risti
Revista Ibérica de Sistemas e Tecnologias de Informação
Revista Ibérica de Sistemas y Tecnologías de Información

Recebido/Submission: 20/12/2022
Aceitação/Acceptance: 28/02/2023

Determinación del mejor algoritmo de detección de matrículas en ambientes controlados y no controlados

Elias Ccoto Huallpa[1], Abel Angel Sullon Macalupu[1], Jorge Eddy Otazu Luque[1], Jorge Sánchez-Garces[2]

elias.ch@upeu.edu.pe; angeli@upeu.edu.pe; jorgeol@upeu.edu.pe; jasg@upeu.edu.pe.

[1] Escuela Profesional de Ingeniería de Sistemas, Universidad Peruana Unión, Carretera Salida ArequipaKm 6 Chullunquiani, Juliaca, Puno, Perú.

[2] Facultad de Ingeniería y Gestión Universidad Nacional Tecnológica de Lima Sur, Lima, Peru.

DOI: 10.17013/risti.49.83–99

Resumen: La seguridad es una prioridad en la gestión y parte de esto es el control y monitoreo para detectar situaciones que agredan el bien público o privado. En este sentido el reconocimiento de matrículas de auto se suma a estos sistemas de monitoreo y control. Este trabajo de investigación aplicó una serie de algoritmos de inteligencia artificial para automatizar dicha detección. En este sentido se utilizó funciones de procesamiento de imágenes con el framework OpenCV considerando que las fuentes de información pudieron tener distitnos escenarios, siendo que el ambiente de detección es abierto y expuesto a las condiciones meteorológicas de la zona. Las tomas de fotos se realizaron en un ambiente ubicado al sur peruano, cuyas condiciones fueron de lluvia, día soleado, día que cayó el granizo. La base de datos de las imágenes entonces se dividió en dos categorías; ambientes controlados donde se consideró una misma distancia, un solo ángulo, pero no necesariamente un mismo clima; y los ambientes no controlados con diferentes ángulos, diferentes distancias y climas. Al procesamiento de imágenes también se utilizó la transformación morfológica, suavizado gaussiano y umbral gaussiano. Con las imágenes segmentadas y extraídos los dígitos de la matrícula; se comparó 3 algoritmos K-NN, SVM y Tesseract. en cada algoritmo se utilizó sus hiperparámetros para el respectivo reconocimiento de caracteres en las imágenes, se obtuvo los siguientes resultados con imágenes tomadas con distintos ángulos y en distintas luminosidades (ambiente no controlado) donde el mejor Overall accuracy con 86 % y el segundo grupo fueron imágenes tomadas con un ángulo similar y distancias similares (ambiente controlado), obtuvo un Overall accuracy de 95.5 %.

Palabras-clave: KNN; SVM; Tesseract; OpenCV; Machine Learning y hiperpárametros.

Determination of the best license plate detection algorithm in controlled and uncontrolled environments.

Abstract: Security is a priority in management and part of this is control and monitoring to detect situations that attack the public or private good. In this sense, the recognition of car license plates is added to these monitoring and control systems. This research work applied a series of artificial intelligence algorithms to automate such detection. In this sense, image processing functions were used with the OpenCV framework, considering that the information sources could have different scenarios, since the detection environment is open and exposed to the weather conditions of the area. The photos were taken in an environment located in southern Peru, whose conditions were rainy, sunny, and the day the hail fell. The image database was then divided into two categories; controlled environments where the same distance, a single angle, but not necessarily the same climate was considered; and uncontrolled environments with different angles, different distances and climates. Morphological transformation, Gaussian smoothing and Gaussian thresholding were also used for image processing. With the segmented images and the number plate digits extracted; 3 algorithms K-NN, SVM and Tesseract were compared. In each algorithm, its hyperparameters were used for the respective recognition of characters in the images, the following results were obtained with images taken with different angles and in different luminosities (uncontrolled environment) where the best Overall accuracy with 86% and the second group were images taken with a similar angle and similar distances (controlled environment), obtained an Overall accuracy of 95.5%.

Keywords: Tesseract; OpenCV; Machine Learning and hyperparameters.

1. Introducción

En la actualidad hay un aumento en vehículos teniendo un aumento proporcional; según (SUNARP, 2020)en Perú hasta el año 2019 hay más de 701 mil vehículos registrados teniendo un aumento del 6.4 %, comparado al 2018, por lo tanto, las infracciones vehiculares y el robo de vehículos incrementaron. Según (INEI, 2021) informa que, hubo más 20000 denuncias de vehículos robados en el Perú durante el 2019 teniendo un promedio de crecimiento de 5 % anual; tan solo en la capital de Perú se registró 17000 denuncias por robo, siendo el departamento del país con más denuncias. Agravando el problema con la clonación de placas según (SEMARNAT, 2006).

El desarrollo de modelos computacionales permiten resolver el problema planteado, automatizando el proceso de detección para identificar los objetos involucrados en posibles actos de delincuencia, junto a esto se evalúan estos modelos para que la detección esta una excelente precisión, pero previo al uso de estos modelos de detección es necesario contar con algoritmos que permiten el procesamiento de las imágenes siendo que estas pueden ser tomadas en distintos escenarios , climas, distancias, ángulos. Se realizó una investigación de los trabajos actuales sobre este tipo de soluciones.

El autor (Agarwal et al., 2018) utilizó el procesamiento morfológico(escala de grises), y el autor (Divya et al., 2021) utilizó tesseract esto para el reconocimiento de manuscritos, el autor (Varma et al., 2020) realizó sus investigaciones para el reconocimiento de matrículas con imágenes capturadas en carreteras de Irak esto en ambientes no controlados usando trasformaciones morfológicas, el autor (Silva & Jung, 2020) para mejorar el reconocimiento de matrículas en ambientes con un mínimo de luminosidad utilizó redes neuronales.

Los autores citados mencionaron el uso de algoritmos de aprendizaje profundo, supervisados y no supervisados, pero no diferenciaron escenarios como controlados y no controlados, que para la implementación en ambientes abiertos y públicos es necesario validar dicho aspecto.

Por lo tanto, la motivación fue hallar un modelo de detección de placas de vehículos para ambientes controlados adecuada y un modelo para detección de placas para ambientes no controlados.

Para esta comparación se usó 2 tipos de algoritmos; siendo el primero el Tesseract con su algoritmo LSTM(Long short-term memory) y los algoritmos supervisados de Support vector machine y K-nn(K-Nearest Neighbors); ambos tuvieron con OpenCv el procesamiento de imágenes con los algoritmos de transformación morfológica, suavizado gaussiano y umbral gaussiano. Con estos algoritmos la precisión con que se detecta los dígitos de la matrícula del auto es mucho mejor se en ambientes controlados o no controlados; logrando otorgar una solución para el monitoreo y acceso de los vehículos a espacios públicos o privado; tales como universidades, supermercado y zonas de aparcamiento con el objetivo de reducir las infracciones y robos.

2. Materiales

- Cámara para la captura de imágenes en ambiente controlados: Cámara YCC365, HD, 1280*1024
- Cámara para la captura de imágenes en ambiente no controlados: Smart phone LG Q60, cámara principal 16MP + 2MP + 5MP y cámara frontal 13MP

3. Muestra

Para hallar un modelo adecuado para cada ambiente ya mencionado en la introducción, siendo el objetivo del estudio se consideró los valores de cada hiperparámetro de los algoritmos para la detección y reconocimiento de caracteres. El proceso se describe en la tabla 01.

Algoritmo	hiperparámetros	Valores
KNN	K.	1,2,3,4,5,6,7,8,9,10
SVM	C.	1, 10, 100, 1000, 0.1, 1
	Gamma.	0.001, 0.0001, 1, 0.1, 0.01, 1
	Kernel.	cv2.ml.SVM_LINEAR,cv2.ml.SVM_RBF, cv2.ml.SVM_CHI2,cv2.ml.SVM_SIGMOID, cv2.ml.SVM_POLY
	Type.	cv2.ml.SVM_EPS_SVR,cv2.ml.SVM_C_SVC, cv2.ml.SVM_NU_SVC,cv2.ml.SVM_ONE_CLASS, cv2.ml.SVM_NU_SVR
	Degree.	1, 2, 3
	P.	0, 0.1
LSMT	Psm.	1,2,3,4,5,6,7,8,9,10,11,12,13

Tabla 1 – Valores de los hiperparámetros de los algoritmos

Una vez obtenido los valores de los hiperparámetros; se tomó la primera muestra, donde se utilizó 80 imágenes en ambientes no controlados estas tomadas con distintos ángulos, distintas distancias. El objetivo fue hallas el mejor hiperparámetro de cada algoritmo descrito en la tabla 02, aplicando los valores mencionados de los hiperparámetros a cada una de las imágenes de las 80.

Algoritmo	hiperparámetros	Cantidad hiperparámetros	Muestra Total
KNN.	K.	10	80*10= 800
SVM.	C, Gamma, Kernel, Type, degree y P.	144	80*144= 11520
LSMT.	Psm.	13	80*13= 1040

Tabla 2 – Muestras por cada algoritmo de la primera ejecución del modelo

Para las muestras en controlados se utilizó 100 imágenes con distancias similares (3 a 5 metros). La cámara que se utilizó fue colocada en control de ingreso de una institución educativa peruana. El objetivo fue hallar los mejores hiperparámetros de cada algoritmo en la totalidad de las muestras descrito en la tabla 03.

Algoritmo	hiperparámetros	Cantidad hiperparámetros	Muestra Total
KNN	K	10	100*10= 1000
SVM	C, Gamma, Kernel, Type, degree y P	144	100*144= 14400
LSMT	psm	13	100*13= 1300

Tabla 3 – Muestras por algoritmo para la segunda ejecución del modelo.

4. Metodología

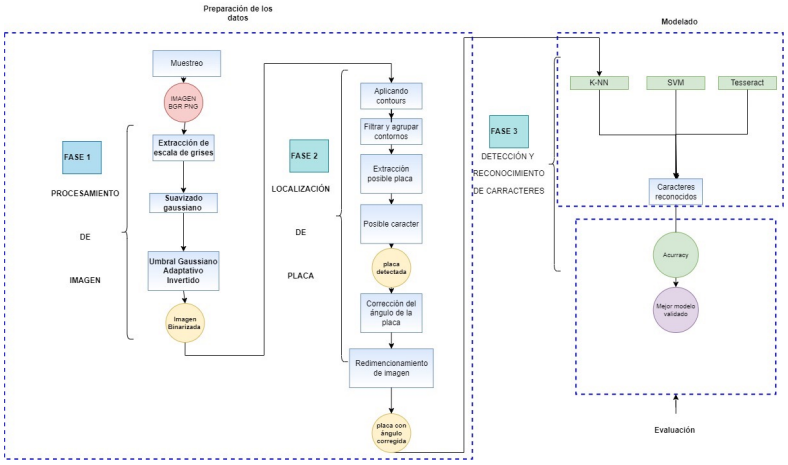

Figura 1 – Método de la investigación

En en la figura 01 se muestra la metodología que se utilizó en esta investigación y donde tiene 3 fases principales utilizadas de la metodología CRISP-DM, se utilizó esta metodología por la flexibilidad que tiene de poder adaptarla al contexto situacional del problema que se abordó. Estas 3 fueron la preparación de los datos y consistió en el procesamiento de imágenes, la localización de placas; la segunda fue el modelado que consistió en la codificación y comparación de tres algoritmos para el reconocimiento de las placas y por último la evaluación que pudo encontrar el mejor hiperparámetro en cada muestra del ambiente controlado y no controlado para obtener el mejor modelo de predicción.

4.1. Nivel de investigación

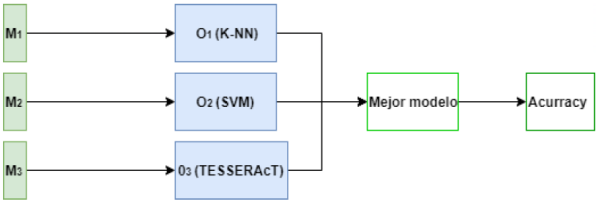

Figura 2 – Nivel de Investigación

Donde:

M = Muestra

O = Observación

La investigación es cuantitativa de nivel descriptiva comparativo porque se realizaron análisis comparativos entre algoritmos computacionales determinando el algoritmo prevalente en función al mejor accuracy así obteniendo el mejor modelo computacional.

4.2. Preparación de los datos: Tratamiento de imagen

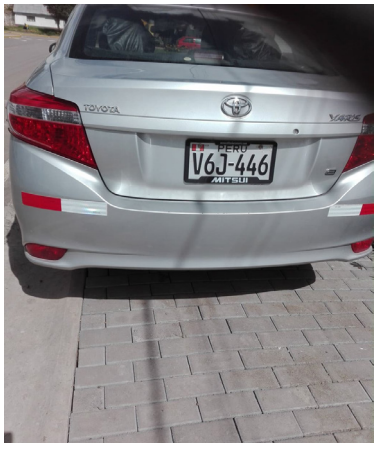

Figura 3 – Vehículo a detectar

Para el tratamiento de imágenes se usa herramientas de OpenCv(The Open Computer Vision). Según (Arévalo et al., 2016), la librería OpenCV permite el procesamiento de imágenes, así como su tratamiento asi pudiendo desarrollar aplicaciones de visión por computador. La entrada fue una imagen RGB, en formato PNG.

I. *Extracción de escala de grises:* en esta fase a la imagen entrante se le aplica la función imgGrayscale, asi se obtinecada píxel tiene un valores de entre 0 y 255, siendo que el valos mas cercano a cero son las más oscuros y los valores más proximo a 255 son las más claros.
II. *Suavizado gaussiano:* Utiliza la función gaussiana lineal. La principal función del filtrado gaussiano es reducir los detalles y ruidos en las imágenes. La utilización del filtro gaussiano a una imagen tiene la ventaja de evitar errores en la detección de la placa.
III. *Umbral Gaussiano Adaptativo Invertido:* El objetimo de aplicar la umbralizacion en una imagen es crear una imagen binaria a partir de la imágen en escala de grises. Este proceso se conoce como binarización de imágenes. Se toma una ventana de tamaño predefinido y se encuentra una suma ponderada de pixeles vecinos para realizar el umbral adaptativo.

```
cv2.adaptiveThreshold(image, 255,

cv2.ADADTIVEtHRESHgAUSSIANc,

cv2.THRESHB INARYiNV,

cBLOCKs IZE, WEIGHT)
```

Aquí:

- ADADTIV4E_THRESH_GAUSSIAN_C El umbral será en función de Gaussian;
- BINARY_INV donde se binariza y se invierte la binarización;
- BLOCK_SIZE es el tamaño de la ventana de umbral.
- WEIGHT se usa para calcular la suma ponderada de los valores en la zona.

4.3. Preparación de los datos: Localización de placa

Luego del ultimo etapa de hacer unos arreglos a las imágenes, Umbral Gaussiano Adaptativo Invertido, dio como resultados una imagen binarizada, con valores de cero y 255. La imagen será utilizada como entrada para la etapa de detección y reconocimiento.

I. *Aplicando contours*: Un contorno es un vínculo de puntos de igual intensidad a lo largo del límite, (Phangtriastu et al., 2017). En OpenCV, encontrar contornos es como encontrar un objeto blanco del negro fondo, por lo tanto, durante la

etapa de Umbral Gaussiano Adaptativo, se tuvo que aplicar la operación de Inversión. La función *cv2.drawContours()* permitió aplicar contornos a una imagen binarizada.

Figura 4 – Imagen binarizada

Figura 5 – Imagen Contorno

II. *Filtrar y agrupar contornos:* Para zonas pequeñas, especialmente en los bordes afilados y valores atípicos de ruido, se aplican contornos. Así, el ojo humano puede calcular fácilmente que tales contornos son innecesarios, pero esto debe ser incorporado en el algoritmo (Abedin et al., 2018). Inicialmente, se aplicaron cuadros delimitadores a cada contorno. Luego, para cada contorno, se consideraron los siguientes factores, como el área mínima del contorno, el ancho y la altura mínimos del contorno, las relaciones de aspecto mínima y máxima posibles.

```
cv2.findContours(imgThreshCopy cv2.

RETR_LIST cv2.CHAIN_APPROX_SIMPLE)
```

Esto resultó en el filtrado de la mayoría de los contornos innecesarios, acercándonos a nuestro objetivo: detectar una placa. La segunda etapa del filtrado consta en comparar cada contorno con cualquier otro contorno en parámetros como la distancia entre los contornos. La tercera etapa es agrupar los contornos, un grupo de contornos satisface todas estas condiciones, se agrupan en uno. Es posible que se puedan obtener dos o más de tales grupos.

III. Extracción de posible placa: para esta fase se enfocó en la ubicación de las posibles placas utilizando la función cv2.boundingRect(self.contour). Esta permite ubicar las placas dentro de rectángulos, donde fueron posicionadas las imágenes.

cv2.boundingRect(selfcontour) la función detecta los rectángulos dentro de nuestra imagen, la función contour devolverá: coordenada X y Y, tanto la altura y el ancho y altura encerrando la placa en un rectángulo así como se ve en la figura 2.

```
IntboundingRectX = intX.

IntboundingRectY = intY.

IntboundingRectWidwt = intWidwt.

IntboundingRectHeight = intHeight.
```

Figura 6 – Rectángulo detectado

Obtenida todas las coordenadas y las medidas se pasan a la siguiente fase que es las extraer la posible placa delimitando y recortada del resto de la imagen como se puede ver en la figura 3.

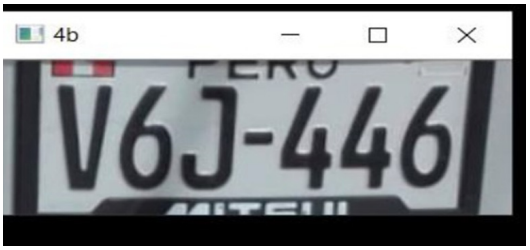

Figura 7 – Placa extraída

IV. *Posible caracter:* En esta fase se presenta otro dificultad que la identificacion de la placa correcta de un n resultados de la face anterior. En esta fase se ubicó la placa correcta, para se identifica los posibles caracteres dentro de la placa, la posible palca sera la que contenga mas caracteres para esto se emplea la función.

```
Cv2.findContours(imgThresehCopy Cv2.RETR_LIST

cv2.CHAIN_APPROX_SIMPLE)
```

La función findContours identifico y nuestros los contornos de la placa, para esto se usa parámetros: la primera Cv2.RETR_LIST ayuda a solo obtener los contornos que no sean dependientes de otros; el otro parámetro que se usa es cv2.CHAIN_APPROX_SIMPLE que mostrará las coordenadas finales de cada contorno, logrando segmentar y hacer el conto de los caracteres, así como se muestra en la figura 4; una vez realizado se obtiene las posibles placa para la identificación de los caracteres.

Figura 8 – Caracteres detectados

V. *Corrección del ángulo de la placa:* El proposito es corregir los angulos estandarizando las posibles placas para la su facil deteccionde. Se mantiene la relación de longitudes entre los puntos que residen en una línea recta. Sin embargo, los ángulos dentro de las líneas y las longitudes dentro de los puntos no se conservan. En OpenCV, la corrección del ángulo de la placa se puede lograr mediante la función getRotationMatrix2D.

```
Cv2.getRotationMatrix2D(tupleCenter)

fltCorrectionAnglenDeg 1.0)
```

VI. *Redimensionamiento de la imágen:* en estas fases se cambiaron los tamaños de la placa extraida de 20x30 en las variables RISIZED_CHAR_IMAGE_WIDTH y RISIZED_CHAR_IMAGE_HEIGHT.

Esta función garantizó que todas las imágenes sean uniformes para formato de entrada para el modelo de aprendizaje, este proceso es necesario ya que ls imágenes tomadas en ambientes no controlados fueron en distintos ángulos y distancias así perdiendo su uniformidad.

4.4.Funciones de código detección de los caracteres

Modelado: Detección y reconocimiento de caracteres

Se utilizó subprocesos en donde se usaron los algoritmos de OpenCv y los de Tesseract para el reconocimiento de caracteres de la placa.

K-NN: Según (Kumar Sahoo, 2020) es el algoritmo más sencillo para la técnica de aprendizaje automático es k-NN. La clasificación de los caracteres se realiza mediante el cálculo de la distancia Euclidiana.

La ventaja de un valor de k mayor es que se suprime el ruido, pero a la vez los límites de los caracteres serán menos distinguibles. La búsqueda aleatoria toma como entrada el modelo y una lista de hiperparámetros que se prueban durante el entrenamiento. En esta fase, se probaron valores positivos diferentes para los parámetros K, Hallando el valor para optimo K para el modelo.

SVM: Support Vector Machine es un método de aprendizaje supervisado para el análisis de clasificación y regresión (Tabrizi & Cavus, 2016). En SVM, los datos se representan en un espacio n-dimensional donde se puede predecir si un nuevo ejemplo de entrenamiento cae en la misma categoría o en otra diferente. El objetivo principal de SVM es encontrar un hiperplano en el espacio n-dimensional que pueda clasificar los puntos de datos. (Singh et al., 2021)menciona que se podría elegir varios hiperplanos potenciales para distinguir las dos clases de puntos de datos. Pero el hiperplano ideal es el que maximiza el margen entre los puntos de datos de ambas clases. Los hiperplanos son límites para la toma de decisiones que ayudan a distinguir los puntos de datos. Los puntos de datos que caen a ambos lados del hiperplano pueden atribuirse a varias clases. La dimensión del hiperplano depende del número de características. El hiperplano se puede encontrar usando la siguiente ecuación.

Para obtener la mejor predicción para SVM se utilizaron distintas combinaciones de hiperparámetros siendo: Kernel, gamma, C, type1, degree y P obteniendo 144 de posibles combinaciones el valor de estos hiperparámetros

#	Hiperparámetros	Valores
1	C	1, 10,100, 1000 y 0.1
2	Gamma	0.001, 0.0001, 0.1, 0.01 y 1
3	Kernel	Lineal, Polynomial, Sigmoid, RBF y CHI2
4	Type	EPS_SVR, C_SVC, NU_SVC, ONE_CLASS y UN_SVR
5	Degree	1, 2 y 3
6	P	0 y 0.1

Tabla 4 – Valores de los hiperparámetrosde svm

TESSERAC: Tesseract es un motor OCR y es un código abierto más usado sobre todo en imágenes que contienen textos(Divya et al., 2021).

Se usó el parámetro PSM (page segmentation modes, en inglés) de la librería Pytesseract que hace referencia a la forma de segmentación en este caso de la placa. Cada número refiere a una forma de segmentación.

Estos parámetros se utilizaron para hallar el modelo optimo mediante el acurracy para la detección y reconocimiento de placas.

5. Resultados

En esta parte se presentaron los resultados experimentales de la localización de matrículas y, además, se presentó una comparación de los algoritmos SVM. KNN y el algoritmo de Tesseract. Para cada algoritmo se utilizó las mismas funciones de procesamiento de imágenes con OpenCV; donde se obtuvo imágenes segmentadas y binarizadas, Estas imágenes fueron tomadas como input para cada algoritmo, así como se muestra en la figura 7.

5.1. Primera Ejecución del Modelo

Para esta primera ejecución se utilizó ochenta imagenes con ángulos y distancias distintas

KNN: Se uso el hiperparámetro k con 10 valores que se puede ver en la tabla 1, opteniendo el valos del hiperparámetro con el valor de 4 con el mejor valor Overall Accuracy de 0,841 tal como se muestra en la tabla 5.

#	K	Accuracy
1.	K4.	0,841
2.	K3.	0,818
3.	K1.	0,807
4.	K5.	0,798
5.	K2.	0,782
6.	K7.	0,763
7.	K6.	0,760

#	K	Accuracy
8.	K8.	0,751
9.	K9.	0,747
10.	K10.	0,733

Tabla 5 – Acurracy de los hiperparámetros de knn.

Hecha las pruebas el k-vecinos resulto tener un valor de 4 como el más óptimo, este favor fuel el más óptimo para el reconocimiento de caracteres, esto tiene sentido ya que cuando K es más cercano al valor de 1, la predicción puede ser más exacta, pero a la vez es más probable que pueda confundir caracteres similares como por ejemplo la "0" con la "O" ya no tiene más K-vecinos para validar su predicción; al contrario, cuando K es más alejado de 1 tiende a errar la predicción ya que tiene demasiados K-vecinos que confunde al algoritmo, esto fue demostrado en los resultados detallados en la tabla 2; por lo cual el valor adecuado para K fue 4 por alejado al 10 y cercano a 1.

*SVM: S*e usaron 6 hiperparametros con valores diferentes detallado en la tabla 1, obteniendo 144 combinaciones de hiperparametros,estor fueron probadas con las 100 imágenes y resultó un total de 11520 muestras para la evaluación del modelo SVM (ver tabla 3). La mejor convinacion de hiperparámetro tuvo los siguientes valores: C = 1, Gamma = 1, Kernel = Lineal, Type1 = C_SVC, Degree = 1 y P = 0 teniendo un Overall accuracy de 0,864 asi como se detalla ver tabla 5.

#	C	gamma	kernel	type1	degree	P	Accuracy
1	1	1	Lineal	C_SVC	1	0	0,864
2	1	0,001	Lineal	C_SVC	3	0	0,844
3	100	1	Lineal	C_SVC	1	0	0,842
4	1	0,001	Lineal	C_SVC	1	0	0,842
5	1	1	Lineal	C_SVC	2	0	0,841
6	100	1	Lineal	C_SVC	2	0	0,840
7	100	1	Lineal	C_SVC	3	0	0,840
8	10	0,0001	Lineal	C_SVC	1	0	0,840
9	10	0,0001	Lineal	C_SVC	2	0	0,840

Tabla 6 – Acurracy de los hiperparámetros de svm.

Resultó con un Kernel lineal, siendo este fue el más rápido y sencillo a la hora del mapeo de muestras; type1 fue C_SVC, el hiperparámetro trabaja adecuadamente con un Kernel Lineal y depende de los valores del hiperparámetro C; si C es cercano a cero no se consideraran puntos cercanos del núcleo en el mapeo así como se muestra en la figura 7 y cuanto más alejado de cero los puntos más cercanos serán considerados en el mapeo; El hiperparámetro C dió como valor 1 siendo el óptimo para nuestro modelo ya que este no cuenta con muchas penalizaciones en el mapeo siendo más preciso en los resultados.

TESSERACT: se considero el parametro (psm), teniendo valores del 1 al 13, donde se consiguió 1040 muestras para su evaluación. Resultó tener parametro psm con un valor 9 con un Overall accuracy de 0.479 siendo el más adecuado para este framework así como se detala en la tabla 7.

#	psm	Accuracy
1	9	0,479
2	12	0,471
3	7	0,470
4	5	0,467
5	6	0,466
6	8	0,431
7	13	0,376
8	11	0,371

Tabla 7 – Acurracy del hiperparámetros de tesseract.

Estevalor 9 de PSM considera a la imagen extraída como una sola palabra en un círculo, siendo este el parámetro adecuado en el reconocimiento de caracteres, pero no alcanzó un Overall accuracy alto para ser óptimo para el modelo, esto porque Tesseract fue desarrollado para detección de textos donde no haya ruidos o manchas en la imagen y en fondos uniformes.

Segunda Ejecución del Modelo

Las Imágenes utilizadas para esta segunda ejecución se tomó 100 imágenes con distancias y ángulos similares resultando un nuevo Overall accuracy para cada uno del algoritmo se detalla en la tabla 8.

Algoritmo	Muestras	Accuracy
SVM.	100	0,955
K-NN.	100	0,863
Tesseract.	100	0,292

Tabla 8 – Overall accuracy de los mejores hiperparámetros

Se muestra en tabla 8 los resultados que variaron específicamente en uno de los algoritmos que fue SVM donde se logró obtener un 95.5% de precisión en el reconocimiento de caracteres obteniendo una mejora a comparación con las muestras de la primera ejecución, SVM tuvo una mejora considerable en la precisión al momento de discernir caracteres confusos o similares(0,O, I, 1, M, V, L,), Pero KNN tuvo más dificultad con en estos caracteres, aun así experimentó un aumento en la precisión al 86.3% en cambio, en Tesseract hubo disminución en su Overall accuracy, esto debido a que tuvo problemas con los caracteres especiales (!,",, $, %, , /,(,),=,?, !); tal como ya se explicó este último

algoritmo es mayormente usado para manuscritos o textos en superficies homogéneas. Se describe mayores detalles en la tabla 9.

Modelo	Hiperpárametro	Acc	Observaciónes
SVM,	C=1, Gamma=1, Kernel=lineal, Type1=SVM_CSVC; degree =1y P = 0	95.5%	El algoritmo obtuvo un resultado adecuado con caracteres similares y confusos como son: 0, O, L, I,1, J entre otros. Obtenido un Overall accuracy de 95.5% y siendo el adecuando para usar en el modelo.
KNN,	K=4	86.3%	El algoritmo de KNN experimento dificultades a la hora de discernir caracteres similares como por ejemplo la 0 con la O obteniendo un accuracy de 86.3% Overall esto depende de la forma que se captura la imagen que ya que influye los ruidos de las imágenes.
Tesseract	psm=9	29.2%	Tesseract experimento muchos problemas con los ruidos en la imagen, esto porque cualquier mancha en la imagen es considerada como caracteres especiales como son ¨$.+, *, -¨ , entre otros por lo cual se obtuvo un Overall accuracy de 29.2%

Tabla 9 – comparación de algoritmos.

6. Discusiones

La motivación del trabajo fue la identificación las placas de los vehículos en ambientes controlados y para ambientes no controlados; en ambos casos se utilizaron funcionalidades de procesamiento de imágenes que pudieron mejorar el reconocimiento de los caracteres eliminando los ruidos.

Según los resultados mostrados se pudo obtener el mejor modelo para el reconocimiento de placas, este modelo fue con el algoritmo de SVM, con sus mejores hiperparámetros, siendo el Kernel lineal como el más adecuado, según (Tabrizi & Cavus, 2016)el Kernel lineal es uno de los más simples, pero, el más rápido al momento del mapeo de muestras; también menciona que el kernel lineal es utilizado cuando las muestras son linealmente separables siendo el caso en las muestras utilizadas.

Así mismo los resultados mostraron que el hiperparámetro C tuvo un valor de 1, el uso de este valor del hiperparámetro fue comprobado por (Chougule & Shah, 2019) obteniendo un acierto de 98% en la clasificación y detección de huellas dactilares; por otra parte uno de los factor que influyeron según (Salau et al., 2021)es la forma en la que las imágenes fueron capturadas, también (Khare et al., 2019)indica que otro factor es el clima al momento de capturar las imágenes, esto es validado por (Silva & Jung, 2020) donde su investigación fue realizada en climas variados y pudo obtener un porcentaje alto al momento de reconocer placas de vehículos. Para tener un mejor overall accuracy, se tuvo que realizar procesamiento de imágenes por ejemplo el autor (Agarwal et al., 2018) logra un 93% realizando procesamiento morfológico, del mismo modo (Farhat et al., 2018) que obtuvo un 95% y (Varma et al., 2020) tuvo un 98%.

7. Conclusiones

El desarrollo de modelos computacionales permite resolver uno de los problemas bandera de la sociedad actual que es la inseguridad pública, entonces nace la pregunta ¿es posible conseguir una buena precisión para que el modelo pueda detectar de forma óptima los dígitos de una matrícula de autoen condiciones controladas y no controladas? Para el problema planteado se determinaron una serie de elementos importantes para lograr un modelo de detección computacional con una métrica de Overall accuracy (OA) de excelente.

Primero se dividió el trabajo en tres fases (procesamiento de imágenes, localización de la placa de auto en toda la imagen, detección de los dígitos de la matrícula); fue importante ubicar el momento en que se aplicaron las distintas técnicas; esto lo ofreció el primer elemento metodológico; el segundo fueron las técnicas algorítmicas para procesar la imagen y estas fueron transformación morfológica, suavizado gaussiano y umbral gaussiano; consiguiendo una imagen visiblemente clara y con el menor ruido a pesar de las condiciones adversas climatológicas o distancias y ángulos diferentes.; el tercer elemento fue la aplicación del muestreo de las imágenes tomadas, a pesar que fueron 80 imágenes en la primera categoría de ambiente controlado y 100 imágenes en la categoría de ambiente no controlado; al aplicar el muestreo; la muestra llegó a tener 1000 combinaciones para KNN, 14400 para SVM y 1300 para Tesseract, haciendo las corridas con esta combinaciones de los hiperparámetros a cada imagen, con esto aumentó la posibilidad de tener un mejor OA y conseguir una detección excelente.

Se llegó a la conclusión que el mejor algoritmo fue el de SVM, cuyo objetivo principal es encontrar un hiperplano en el espacio n-dimensional que pueda clasificar los puntos de datos; logrando agrupar los puntos y determinando el dígito a ser predecido. La tabla 7 describe una tasa de reconocimiento con promedio de aproximadamente el 86.4% con imágenes tomadas de distintos ángulos y distancias, pero con las muestras con un ángulo recto y distancia similares hubo un aumento considerable obteniendo un Overall accuracy del 95.5 (tabla 9)

Referencias

Abedin, M. Z., Nath, A. C., Dhar, P., Deb, K., & Hossain, M. S. (2018). License plate recognition system based on contour properties and deep learning model. 5th IEEE Region 10 Humanitarian Technology Conference 2017, R10-HTC 2017, 2018-Janua, 590–593. https://doi.org/10.1109/R10-HTC.2017.8289029

Agarwal, P., Chopra, K., Kashif, M., & Kumari, V. (2018). Implementing ALPR for detection of traffic violations: A step towards sustainability. Procedia Computer Science, 132, 738–743. https://doi.org/10.1016/j.procs.2018.05.085

Arévalo, V., González, J., & Ambrosio, G. (2016). La librería de visión artificial OPENCV. 1–6. https://www.researchgate.net/publication/236668252_La_Libreri-a_de_Vision_Artificial_OpenCV_Aplicacion_a_la_Docencia_e_Investigacion_in_spanish

Chougule, A., & Shah, M. (2019). Local Binary Pattern with Hyperparameter Tuned Support Vector Machine for Fingerprint Classification. 2019 International Conference on Intelligent Computing and Control Systems (ICCS), Iciccs, 1084–1087. https://doi.org/10.1109/ICCS45141.2019.9065509

Divya, P., Varma, M., Ratna Mouli, U., Srinivas, Garima, Nikhil, & Vishistha. (2021). Web based optical character recognition application using flask and tesseract. Materials Today: Proceedings https://doi.org/10.1016/j.matpr.2020.10.850

Farhat, A., Hommos, O., Al-Zawqari, A., Al-Qahtani, A., Bensaali, F., Amira, A., & Zhai, X. (2018). Optical character recognition on heterogeneous SoC for HD automatic number plate recognition system. Eurasip Journal on Image and Video Processing, 2018(1). https://doi.org/10.1186/s13640-018-0298-2

INEI. (2021). PERU Instituto Nacional de Estadística e Informática INEI. https://www.inei.gob.pe/estadisticas/indice-tematico/vehicle-theft/

Khare, V., Shivakumara, P., Chan, C. S., Lu, T., Meng, L. K., Woon, H. H., & Blumenstein, M. (2019). A novel character segmentation-reconstruction approach for license plate recognition. Expert Systems with Applications, 131, 219–239. https://doi.org/10.1016/j.eswa.2019.04.030

Kumar Sahoo, A. (2020). Automatic recognition of Indian vehicles license plates using machine learning approaches. Materials Today: Proceedings. https://doi.org/10.1016/j.matpr.2020.09.046

Phangtriastu, M. R., Harefa, J., & Tanoto, D. F. (2017). Comparison between Neural Network and Support Vector Machine in Optical Character Recognition. Procedia Computer Science, 116, 351–357. https://doi.org/10.1016/j.procs.2017.10.061

Salau, A. O., Yesufu, T. K., & Ogundare, B. S. (2021). Vehicle plate number localization using a modified GrabCut algorithm. Journal of King Saud University - Computer and Information Sciences, 33(4), 399–407. https://doi.org/10.1016/j.jksuci.2019.01.011

SEMARNAT. (2006). Cuidado con los plaguicidas.

Silva, S. M., & Jung, C. R. (2020). Real-time license plate detection and recognition using deep convolutional neural networks. Journal of Visual Communication and Image Representation, 71. https://doi.org/10.1016/j.jvcir.2020.102773

Singh, K. R., Neethu, K. P., Madhurekaa, K., Harita, A., & Mohan, P. (2021). Parallel SVM model for Forest Fire Prediction. Soft Computing Letters, 100014. https://doi.org/10.1016/j.socl.2021.100014

SUNARP. (2020). Sunarp: número de autos que circulan en el país acumula una década de crecimiento continuo. https://www.sunarp.gob.pe/PRENSA/inicio/post/2020/01/08/sunarp-numero-de-autos-que-circulan-en-el-pais-acumula-una-decada-de-crecimiento-continuo

Tabrizi, S. S., & Cavus, N. (2016). A Hybrid KNN-SVM Model for Iranian License Plate Recognition. Procedia Computer Science, 102, 588–594. https://doi.org/10.1016/j.procs.2016.09.447

Varma, P. R. K., Ganta, S., Hari Krishna, B., & Svsrk, P. (2020). A Novel Method for Indian Vehicle Registration Number Plate Detection and Recognition using Image Processing Techniques. Procedia Computer Science, 167, 2623–2633. https://doi.org/10.1016/j.procs.2020.03.324