinaciones, transpuesta de
combinaciones]
```

2.3.5 Diseño de archivos de etiquetas

Se crearon vectores que contemplen ciertos aspectos de almacenamiento para valores que fueron utilizados en otras clases a través

de métodos. Los cuales se representaron mediante etiquetas de números para cada modelo de automóvil.

2.3.6 Entrenamiento mediante algoritmos de clasificación

Para validar las distintas etiquetas creadas mediante la combinación de los algoritmos se crearon varias de ellas, teniendo en cuenta las que se utilizaron para el testeó y para las pruebas respectivas.

En la implementación de los algoritmos de clasificación K-NN y K-Means se utilizaron funciones, los cuales permitieron la identificación de objetos, mediante vectores característicos que exploraron las distancias que se encontraban en la cercanía de los modelos vehiculares que permitieron su clasificación de acuerdo a su interpretación.

Mientras que el algoritmo SVM diseñó hiperplanos para su respectiva clasificación, también permitió crear observaciones de entrenamiento para los distintos modelos a clasificar, teniendo un margen más amplio para el análisis, el cual procedió a entregar una respuesta definitiva.

El algoritmo de clasificación K-NN agrupó los 10 modelos y marcas vehiculares mediante el uso del vector Y de etiquetas, el cual permitió predecir o clasificar los nuevos automóviles que se ingresaban en el sistema. Se puede observar el algoritmo utilizado en la Tabla 11.

Tabla 11. Algoritmo K-NN de clasificación.

`Kn=10`
`clasificador.aprendizaje(n_ft,`
`vectores de etiquetas)`

El algoritmo K-Means determinó el número de clusters o centroides con los cuales se trabajó. En este caso se ingresaron los 10 modelos de vehículos, el cual permitió determinar la marca del auto en cada centroide localizado. Se puede observar en la Tabla 12.

Tabla 12. Algoritmo de ingreso de los clusters para el entrenamiento.

`vector_kmeans = KMeans(número de`
`centroides).fit(matriz X para`
`pruebas)`

Mientras que el algoritmo SVM necesitó de una matriz de datos para la obtención del número de autos disponibles, los cuales fueron entrenados y clasificados mediante un hiperplano que mostró sus resultados, así permitió ser incorporado al grupo de combinaciones realizadas con anterioridad.

2.3.7 Implementación de métricas de evaluación

Para la implementación de métricas que evaluaron los distintos algoritmos, se necesito de incorporar métodos que vinieron incorporados en las librerías de Python, como es el caso de la sensibilidad “**recall_score**”, para la precisión “**precision_score**” y por último, la exactitud fue realizado con “**accuracy_score**”, así permitió mostrar cada uno de sus resultados por consola con tan solo imprimirlo.

2.3.8 Creación de la interfaz gráfica del sistema

Primero se necesitó crear la ventana o interfaz gráfica para que el usuario interactúe sobre ella, permitiendo agregar ciertas características como el tamaño, las dimensiones, las posiciones de los ComboBox, los botones, los labels, así como el diseño del menú de configuraciones y para subir o cargar la imagen al sistema. Todo esto se puede observar en el código de la interfaz gráfica ubicada en la sección de Anexos.

3. Resultados y discusión

3.1 Resultados obtenidos

El sistema presenta varios algoritmos implementados en las fases de preprocesamiento, extracción de características y clasificación. Esto quiere decir que van a existir varias soluciones que

van a responder a la finalidad del sistema, el cual es la predicción de los distintos modelos de automóvil seleccionados.

Al contar con 3 algoritmos distintivos para cada fase, se obtuvo un resultado de 27 combinaciones de los cuales generaron 14 imágenes, que permitieron ser probadas para la clasificación de las 386 imágenes restantes que se implementaron para el conjunto de entrenamiento obteniendo así, los resultados presentados en la siguiente Tabla 13.

Tabla 13. Resultados de las combinaciones de algoritmos.

#	Algoritmos de preprocesamiento	Algoritmos de extracción de características	Algoritmos de clasificación	Error de clasificación
1	Escala de grises	HOG	KNN	5/14 = 0.36
2	Escala de grises	HOG	K-Means	7/14 = 0.5
3	Escala de grises	HOG	SVM	9/14 = 0.64
4	Escala de grises	CANNY	KNN	7/14 = 0.5
5	Escala de grises	CANNY	K-Means	11/14 = 0.79
6	Escala de grises	CANNY	SVM	11/14 = 0.79
7	Escala de grises	SIFT	KNN	6/14 = 0.43
8	Escala de grises	SIFT	K-Means	11/14 = 0.79
9	Escala de grises	SIFT	SVM	9/14 = 0.64
10	Binarización	HOG	KNN	7/14 = 0.5
11	Binarización	HOG	K-Means	2/14 = 0.14
12	Binarización	HOG	SVM	7/14 = 0.5
13	Binarización	CANNY	KNN	6/14 = 0.43
14	Binarización	CANNY	K-Means	8/14 = 0.57
15	Binarización	CANNY	SVM	10/14 = 0.71
16	Binarización	SIFT	KNN	6/14 = 0.43
17	Binarización	SIFT	K-Means	9/14 = 0.64
18	Binarización	SIFT	SVM	7/14 = 0.5
19	Ecualización de histogramas	HOG	KNN	6/14 = 0.43
20	Ecualización de histogramas	HOG	K-Means	9/14 = 0.64
21	Ecualización de histogramas	HOG	SVM	5/14 = 0.36
22	Ecualización de	CANNY	KNN	5/14 =

	histogramas			0.36
23	Ecualización de histogramas	CANNY	K-Means	11/14 = 0.79
24	Ecualización de histogramas	CANNY	SVM	10/14 = 0.71
25	Ecualización de histogramas	SIFT	KNN	8/14 = 0.57
26	Ecualización de histogramas	SIFT	K-Means	11/14 = 0.79
27	Ecualización de histogramas	SIFT	SVM	11/14 = 0.79

Antes de explicar los resultados obtenidos se quiere indicar sobre el error de clasificación. Este error no es más que la anotación de los resultados que salieron mal clasificados es decir, de las 14 pruebas que se realizó con las imágenes, solo se anotaron las combinaciones que no lograron acertar a la marca y modelo del vehículo.

Los resultados que obtuvieron aproximaciones o a exceder al 60% de error, afirmaron que los algoritmos presentaron ciertos inconvenientes al tratar de clasificar o de predecir un modelo de automóvil, esto pudo deberse principalmente a las combinaciones que se realizó con todos los algoritmos de preprocesamiento, de características y de clasificación. Otro factor importante que hubo para obtener resultados tan altos fue por las imágenes que se utilizaron en el conjunto de entrenamiento, todos estos resultados se pueden observar en la Tabla 14.

Tabla 14. Resultados con errores de clasificación altos.

1	Escala de grises	HOG	K-Means	0.5
2	Escala de grises	HOG	SVM	0.64
3	Escala de grises	CANNY	KNN	0.5
4	Escala de grises	CANNY	K-Means	0.79
5	Escala de grises	CANNY	SVM	0.79
6	Escala de grises	SIFT	K-Means	0.79
7	Escala de grises	SIFT	SVM	0.64
8	Binarización	HOG	KNN	0.5
9	Binarización	HOG	SVM	0.5
10	Binarización	CANNY	K-Means	0.57
11	Binarización	CANNY	SVM	0.71

NY				
12	Binarización	SIFT	K-Means	0.64
13	Binarización	SIFT	SVM	0.5
14	Ecualización de histogramas	HOG	K-Means	0.64
15	Ecualización de histogramas	CAN NY	K-Means	0.79
16	Ecualización de histogramas	CAN NY	SVM	0.71
17	Ecualización de histogramas	SIFT	KNN	0.57
18	Ecualización de histogramas	SIFT	K-Means	0.79
19	Ecualización de histogramas	SIFT	SVM	0.79

Sin embargo no se logró afirmar o negar que las combinaciones o imágenes pudieran ser el principal problema, debido a que existieron otros casos en particular, que obtuvieron resultados por debajo del 45%, siendo resultados más favorables que llegaron a la aproximación del 10 % del error de clasificación, como se observa en la Tabla 15.

Tabla 15. Resultados con errores de clasificación bajos.

1	Escala de grises	HOG	KNN	0.36
2	Escala de grises	SIFT	KNN	0.43
3	Binarización	HOG	K-Means	0.14
4	Binarización	CANNY	KNN	0.43
5	Binarización	SIFT	KNN	0.43
6	Ecualización de histogramas	HOG	KNN	0.43
7	Ecualización de histogramas	HOG	SVM	0.36
8	Ecualización de histogramas	CANNY	KNN	0.36

Detallando y separando estos resultados en tablas distintas se percató la mejor combinación que se aproximó al margen de error del 10%, el cual se detalla en la Tabla 16. Siendo uno de los resultados más favorables y acertados, demostraron que los algoritmos en ciertas combinaciones pueden detallar resultados esperados y deseados para dar notoriedad que el sistema funciona correctamente.

Tabla 16. Resultado con el mejor margen de error.

1	Binarización	HOG	K-Means	0.14
---	--------------	-----	---------	------

3.2 Resultados de la clasificación

En las distintas imágenes que se presentan a continuación, se puede observar el proceso de clasificación mediante la selección de los distintos modelos de algoritmos que fueron implementados, tanto como son los de preprocesamiento, los de extracción de características y los de clasificación, teniendo los siguientes resultados mostrados en las respectivas Figuras 14 y 15. Cabe mencionar que cada fotografía presenta una etiqueta correspondiente a su tipo de modelo y marca vehicular, el cual para estos 4 ejemplos en particular, el algoritmo ha clasificado con éxito, cada uno de ellos.



Figura 14. Camioneta JAC HFC.

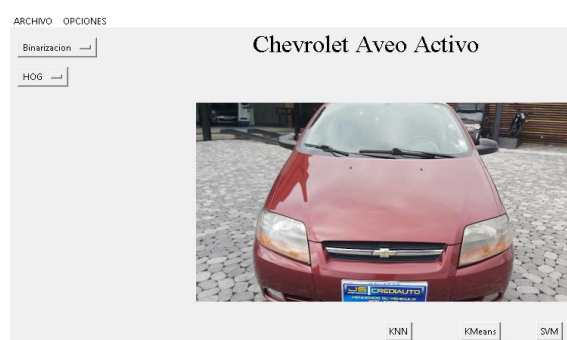


Figura 15. Chevrolet Aveo Activo.

3.3 Métricas de evaluación

Las métricas que se implementaron para la evaluación de las distintas combinaciones de algoritmos, se presentaron a través del mejor resultado en la Tabla 16. Estas métricas obtuvieron márgenes de diferencia entre los distintos porcentajes resultantes para cada imagen clasificada.

El auto es de la marca: [7] = Kia Picanto
Sensibilidad: 0.9643
Precision: 0.9643
Exactitud: 0.9643

Figura 16. Ejemplo 1, resultado de la implementación de métricas.

El auto es de la marca: [7] = Kia Picanto
Sensibilidad: 0.8534
Precision: 0.8503
Exactitud: 0.8543

Figura 17. Ejemplo 2, resultado de la implementación de métricas.

En las Figuras 16 y 17 se pueden observar que en ambos ejemplos, las métricas de evaluación reflejaron resultados distintos, siendo que en cada caso se utilizó el mismo algoritmo de combinación pero para un tipo de imagen de prueba diferente, esto permitió analizar cómo las métricas pueden evaluar la interacción de los distintos algoritmos, el cual procedió en la clasificación de la imagen otorgando un resultado asertivo.

También se requiere detallar la última métrica que se implementó, el cual fue la matriz de confusión. Esta métrica permitió encontrar resultados en lo referente al desempeño y valoración de los algoritmos con la mejor combinación para la clasificación, siendo estos **“Binarización, Hog y K-Means”**. Esta herramienta se conformó por etiquetas reales y de predicción (fueron los que el algoritmo tomó para realizar el entrenamiento), definido por sus 10 tipos de modelos y marcas vehiculares, se decidió crear un diccionario de etiquetas, que se presenta a continuación:

- 0 - Camioneta JAC HFC.
- 1 - Chevrolet Aveo Activo.
- 2 - Chevrolet Beat.
- 3 - Chevrolet Captiva.
- 4 - Chevrolet D Max.
- 5 - Great Wall Wingle.
- 6 - Kia Picanto.
- 7 - Kia Rio.
- 8 - Kia Sportage.
- 9 - Toyota Prius.

Como se observa en la Tabla 17, el número de etiquetas reales y de predicción fueron iguales al número de pruebas totales. Estos resultados comprobaron la efectividad de la automatización de los algoritmos, al predecir con efectividad cada modelo y marca del vehículo, teniendo como error un 14% que no representó más de 2 imágenes mal clasificadas.

Tabla 17. Resultados de la Matriz de Confusión.

n=14	Etiquetas de predicción										
	0	1	2	3	4	5	6	7	8	9	
Etiquetas reales	0	3	0	0	0	0	0	0	0	0	3
	1	0	5	0	0	0	0	0	0	0	5
	2	0	0	0	0	0	0	0	0	0	0
	3	0	0	0	2	0	0	0	0	0	2
	4	0	0	0	0	0	0	0	0	0	0
	5	0	0	0	0	0	0	0	0	0	0
	6	0	0	0	0	0	0	0	0	0	0
	7	0	0	0	0	0	0	0	1	0	1
	8	0	0	0	0	0	2	0	0	0	2
	9	0	0	0	0	0	0	0	0	0	1
											14/14

Para terminar y dar cierre a esta sección, se muestran los resultados de las métricas de evaluación que se realizaron a las distintas combinaciones de algoritmos presentadas en la Tabla 18.

Tabla 18. Resultados de las métricas de evaluación por algoritmos de combinación.

Algoritmos				Métricas de evaluación		
#	1- 2- 3-	Preprocesamiento Extracción de características Clasificación	4- 5- 6-	Sensibilidad Precisión Exactitud		
	1	2	3	4	5	6
1	Grises	HOG	KNN	79.01	79.01	78.18
2	Grises	HOG	KMeans	61.36	60.25	59.03
3	Grises	HOG	SVM	64.45	63.99	64.00
4	Grises	CANNY	KNN	74.02	73.78	73.78
5	Grises	CANNY	KMeans	55.89	52.00	55.00
6	Grises	CANNY	SVM	45.57	46.28	47.12
7	Grises	SIFT	KNN	80.58	81.11	79.95
8	Grises	SIFT	KMeans	50.98	54.10	49.02
9	Grises	SIFT	SVM	68.69	69.36	67.74
10	Bina.	HOG	KNN	75.47	75.18	75.99
11	Bina.	HOG	KMeans	98.11	96.63	98.00
12	Bina.	HOG	SVM	79.98	79.98	79.98
13	Bina.	CANNY	KNN	87.97	88.70	88.70
14	Bina.	CANNY	KMeans	63.52	62.58	62.58

15	Bina.	CANNY	SVM	52.14	50.15	53.27
16	Bina.	SIFT	KNN	73.30	73.30	73.30
17	Bina.	SIFT	KMeans	60.28	61.36	61.99
18	Bina.	SIFT	SVM	71.96	70.77	69.88
19	Ecua.	HOG	KNN	75.00	74.22	76.52
20	Ecua.	HOG	KMeans	77.87	75.99	77.99
21	Ecua.	HOG	SVM	81.27	80.04	79.58
22	Ecua.	CANNY	KNN	77.00	80.60	75.22
23	Ecua.	CANNY	KMeans	54.01	51.10	51.00
24	Ecua.	CANNY	SVM	60.77	62.64	60.07
25	Ecua.	SIFT	KNN	75.78	75.99	78.00
26	Ecua.	SIFT	KMeans	57.10	57.89	57.00
27	Ecua.	SIFT	SVM	62.55	64.62	61.89

4. Conclusiones

Los algoritmos de preprocesamiento permitieron la modificación automática del número de píxeles para distribuir las partes más notorias de la imagen, eliminando el ruido de fondo y permitiendo el enfoque a los rasgos más generales de cada modelo vehicular.

El redimensionamiento que se aplicó a cada imagen que forma el conjunto de entrenamiento, permitió la eficaz extracción de los vectores y puntos característicos de una fotografía para que así, pueda ser reconocida y diferenciada de todos los 10 modelos pertinentes.

Los algoritmos de clasificación permitieron agrupar a los distintos modelos y marcas de vehículos, comparándolos con todas las imágenes que fueron modificadas y segmentadas, otorgando resultados que interpretan la cercanía y similitud, emparejando a una etiqueta perteneciente a un modelo de vehículo en particular.

El uso de métricas de evaluación permite observar un grado de aceptación pertinente y destacar que los resultados que se obtengan van a permitir concluir si los algoritmos implementados están

funcionando correctamente, discrepando sus resultados variantes. Esto quiere indicar que las calificaciones que son predecidas correctamente deben tener un margen mínimo para que exista prudencia con sus resultados, en caso contrario demostraría que algo anda mal en el algoritmo de clasificación, sin descartar que la calidad de una imagen pueda influir en esos resultados.

5. Referencias

[1] Acervo Lima. (27 de 12 de 2021). Acervo Lima. Recuperado el 20 de 04 de 2022, de Acervo Lima: <https://es.acervolima.com/procesamiento-de-imagenes-sin-opencv-python/>

[2] Nelli, F. (13 de 4 de 2017). MECCANISMO COMPLESSO. Recuperado el 21 de 4 de 2022, de MECCANISMO COMPLESSO: <https://www.meccanismocomplesso.org/en/opencv-python-the-otsus-binarization-for-thresholding/>

[3] ACODIGO. (17 de 8 de 2017). ACODIGO. Recuperado el 21 de 4 de 2022, de ACODIGO: <http://acodigo.blogspot.com/2017/08/histog-ramas-opencv-python.html>

[4] Castrillon, W. A., Alvarez, D. A., & López, A. F. (2008). Técnicas de extracción de características en imágenes para el reconocimiento de expresiones faciales. Pereira: Universidad Tecnológica de Pereira.

[5] Ysiquio, A. N. (2004). Modelado de Sistemas de control de un robot manipulador basado en procesamiento digital de imágenes. Cholula, Puebla, México: Universidad de las Américas Puebla.

[6] Pardo, C. J. (4 de 12 de 2017). Visión por computador. Recuperado el 21 de 4 de 2022, de Visión por computador: <https://carlosjuliopardoblog.wordpress.com>

[7] Unipython. (5 de 4 de 2018). Unipython. Recuperado el 22 de 4 de 2022,

de Unipython: <https://unipython.com/algoritmo-de-canny/>

[8] Pizarro, D. A., & Alvarado, N. D. (12 de 2013). Detección de puntos claves mediante SIFT paralelizado en GPU. Chile: Escuela Universitaria de Ingeniería Industrial, Informática y Sistemas.

[9] Garcia, C., & Gómez, I. (s.f.). Algoritmos de aprendizaje: KNN y KMEANS. Madrid, España: Universidad Carlos III de Madrid.

[10] Camacho, J. A. (12 de 8 de 2021). jacobsoft.com. Recuperado el 22 de 4 de 2022, de jacobsoft.com: https://www.jacobsoft.com.mx/es_mx/k-nearest-neighbors/

[11] Adrimelus. (30 de 10 de 2020). adrimelus. Recuperado el 23 de 4 de 2022, de adrimelus: <https://adrimelus.com/blog/algoritmo-kmeans-explicacion-e-implementacion-en-python/>

[12] IBM. (17 de 8 de 2021). IBM. Recuperado el 23 de 4 de 2022, de IBM: <https://www.ibm.com/docs/es/spss-modeler/SaaS?topic=networks-neural-model>

[13] IBM. (17 de 8 de 2021). IBM. Recuperado el 23 de 4 de 2022, de IBM: <https://www.ibm.com/docs/es/spss-modeler/SaaS?topic=models-how-svm-works>

[14] Shin, T. (5 de 2020). DATASOURCE.AI. Recuperado el 23 de 4 de 2022, de DATASOURCE.AI: <https://www.datasource.ai/es/data-science-articles/compreension-de-la-matriz-de-confusion-y-como-implementarla-en-python>

[15] Álvarez Durán, M. A. (2014). Análisis, diseño e implementación de un sistema de control de ingreso de vehículos basado en visión artificial y reconocimiento de placas en el parqueadero de la

Universidad Politécnica Salesiana-Sede Cuenca (Bachelor's thesis).

[16] Herrera González, N. M., & Chamaidan Asencio, D. A. (2019). Plataforma tecnológica para contribuir a la planeación urbana de la ciudad de Guayaquil dirigido a la transportación (Bachelor's thesis, Universidad de Guayaquil Facultad de Ciencias Matemáticas y Físicas Carrera de Ingeniería en Sistemas Computacionales).

[17] Torres Cabrera, M. D. (2020). Reconocimiento automático de la placa de un vehículo de Ecuador (Bachelor's thesis).

[18] User, S. (2021, 29 mayo). Inteligencia Artificial AI - Ecuador. [www.Grupomundodigital.Ec.](https://www.grupomundodigital.ec/corporativo/inteligencia-artificial-ai.html)
<https://www.grupomundodigital.ec/corporativo/inteligencia-artificial-ai.html>

[19] Peña, J. A. (29 de 4 de 2014). EOIES. Recuperado el 25 de 4 de 2022, de EOIES:
<https://www.eoi.es/blogs/embacon/2014/04/29/las-5-fases-en-gestion-de-proyectos/>

[20] Pallares, M. G. (2021). Detección de incidentes automovilísticos usando técnicas de aprendizaje de máquina. Quito, Ecuador: Escuela Politécnica Nacional.

[21] Jarrín, D., & Riofrío, D. (2019). Estudio comparativo entre modelos de aprendizaje profundo, desarrollados a partir de redes neuronales recurrentes a redes neuronales convolucionales, para la detección de intrusos de red. Universidad Internacional SEK.

[22] Maldonado, D. F. (2006). Diseño de un módulo de preprocesamiento y extracción de características físicas de imagen para una aplicación de reconocimiento automático de número de placa de automóviles. Quito, Ecuador: Escuela Politécnica Nacional.

[23] Agudelo, D., Silva, J., & Gil, J. D. (2013). ESTIMACIÓN DE LA LOCALIZACIÓN DE UN VEHÍCULO.

Pereira: Universidad Tecnológica de Pereira.

[24] Justin, R., & Kumar, D. (2018). Vehicle Detection and Counting Method Based on Digital Image Processing in. IJEECS.

[25] Zuenko, D., & Makarov, I. (2020). Real-Time Vehicle Type Detection and Counting from. Moscow, Russia: CEUR Workshop Proceedings.

[26] Mustafa, S. M. (2019). VEHICLE DETECTION AND TRACKING USING. School of Applied Sciences of Near East University.

[27] Zamora, M. I. (2015). Diseño de algoritmos para la detección de adelantamientos de vehículos en plataforma Android. España: Atribución-NoComercial-SinDerivadas 3.0.

[28] Martínez, R. (2018). Implementación de un sistema de visión artificial para detección de coches en tiempo real mediante Caffe2 y C++. Madrid, España: E.T.S.I. Telecomunicación (UPM).

[29] Comas, S., & Taborda, F. (2019). Investigación e implementación de algoritmos de visión asistida aplicados a estimación de flujo vehicular y velocidad de automóviles.

[30] Olafenwa, M., & Olafenwa, J. (2021). ImageAI. Recuperado el 25 de 4 de 2022, de ImageAI:
<https://imageai.readthedocs.io/en/latest/prediction/>

[31] ImageAI. (2021). ImageAI. Recuperado el 25 de 4 de 2022, de ImageAI:
<https://imageai.readthedocs.io/en/latest/index.html>

[32] Myrianthous, G. (11 de 4 de 2021). Towardsdatascience. Recuperado el 25 de 4 de 2022, de Towardsdatascience:
<https://towardsdatascience.com/how-to->

[split-a-dataset-into-training-and-testing-sets-b146b1649830](#)

[33] Yin, S. W. (s.f.). programador clic. Recuperado el 25 de 4 de 2022, de programador clic: <https://programmerclick.com/article/1169747909/>

[34] Mi Diario Python. (11 de 1 de 2019). Mi Diario Python. Recuperado el 25 de 4 de 2022, de Mi Diario Python: <https://pythondiario.com/2019/01/matrices-en-python-y-numpy.html>

[35] Emmanuelle Gouillart. (s.f.). claudiovz. Recuperado el 26 de 4 de 2022, de claudiovz: https://claudiovz.github.io/scipy-lecture-notes-ES/advanced/image_processing/index.html

[36] facundoq. (2018). facundoq. Recuperado el 26 de 4 de 2022, de facundoq: https://facundoq.github.io/courses/aa2018/res/04_imagenes_numpy.html

[37] Conexión ESAN. (11 de 10 de 2018). esan. Recuperado el 25 de 4 de 2022, de esan: <https://www.esan.edu.pe/conexion-esan/la-importancia-del-pmbok-y-su-influencia-en-un-proyecto>

[38] JUANBARRIOS. (s.f.). JUANBARRIOS. Recuperado el 27 de 4 de 2022, de JUANBARRIOS: <https://www.juanbarrios.com/redes-neurales-convolucionales/>

[39] datacentric. (19 de 5 de 2021). datacentric. Recuperado el 27 de 4 de 2022, de datacentric: <https://www.datacentric.es/blog/insight/red-neuronal-artificial-aplicaciones/>

[40] Arrabales, R. (29 de 3 de 2016). Xataka. Recuperado el 26 de 4 de 2022, de Xataka: <https://www.xataka.com/robotica-e-ia/deep-learning-que-es-y-por-que-va-a-ser-una-tecnologia-clave-en-el-futuro-de-la-inteligencia-artificial>

6. ANEXOS

Segmentos de código

Tabla 19. Código de la interfaz gráfica.

```
#creacion de ventana
ventana=Tk()
w=1000
h=500

extraW=ventana.winfo_screenwidth()-w
extraH=ventana.winfo_screenheight()-h
r = StringVar()

ventana.geometry("%dx%d%d%d"%(w,h,extraW/2,extraH/2))
canvas=Canvas(ventana,width=1000, height=650)
canvas.pack()

itemsGBH = ['Escala de grises', 'Binarizacion', 'Ecuilizacion de histogramas']
itemsHCS = ['HOG', 'CANNY', 'SIFT']

tipo_imagen1 = StringVar(ventana)
tipo_imagen2 = StringVar(ventana)

tipo_imagen1.set('Escala de grises')
tipo_imagen2.set('HOG')

gbh_opciones= OptionMenu(ventana, tipo_imagen1, *itemsGBH)
hcs_opciones= OptionMenu(ventana, tipo_imagen2, *itemsHCS)

gbh_opciones.place(x=10, y=10)
hcs_opciones.place(x=10, y=50)

ventana.title("Clasificación de imagenes")
Label(textvariable=r, font=("Times New Roman",25)).place(x=320,y=0)

btn_clasificar = Button(text="KNN", command=clasificar_KNN)
btn_clasificar.place(x=500, y=400)
btn_clasificar = Button(text="KMeans", command=clasificar_KMeans)
btn_clasificar.place(x=600, y=400)
btn_clasificar = Button(text="SVM", command=clasificar_SVM)
btn_clasificar.place(x=700, y=400)
```

Tabla 20. Creación de menús del sistema.

```
#creacion de menus
barraMenu=Menu(ventana)
mnu1OpFunda=Menu(barraMenu)
mnuOpciones=Menu(barraMenu)

mnu2FiltroR=Menu(barraMenu)

mnu3OperacMorfolog=Menu(barraMenu)
mnu4OperacSeg=Menu(barraMenu)

#submenus unidad-operaciones fundamentales
mnu1OpFunda.add_command(label="Abrir imagen", command=abrir)
mnu1OpFunda.add_separator()

#SEGMENTACION
mnuOpciones.add_command(label="Limpiar", command=limpiar)
mnuOpciones.add_separator()
mnuOpciones.add_command(label="Salir", command=ventana.destroy)
mnuOpciones.add_separator()

#carga de la ventana
barraMenu.add_cascade(label="ARCHIVO", menu=mnu1OpFunda)
barraMenu.add_cascade(label="OPCIONES", menu=mnuOpciones)

ventana.config(menu=barraMenu)
ventana.mainloop()
```
