Comparativa de algoritmos de Inteligencia artificial para la detección de personas a través de una cámara instalada en el interior de un vehículo

RESUMEN: En este trabajo se muestra la comparativa realizada entre diferentes algoritmos de clasificación para la detección de personas usando una cámara al interior de un automóvil, para ello, la cámara elegida cuenta con 105° de visión, en donde el objeto (persona) debe de estar en un rango entre 1 y 4 metros de distancia del lente de la cámara para que sea detectado con éxito.

Los algoritmos de: árboles de decisión, Naive Bayes, KNN y SVM se aplicó el análisis de la textura aplicando GLCM con el fin de extraer las características. Otro algoritmo utilizado fue Redes Neuronales Convolucionales, en él, se fue variando las épocas durante el entrenamiento entre 50 y 250 hasta obtener la mejor precisión.

Así mismo se utilizó un Dataset de 931 imágenes bajo las condiciones antes mencionadas en los 5 algoritmos utilizados.

Hasta el momento, se tiene una precisión del 94% usando la métrica Recall con un nivel de confianza de 85%; donde Redes Neuronales Convolucionales obtuvieron el mejor resultado.

Seguiremos trabajando en mejorar el nivel de confianza y así poder utilizar este algoritmo en un prototipo que permita alertar al propietario cada vez que se detecte una persona merodeando la unidad motriz.

PALABRAS CLAVE: Inteligencia Artificial, extracción de características, CNN, KNN, Árboles de Decisión, Naive Bayes, SVM, GLCM.

Colaboración
Christian Ivan Onofre Ramos; José Antonio Montero Valverde; Miriam Martínez Arroyo, Instituto Nacional de México / campus Acapulco

Fecha de recepción: 25 de agosto del 2022
Fecha de aceptación: 18 de abril del 2023

ABSTRACT: This work shows the comparison between different classification algorithms for the detection of people using a camera inside a car, for this, the camera chosen has 105° of vision, where the object (person) must be in a range between 1 and 4 meters away from the camera lens to be detected successfully.

The algorithms of: decision trees, Naive Bayes, KNN and SVM were used for texture analysis applying GLCM in order to extract the features. Another algorithm used was Convolutional Neural Networks, in which the epochs were varied during training between 50 and 250 until the best accuracy was obtained.

Likewise, a Dataset of 931 images was used under the aforementioned conditions in the 5 algorithms used.

So far, we have an accuracy of 94% using the metric Recall with a confidence level of 85%; where Convolutional Neural Networks obtained the best result.

We will continue working on improving the confidence level and thus be able to use this algorithm in a prototype that will alert the owner every time a person is detected prowling around the motor unit.

KEYWORDS: Deep Learning, Feature extraction, CNN, KNN, Decision tree, SVM, Naive Bayes, GLCM.

INTRODUCCIÓN

La delincuencia organizada a nivel mundial va íntimamente ligada a delitos relacionados con autos y es que la presencia de

autos en el mundo es tan importante a tal grado que en ocasiones son vinculados con el terrorismo según datos de la Interpol (Policía Internacional) [1]. En ese sentido, el aumento de robo de autos, robo de partes, al igual que objetos al interior del mismo, es una actividad que afecta en primer plano a los propietarios y termina por mermar una sociedad, la cual se siente insegura incluso dentro de su propio automóvil de acuerdo a la Encuesta Nacional de Victimización y Percepción sobre seguridad pública en México (ENVIPE) [2]. En concordancia con las cifras que muestra el Secretariado Ejecutivo del Sistema Nacional de Seguridad Pública (SESNSP), tan sólo en el primer bimestre del presente año fueron hurtadas alrededor de 8171 unidades en todo el país, esto sin uso de violencia, es decir, autos que se encontraban estacionados en lugares como una calle o espacios públicos [3].

Por tal motivo y en apoyo a los usuarios que a diario hacen uso de sus coches, lo que se propone es el uso de un algoritmo que permita la detección de personas que merodean un auto por medio de visión computacional a través de una cámara instalada al interior del mismo, así, posteriormente en otra etapa del proyecto, dicho algoritmo se pueda integrar a un sistema más completo y hacer que el prototipo envíe una alerta al propietario en base a la detección de personas.

MATERIAL Y MÉTODOS

La metodología aplicada referente a los algoritmos utilizados en el presente trabajo se muestra a continuación en la Figura 1, la cual, consta de 5 fases.

Figura 1. Metodología.
Fuente: Elaboración propia.

Adquisición de la imagen

En esta primera etapa, se capturaron un total de 931 imágenes subdivididas en 3 secciones, las cuales son 634 para entrenamiento, 286 para validación, y 10 más para pruebas; las imágenes fueron capturadas desde el interior de un vehículo con una cámara USB de 6.0 Mpx, el sensor cuenta con una resolución de 3264px por 1836px; en ella, se tomaron en cuenta las siguientes condiciones para la construcción del set de imágenes (Dataset) en la Figura 2 se observa la cámara utilizada.

Figura 2. Cámara web utilizada.
Fuente: Elaboración propia.

- La distancia del objeto (persona) con respecto al lente de la cámara debe de ser entre 1 y 4 metros, esto, con el fin de apreciar lo mejor posible el dorso y rostro de la persona.

- La posición de la cámara es fija ubicada del lado del copiloto bajo la esquina inferior del parabrisas, tal como se aprecia en la Figura 3.

- El ángulo de visión de la cámara es de 105°, nuevamente se puede observar en la Figura 3.

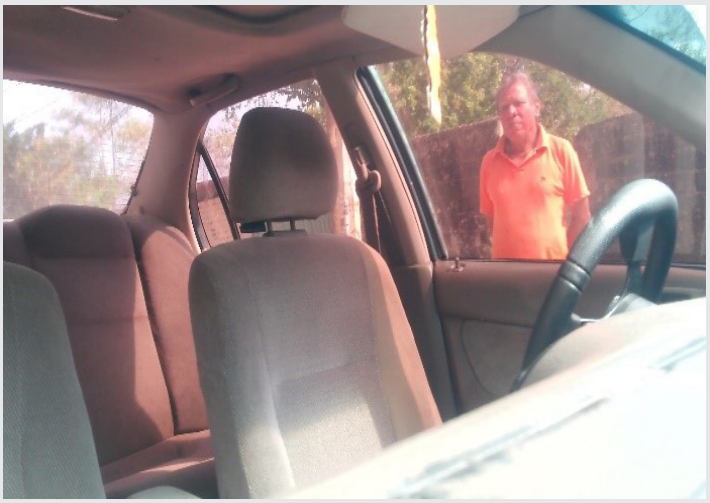

Figura 3. Imagen Original.
Fuente: Elaboración propia.

- Las imágenes capturadas se realizaron entre las 10:00 a.m. y 4:30 de la tarde, procurando obtener escenarios bajo ese lapso de tiempo.

- Las personas que aparecen en las imágenes están de pie y en forma vertical lo más cercano a 90° procurando una postura natural.

- Las personas mantienen diferentes posiciones (perfil, espalda y frente), por ello, sólo se está considerando imágenes donde aparece como mínimo el 80% entre el dorso y el rostro, así los objetos fuera de las anteriores condiciones no garantizan una detección efectiva.

Preprocesamiento

En esta segunda etapa cada una de las 920 imágenes destinadas para entrenamiento y validación se les aplicó las siguientes transformaciones que se describen a continuación, cuyo objetivo es resaltar y evitar la pérdida de información.

1.- En un principio se probó con una dimensión de 180px por 200 px, pero cuando se llegó a la etapa de clasificación los resultados no fueron tan favorables en cuanto a precisión, por ello, nos dimos a la tarea de retornar a la presente etapa y probar con diferentes redimensiones para los algoritmos de (Naive Bayes, SVM, Árboles de Decisión y KNN) esto se realizó con el objetivo de mantener la mayor cantidad de información en las imágenes sin perder la geometría de las mismas y por ende un entrenamiento más rápido.

Las redimensiones probadas fueron las siguientes: (200 por 200 px, 180 por 200 px, 300 por 300 px y 256 por 256 px).

Esto, nos llevó a concluir que de todas las combinaciones probadas, la que mejor resultados nos arrojó al realizar el entrenamiento y clasificación fue la redimensión de 256 por 256 px, es decir; se pudo observar que al final de cada clasificación el tiempo de entrenamiento mejoraba por mucho al igual que el nivel de precisión era mejor; cabe mencionar que en puntos posteriores mostraremos más a detalle todo lo relacionado a la clasificación.

2.- La redimensión no fue la única, también, se aplicó una transformación a escala de grises con el fin de trabajar en un solo canal y dejar de lado el modelo de color RGB (Red, Green y Blue) lo realizado fue por dos razones, la primera es que la técnica que se utilizó para la extracción de las características basadas en la textura (GLCM), obtiene los valores conforme a los niveles de gris de cada imagen, los cuales, estarán oscilando entre 0 y 255, de ahí el principal motivo de aplicar esta transformación, la segunda es que computacionalmente hablando es más sencillo trabajar en niveles de gris en comparación de usar los 3 canales RGB, de esta manera podemos observar algunas variaciones en gris a modo de ejemplo en la siguiente Figura 4 [4], así, cada pixel de las 920 imágenes de entrenamiento y validación son transformados entre los valores 0 y 255.

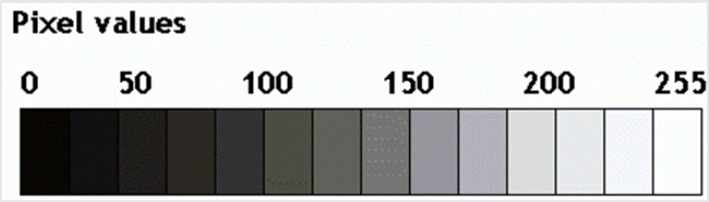

Figura 4. Muestra de variación de pixeles en escala de grises que van de 0 a 255.
Fuente: Elaboración propia.

3.- Por último se aplicó una ecualización al histograma de las 920 imágenes con el fin de tener una distribución uniforme de los pixeles anteriormente transformados, dicha variación ya mencionamos que va de 0 hasta 255 en escala de grises, por esta razón y dicho en otras palabras; al aplicar la ecualización se trata de mantener de manera uniforme la distribución de niveles de gris, dejando de lado una imagen con niveles de gris demasiado oscura o demasiado clara, así el resultado es una imagen con tonos equilibrados en toda su estructura.

Extracción de características

En esta tercera etapa se utilizaron 11 características basadas en la textura de las 14 propuestas por Haralick en 1973, utilizando la técnica de Matriz de co-ocurrencia de niveles de gris (GLCM), con esta técnica, se obtiene dicha matriz de ocurrencia de niveles de gris y con ella las siguientes características que se listan en la Figura 5 [5].

Figura 5. Características a utilizar usando GLCM.
Fuente: Elaboración propia.

Estas características se usaron con las técnicas de KNN (Vecinos Cercanos), Naive Bayes, Árboles de Decisión y SVM (Máquina de Soporte de Vectores), cabe mencionar que más adelante hablaremos de otra técnica más que lleva por nombre Redes Neuronales Convolucionales, la cual; no necesariamente trabaja con estas características.

De esta manera, lo realizado en esta etapa es precisamente la extracción de las características de cada imagen y para ello, fue necesario no solo extraer las características sino ir variando la separación entre pixeles en un rango de (1, 2, 3, 4, 5) px. Así mismo, se consideró usar los ángulos 0°, 45°, 90° y 135° tal como lo propuso Haralick en su trabajo de análisis de textura para la clasificación de imágenes, así, al igual que Haralick, no se consideraron los otros 4 ángulos opuestos, ya que estos tienen una relación invariante [5].

Al final de esta etapa se utilizó la métrica de Recall para determinar la mejor de las combinaciones probadas y con ellos se concluyó que el mejor resultado fue usando 1 pixel de separación con 0° de inclinación, dicho de otra manera, la mejor relación espacial fue de (1, 0) y esta, fue la utilizada en esta etapa de extracción de características; Es de importancia mencionar que obtuvimos 16 Dataset diferentes con las combinaciones de separación y ángulos antes mencionadas y (1, 0) fue la mejor relación espacial.

Para más detalle y a modo de ejemplo, en la Figura 6, vemos un Dataset (Conjunto de Características) el cuál es 1 de los 16 obtenidos en las pruebas de extracción, en dicha Figura, sólo se aprecia una muestra de 8 características de 11 totales que obtuvimos en cada Dataset, así mismo, es posible observar las primeras 10 imágenes listadas por fila recordando que son un total de 920 filas para cada Dataset, así mismo los datos mostrados en dicha Figura están sin normalizar puesto que en las pruebas realizadas se obtuvo una mejor precisión usando los datos sin aplicar la normalización.

Energia	Corr	Diss_sim	Homogen	Contraste	Asimilitud	Varianza	Media
0.04238511	0.98093098	4.26304134	0.41531696	97.5127953	0.0017965	1035921.8	8.486816
0.03311273	0.97726231	5.18128691	0.35539415	124.171444	0.00109645	1182064	11.37085
0.17493874	0.98356895	5.76125738	0.42193089	193.917876	0.03060356	519824.78	8.470215
0.03394131	0.95367901	6.73492864	0.24702564	136.926858	0.00115201	3060042.5	54.43701
0.05867145	0.96118226	9.68122539	0.29021574	389.304749	0.00344234	1148603.6	15.820801
0.02056522	0.9428114	8.92550443	0.17490928	196.286848	0.00042293	7066253	71.91821
0.02751412	0.95616297	6.58409203	0.23012115	126.945928	0.00075703	37921036	100.25537
0.02721931	0.97786484	5.10900591	0.25239915	71.5129183	0.00074089	13576736	80.39404
0.03688873	0.95406807	4.55647146	0.29022492	60.8793061	0.00136078	31711022	93.08081
0.01950311	0.96386042	8.76371801	0.18010125	220.652375	0.00038037	1954559.5	56.39966

Figura 6. Muestra del Dataset con 8 características de 11 usando GLCM con una relación espacial (1,0).
Fuente: Elaboración propia.

Entrenamiento

Durante esta etapa de entrenamiento como es de suponer se entrenaron los algoritmos (Naive Bayes, SVM, KNN, Arboles de Decisión) usando un Dataset como el mostrado en la Figura anterior, la número 6 con las 11 características de GLCM bajo una relación espacial de (1, 0), cabe mencionar que para esta etapa se utilizaron las 634 imágenes destinadas para el entrenamiento tal como se describió en la etapa de adquisición de la imagen.

De esta forma es como pusimos a prueba cada uno de los algoritmos aprovechando sus ventajas, a continuación, mostraremos las configuraciónes y pruebas realizadas con cada uno.

Algoritmo KNN

Este algoritmo es uno de los más utilizados en los últimos años en lo que se refiere a aprendizaje automático, ya que aparte de ser un algoritmo simple y de fácil comprensión, es muy preciso al trabajar con nuevos datos.

Básicamente este algoritmo consiste en seleccionar el número de K vecinos y obtener la distancia entre pixeles a la que se encuentran esos vecinos del nuevo dato basados en la distancia euclidiana [6].

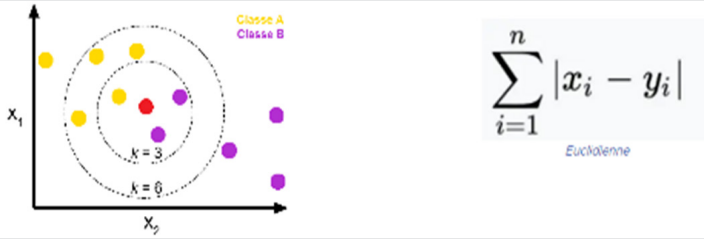

Figura 7. Distancia Euclidiana.
Fuente: Elaboración propia.

Las pruebas realizadas con este algoritmo fue variar el número de vecino entre los valores de (3, 4, 5, 10 y 25), de esta manera, considerando una distancia de K=5 obtuvimos una métrica en Recall con buenos resultados, ya que a medida que asignábamos un valor de k mayor, el algoritmo tendía a ser más complejo por lo que se consideró usar un valor no tan alto para K asignando el valor de 5.

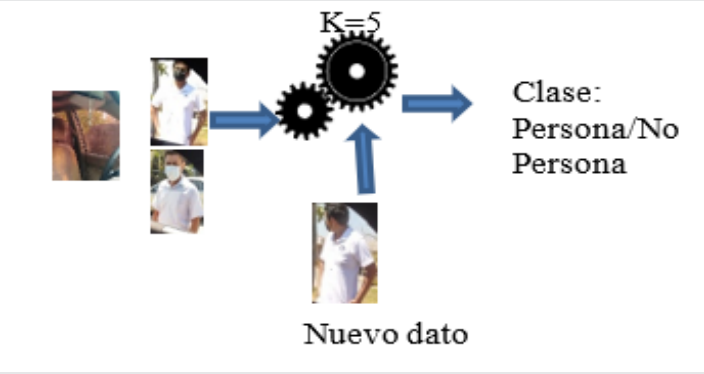

Figura 8. Algoritmo KNN.
Fuente: Elaboración propia.

Algoritmo Naive Bayes

Este algoritmo es de tipo probabilístico, por lo que a pesar de que algunos lo consideran un tanto simple,

este, trabaja de manera efectiva cuando se ingresan en el dato en tiempo real; una de las ventajas que lo hace destacar de varios algoritmos es que en ocasiones no necesita muchos datos de entrada para arrojar resultados, sin embargo, lo cierto es que entre más datos se le presenten, es mayor la probabilidad de obtener mejores resultados.

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

Figura 9. Fórmula probabilística de Naive Bayes [13].
Fuente: Elaboración propia.

La fórmula anterior es la utilizada por el algoritmo, en la cual, podemos decir que h es la clase y D los datos, de esta manera se trata de determinar la probabilidad de una clase dado un conjunto de dato, para llegar a ello, se multiplica la probabilidad de dicha clase por la probabilidad de los datos dada la clase, todo esto entre la probabilidad de los datos siguiendo el procedimiento de Bayes [7].

Para utilizar este algoritmo, existen 3 tipos de distribución diferentes, cada una tiene sus propias ventajas, por tal motivo en este trabajo se utilizó la implementación Gausianna de Naive Bayes, con el objetivo de que los datos siguieran una distribución del mismo tipo, es decir, (Gausianna) dicho en otras palabras, una distribución normal; las otras dos distribuciones son Bernoulli la cual es para datos boleanos y Multinomio se usa con datos de distribución de tipo nominal, por lo que sus aplicaciones de esta última son para fines de clasificación de textos; de esta manera es como se configuraron los parámetros que se utilizaron durante la etapa de entrenamiento para este algoritmo.

Algoritmo de Arboles de Decisión

Los árboles de decisión son algoritmos de tipo estadísticos para la construcción de modelos de análisis predictivo, por lo que es utilizado para la clasificación o regresión, este, se utiliza para obtener determinado resultado en base a la incertidumbre que se tiene ante un problema dado, una de las ventajas que se tiene al aplicar dicho algoritmo, es la simplicidad y comprensión de los datos, esto, gracias a la visualización del árbol que genera el algoritmo, ya que como es de esperar, su estructura es muy similar a los árboles tal como podemos observar en la Figura 10.

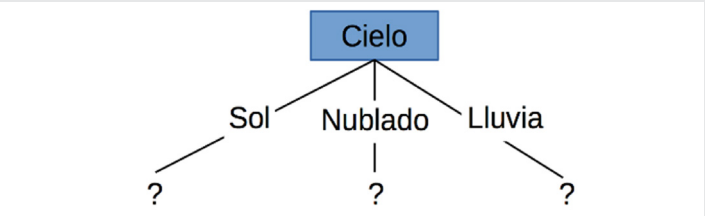

Figura 10. Árbol de decisión.
Fuente: Elaboración propia.

Así, los nodos internos representan las características o propiedades que se tienen que considerar para poder tomar cada una de las decisiones, las ramas representan la decisión que se toma en base a cada condición (probabilidad de que ocurra esa condición), los nodos finales representan el resultado de la decisión tomada conforme a cada una de las condiciones o características.

Existen diferentes escalas que es posible aplicar a los árboles de decisión las cuales son: (Tasa de error de clasificación, Gini index, entropía cruzada y chi-square) de las alternativas descritas, las más óptimas para la clasificación son Gini index y la entropía cruzada, esta última, es la responsable de cuantificar el desorden de un sistema, así, dada una clase pura, su entropía será 0, caso contrario en donde la frecuencia de una clase es la misma, la entropía es igual a 1 [8].

Dicho lo anterior es la razón principal por lo que al aplicar esta técnica de árboles de decisión se consideró aplicar la entropía cruzada y con ello se obtuvieron los mejores resultados de clasificación con este algoritmo.

SVM

Las máquinas de soporte de vectores (SVM) son utilizadas para la resolución de problemas de clasificación y regresión, estas son consideradas como una especie de caja negra en donde la información entra al algoritmo y este devuelve una predicción basada en la información de entrada. Un dato curioso es que, en un inicio cuando apareció este algoritmo sólo atacaba problemas de clasificación lineal, pero hoy en día con las funciones kernel no lineales es posible utilizar SVM para resolver un mayor rango de problemas.

En el presente se consideró el utilizar un Kernel de función de Base Radial (RBF) con el fin de aumentar las dimensiones, este, sólo es uno de muchos existentes, dicho kernel considera la distancia euclidiana entre los dos vectores de características, por lo que es uno de los más poderosos cuando se tienen datos no lineales, es decir, el valor de Gamma y la penalización C, las configuraciones que mejor resultado nos arrojó fue establecer el valor de Gama de forma automática, esto quiere decir que el algoritmo tiene la finalidad de adaptarse a la estructura de los datos con el fin de lograr la clasificación.

De esta forma es como actualmente es posible utilizar las Máquinas de Soporte de Vectores en problemas de clasificación con la ayuda de los Kernel.

Redes Neuronales Convolucionales

Una red neuronal convolucional es una especie de red que se asemeja a las neuronas cerebrales, de ahí su nombre Redes Neuronales.

Para poder trabajar con este algoritmo, previamente es necesario etiquetar todas las imágenes que se van a utilizar en las etapas de entrenamiento y validación re-

cordando que para este proyecto se cuenta con 920 imágenes, así cada imagen puede tener más de una instancia y cada una de ellas debe de ser etiquetada bajo el formato de YOLO, para este proyecto se utilizó el software labelImg para lograr el etiquetado de clases en donde a modo de ejemplo se observa la Figura 11.

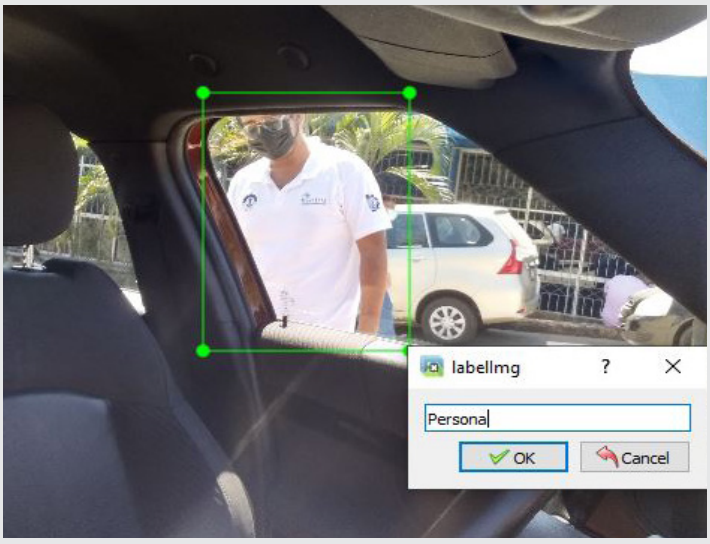

Figura 11. Etiquetado de clases usando LabelImg.
Fuente: Elaboración propia.

Una vez realizado el etiquetado, este algoritmo trabaja en 3 etapas las cuales describiremos a continuación:

1.- En la primera etapa el mismo algoritmo se encarga de redimensionar las imágenes de entrada a 640 px por 480 px así la búsqueda de características será más rápida.

2.- Dividir una imagen de entrada en secciones de igual tamaño cuadriculada, así un Kernel recorre toda la imagen en donde cada cuadro de dicha cuadrícula es responsable de intentar detectar la o las clases que fueron etiquetadas previamente, es aquí donde se realizan todas las operaciones convolucionales con el fin de obtener un vector de características.

3.- Se utiliza el vector de características obtenido con el fin de reducir cada uno de los nodos e ir agrupando estas características en clases.

Así cuando una imagen nueva entra a la red, esta recorrerá neurona a neurona la red y en base al vector de características, esta se irá acercando más en cada una de las capas ocultas hasta llegar a clase a la que pertenece la imagen, así una red neuronal es capaz de reconocer más de una clase.

4.- En la última etapa llamada Capa de salida o de clasificación, se determinan la o las clases asignando un porcentaje de predicción a cada una de ellas, en la siguiente Figura 12, se muestra un ejemplo de la arquitectura de una red neuronal convolucional.

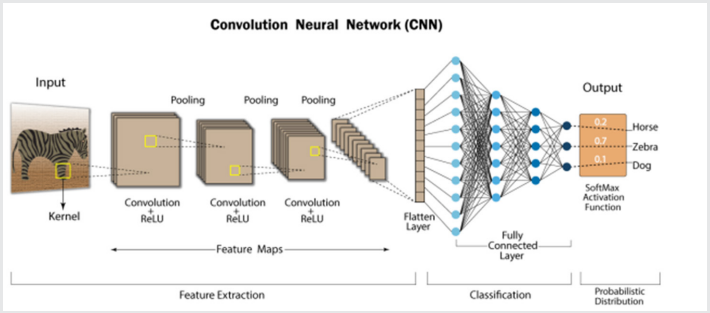

Figura 12. Estructura de una Red Neuronal Convolucional [9].
Fuente: Elaboración propia.

CNN YOLO versión 5

Para este trabajo se utilizó la red neuronal de YOLO, específicamente YOLOv5s (you only look once) [10] el cual, es uno de los algoritmos más utilizados en la actualidad en la detección de objetos, principalmente por su velocidad y precisión al ser de sus principales características; cabe mencionar que existe ya la versión 6 y 7 en modo Beta, pero no es utilizada en este trabajo ya que aún sigue en fase de pruebas, por lo que se consideró trabajar con una versión estable; así mismo existen diferentes sub versiones de YOLOv5 tal como lo vemos en la Figura 13, las cuales van desde una versión ligera hasta llegar a una compleja, la cual, cada una requiere distintos grados de fuerza computacional distinta, la versión utilizada en este trabajo es la versión Small esto de acuerdo a la página oficial es recomendable para dispositivos móviles entre ellos los Raspberry, este último es el principal candidato para llevar a producción el algoritmo que se presenta en esta trabajo para futuras investigaciones.

Por otro lado, una de las ventajas de este tipo de algoritmos es que al utilizar pesos Pre-entrenados se aplica lo que se conoce como transferencia de aprendizaje, logrando que la red neuronal identifique líneas y formas con una estructura similar a lo que pretendemos que aprenda usando nuevos datos, así, con el uso de estos pesos Pre-entrenados el algoritmo no partirá de cero en su aprendizaje sino más bien de lo que ya conoce, con el fin de hacer la transferencia del mismo logrando que aprenda a reconocer patrones de forma rápida.

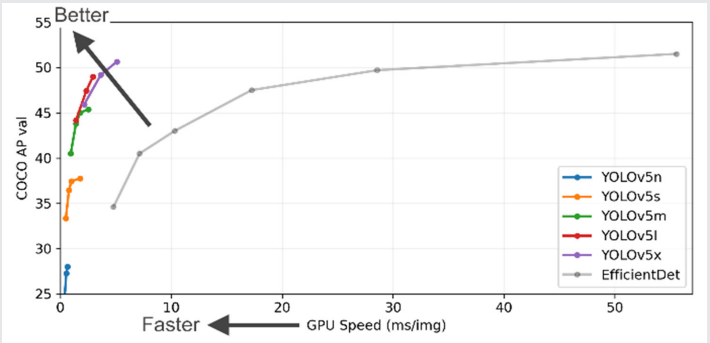

Figura 13. Versiones YOLO [10].
Fuente: Elaboración propia.

Al ser YOLO una red neuronal convolucional, su estructura está apegada a dicha ideología tal como lo explicamos anteriormente, Así lo vemos en la Figura 13.

1.- En el BackBone: (Columna vertebral) es la encargada de preparar e integrar las características de la imagen de entrada agregándolas a un vector con el fin de utilizarlas en etapas posteriores.

El kernel va recorriendo toda la imagen para construir el Mapa o Vector de Características, y obtenerlo es el principal objetivo de esta etapa.

2.- PANet: en esta etapa, se utiliza una serie de capas por lo que es la encargada de mezclar y combinar las características que ya se definieron en el BackBone, agrupándolas en clases, también se aplica la función de activación mejor conocida como ReLU (Unidad Lineal Rectificada) la cual: sirve para aligerar la carga de la red y eliminar los valores no esperados, es decir, valores por debajo de 0 se convierten en 0. Otra de las funciones que son muy importantes en esta etapa es la de Non-Max Supression (Supresión del no máximo) el cual se encarga de agrupar los cuadros delimitadores duplicados de cada una de las clases, es decir, conforme se va haciendo la convolución a la par usando el vector de características es posible que se genere más de un cuadro delimitador para un mismo objeto, por lo que en más de una ocasión se tendrá un solo objeto con más de un cuadro así Non-Max se encarga de suprimir los cuadros repetidos para un mismo objeto dejando la información de las clases lo más limpia posible sin cuadros repetidos.

3.- Salida: en la última etapa del algoritmo, se asigna el porcentaje de confianza de cada clase, es decir, la red neuronal determina que tan segura ésta de que los objetos que logró localizar fueron conforme a lo aprendido, por tanto, esta red va asignando un porcentaje de 0 a 100 en la capa de salida.

Todo lo antes mencionado se puede observar en la Figura 14.

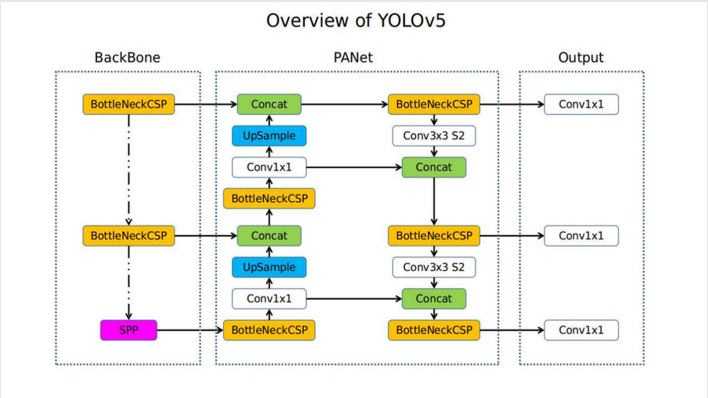

Figura 14. Estructura de YOLOv5 [10].
Fuente: Elaboración propia.

A modo de resumen, para este algoritmo
1.- Se utilizaron las mismas 920 imágenes destinadas para entrenamiento y validación, previamente fueron etiquetados los objetos (Persona).

2.- Se utilizó los pesos Pre-entrenados (YOLOv5s) de la página oficial con el fin de agilizar la transferencia de aprendizaje.

Después de varias pruebas se trabajó con 50, 100, 150 y 250 épocas con el fin de obtener la efectividad del algoritmo, así, se determinó una duración de 150 épocas logrando con estas la mejor efectividad, esto nos arrojó como resultado un modelo entrenado con extensión .pt este archivo contiene toda la lógica necesaria para realizar predicciones; de esta manera la red neuronal será capaz de reconocer en una imagen la clase persona usando el archivo .pt.

Validación
En esta etapa se utilizaron 286 imágenes diferentes a las utilizadas durante el entrenamiento, esto, con el fin de que el algoritmo fuera validando lo aprendido; es tan esencial el entrenamiento con la validación puesto que es aquí donde realmente se aprecia si todo lo realizado anteriormente fue lo mejor para obtener una buena precisión.

En la Figura 15 se muestra una prueba de validación realizada con YOLOv5s una vez entrenado el algoritmo.

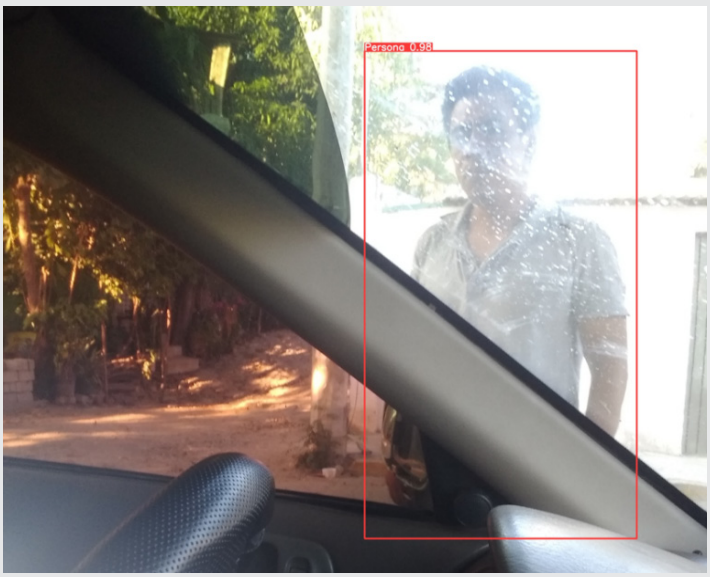

Figura 15. Etapa de validación YOLOV5s.
Fuente: Elaboración propia.

RESULTADOS

A continuación, se muestra la siguiente tabla comparativa con los resultados obtenidos en cada uno de los clasificadores usando la métrica Recall.

De esta manera es posible determinar la efectividad de los mismos reconociendo las ventajas y desventajas

que nos ofrecen con el objetivo de determinar el que mejor nos servirá para el presente trabajo.

Comparativa de clasificadores

Tabla 1. Comparativa Métrica Recall: KNN, Bayes, Arbol de Decisión, SVM, CNN.

Espaciado (Pixeles, grados)	KNN	NAIVE BAYES	ARBOL DE DECISIÓN	SVM	CNN YOLO 94%
1 ,0°	92.90%	91.61%	93.19%	93.25%	50 Épocas 87%
1,45°	90.32%	88.38%	93.12%	67.39%	
1,90°	92.90%	91.61%	93.77%	67.39%	
1,135°	92.25%	90.96%	92.83%	68.58%	
2,0°	89.03%	87.09%	91.61%	67.39%	
2,45°	93.54%	88.38%	92.90%	67.39%	100 Épocas 88.5%
2,90°	90.96%	91.61%	92.83%	67.39%	
2,135°	90.32%	90.96%	91.83%	68.58%	
3,0°	90.96%	83.87%	90.96%	67.39%	
3,45°	92.25%	84.51%	90.32%	67.39%	
3,90°	89.67%	88.38%	93.77%	67.39%	150 Épocas 94%
3,135°	91.61%	89.03%	92.54%	67.39%	
4,0°	92.25%	83.22%	90.32%	67.39%	
4,45°	90.96%	82.58%	92.50%	67.39%	
4,90°	90.96%	88.38%	93.19%	67.39%	
4,135°	91.61%	85.80%	93.50%	67.39%	250 Épocas 85%
5,0°	92.25%	80.64%	90.96%	67.39%	
5,45°	90.32%	78.70%	92.58%	67.39%	
5,90°	93.19%	87.09%	93.04%	67.39%	
5,135°	93.19%	81.29%	93.41%	67.39	

Recordemos que se usaron 920 imágenes donde los clasificadores de Naive Bayes, Arboles de Decisión, KNN y SVM se aplicó con 5 tipos de espaciado de pixeles y 4 grados de inclinación diferentes, por lo que se puede apreciar en la Tabla 1.

También el algoritmo de redes neuronales se sometió a pruebas en las cuales la que mejor resultados nos arrojo fue usando 150 épocas mostradas en la Figura 16.

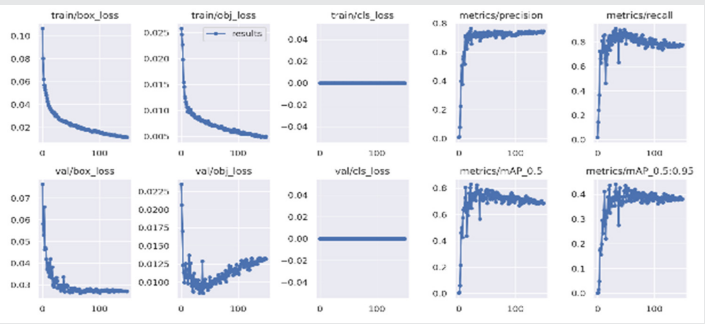

Figura 16. Resultados usando CNN (YOLOv5s) a 150 Épocas con 920 Imágenes redimensionadas a 640px por 480px.
Fuente: Elaboración propia.

CONCLUSIONES

Dados los resultados mostrados en las pruebas anteriores al aplicar distintos clasificadores y con ello logrando aprovechar al máximo las ventajas que cada uno nos ofrece, pudimos llegar a la conclusión de que es factible utilizar el algoritmo de YOLOv5 basado en Redes Neuronales Convolucionales para la detección de personas usando una cámara al interior de un vehículo tal como lo podemos ver en la Figura 17.

Figura 17. Resultados.
Fuente: Elaboración propia.

Si bien es cierto la métrica de Recall nos arroja un 94%, eso no significa que siempre dará buenos resultados, puesto que aun el nivel de confianza del algoritmo se puede mejorar puesto que aún se encuentra alrededor del 82% hasta el momento, por este hecho, seguiremos trabajando en la mejora de dicha métrica, con el fin de obtener un nivel de confianza y predicción mayor, y con ello, poder migrar el uso del algoritmo implementándolo en un Raspberry o dispositivo móvil en investigaciones futuras.

BIBLIOGRAFÍA

[1] Interpol, «Interpol.int,» [En línea]. Available: https://www.interpol.int/es/Delitos/Delincuencia-relacionada-con-los-vehículos. [Último acceso: 01 07 2022].

[2] INEGI, «Encuesta Nacional de Victimización y percepción sobre seguridad pública 2021,» INEGI, Ciudad de México, 2021.

[3] S. E. d. S. N. d. s. Pública, «Incidencia delictiva del fuero común,» Ciudad de México, 2022.

[4] ESA, «Pixel Values,» [En línea]. Available: https://www.esa.int/images/greylevel.gif. [Último acceso: 24 08 2022].

[5] K. S. y. D. Robert M. Haralick, «Análisis de textura para la clasificación de imagenes,» IEEE, Kansas City, 1973.

[6] L. Salcedo, «pythondiario,» 31 01 2018. [En línea]. Available: https://pythondiario.com/2018/01/intro-

ducción-al-machine-learning-9-k.html. [Último acceso: 24 08 2022].

[7] C. M. Luque, «Clasificadores bayesianos El algoritmo Naive Bayes,» p. 3, 2003.

[8] J. A. Rodrigo, «www.cienciadedatos.net,» 01 10 2020. [En línea]. Available: https://www.cienciadedatos.net/documentos/py07_árboles_decisión_python.html. [Último acceso: 15 07 2022].

[9] Platon, «https://es.platodata.ai/,» [En línea]. Available: https://zephyrnet.com/es/fundamentos-de-cnn-en-aprendizaje-profundo/. [Último acceso: 25 08 2022].

[10] G. Jocher, «YOLOv5,» 07 01 2020. [En línea]. Available: https://docs.ultralytics.com/. [Último acceso: 14 06 2022].

[11] AMIS, «Reporte anual Asociación Mexicana de Instiruciones de Seguros,» Centro estadístico de sector asegurador, Ciudad de México, Tlacopac, Delegación Alvaro Obregón, 2022.

[12] concyteq, «http://www.concyteq.edu.mx/,» 01 06 2017. [En línea]. Available: http://www.concyteq.edu.mx/concyteq/uploads/convocatoria/2017-11-10351.pdf. [Último acceso: 07 07 2022].

[13] aprendeia, «aprendeia,» [En línea]. Available: https://aprendeia.com/naive-bayes-teoria-machine-learning/. [Último acceso: 24 08 2022].

[14] V. Escotto, «Espejos, faros y otras de las autopartes mas robadas en México,» businessinsider, 2021.

ANEXO

CARACTERÍSTICAS GLCM	
1.- Varianza: determina que tan dispersos están los datos con respecto a la media	$VAR = \sum_{i,j=0}^{N-1} P_{i,j}\,(i-\mu)^2$
2.- Entropía: determina la cantidad de ruido o desorden de la información que se tiene por lo que a mayor entropía mayor es la complejidad para el algoritmo.	$\sum_{i,j=0}^{N-1} P_{i,j}\left(-\ln P_{i,j}\right)$
3.- Autocorrelación: nos permite analizar la relación espacio tiempo entre los elementos cercanos y los que están más alejados.	$I = \frac{n}{s}\frac{\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} z_i z_j}{\sum_{i=1}^{n} z_i^2}$
4.- Correlación: muestra la dependencia lineal entre un valor no correlacionado 0 y un valor perfectamente relacionado 1.	$\sum_{i,j}\frac{(i-\mu i)(j-\mu j)p(i,j)}{\sigma_i \sigma_j}$
5.- Contraste: Se encarga de medir las variaciones locales conforme a los niveles de gris de la matriz de co-ocurrencia.	$\sum_{i,j}\lvert i-j\rvert^2\, p(i,j)$
6.- Energía: Provee la suma de cuadrados de los elementos en GLCM también se le conoce como segundo momento angular o uniformidad.	$\sqrt{\sum_{i,j} p(i,j)^2}$
7.- Homogeneidad: Mide que tan cercanos están los elementos conforme a la diagonal GLCM	$\sum_{i,j}\frac{p(i,j)}{1+\lvert i-j\rvert}$
8.- Desviación estándar: Determina la cantidad de variación o dispersión de los datos con respecto a la media	$standard\ deviation = \sqrt{\frac{1}{N_p}\sum_{i=1}^{N_p}(\mathbf{X}(i)-\bar{X})^2}$
9.-ASM o uniformidad: Es la probabilidad en la que aparecen los niveles de gris para una distancia y dirección.	$\sum_{i,j=0}^{N-1} P_{i,j}^2$
10.- Disimilaridad: Determina la falta de relación entre pixeles	$dissimilarity = \sum_{i=1}^{N_g}\sum_{j=1}^{N_g}\lvert i-j\rvert p(i,j)$
11.- Media: Determina la intensidad promedio de los niveles de gris.	$mean = \frac{1}{N_p}\sum_{i=1}^{N_p}\mathbf{X}(i)$