ESCUELA DE INGENIERÍA DE FUENLABRADA

GRADO EN INGENIERÍA EN SISTEMAS AUDIOVISUALES Y MULTIMEDIA

TRABAJO FIN DE GRADO

Conteo automatizado mediante visión artificial con herramientas de deep learning para el cálculo de la medida IMD.

Autor: Roberto Ramos Castro

Tutor: Jorge Calero Sanz

Curso académico 2022/2023

Agradecimientos

Agradezco el tiempo empleado a los profesores de la carrera, su dedicación y compromiso para darnos una educación de calidad. Su pasión por compartir los conocimientos y su continuo apoyo son fundamentales para mi crecimiento personal y académico.

También me gustaría agradecer a mi familia por su constante lucha en ayudarme a lo largo de este camino. Su amor, su paciencia y su manera de facilitarme la vida ha sido mi fuente de motivación durante los momentos más difíciles.

Por supuesto, a mi aliento dentro de las aulas, mis compañeros de universidad que me han apoyado durante estos años. Juntos compartimos risas, desafíos y logros, forjando lazos que perdurarán durante este período de aprendizaje.

Este proyecto de final de carrera no hubiera sido posible sin la cooperación y el esfuerzo conjunto de todos. Estoy verdaderamente agradecido por haber tenido la ocasión de aprender y crecer junto a todos ellos. Gracias por ser parte de este viaje, por su continuo apoyo y por creer en mí. Su influencia ha tenido un profundo efecto en mi vida y siempre les estaré agradecido.

Resumen

El parámetro *Intensidad Media Diaria* mide el número de vehículos por día que circulan por un tramo de carretera específico. Es importante a la hora de gestionar y controlar infraestructuras de transporte de una manera inteligente, pudiendo así mejorar la seguridad en la carretera.

El foco del estudio está en la automatización del conteo de los vehículos para conocer y controlar el impacto acústico de una carretera en edificios colindantes ya sean para ocio, viviendas, trabajo, etc. Para conseguir esta automatización a la hora de realizar el conteo, se hace uso de la inteligencia artificial y, particularmente, de modelos de *deep learning* para facilitar el cálculo de un parámetro perteneciente al ámbito de la acústica.

Este trabajo comienza con una introducción sobre el aprendizaje automático, presentando las redes neuronales artificiales y sus diferentes casos de uso. Dentro de este ámbito, se estudiará con mayor profundidad las redes de neuronas convolucionales las cuales se aplicarán para poder identificar y clasificar vehículos en vídeos o circuitos cerrados de televisión (CCTV). Estas redes neuronales recogerán información de entrada y producirán una salida la cual se compondrá de la clase del vehículo identificado y unos valores que delimitarán la posición del objeto en la imagen.

Para ello, se ha hecho uso de varios modelos pre-entrenados que fueran capaces de detectar las distintas clases de vehículos de interés (vehículo ligero, vehículo pesado y turismo), tanto en un vídeo de referencia como en un vídeo grabado por el alumno. Posteriormente, se obtuvieron los resultados de cada modelo y se realizó una comparación en cuanto al acierto, precisión y velocidad entre todos los modelos.

Índice general

Índice de figuras

Índice de tablas

1 Introducción

1.1 Contexto histórico.

La inteligencia artificial (IA) es un campo de la investigación en informática e ingeniería que se ocupa del desarrollo de máquinas que pueden realizar tareas que, normalmente, requieren inteligencia humana. Estas tareas incluyen el aprendizaje, que incluye la obtención de información y reglas sobre la aplicación; razonamiento basado en reglas para llegar a conclusiones aproximadas o definitivas; corrección automática o resolución de problemas [1].

A mediados del siglo XX, los investigadores iniciaron el estudio de la posibilidad de creación de máquinas que fueran capaces de simular la inteligencia humana. Algunos hitos claves de su historia se encuentran recogidos en la Tabla 1:

Tabla 1. Tabla que recoge los avances desde los inicios de la IA hasta el presente.

Fecha	Evento/Avance
1943	Warren McCulloch y Walter Pitts proponen el concepto de red neuronal por primera vez [2].
1950	Alan Turing expone el "Test de Turing" como una medida de la habilidad de una máquina para exhibir un comportamiento inteligente [3].
1956	Conferencia de Darthmouth, considerada el lugar de nacimiento de la IA como campo de estudio.
1957	Rosenblatt introduce el perceptrón, una red neuronal artificial fundamental en el aprendizaje automático.
1960-1970	Investigación en sistemas basados en reglas y sistemas expertos.
1980-1990	Con la aparición de Internet en 1993 sucedieron hitos como el desarrollo del Big Data, la red 5G, avances en IA como el aprendizaje automático y el PLN.
2000-presente	Avances en IA impulsados por grandes conjuntos de datos, computación en la nube y el aprendizaje profundo y por refuerzo.

En el camino, ha habido muchos avances y parones en el desarrollo de la IA, así como debates sobre las implicaciones éticas y sociales de esta. Sin embargo, la historia de la IA se ha caracterizado por un esfuerzo continuo por crear máquinas que puedan simular la inteligencia humana y mejorar nuestra capacidad para resolver problemas complejos y tomar decisiones.

La IA se adentró en nuestra vida cotidiana con aplicaciones como el buscador de Google que aprende nuestras preferencias, anuncios relacionados con dichas búsquedas, recomendaciones de series en plataformas como Netflix, seguridad vial, medicina, etc.

1.1.1 El Perceptrón

En el trabajo de Rosenblatt [4], el perceptrón es un tipo de red neuronal artificial desarrollada por el psicólogo e informático estadounidense Frank Rosenblatt a finales de la década de 1950. Es uno de los primeros y más influyentes algoritmos en el entorno del aprendizaje automático.

El perceptrón reconoce patrones en el conjunto de datos de entrada. Se trata de un conjunto de unidades de entrada que recoge los datos, un conjunto de pesos que se utilizan para procesar los datos y una unidad de salida que como resultado devuelve si el patrón de entrada pertenece a una determinada categoría o no, véase figura 1.

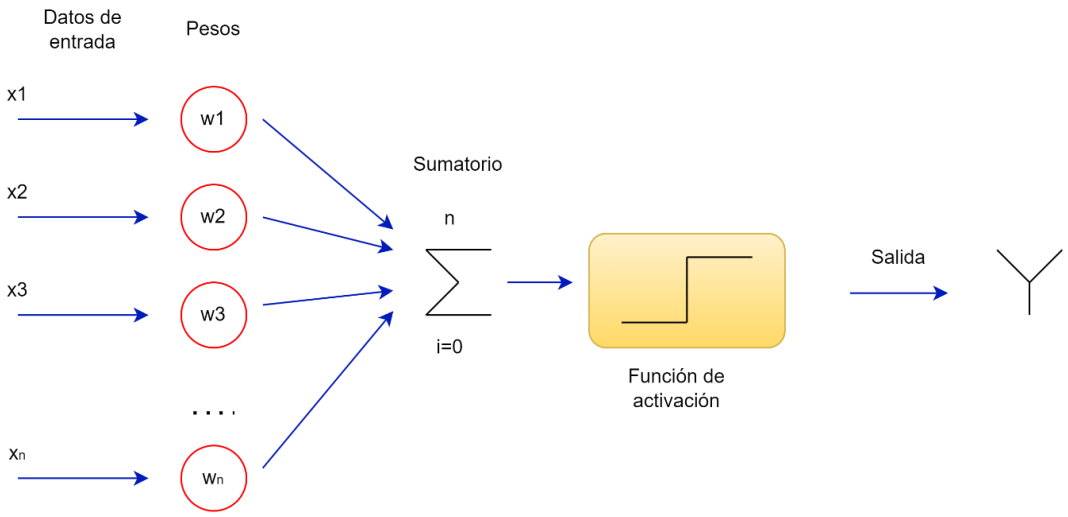

Figura 1. Esquema de los procesos que existen desde la recogida de los datos de entrada hasta devolver la salida binaria del Perceptrón.

El transcurso del entrenamiento del perceptrón es el elemento principal para su capacidad de encontrar patrones en los datos de entrada y así ajustar los parámetros en la salida. Durante el entrenamiento, los pesos del perceptrón van ajustándose en base a los datos de entrada y la salida que se desee, siendo el objetivo minimizar la diferencia entre la salida prevista por el perceptrón y la salida real. El proceso por el cual los pesos se van ajustando se conoce como *back propagation*.

El perceptrón fue innovador porque fue el primer algoritmo que pudo aprender a reconocer patrones en datos del mundo real. Rosenblatt demostró que el perceptrón podía entrenarse para reconocer dígitos y letras escritos a mano, allanando el camino para el desarrollo de otros sistemas de reconocimiento de patrones.

Sin embargo, el perceptrón tiene limitaciones. Sólo puede clasificar datos separables linealmente, debido a que hacía uso de funciones de activación lineales. Esto significa que sólo puede clasificar con precisión los datos que se pueden separar por una línea recta. También es propenso al sobreajuste u *overfitting*, lo que significa que puede tener un rendimiento deficiente en datos nuevos. A pesar de sus limitaciones, el perceptrón sentó las bases para el desarrollo de modelos de redes neuronales más avanzados, como el aprendizaje profundo.

En la actualidad, se utilizan perceptrones de capas múltiples que son capaces de aprender y resolver problemas complejos gracias a la incorporación de funciones de activación no lineales en sus capas ocultas.

1.1.2 Red neuronal

Como conocemos a partir del trabajo de Warren McCulloch y Walter Pitts [2], una red neuronal en inteligencia artificial es un modelo matemático que está diseñado para procesar y analizar datos de manera similar a cómo lo hace el cerebro humano. Se basa en un grupo de neuronas conectadas entre sí que trabajan juntas para resolver un problema específico.

Cada neurona en una red neuronal procesa información de entrada, por separado, y la combina con información de otras neuronas para producir una salida. Dicha salida de la neurona puede

llegar a ser la entrada en otra neurona, hasta que se produce una salida final que representa el valor de salida del modelo como se puede apreciar en la figura 2.

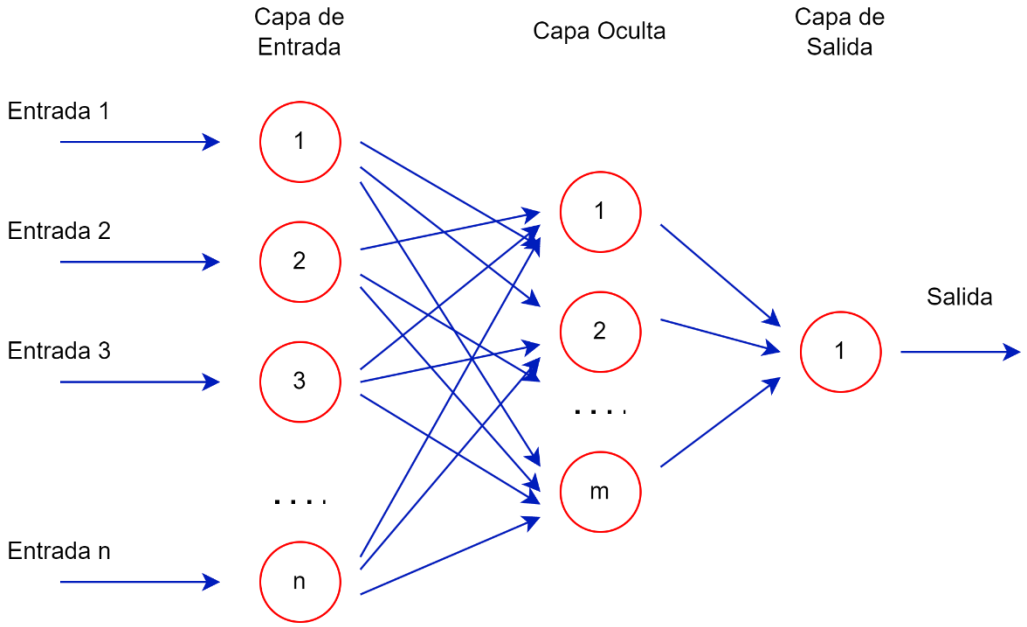

Figura 2. Arquitectura de una red neuronal que recibe parámetros de entrada, los procesa en las capas ocultas y devuelve una salida.

Es amplio el campo de uso de las redes neuronales como en el reconocimiento de patrones [5], la clasificación de imágenes [6], el procesamiento del lenguaje natural [7], la traducción automática y la toma de decisiones. Con el avance de la tecnología y el aumento de la capacidad del procesamiento en los ordenadores, las redes neuronales se han vuelto más sofisticadas y efectivas en la resolución de problemas complejos.

1.2 Descripción de la IA

La finalidad de la IA es crear máquinas capaces de realizar tareas mediante algoritmos y modelos que les permiten aprender datos y experiencias, adaptarse a diferentes situaciones y contextos y tomar decisiones autónomas.

Las máquinas pilotadas por inteligencia artificial tienen dos características intrínsecas: autonomía y adaptabilidad.

- La **autonomía** que es la capacidad de funcionar sin una ayuda constante del usuario, pudiendo realizar tareas por cuenta propia.

- La **adaptabilidad** que es la capacidad de mejora en el trabajo que desempeñe mediante la experiencia adquirida.

La IA se basa en la idea de que las computadoras se pueden programar para simular un comportamiento inteligente [1], [8], lo que incluye aprender de los datos, hacer predicciones y adaptarse a nuevas situaciones.

Hay varios enfoques de IA como el aprendizaje automático y, recogido dentro de este, el aprendizaje profundo, ambos ilustrados en la figura 3. El aprendizaje profundo o *deep learning* (DL) es un tipo de aprendizaje automático que, a diferencia de las redes neuronales tradicionales con una o dos capas ocultas, puede tener decenas, cientos o incluso miles de capas ocultas. Esta estructura en capas múltiples permite capturar y representar de manera más efectiva características y patrones complejos presentes en los datos. Utiliza redes neuronales muy grandes para aprender a partir de una mayor cantidad de datos.

La inteligencia artificial cubre una amplia gama de campos, que incluyen:

- **Aprendizaje automático:** el estudio de algoritmos y modelos estadísticos que permiten que los sistemas informáticos aprendan automáticamente de los datos, sin ser programados explícitamente.

- **Aprendizaje profundo:** es una parte del aprendizaje automático que utiliza redes neuronales artificiales para estructurar y resolver problemas complejos.

- **Visión artificial:** la capacidad de las máquinas para analizar, interpretar y comprender datos visuales del mundo, como imágenes y vídeos. La visión por computadora se utiliza en diversas aplicaciones, como automóviles autónomos, sistemas de seguridad e imágenes médicas.

- **Robótica:** el uso de la IA para controlar y automatizar máquinas físicas y robots, lo que les permite realizar tareas en diversas industrias, como la fabricación, la logística y la atención médica.

- **Sistemas expertos:** el uso de IA para crear sistemas que puedan razonar y tomar decisiones en dominios específicos, como el diagnóstico médico, la planificación financiera y el análisis legal.

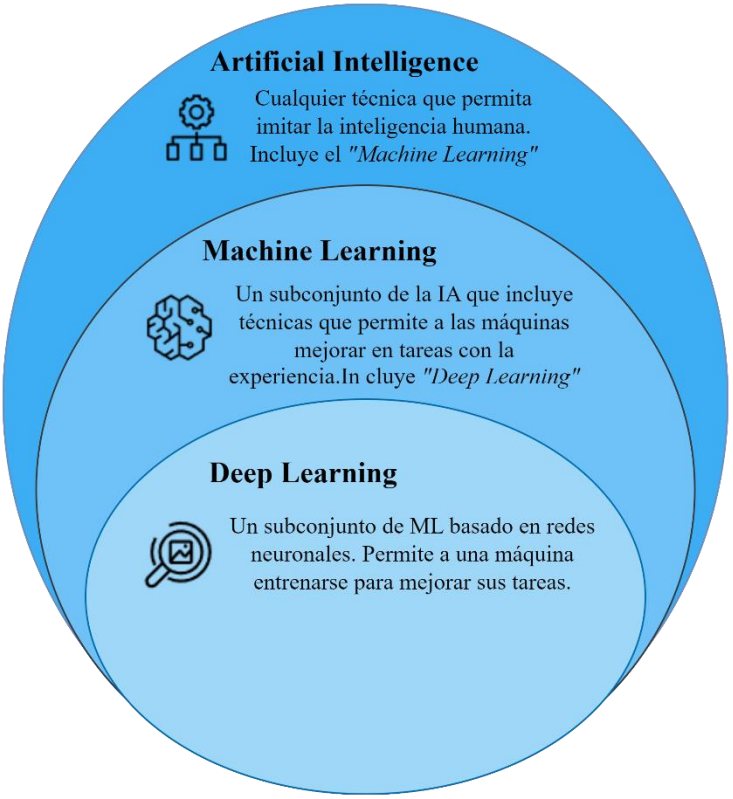

Figura 3. El machine learning es un subcampo de a IA y, a su vez, el deep learning es un subcampo del ML.

La IA tiene muchas aplicaciones, como por ejemplo en diagnósticos médicos, detección de fraudes financieros, vehículos autónomos y aprendizaje personalizado. Sin embargo, también existen preocupaciones sobre el impacto de la IA en la sociedad, incluido el desplazamiento laboral, los sesgos en los algoritmos y el posible uso indebido de la tecnología.

1.3 Aprendizaje automático o *machine learning*

"El machine learning es un subcampo de la inteligencia artificial que implica el uso de algoritmos y modelos estadísticos para permitir que los sistemas informáticos aprendan automáticamente de los datos, sin ser programados explícitamente" [9]. En el aprendizaje automático, las computadoras están capacitadas para reconocer patrones y hacer predicciones basadas en datos de entrada.

Hay tres tipos principales de aprendizaje automático. Dos de ellos, el aprendizaje supervisado [10] y el no supervisado, se diferencian en la clasificación de un conjunto de datos etiquetado o por una agrupación de patrones similares que contenga el propio set de datos tal y como se puede ver en la figura 4.

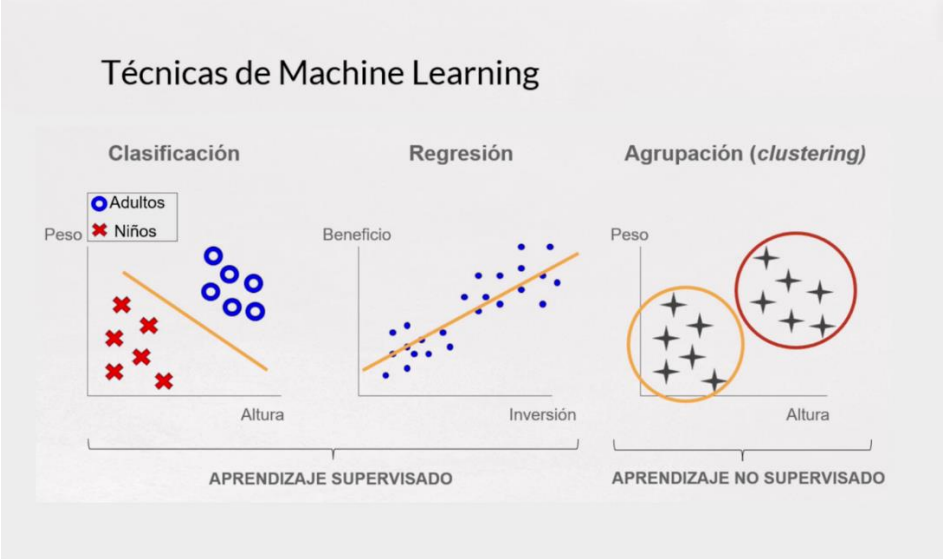

Figura 4. Comparativa de cómo se clasifica una máquina supervisada, la cual ya conoce la etiqueta del dato, frente a una no supervisada que va agrupando conjuntos de datos para clasificarlos en distintas etiquetas [11].

1.3.1 Aprendizaje supervisado

El aprendizaje supervisado es una técnica de *machine learning* que implica entrenar un modelo que contiene sus datos ya etiquetados. En otras palabras, el modelo recibe datos de entrada junto con sus correspondientes etiquetas de salida y utiliza esto para aprender a hacer predicciones precisas en nuevos conjuntos de datos que no conoce.

El proceso de aprendizaje supervisado se puede dividir en dos fases: entrenamiento y evaluación.

Durante la fase de entrenamiento, el modelo se ajusta a los datos de entrenamiento y se utilizan técnicas como la regresión, la clasificación o el análisis de series temporales para hacer predicciones precisa mientras que, durante la fase de evaluación, el modelo se prueba en un conjunto de datos de prueba para medir su capacidad para generalizar y hacer predicciones precisas en nuevos datos.

Es en esta fase de prueba donde entra en juego la importante técnica de validación cruzada (*cross-validation*, en inglés). Esta técnica implica que hay dividir el conjunto de datos en varios subconjuntos donde una parte se dedicará al entrenamiento y la otra a la comprobación de su desempeño. Esta última parte de "*testing"* irá variando en distintos subconjuntos como podemos ver en la figura 5. De este modo, a partir del mismo conjunto de datos, se podrán comprobar cómo fue de bien el entrenamiento en las distintas iteraciones.

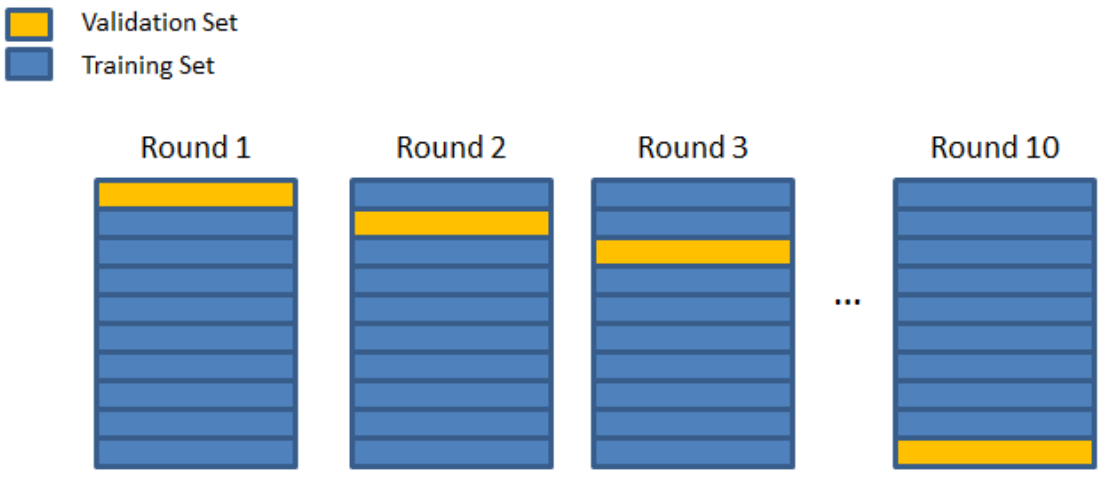

Figura 5. Asignación de la parte de test en el conjunto de datos para comprobar el desempeño de un modelo [12].

Algunos ejemplos comunes de aplicaciones de aprendizaje supervisado incluyen la clasificación de imágenes o la predicción de precios de acciones. Para implementar el aprendizaje supervisado, se necesitan algoritmos de aprendizaje automático adecuados, como regresión lineal, regresión logística, árboles de decisión y redes neuronales.

A su vez, se ha de utilizar métricas de evaluación adecuadas para asegurarse de que el modelo se esté ajustando correctamente a los datos y haciendo predicciones precisas.

1.3.2 Aprendizaje no supervisado

El aprendizaje no supervisado es una técnica de *machine learning* que consiste en entrenar un modelo sin datos etiquetados. En lugar de proporcionar al modelo datos de entrada y salida, se le proporciona datos sin valor de salida y se espera que el modelo descubra patrones y estructuras por sí solo.

Se dedica a detectar patrones con características similares entre los datos que son de entrada para así poder categorizarlos. Por ejemplo, se pueden extraer patrones de datos masivos obtenidos a través de aplicaciones para recogida de datos y así poder crear campañas publicitarias.

Los algoritmos de aprendizaje no supervisado se utilizan principalmente para tareas de *clustering* y reducción de dimensionalidad. El *clustering* implica agrupar los datos en diferentes grupos o *clusters* basados en sus características similares, mientras que la reducción de dimensionalidad codifica las características de los datos que explora pero sin perder información relevante.

Para implementar el aprendizaje no supervisado, se utilizan técnicas como el análisis de componentes principales (PCA), la agrupación *k-means*, la agrupación jerárquica, etc. Es importante tener en cuenta que el aprendizaje no supervisado actúa diferente que el aprendizaje supervisado, ya que el modelo debe encontrar patrones por sí solo sin la ayuda de datos etiquetados.

Por lo tanto, la evaluación del rendimiento del modelo en el aprendizaje no supervisado puede ser más subjetiva y depende en gran medida del juicio humano. Sin embargo, si se implementa

correctamente, el aprendizaje no supervisado puede ser muy útil para descubrir información oculta y patrones en los datos de entrada.

1.3.3 Aprendizaje por refuerzo

Es una técnica de *machine learning* que se centra en la interacción del modelo con un ambiente dinámico. En lugar de proporcionar al modelo datos de entrada etiquetados o sin etiquetar, se le proporciona información de recompensa o castigo después de cada acción que toma en el ambiente.

La finalidad del aprendizaje por refuerzo es que el modelo aprenda a tomar las acciones correctas para maximizar la recompensa acumulada a largo plazo en el ambiente. A medida que el modelo interactúa con el ambiente, aprende a ajustar su política de decisión para tomar mejores decisiones.

Algunos ejemplos comunes de aplicaciones de aprendizaje por refuerzo incluyen los juegos de estrategia, como Go o ajedrez, los vehículos autónomos y los sistemas de recomendación.

Para implementar el aprendizaje por refuerzo, se utilizan técnicas como los algoritmos *Q-Learning*, los métodos de Monte Carlo y la aproximación de función de valor. Además, es importante diseñar un ambiente adecuado y definir correctamente las recompensas y penalizaciones para asegurarse de que el modelo aprenda a tomar las acciones correctas.

El aprendizaje por refuerzo es un área emocionante de la investigación de machine learning, ya que tiene aplicaciones potenciales en una amplia variedad de áreas, incluyendo robótica, economía y salud.

La figura 6 trata de un esquema que ,por medio de una interacción del agente con el ambiente y con la ayuda de unas recompensas, será capaz de autogestionar y aprender por sí mismo la validez de sus acciones.

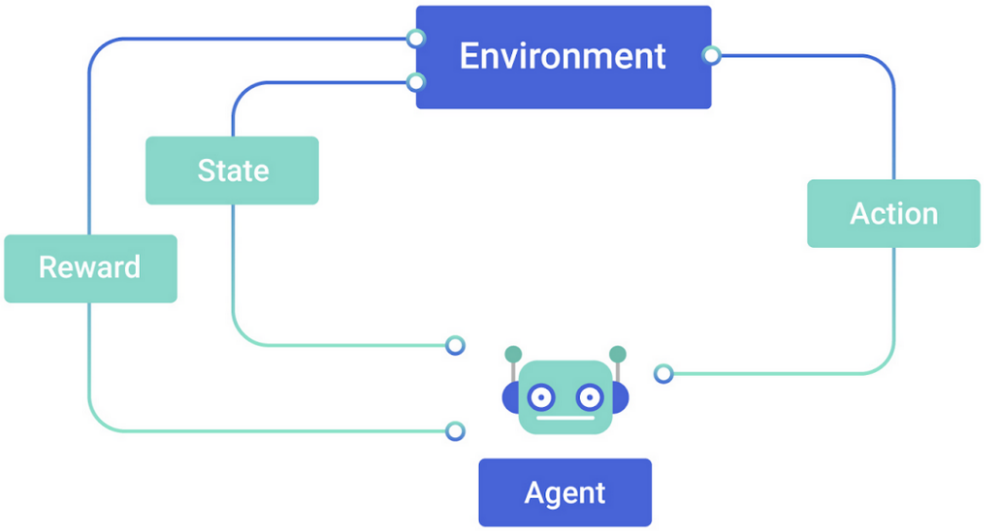

Figura 6. Estructura de la interacción de una máquina que aprende por refuerzo [13].

1.4 Aprendizaje profundo o *deep learning*

El *deep learning* es un fragmento que pertenece al aprendizaje automático basado en redes neuronales que usan múltiples capas para aprender representaciones de datos [14]. Las redes neuronales profundas, debido al uso de múltiples capas, tienen la capacidad de aprender características cada vez más complejas a partir de datos sin procesar. Esto les permite desempeñar tareas de mayor esfuerzo como el reconocimiento de imágenes, reconocimiento de voz y procesamiento del lenguaje natural (PLN).

Las características clave que diferenciar al aprendizaje profundo del aprendizaje automático son: el uso de múltiples capas de neuronas artificiales, la capacidad de aprender representaciones jerárquicas de datos y el uso de retro-propagación para ajustar por su cuenta los pesos de la red durante el entrenamiento.

El aprendizaje profundo ha sido posible gracias a los avances en hardware informático y la disponibilidad de grandes conjuntos de datos, así como al desarrollo de nuevos algoritmos y arquitecturas para redes neuronales.

El DL mejora, con creces, las tecnología empleada por su antecesor ya que cada capa de neuronas trabaja la información y devuelve un resultado ponderado. Por ejemplo, en el presente caso, si ofreciéramos una ingente cantidad de imágenes de coches a un ordenador y buscásemos un tipo en concreto (vehículo pesado, vehículo ligero o turismos) detallando las características principales, un modelo de *deep learning* sería capaz de procesar los resultados y ponderar con un porcentaje la posibilidad de que sea o no el tipo buscado.

El sistema se basa en aprender en cada iteración pudiendo así tener claro qué dato buscar y cuál no en la siguiente iteración y mejorar los resultados. Es fundamental cómo se trabaja con los errores porque cada vez que define que se ha equivocado, añade un nuevo dato a la red que no repetirá en una búsqueda similar.

La definición de *deep learning* apareció, por primera vez, en los estudios sobre el aprendizaje biológico de Warren McCulloch y Walter Pitts [2] y Hebb [15]. Su época de máximo esplendor no fue hasta el año 2010 debido a la gran limitación que tenían los procesos de computación de la época. Gracias al aumento de usuarios en Internet y el constante apoyo en la investigación en el mundo web, se multiplican la generación de datos por personas con métodos como las redes sociales. Es por esto que surge la necesidad inmediata de gestionar y sacar provecho de estos datos.

1.5 Motivación y objetivo

La principal motivación de este trabajo surge de la necesidad de facilitar el conteo de vehículos a las personas en proyectos acústicos que requieran de la intensidad media diaria de tráfico (IMD) como dato en proyectos acústicos.

En la actualidad, el método para la clasificación y el reconocimiento de los distintos tipos de vehículos a través de vídeo ha evolucionado, dando lugar a una multitud de formas para obtener este tipo de información. En la actualidad, se busca no solo la cantidad de vehículos, sino también su tipo y características específicas.

Todo ello se debe a la propia experiencia del alumno en la realización de las prácticas de la asignatura Ingeniería Acústica 2, donde se requería emplear un tiempo considerable para contabilizar el número de vehículos ligeros, pesados y turismos que circulaban por una sección de la carretera que se encuentra frente a la universidad en un período de 15 minutos, para posteriormente realizar un cálculo proporcional a 1 hora.

Por consiguiente, la meta de este proyecto consiste en la automatización del conteo de vehículos, empleando técnicas de *deep learning* y visión artificial con el fin de realizar un conteo más efectivo y dinámico aprovechando las ventajas que proporciona la tecnología actual. Estas ventajas que se proporcionan se verán más en profundidad en el siguiente capitulo, donde se muestra una comparación entre las técnicas tradicionales y las técnicas de aprendizaje profundo en cuanto a la precisión y el tiempo de procesamiento, costo y mantenimiento.

2 Estado del arte

2.1 Intensidad media diaria

La intensidad media diaria de tráfico (IMD) es el número de vehículos que circulan por un tramo/punto kilométrico (PK) de la vía, partido por el número total de días que tiene un año (365). Se expresa en términos de número de vehículos que pasan por un punto determinado de la carretera en un día [16].

El valor IMD es necesario a la hora de planificar o diseñar vías de transporte, gracias a que ayuda a definir la capacidad y las necesidades de la red vial. A su vez es útil para el seguimiento, la evaluación del tráfico y seguridad del tráfico.

En la actualidad, existen diferentes métodos para la recolección de los datos de interés acerca del estado del tráfico. Las metodologías expuestas en la figura 7, hacen uso de una serie de dispositivos capaces de recabar datos como la intensidad de la vía, la velocidad de circulación y el tipo de vehículo (ligero, pesado o turismos). A continuación, se mencionan dichas metodologías.

- **Aforos manuales:** un usuario se sitúa frente al PK que se quiera medir la intensidad de tráfico y. junto a pulsadores o tableros electrónicos, lleva un conteo manual de los vehículos que circulan.

- **Aforos Neumáticos:** se sitúan dos láminas metálicas (detectores) que se activan cada vez que un coche cada vez que circula un coche por encima de un captador, registrado así cada vehículo que pasa [17].

- **Visión artificial:** se trata digitalmente por fotogramas los vídeos recogidos en cámaras CCTV y, junto con algoritmos dedicados, se analizan esta imágenes para obtener valores como la intensidad de la vía, velocidad y longitud de los vehículos.

- **Radar de microondas:** se emite energía de alta frecuencia directamente al carril por donde transitan los vehículos detectando así la intensidad y velocidad de los vehículos.

- **Radar de ultrasonido:** al igual que el rada de microondas, se emiten ondas de sonido de manera perpendicular sobre la carretera. En función a los tiempos de ida y vuelta de dichas ondas si circula un vehículo o si chocan contra el pavimento, se detecta un vehículo o no [18].

Existen más tecnologías para la medición de vehículos como: basadas en Bluetooth, detectores infrarrojos, detectores magnéticos, por inducción electromagnética… Los cuales no se consideran relevantes para el presente proyecto ya que no aportan la suficiente información ni están relacionados con nuestro método.

	Sistemas de captación	Ventajas	Desventajas
Métodos basados en visión	Aforos Manuales	• Alta disponibilidad • No interfieren en la carretera • Empleo no especializado	• Poca fiabilidad • Se puede ver alterado por las condiciones ambientales
	Visión Artificial	• No interfieren en la carretera • Alta fiabilidad • No sufren desgasto por el paso de vehículos • No sufren desgasto por labores de mantenimiento en la carretera • Recogen movimientos complejos de intersecciooón	• Alto conste de instalación • Requieren un punto de visión perpendicular • Su precisión cae en función de las condiciones de visibilidad de la vía
Métodos basados en sensores magnéticos	Radar de Microondas	• No interfieren en la carretera • Buen funcionamiento con meteorología adversa • Gran precisión para medir la velocidad	• No capta los vehículos que circulan a una velocidad <15km/h • Mala adaptación si hay congestión en el tráfico
	Radar de Ultrasonido	• Fáciles de instalar	• Detectores MUY sensibles a la temperatura y al viento
	Aforos Neumáticos	• Funcionamiento simple • Bajo coste	• Cortes en los tramos para su instalación • Duración limitada por el uso de baterías

Figura 7. Ventajas y desventajas de los distintos métodos empleados en captación de vehículos.

2.1.1 Cálculo de la IMD

Para calcular la intensidad media diaria de tráfico se puede utilizar la ecuación (1):

$$IMD = \frac{V_1 + V_2 + V_3 + \ldots + V_n}{n}, \qquad (1)$$

donde:

- $V_1, V_2, \ldots, V_n$ son las medidas de tráfico registradas en cada uno de los n días de conteo.

- **n** es el número de días de conteo.

Es importante destacar que el conteo de tráfico debe realizarse en días típicos y representativos del tráfico en la zona que se quiere medir. Asimismo, es recomendable realizar varios conteos de tráfico para tener una medida más precisa de la IMD.

2.1.2 Aumento de tráfico en carreteras de España

El aumento de la movilidad en carretera producido alrededor de los años 70 en España se debe a las variaciones en la actividad económica, social y territorial. Multiplicándose así la distancia recorrida por cada ciudadano. Por ello fue necesario cuantificar el tráfico existente.

Fue gracias a esto que se realizó el primer Plan Anual de Aforos sobre los 80.000 Km que componía la Red de Carreteras del Estado de entonces por la Dirección General de Carreteras del Ministerio de Fomento, uniéndose a partir de 1990 los planes de aforos autonómicos y provinciales.

En la Red de Carreteras de España (RCE) existen aforos que captan el tráfico que circula por cada tramo de carretera medida, aportando importantes datos como la medida IMD, el porcentaje de vehículos pesados, vehículos ligeros, turismos, la intensidad diaria de vehículos

extranjeros y más.

Los métodos del cálculo de la medida IMD, en su combinación con *deep learning*, pueden proporcionar ayuda en situaciones como: evitar congestión de tráfico, prevención de accidentes, contaminación acústica de una carretera a edificios lindantes, peaje de carreteras, la vigilancia/planificación de carreteras, la seguridad del tráfico, la conducción autónoma, los sistemas de gestión de estacionamientos, etc.

2.2 Visión artificial

La visión artificial por computadora es un campo de estudio en continuo desarrollo. Los pilares sobre los que se fundamenta son: la adquisición, procesamiento y análisis de imágenes tomadas del mundo real. Tiene como objetivo final el de obtener información relevante acerca de dichas imágenes para poder trabajar en disciplinas como la detección de objetos, seguimiento del movimiento, reconocimiento de eventos, entre otros.

Desde la perspectiva ingenieril, los sistemas basados en visión artificial son los sistemas capaces de imitar acciones del sistema de visión humano. Algunos ejemplos de dichos sistemas se encuentran en el campo de la robótica, en los procesos de inspección automática, el análisis de imágenes médicas, entre otros. La manera más cercana que tenemos de explicar su uso diario y cotidiano es en el reconocimiento de rostros tanto en una escena capturada como con nuestro teléfono móvil gracias al uso de técnicas de reconocimiento de patrones.

Para poder utilizar la información visual, es necesario transformarla a un formato en el que pueda ser procesada. En la visión humana, el ojo humano realiza esta tarea, mientras que, en la visión artificial, una cámara se encarga de convertir la energía luminosa en impulsos eléctricos que pueden ser muestreados y digitalizados para su procesamiento en un ordenador.

Asimismo, las diferentes partes que componen la estructura del ojo humano se pueden ver en la figura 8. Estas pueden representarse como componentes de una máquina para la visión artificial:

- Una posible semejanza es la del **cristalino** del ojo y el **microchip** en la visión artificial. Ambos funcionan como elementos ópticos que se encargan de centrar la luz y procesar la información visual para enfocar y mejorar la calidad de la imagen.

- De igual manera, tanto el **iris** como el **diafragma** en una cámara de visión artificial tienen la función de controlar la cantidad de luz que llegar al sensor o a la retina ajustando su "apertura" de acuerdo con las condiciones de iluminación.

- La función que desempeñan tanto la **retina** como el **sensor,** en visión artificial, es que ambos actúan como el receptor de la información visual y la convierten en una señal eléctrica para su procesamiento posterior.

- En el caso de la visión, el **lóbulo parietal** ayuda a interpretar la información visual, como la ubicación de los objetos en el espacio y la orientación de estos. Los datos que procesa el lóbulo parietal le llegan por medio del **nervio óptico**, el cual actúa como **canal de transmisión** para enviar contenido al **sistema de procesamiento**.

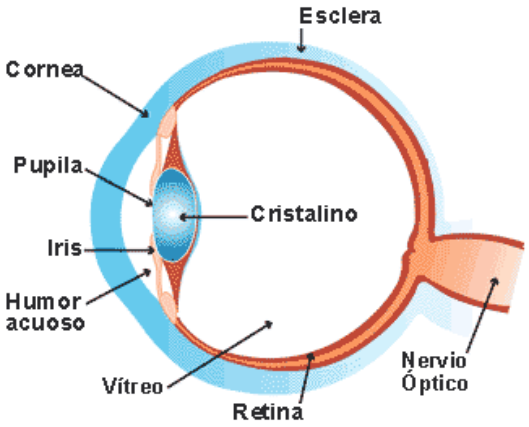

Figura 8. Demostración gráfica de cada componente del ojo humano que se puede asemejar a cómo trabajan los componentes de una máquina en visión artificial [19].

2.3 Clasificación de objetos

2.3.1 ¿Qué es una red neuronal convolucional?

Una red neuronal convolucional (CNN, por sus siglas en inglés) es un tipo de red neuronal artificial que utiliza una arquitectura especializada para adaptarse bien al procesamiento de datos de alta dimensión como imágenes y vídeos. Este tipo de redes neuronales se utilizan principalmente en el procesamiento de imágenes y vídeos. Esto se debe a que las capas de convolución en las redes neuronales convolucionales aplican filtros a pequeñas regiones del campo visual, permitiendo que la red identifique patrones visuales locales. A medida que la información fluye a través de las capas convolucionales, se capturan características de mayor nivel y se combinan para formar representaciones más complejas y abstractas

La arquitectura de una CNN se compone de varias capas, como se puede apreciar en la figura 9, que se encargan de la extracción de características y clasificación de las imágenes.

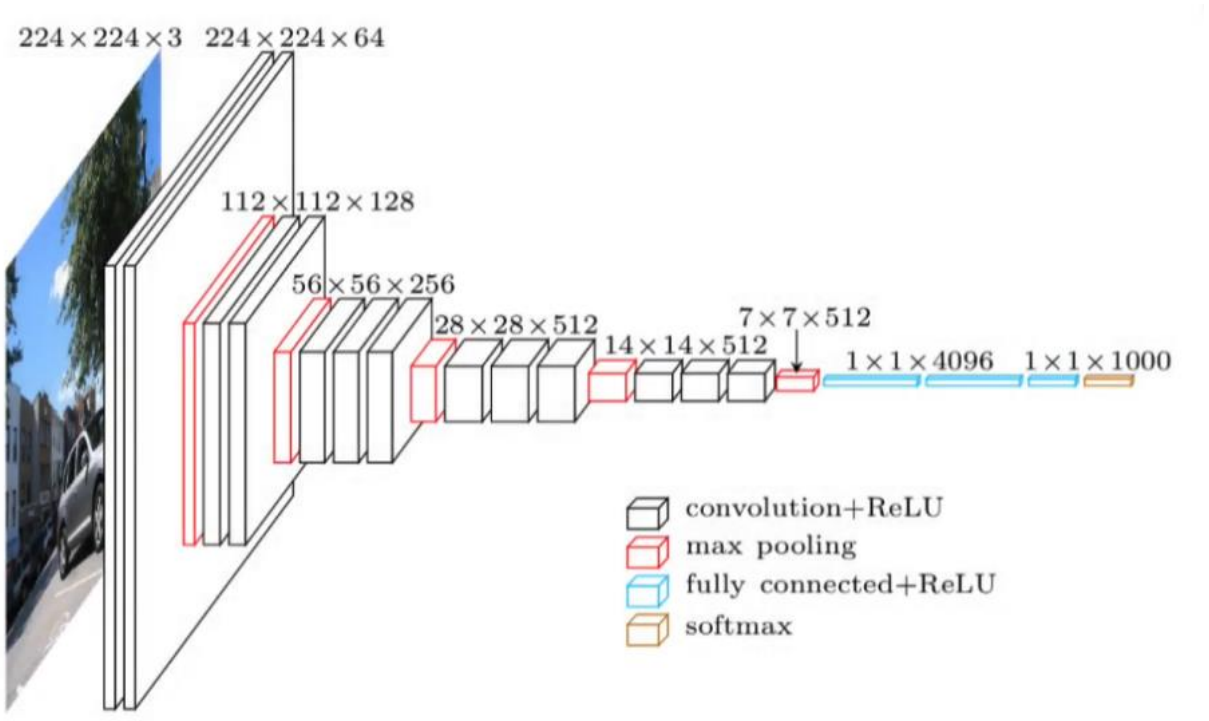

Figura 9. Ejemplo de la estructura de una red neuronal convolucional. [20]

A continuación, se describen las capas más comunes que componen una CNN:

- **Capa de entrada:** esta capa se encarga de recibir la imagen de entrada y prepararla para ser procesada por la red neuronal. Por lo general, se utiliza una capa de convolución o una capa de submuestreo como primera capa de procesamiento.

- **Capas convolucionales:** estas capas se encargan de extraer características de la imagen. Cada capa convolucional consta de varios filtros que se aplican a la imagen de entrada para detectar patrones específicos, como bordes, texturas y formas.

- **Capas de submuestreo:** estas capas se utilizan para reducir la dimensión de la imagen de entrada y reducir la complejidad de la red neuronal. La operación más común utilizada, detallada en la figura 10, en estas capas es el *max-pooling*, que selecciona el valor máximo de una región determinada de la imagen.

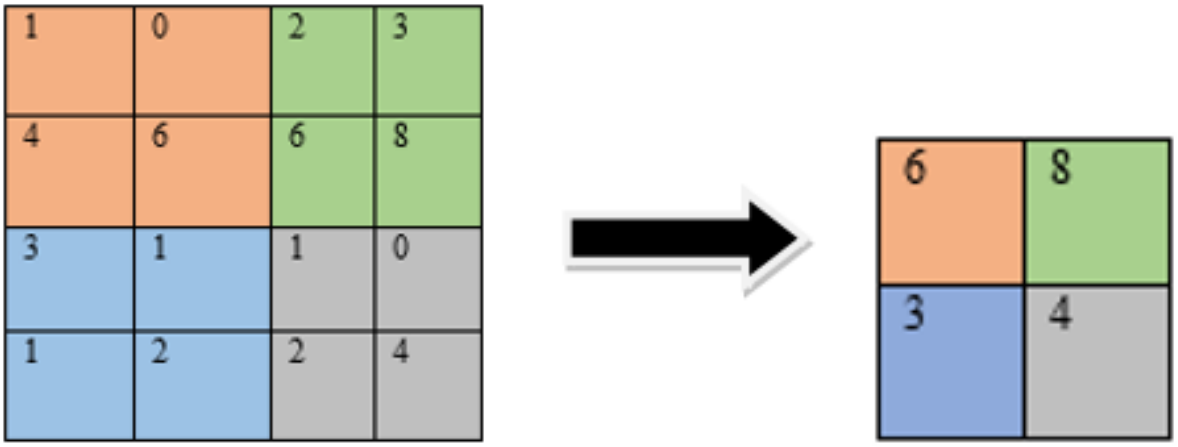

Figura 10. Demostración de cómo es el funcionamiento de la función max-pooling.

- **Capas totalmente conectadas:** estas capas se utilizan para clasificar la imagen de entrada en una de las categorías de salida. Cada neurona en se encuentra interconectada con todas las neuronas de la capa anterior.

- **Capa de salida:** esta capa es la última capa de la red neuronal y se utiliza para producir la salida final de la red, que puede ser una clasificación de la imagen de entrada o un valor numérico.

Existen varias arquitecturas de redes neuronales convolucionales que se han propuesto a lo largo de los años y que han demostrado un gran rendimiento en diferentes aplicaciones de procesamiento de imágenes y visión por computadora. Las arquitecturas que se muestran a continuación son solo algunas de las arquitecturas de CNN más conocidas. Cada una de ellas tiene sus propias características y ventajas, y la elección de la arquitectura dependerá de la naturaleza del problema que se esté abordando:

- **LeNet:** Fue una de las primeras arquitecturas de CNN desarrolladas por Yann LeCun en 1998. Fue diseñada para reconocimiento de dígitos escritos a mano [21] y consiste en varias capas convolucionales y de submuestreo seguidas de capas totalmente conectadas.

- **AlexNet:** Fue presentada en el concurso ImageNet en 2012 y inauguró la época actual de las CNN. La arquitectura abarca varias capas convolucionales, de submuestreo y *fully connected* (totalmente conectadas), utiliza la función de activación ReLU y la técnica de regularización de abandono (dropout) [22].

- **VGGNet:** Fue desarrollada por el grupo de investigación *Visual Geometry Group* en 2014. La arquitectura es muy profunda, consta de 19 capas convolucionales y utiliza filtros de tamaño 3x3. Fue diseñada para ser muy fácil de entender y de implementar [23].

- **GoogLeNet/Inception:** Fue presentada en el concurso ImageNet en 2014 y es conocida por su arquitectura en forma de *Inception*. La arquitectura consta de múltiples módulos que combinan diferentes tamaños de filtro y agrupaciones en una sola capa [24].

- **ResNet:** Fue desarrollada por Microsoft Research en 2015. La arquitectura se basa en el concepto de "residual learning", que permite que la red aprenda diferencias en lugar de características directamente. Es una arquitectura muy profunda, que consta de más de 100 capas [25].

- **MobileNet:** Fue presentada en 2017 y está diseñada para ser muy eficiente en términos de cálculo y memoria. La arquitectura utiliza filtros de tamaño reducido y separables para reducir el número de parámetros [26].

2.4 Detección de objetos

La detección de objetos es una tarea importante en visión por computadora que implica identificar y localizar objetos específicos en una imagen o video. El objetivo es detectar los objetos en la imagen tal y como se puede apreciar en la figura 11 y, en algunos casos, también proporcionar información sobre su posición en la propia imagen.

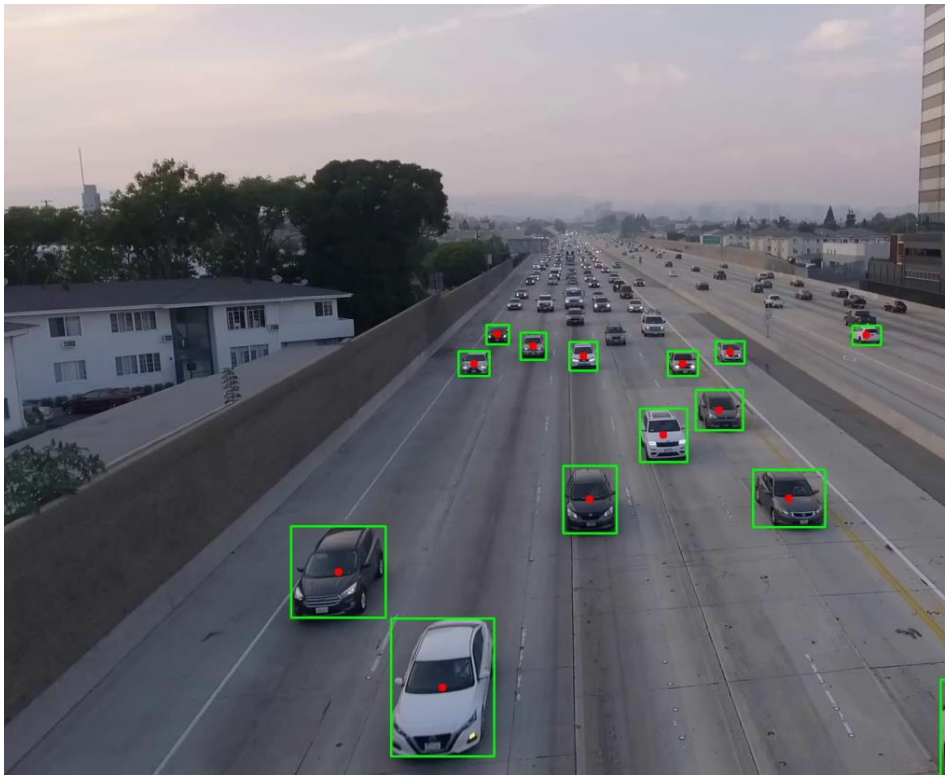

Figura 11. Detección de vehículos en una carretera con la herramienta de cuadro delimitador. [27]

La detección de objetos se utiliza en muchas aplicaciones prácticas, como sistemas de vigilancia, conducción autónoma, clasificación de imágenes, entre otras.

Las técnicas utilizadas para la detección de objetos suelen ser algoritmos de aprendizaje profundo que combinan capas de redes neuronales convolucionales con capas de regresión lineal. La parte convolucional se dedica al tratamiento de las imágenes mientras que las capas de regresión tratan la parte numérica, donde se encuentran las cajas delimitadoras que sitúan en la imagen el objecto detectado.

Existen varias herramientas que se utilizan para la detección de objetos en imágenes y vídeos. Algunas de las más populares son:

- **You Only Look Once (YOLO):** es un algoritmo de detección de objetos en tiempo real que utiliza redes neuronales convolucionales para identificar y localizar objetos en una imagen o un video. YOLO fue desarrollado por Joseph Redmon, Santosh Divvala, Ross Girshick y Ali Farhadi en 2016.

  El algoritmo YOLO es capaz de detectar varios objetos en una sola imagen o cuadro de video y proporciona la localización precisa de cada objeto en términos de su caja delimitadora y una probabilidad de que el objeto detectado sea de una determinada clase.

  YOLO utiliza una única red neuronal para realizar tanto la detección como la clasificación de objetos, lo que lo hace más rápido que otros enfoques de detección de objetos que requieren múltiples etapas de procesamiento.

  Desde su lanzamiento, YOLO ha sido ampliamente utilizado en aplicaciones de visión por computadora que requieren detección de objetos en tiempo real, como sistemas de seguridad, conducción autónoma, vigilancia de la seguridad pública, detección de objetos en imágenes médicas y mucho más. YOLO ha sido actualizado varias veces, con versiones como YOLOv2, YOLOv3 y YOLOv4 que han mejorado aún más la precisión y el rendimiento del algoritmo.

- **Single Shot Detector (SSD):** es un algoritmo de detección de objetos que utiliza una única red neuronal para detectar objetos en una imagen y realiza una única "pasada" a través de la imagen. Al igual que YOLO, SSD es capaz de realizar la detección de objetos en tiempo real y proporciona alta precisión.

- **Faster R-CNN:** es otro algoritmo de detección de objetos que utiliza una red neuronal convolucional y una técnica de regiones propuestas de objetos. Aunque es más lento que YOLO y SSD, Faster R-CNN proporciona una mayor precisión en la detección de objetos.

- **RetinaNet:** es una red neuronal diseñada específicamente para la detección de objetos en imágenes con una gran cantidad de objetos pequeños. Al igual que YOLO, RetinaNet utiliza una única red neuronal para la detección de objetos y proporciona una alta velocidad y precisión.

- **Mask R-CNN:** es una extensión del algoritmo Faster R-CNN que también proporciona información sobre la segmentación de objetos. Además de detectar objetos, Mask R-CNN es capaz de generar máscaras precisas para cada objeto detectado.

- **Tensorflow Object Detection API:** es una biblioteca de aprendizaje automático de código abierto que ofrece recursos para el reconocimiento de objetos en tiempo real y el seguimiento de objetos en video. Esta API se basa en Tensorflow, una biblioteca de aprendizaje automático de Google, y utiliza modelos de detección de objetos pre-entrenados para detectar objetos en imágenes y vídeos.

2.4.1 YOLOv8

En esta ocasión, se ha utilizado una versión no oficial de YOLO, la cual ha sido YOLOv8.
Yolov8 es una versión no oficial desarrollada por la empresa Ultralytics de una manera abierta y tiene toda la información de cómo usarlo e información relevante en su repositorio de GitHub.

La manera de usarlo es importándolo como si fuera una librería más de Python tal y como podría ser la librería NumPy y desde ahí llamar a las clases creadas dedicadas a la detección de objetos.

El proceso que siguió Ultralytics fue entrenar la última versión oficial existente, YOLOv5, con el conjunto de datos COCO para obtener una precisión alta en detección de objetos. La arquitectura de la red convolucional usada se llama CSP (*Cross-Stage Partial connections*) y es una versión mejorada de la arquitectura Darknet utilizada en versiones anteriores de YOLO [28].

El proceso de entrenamiento de YOLOv5 consistió en recopilar y etiquetar imágenes en el conjunto de datos COCO, y luego utilizar estas imágenes etiquetadas para entrenar la red

neuronal convolucional CSP en el marco de detección de objetos. Durante el entrenamiento, la red ajusta sus pesos y varianza para minimizar la función de pérdida y así lograr una alta precisión en la detección de objetos en nuevas imágenes.

Después del entrenamiento, la red neuronal CSP de YOLOv5 se puede utilizar para detectar objetos en imágenes y videos en tiempo real con una alta precisión y velocidad.

Tal y como dice Ultralytics en su repositorio: "*YOLOv8 está diseñado para ser rápido, preciso y fácil de usar, lo que lo convierte en una excelente opción para una amplia gama de tareas de detección y seguimiento de objetos, segmentación de instancias, clasificación de imágenes y estimación de poses".* Es por esto por lo que se ha decidido usar en este trabajo, debido a su fácil implementación e intuitivo uso, alto nivel de precisión y eficacia.

Asimismo, Ultralytics deja a nuestra disposiciones diferentes modelos pre-entrenados. La diferencia entre estos modelos se muestra en la figura 12. Cada modelo tendrá una precisión y una velocidad distintas, por lo que se buscará un término medio entre estas dos características para hacer funcionar nuestro modelo.

Model	size (pixels)	mAP^{val} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

Figura 12. Valores de los modelos pre entrenados de YOLOv8 por Ultralytics [29].

- Los valores mAP (*mean average precission*) son para un solo modelo a una sola escala en el conjunto de datos val2017 COCO.

- La velocidad es promediada sobre imágenes de validación COCO utilizando una instancia de Amazon EC2 P4d. La instancia de Amazon EC2 P4d es una máquina virtual que se crea y administra a través del servicio de AWS, y está dotada con GPU NVIDIA A100 Tensor Core, que son unidades de procesamiento gráfico especializadas en cálculos intensivos y aceleración de tareas de aprendizaje automático.

- El valor de *"params"* hace referencia al número de parámetros con los que ha sido entrenado el modelo. Es por ello por lo que a medida que tenemos menos parámetros en la red neuronal, disminuye significativamente la precisión en la detección de objetos, pero aumenta la velocidad de procesado.

En la figura 13 podemos comprobar como a medida que va disminuyendo en número de parámetros usados será más rápido, pero menos preciso. A su vez, se muestra una comparativa entre los modelos pre-entrenados de YOLOv8 y versiones anteriores desarrolladas por Ultralytics también.

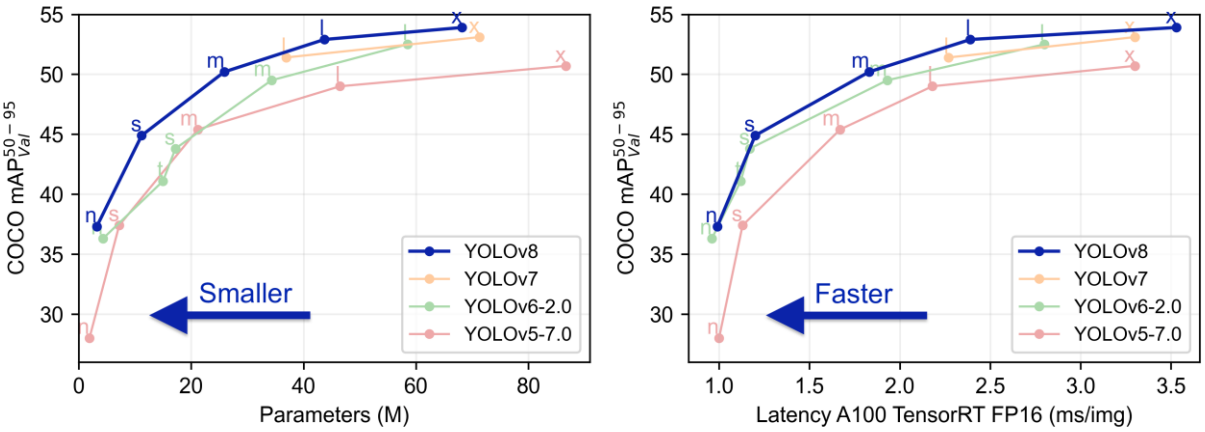

Figura 13. Comparativa entre las distintas versiones pre-entrenadas de YOLOv8 [29].

2.5 Seguimiento de objetos

El seguimiento de objetos es una técnica utilizada en visión por computadora para rastrear y localizar la posición de un objeto en movimiento a lo largo del tiempo en una secuencia de imágenes.

El objetivo es identificar y seguir el mismo objeto a través de múltiples fotogramas, incluso si el objeto cambia de forma, tamaño, posición o apariencia en el transcurso de la secuencia de imágenes. Para ello se podrán aplicar técnicas como representar mediante puntos la dirección del movimiento que tiene un objeto a lo largo de los fotogramas de un video como en la figura 14.

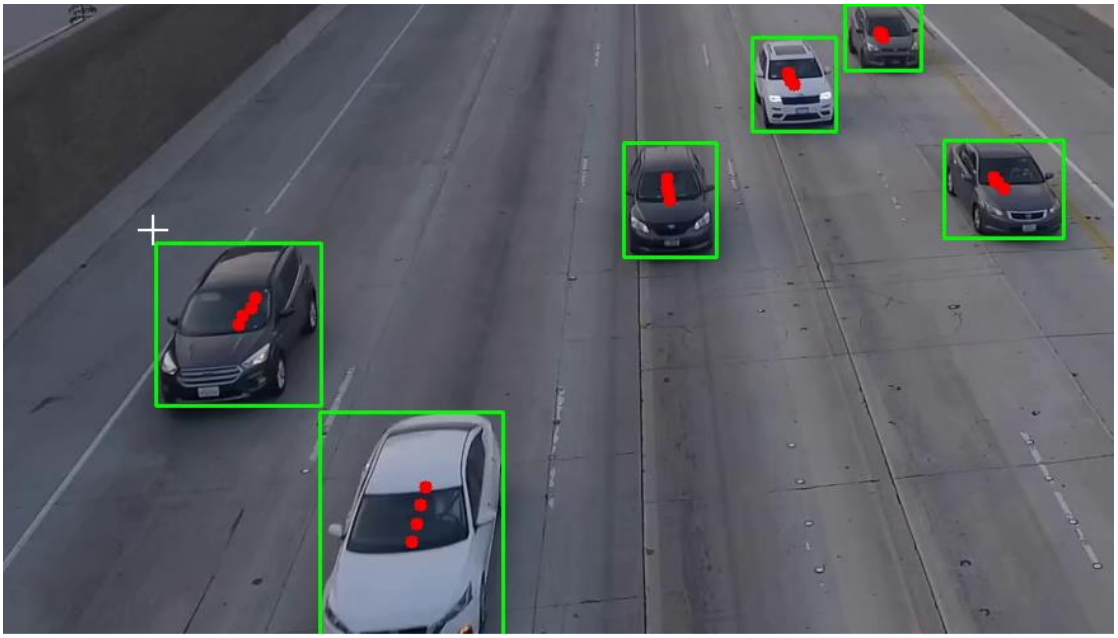

Figura 14. Seguimiento de vehículos dibujando en cada frame su centro para conocer la dirección de su movimiento [27].

Desde la figura 15 hasta la figura 18 se aprecia una secuencia de *frames* que representan cómo se realiza el seguimiento a determinados vehículos, dentro de una región de interés (ROI, por sus siglas en inglés) donde con la ayuda de asignarles un identificador se podrán reconocer y distinguir a través de la progresión de los fotogramas en un vídeo.

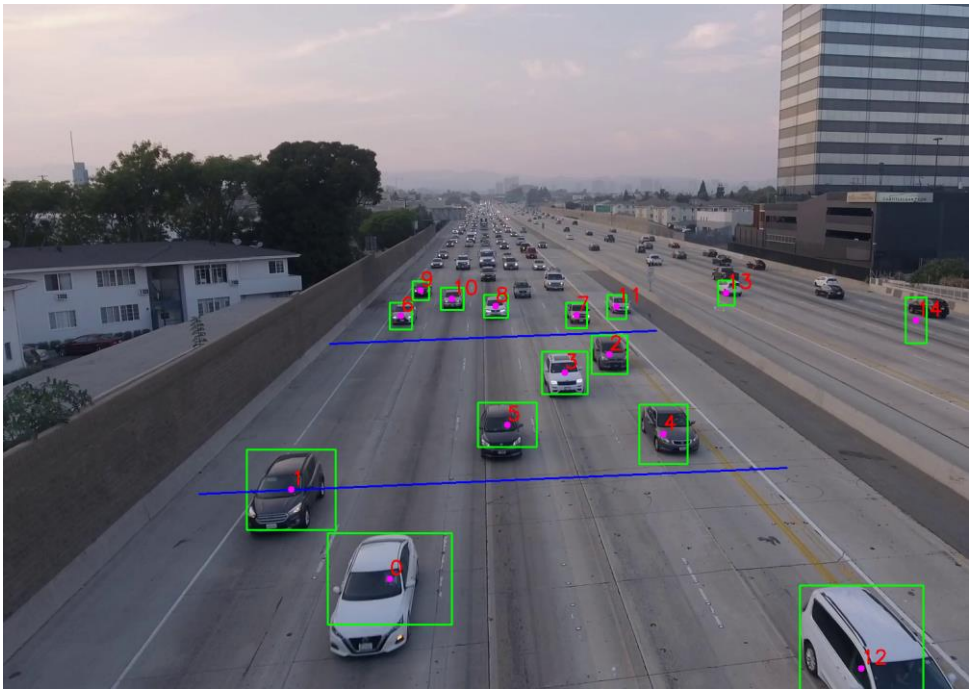

Figura 15. Frame 0 que etiqueta los distintos vehículos de un frame.

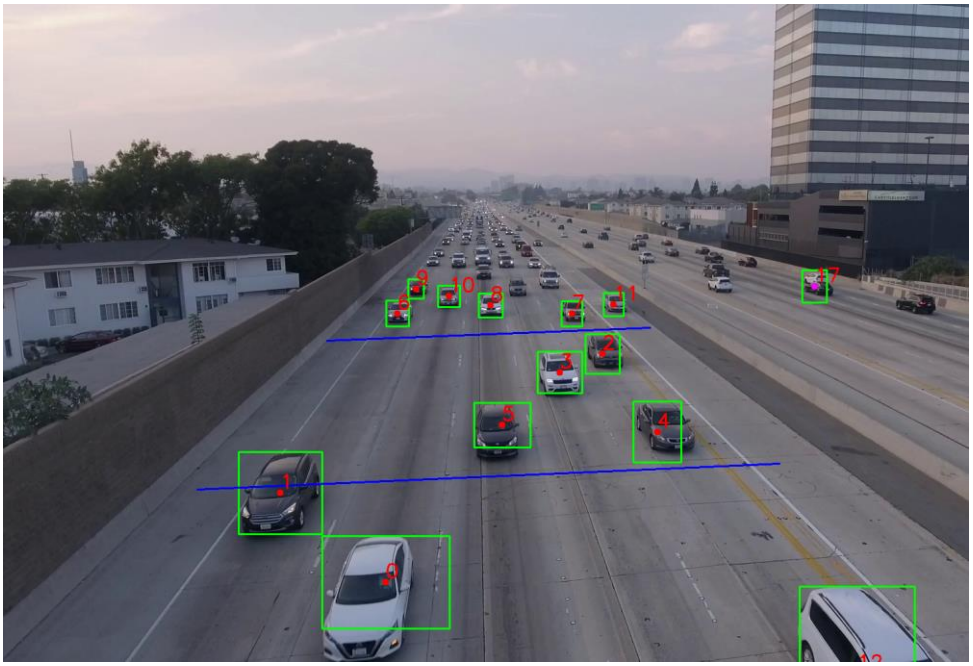

Figura 16. Frame 1 que muestra el seguimiento de los objetos y su correspondiente identificador

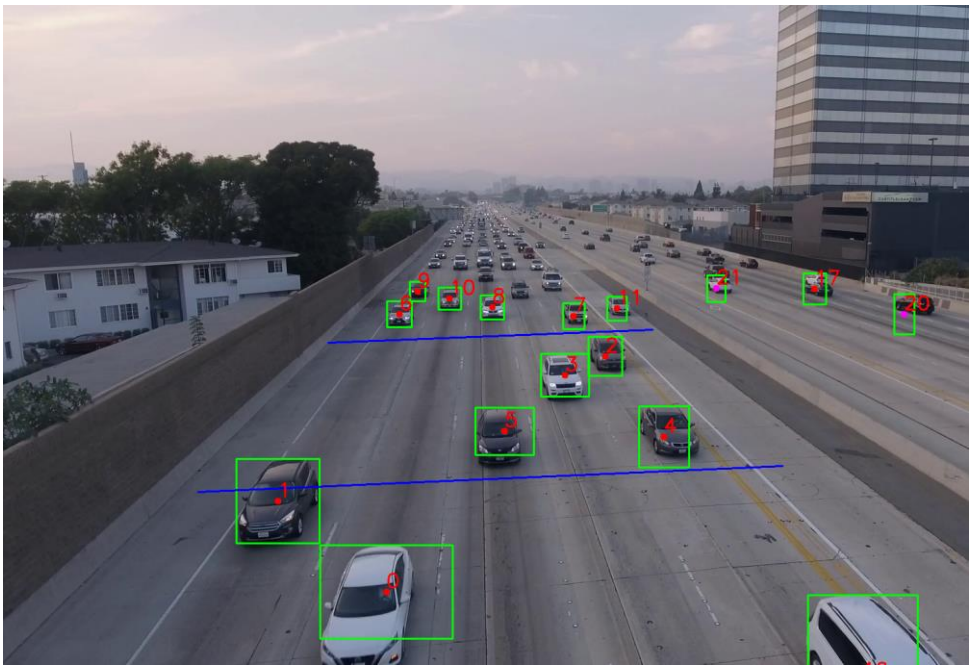

Figura 17. Frame 2 que muestra el seguimiento de los objetos y su correspondiente identificador

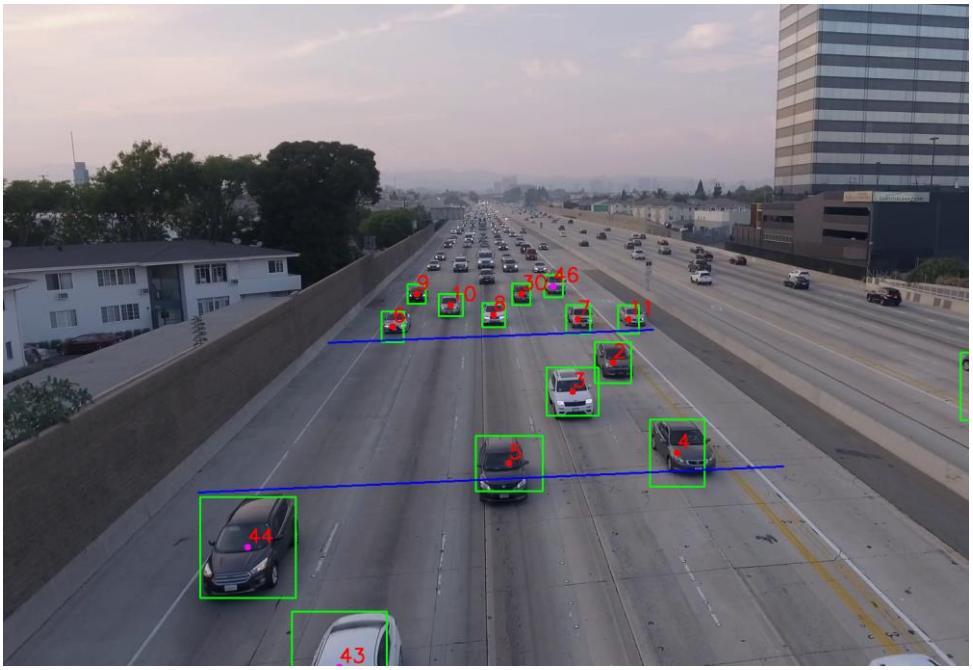

Figura 18. Frame 8 que muestra cómo se mantiene el identificador en nuestra ROI de los distintos vehículos que circulan por él.

El seguimiento de objetos puede ser realizado mediante diferentes métodos, como el seguimiento basado en características, el seguimiento de contornos o el seguimiento basado en

la correlación.

Existen diversas herramientas y bibliotecas de software que se utilizan para realizar el seguimiento de objetos en secuencias de video. Algunas de las más populares son:

- **OpenCV:** es una librería de visión artificial de código de uso libre que proporciona una amplia variedad de recursos para el procesado de imágenes y el seguimiento de objetos.

- **DeepSORT:** es un sistema de seguimiento de objetos basado en aprendizaje profundo que utiliza un enfoque basado en detección y seguimiento. Es uno de los algoritmos más precisos en el campo del seguimiento de objetos y ha sido utilizado en diversas aplicaciones de seguimiento de objetos en tiempo real.

- **Simple Online and Realtime Tracking (SORT):** es un algoritmo de seguimiento de objetos en tiempo real que utiliza una técnica basada en asociación de datos para realizar un seguimiento preciso y confiable de objetos en secuencias de video. Es muy rápido y fácil de implementar y ha sido utilizado en diversas aplicaciones de seguimiento de objetos.

- **Kalman Filter: e**s una técnica de seguimiento de objetos que utiliza un modelo de predicción basado en el movimiento del objeto y un modelo de observación basado en las mediciones de los sensores. Este filtro se compone de dos etapas: la etapa de predicción y la etapa de corrección.

    - La etapa de predicción se utiliza para predecir el estado del sistema en el siguiente paso de tiempo y se basa en el estado estimado en el tiempo anterior y el modelo dinámico del sistema.

    - La etapa de corrección se utiliza para actualizar la estimación del estado del sistema en función de las mediciones observadas.

  Es decir, el filtro de Kalman se utiliza para estimar el estado de un sistema dinámico a partir de mediciones incompletas y ruidosas, utilizando un modelo del sistema y un

modelo del ruido utilizando consigo matrices y vectores para representar el estado y las mediciones del sistema.

- **Minimum Output Sum of Squared Error (MOSSE):** es un método de seguimiento de objetos basado en el análisis de correlación que utiliza una red de filtrado adaptativo para realizar el seguimiento de objetos en tiempo real.

- **Kernelized Correlation Filter (KCF):** es un enfoque de seguimiento de objetos basado en el análisis de correlación que utiliza características de imagen y filtros de correlación para realizar el seguimiento de objetos con alta precisión.

Estas son solo algunas de las herramientas y bibliotecas de software que se utilizan para realizar el seguimiento de objetos en secuencias de video. La elección de la herramienta adecuada dependerá de las necesidades específicas de la aplicación y del nivel de precisión y velocidad requerido.

2.5.1 SORT

Como se ha mencionado antes, uno de los principales algoritmos de seguimiento de objetos es SORT [30]. Se decidió desde un primer momento usar este algoritmo por su sencillez de implementación en el proyecto.

SORT fue desarrollado y promovido por el usuario de GitHub *"abewley"*. Se puede encontrar el algoritmo y una guía de su uso en su repositorio, para poder implementarlo en proyectos de *object tracking.* Tal y como tiene escrito dicho usuario en su repositorio: *"SORT es una implementación básica de un marco de seguimiento visual de múltiples objetos basado en técnicas rudimentarias de asociación de datos y estimación de estados. Está diseñado para aplicaciones de seguimiento en línea donde solo se dispone de fotogramas pasados y actuales, y el método produce identidades de objetos sobre la marcha. Si bien este rastreador minimalista no maneja la oclusión ni objetos que vuelven a entrar, su propósito es servir como punto de referencia y plataforma de pruebas para el desarrollo de rastreadores futuros."*

Desde su publicación inicial, SORT fue clasificado como el mejor rastreador de objetos

múltiples de código abierto en el banco de pruebas MOT (*multiple object tracking*). Asimismo, ha sido utilizado en diversos campos de estudios como en seguimiento de humanos bajo el agua [31], o seguimiento de objetos aéreos en [32].

Más adelante, en el punto 3.1.3, se destacará el conjunto de modificaciones que se aplicaron sobre el algoritmo inicial, ya que su principal funcionalidad se centraba en resolver problemas MOT. Sin embargo, el objetivo del proyecto era de resolver un problema real de MOT + MOC (*multiple object counting*) por lo que se realizó un exhaustivo trabajo de entendimiento del código y modificaciones para que se ajustara a la meta propuesta.

Principalmente, hace uso únicamente de una combinación rudimentaria de técnicas familiares como el Filtro de Kalman y el algoritmo Húngaro para los componentes de seguimiento y este enfoque logra una precisión comparable a la de los rastreadores en línea de última generación.

Debido a la simplicidad de este método de seguimiento, el rastreador es más de 20 veces más rápido que otros rastreadores de última generación como se muestra en la figura 19.

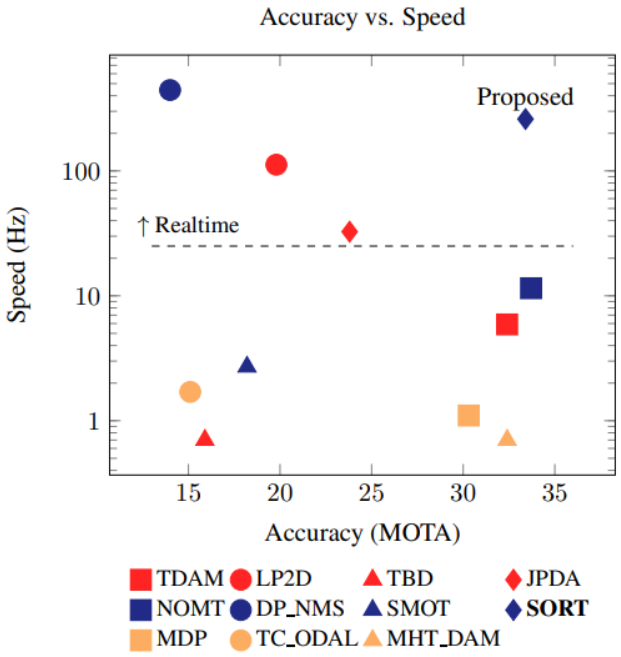

Figura 19. Comparación en términos de velocidad de procesado y precisión del algoritmo propuesto con trackers del estado del arte [33].

2.6 Trabajo relacionado

Entre las diversas técnicas empleadas para resolver el problema que se afronta en el presente proyecto se encuentra el estudio [34]. Donde los autores proponen un método de clasificación de vehículos por medio del procesamiento de imágenes, recreando así el *background* de los *frames* de un video a partir de una cantidad de *frames* del propio.

Para poder extraer los objetos en movimiento de una imagen, a los distintos frames le sustraen este *background* recreado para obtener los objetos estacionarios, para la compresión de la imagen utilizan la transformada wavelet y luego aplican NeuralSIM que permite probar rápidamente redes neuronales sin tener que escribir código para neuronas individuales.

La investigación propuesta en [35], Saeid Fazli et al. demuestra un algoritmo basado en dos partes: primero extraen el *background* y después encuentran los vehículos que circulan cerca de la cámara, pudiendo extraer características especiales (área del objeto, altura, anchura, cantidad de píxeles que ocupa en el frame…) las cuales serán la entrada de una red neuronal perceptrón multicapa con dos capas ocultas que devolverán a su salida el tipo de vehículo.

Se puede ver cómo en el artículo [36], los autores proponen un novedoso algoritmo el cuál lo comparan con el *back-propagation* estándar y el DSM (*direct solution method*) estándar. El novedoso método, DSM-AN, logra un 62% de efectividad en el set de datos de test y un 100% en el entrenamiento, pero aun así obtiene una buena generalización. La extracción de características se realiza en un primer preprocesamiento de imágenes con HIPS [37] el cuál recoge una colección de características independientes de la escala del vehículo particular.

En línea con estas técnicas de procesamiento de imágenes, la investigación realizada en [38], utiliza sustracción del *background* y la segmentación de la imagen con transformaciones morfológicas tales como erosión y, seguidamente, dilatación de los *frames* para así poder seguir y contar vehículos. Es empleado junto a un *background* adaptativo que le hace aprender de manera más rápida y eficiente, así como su adaptación a cambios en el entorno.

Otras técnicas como ViBe [39] y SURF [40] son utilizas en trabajos como [41], donde se usará un vehículo aéreo no tripulado (UAV, en inglés) para mejorar la vigilancia y el seguimiento de

vehículos en una determinada área. Se tendrá en cuenta que el fondo puede ser estático o presentar objetos en movimiento y para ello se utilizarán las técnicas mencionadas para extraer características estáticas y en movimiento del fondo. Se desarrollará un módulo de gestión de vehículos múltiples utilizando la tecnología *multi-threading* para permitir el seguimiento y el conteo de vehículos que crucen una LOI (línea de interés o *line of interest* en inglés) virtual en ambas direcciones.

También se puede encontrar arquitecturas como en la publicación [42], donde los autores desarrollan una red neuronal simple para la clasificación de vehículos en imágenes de baja resolución.
La precisión del modelo propuesto (92.9%) es algo inferior a la precisión de la arquitectura VGG-16 (96%) y VGG-16 *fine-tuned* (99.2%), aunque es suficiente ya que le supera con creces en términos de eficiencia y rapidez debido a que solo consta de dos capas convolucionales con 16 filtros de 5x5 cada una y ReLU como función de activación, seguidas de cuatro capas MaxPool con filtros de 2x2 y un *stride*, el cual se refiere a los saltos que dará el filtro sobre la imagen de entrada, de valor 2 para la extracción de características.

En el ensayo presentado en [43], se aborda el problema de la clasificación de vehículos desde una perspectiva amplia y se revisan las diferentes técnicas disponibles para evaluar su utilidad en la resolución de este problema crítico. Comparan diferentes técnicas, incluyendo las redes neuronales (NN) para, finalmente, destacar el modelo Hybrid Dynamic Bayesina Network (HDBN) gracias a su capacidad para estimar características simples de los vehículos a través de vídeos. Es importante destacar que este estudio se realizó antes del auge del *deep learning* y las CNN, lo que sugiere que podrían existir nuevas técnicas que superen al HDBN en la actualidad.

Técnicas como las que presenta [44] para la clasificación y conteo de vehículos gracias al sistema *Gaussian Mixture Model* (GMM) para sustraer el *background* y luego clasifica los vehículos comparando el área con los valores supuestos. La clasificación se basa en tres métodos: comparación del contorno, bolsa de características (BoF) y SVM. El modelo propuesto se divide en tres módulos: aprendizaje del fondo, extracción del primer plano y clasificación.

A medida que iban avanzando las técnicas iban apareciendo nuevos modelos y metodologías para facilitar esta tarea como la innovadora CNN *Multi-Level* diseñada en [45] donde, con 4 puntos de vista diferentes para obtener una mayor cantidad de datos, aplicaba una técnica de

sustracción de fondo y un seguimiento de múltiples vehículos para evitar contarlos dos veces.
El objetivo del proyecto es lanzar un *framework* de red neuronal convolucional multinivel para estimar el tráfico en configuraciones más complejas, como escenarios con iluminación variada y montajes complicados.
El *framework* propuesto consta de cinco módulos: preprocesamiento de imágenes, detección de objetos, seguimiento, clasificación de objetos y cuantificación. Se comparará la precisión del modelo entre las técnicas de detección de objetos YOLO [46] y SSD [47], para determinar la eficacia del enfoque propuesto siendo YOLO la mejor opción entre ellas.

Otro proyecto que refuerza el uso de YOLO en trabajos anteriores es [48], en el que tratan de resolver el problema de conteo y clasificación de vehículos en países en vía de desarrollo donde las carreteras carecen de carriles y los vehículos tienen una gran homogeneidad. Para resolver este problema, se utiliza una red CNN simple del modelo YOLO entrenada en MS-COCO y ajustada posteriormente con los conjuntos de datos de PASCAL VOC [49] y KITTI [50]. Después de entrenar la red, se comparan las vistas utilizadas con los modelos de YOLO 1-5 para determinar cuál obtiene el mejor MAP (*mean average precission*).
El objetivo final es lograr la clasificación en tiempo real de los vehículos, aunque el conteo de estos se realizará de forma manual a partir de las imágenes recogidas en los vídeos para tener así un modelo comparativo *ground-truth*.

Los autores de [51], gracias a la implementación de múltiples puntos de vista (6) y un novedoso *framework, OverFeat*, obtienen una precisión de 96.55%. En el trabajo se combina una CNN para la extracción de características y un clasificador de aprendizaje automático como Regresión Logística porque este requiere menos datos y tiempo que la CNN para la etapa de entrenamiento. Además, utilizan el método de sustracción de fondo (BSM), que utiliza el modelo de mezcla de GMM para extraer el fondo de las imágenes. Para medir la efectividad del framework comparan con la página *Placemeter*, y el conteo manual de un alumno, obteniendo unos resultados satisfactorios.

De acuerdo con el novedoso desarrollo presentado en [52], el enfoque de aprendizaje activo (ALVeRT) permitirá mejorar la eficacia del sistema de reconocimiento y seguimiento de vehículos, lo que permitirá detectar situaciones de riesgo y amenazas en tiempo real.
Se divide en dos fases donde la primera fase es similar al aprendizaje pasivo, donde se utiliza un conjunto de imágenes positivas para entrenar el sistema. En la segunda fase, se utiliza una

función de consulta para extraer ejemplos sin etiquetar, y mediante un mecanismo humano de *ground-truth* se asignan las etiquetas correspondientes a los nuevos ejemplos.
A continuación, se reentrena el clasificador con los nuevos ejemplos etiquetados. Para la clasificación hará uso de Adaboost lo que le permitirá ser eficiente en tiempo real.

Una inexplorada manera de usar YOLO se presenta en [53], es una combinación con 2 pares redes neuronales siamesas. El primer par se encarga de extraer el modelo, color y tipo de vehículo y de extraer la información de la matricula. Se usan redes siamesas porque son una simple y eficiente manera de resolver problemas de coincidencia. Las características profundas se extrajeron de VGG-16 donde redujeron su número de capas convolucionales para que no haga tanto esfuerzo computacional. La combinación siamesa que se encarga de las similitudes de las matrículas se basa en OCR que es una arquitectura CR-NET donde las 11 primeras capas convolucionales son las mismas que en YOLO y añade 4 capas más para mejorar la no linealidad.

3 Diseño e implementación

El objetivo principal del proyecto es desarrollar un programa capaz de identificar distintos tipos de vehículos en un tramo de carretera, clasificarlos y poder contarlos. Una vez contados, realizar el cálculo de la medida IMD de una carretera con los datos arrojados por dicho programa.

Para ello, se ha utilizado la plataforma de código abierto Anaconda. Anaconda es una plataforma de ciencia de datos y distribución de software libre y de código abierto que simplifica la gestión e implementación de paquetes y entornos de trabajo en Python.

Es por ello por lo que se creó un entorno virtual donde se instalaron todos los paquetes de software necesarios para trabajo en el campo de la visión y se han ido realizando ahí las distintas pruebas con imágenes y videos de test.

3.1 Desarrollo del proyecto

Al comienzo del proyecto se estudiaron diferentes métodos de identificación y clasificación de objetos como los que se comentaron en el punto 2.4 del mismo. Tras estas iteraciones, se acabó utilizando una versión no oficial del algoritmo de detección de objetos en imágenes y video, YOLO.

En la primera fase, se trató de implementar el algoritmo de YOLO en una imagen. Una vez implementado, fue necesario cerciorarse de que tenía un alto porcentaje de acierto en detección de los objetos de nuestro interés. Posteriormente, se procedió a la segunda fase la cual trata implementar el algoritmo en un video de internet el cual fue de vital importancia en la evaluación de la calidad del algoritmo en los vídeos de test.

Al conseguir que el script generado captara los objetos de interés en los distintos fotogramas del video, se usó un algoritmo de tracking para añadir un identificador a cada vehículo y así poder seguirlo a lo largo de los fotogramas.

El algoritmo en cuestión fue obtenido de una fuente en línea y lograba asignar un identificador a cada vehículo y funcionaba correctamente si el objetivo hubiera sido contar el número de vehículos que circula por una carretera. Al ser diferente el propósito, en la tercera fase el alumno tuvo que realizar modificaciones en el código utilizado para adaptarlo a sus necesidades.

Dichas modificaciones conseguirían resolver un problema de *multi-object counting*, es decir, ahora el algoritmo sería capaz de añadir un identificador y la clase del vehículo, mostrando así por pantalla qué tipo de vehículo está circulando por la carretera y su identificador como se puede apreciar en la figura 20.

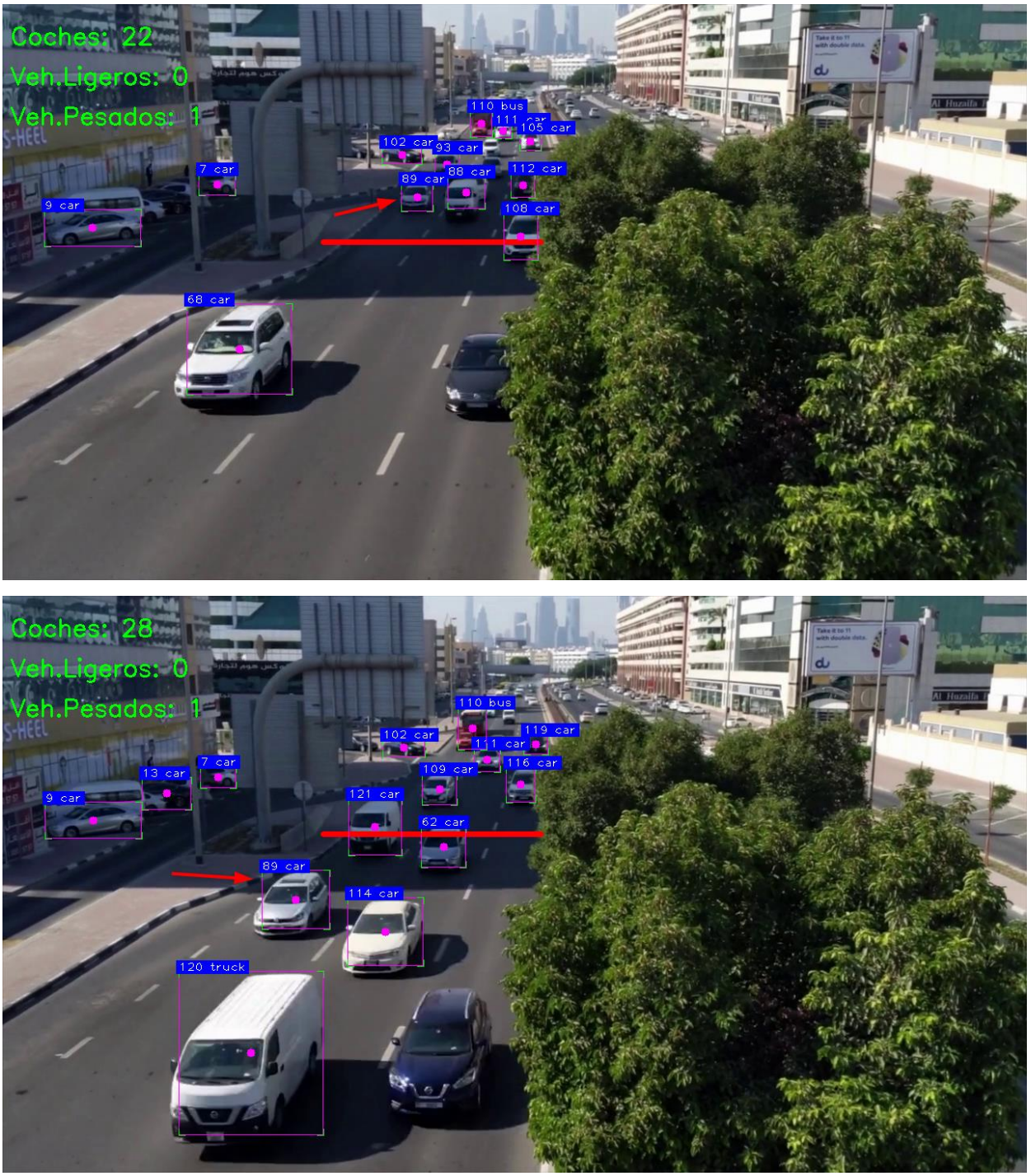

Figura 20. Identificador asignado en la primera vez que aparece el vehículo (89) y se mantiene en los frames.

En la cuarta fase se grabó un video en una carretera que recogiera las características necesarias para que el algoritmo funcionara y devolviera resultados coherentes.

Y, por último, en la fase final, se desarrolló un script que fuera capaz de tratar los resultados del conteo de vehículos creando un archivo CSV que recoge el número de vehículos referentes a cada tipo que habían circulado en un periodo de tiempo concreto e imprimiera por pantalla la efectividad del modelo, comparando el número de vehículos que han sido contados por el algoritmo y el número de vehículos contados de manera manual.

A continuación, se explicará detalladamente y por separado como se ha llevado a cabo cada una de las fases del proyecto.

3.1.1 Detección de objetos en imágenes

Para la detección de objetos en imágenes se pensó en utilizar una red neuronal propia, creada por el alumno que fuera capaz de clasificar objetos a partir de que fuera entrenada con bases de datos de vehículos que se muestran a continuación en la figura 21.

Figura 21. Tipos de vehículos en los que se centra el algoritmo.

Una vez se logró, surgió el problema de combinar capas convolucionales con capas de regresión para tratar los valores de la caja delimitadora que sitúan y determinan la posición de un vehículo en una imagen.

Es por esto por lo que para una primera detección de objetos en imágenes se utilizó YOLOv8, la cual es una versión mejorada de la última versión oficial YOLOv5. Esta versión combina varias técnicas de detección y seguimiento de objetos para mejorar la precisión y la velocidad, como muestra la figura 22.

En ella se puede apreciar cómo se han identificado los coches con una caja delimitadora de color naranja y los camiones con una de color verde. Además de mostrar la clase, se muestra la fiabilidad del programa de que ese objeto detectado sea dicha clase.

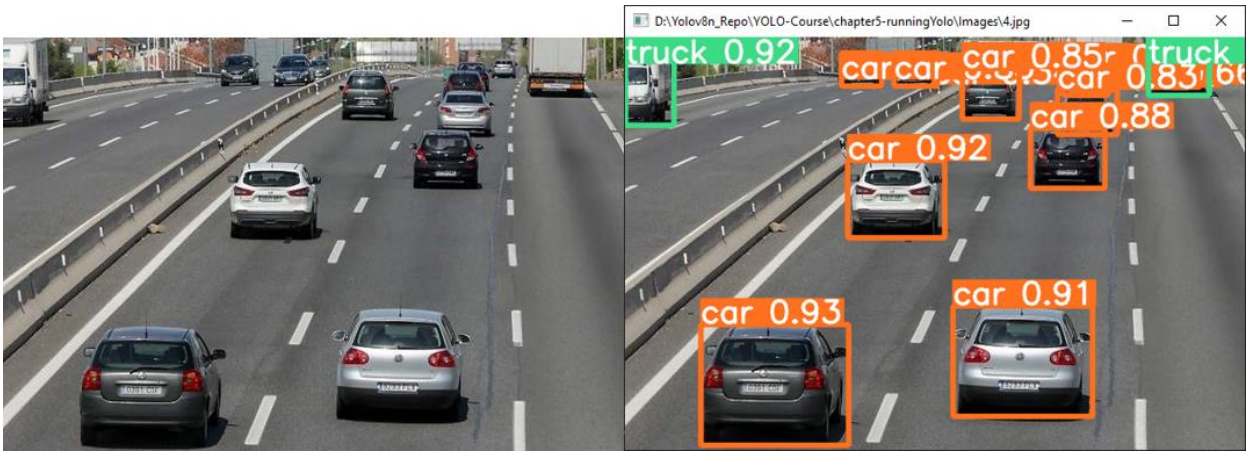

Figura 22 Coches circulando y la detección de ellos sobre la misma imagen.

Una de las principales mejoras de YOLOv8 es la inclusión de la técnica de detección de objetos *Panoptic Feature Pyramid Networks* (PFPN), que permite al modelo detectar objetos a diferentes escalas y niveles de detalle.

YOLOv8 utiliza la red neuronal convolucional Darknet para realizar la detección de objetos y cuenta con diferentes tamaños de modelos pre-entrenados (n, s, m, l, x) para adaptarse a diferentes requisitos de precisión y velocidad.

Para que el algoritmo YOLOv8 lograse funcionar, fue necesario crear un entorno virtual en Anaconda, con una versión especifica de Python (3.9) ya que los paquetes de software que se utilizaron funcionan con esta versión en concreto.

Es por ello por lo que, dentro de la carpeta, existe un archivo de texto que contiene librerías tales como *cvzone*, que nos permite dibujar cajas dentro de una imagen, la versión de *Ultralytics* que contiene YOLOv8, *matplotlib* y *numpy* para operaciones matriciales y de tensores, *OpenCV* que sirve para poder visualizar videos en Python, y más.

En estos casos, se hace uso de una popular métrica, IoU (*intersection of union*) para medir cómo de bien se superponen dos regiones de objetos, es decir, si el IoU es cercano a 1, significa que las regiones se superponen casi por completo, lo que indica una alta precisión en el seguimiento. Por otro lado, si el IoU es cercano a 0, significa que las regiones no se superponen en absoluto, lo que indica una baja precisión en el seguimiento.

3.1.2 Detección de objetos en video

En este apartado, se hizo uso de un vídeo de prueba que se aportaba en un curso realizado por el alumno en Internet [54]. Este se ponía a su disposición en un repositorio de GitHub y, después, se procedió a detectar los vehículos en los distintos frames del propio vídeo.

En este momento surgió la necesidad de crear un script donde fuéramos a desarrollar el código principal. El desempeño de dicho script sería el de iterar sobre los fotogramas del video con un bucle *for* para poder realizar exactamente el mismo trabajo que en la detección de imágenes.

El programa es capaz de detectar en cada fotograma los distintos objetos e indicar la posición exacta del objeto en cada fotograma tal y como se puede ver en las figuras 23 y 24. Pudiendo así comprobar que el objeto con una clase cualquiera ha sido detectado con precisión.

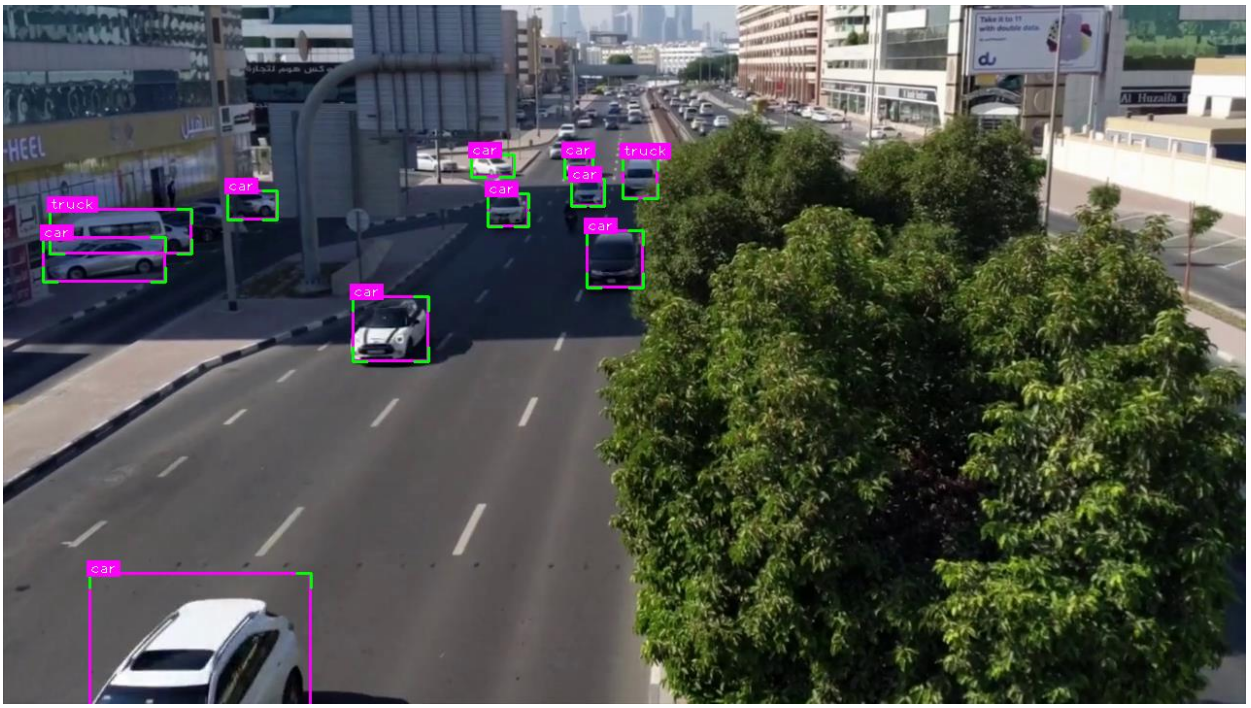

Figura 23. Fotograma 10 donde están todos los objetos localizados

```
frame 10
bbox:(638, 155, 674, 199)
bbox:(48, 210, 194, 256)
bbox:(578, 154, 607, 178)
bbox:(231, 191, 282, 221)
bbox:(482, 154, 526, 177)
bbox:(585, 179, 619, 207)
bbox:(601, 232, 659, 291)
bbox:(499, 194, 541, 228)
bbox:(41, 239, 167, 285)
bbox:(360, 300, 438, 367)
bbox:(89, 585, 317, 748)
```

Figura 24. Valores de las cajas delimitadoras de cada objeto del localizado del fotograma 10.

Al comprobar la exactitud al localizar la posición exacta de los objetos en cada frame, se decidió dibujar dentro de la caja delimitadora un círculo que representaría el centro de cada objeto.

Dicho círculo permitiría seguir la trayectoria del objeto y asegurar que se había detectado en todos los fotogramas del video ya que posteriormente se dibujó una LOI la cual sirve para demostrar que el centro de un objeto ha cruzado por ese tramo.

En la figura 25 y 26 se muestran unos fotogramas del vídeo en los cuales se puede apreciar como los objetos tienen un centro dibujado y como al cruzar ese centro por la LOI, cambia de color indicando que se ha contabilizado el vehículo. Para ello, se eligió una zona de la carretera grabada donde se obtuviera una mayor exactitud en la detección de los vehículos que circulan.

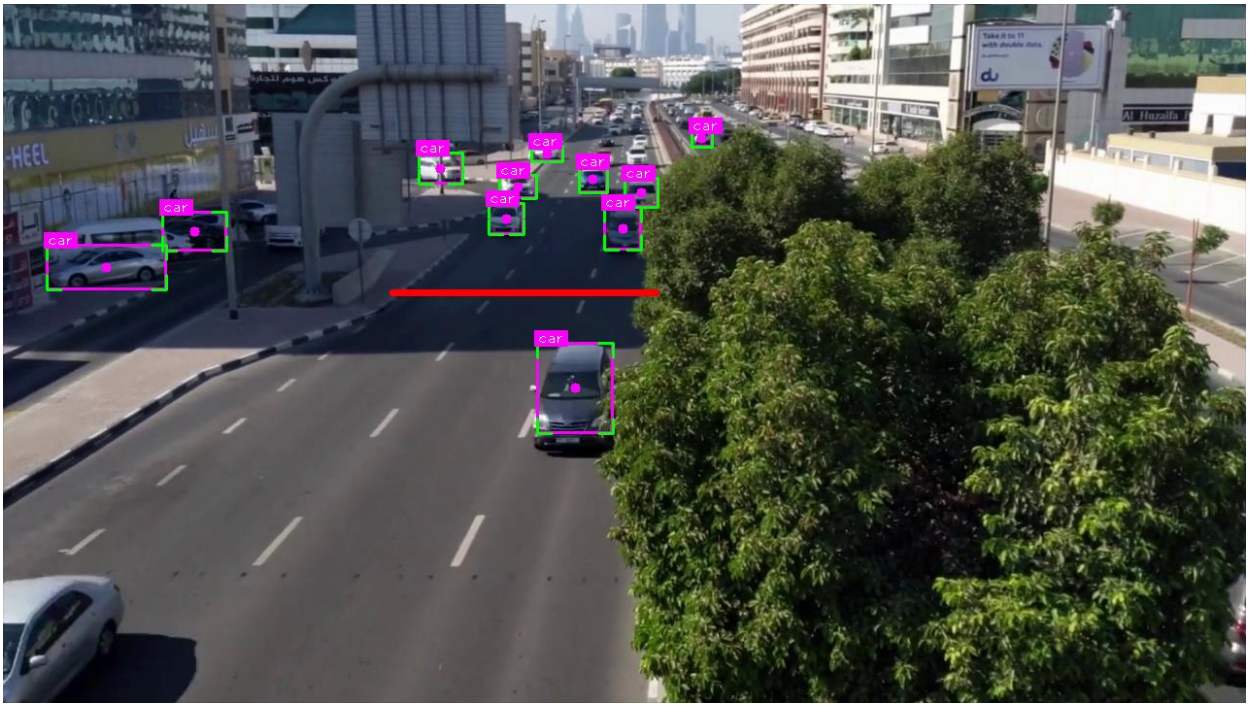

Figura 25. Vehículos detectados con el centro en su cada delimitadora.

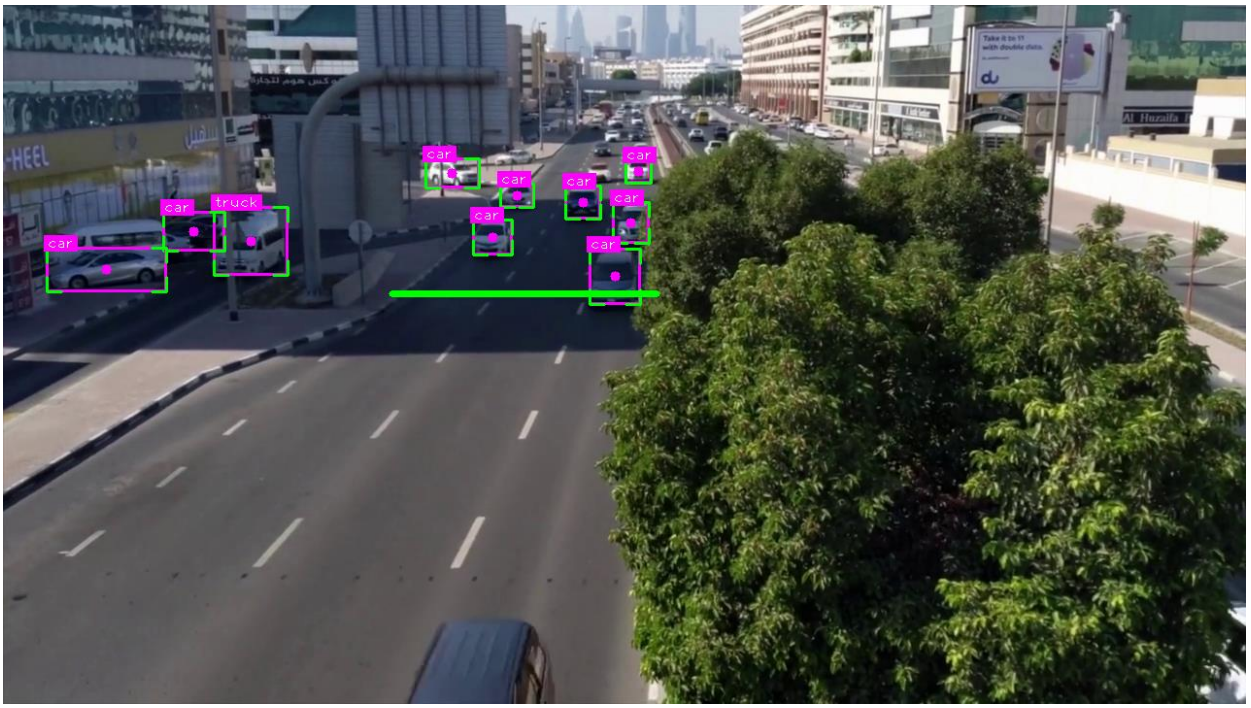

Figura 26. LOI cambiando de color al circular el centro de un vehículo dentro del rango de aceptación.

3.1.3 Modificación algoritmo SORT

El desarrollo del alumno parte de un famoso algoritmo de tracking llamado SORT al que ya se ha hecho referencia en este documento. Tal algoritmo es capaz de asignar un identificador a medida que se van detectando vehículos en los distintos fotogramas gracias al filtro de Kalman, el cual se explica en el punto 2.5.

Además, tiene en cuenta adversidades tales como que un objeto deje de visualizarse en un numero definido de frames, que tenga un porcentaje bajo de *intersection of union*, baja fiabilidad a la hora detectar si es una clase o no, etc.

El algoritmo de SORT está desarrollado con programación orientada a objetos (POO), por lo que, para llamarlo en el script del alumno de detección de objetos, se debe importar y pasarle un argumento el cual es un array que contiene en sus primeras 4 posiciones los valores de las cajas delimitadoras de cada objeto y, además, la fiabilidad que tiene el modelo al detectar un objeto.

Después de aplicar el filtro de Kalman con sus respectivas máscaras, SORT devolvía como resultado la estimación de la caja delimitadora actual y el identificador asignado. Al suceder esto, el filtro de Kalman eliminaba el valor de fiabilidad de la detección por lo que fue ahí donde se identificó la raíz del problema.

Después de trabajar en la comprensión del código, se consiguió que, además de devolver los valores estimados de las cajas delimitadoras y su id, devolviera la clase.

El siguiente problema con el que se encontró el alumno fue que el algoritmo ahora asignaba la misma clase a todos los objetos. Se continuó realizando ciertas modificaciones en el array de detecciones que se pasaba a la clase SORT, modificaciones en las máscaras del filtro de Kalman y en las variables creadas dentro de la clase dedicada a dicho filtro.

Finalmente, se percibió que el algoritmo, para realizar las estimaciones, primero hacía un cambia de formato a los valores de la caja delimitadora, la cual la pasaba del formato $[x_1, y_1, x_2, y_2]$ a $[x, y, s, r]$ (siendo 'r' el *aspect ratio* y 's' el área) y luego a la inversa para recuperar los valores de la *bounding box* y devolver la predicción. Era en este punto donde se "perdía" la clase del objeto.

En definitiva, se decidió crear una variable *'self.class'* dentro de la clase del filtro de Kalman que almacenara la clase del array de detecciones para que, al hacer la conversión de los ejes, se perdiera, pero luego antes de añadir esta variable al array definitivo se le pudiera añadir la clase correspondiente que previamente se había almacenado.

Para terminar, en el script generado por el alumno se creó un bucle *for* que itera sobre los resultados que arroja el nuevo algoritmo de SORT y tratar por separado los valores del array resultado. Con ello se podrá dibujar en cada caja delimitadora de cada objeto el identificador y la clase correspondiente a dicho objeto.

Ahora, al tener cada objeto definido con su clase en específico, se podrán contar los objetos por separado cuando cruzan la LOI creada en la fase anterior y así almacenar ese resultado en variables diferentes en función si el vehículo pertenece a un automóvil, un vehículo pesado o un vehículo ligero.

3.1.4 Grabación del vídeo en una carretera española

Para esta fase se ha utilizado un vídeo representativo grabado por el alumno en un puente para poder tener cierta altura y así evitar oclusiones entre vehículos.

Con el objetivo de complementar el video de Internet proveniente de la India, se busca obtener un video de una carretera española donde los vehículos detectados sean modelos de vehículos que se utilizan en el día a día en España. La intención es tener un video que refleje la realidad y las características de las carreteras españolas, con vehículos representativos de los que se encuentran en circulación.

El alumno se personificó en el lugar de interés, en la carretera A-2 E-90 Autovía del nordeste, pasado el kilómetro 12, figura 27.

Figura 27. Imagen de la carretera donde está situado el puente en el que fue grabado el vídeo. Punto 1 donde se grabó el vídeo y punto 2 edificio al que afecta acústicamente la carretera.

Se eligió este lugar por la altura comentada, por tener alto tránsito de tráfico ya que se pretende exponer el algoritmo a una alta afluencia de vehículos y que tenga edificios en las partes laterales.

No se consiguió encontrar una carretera de estas características con un punto elevado que tuviera edificios de viviendas colindando con la carretera. Aunque la posición que se ha utilizado cuenta con oficinas de trabajo algo más apartadas que se identificaran como viviendas ya que en cuanto a valores acústicos permitidos, es distinto el trato con oficinas al trato con viviendas.

El RD 1367/2007, en mediciones en el exterior, muestra los valores que no pueden ser superados, diferenciando entre suelo industrial o residencial, tal y como se da a conocer en la figura 28.

Tipo de área acústica		Índices de ruido		
		L_d	L_e	L_n
e	Sectores del territorio con predominio de suelo de uso sanitario, docente y cultural que requiera una especial protección contra la contaminación acústica	55	55	45
a	Sectores del territorio con predominio de suelo de uso residencial.	60	60	50
d	Sectores del territorio con predominio de suelo de uso terciario distinto del contemplado en c.	65	65	55
c	Sectores del territorio con predominio de suelo de uso recreativo y de espectáculos.	68	68	58
b	Sectores del territorio con predominio de suelo de uso industrial	70	70	60

Figura 28. Tabla de valores de inmisión de ruido aplicables a infraestructuras portuarias y a actividades del RD 1367/2007

En la figura 29, se puede contemplar la cercanía a las oficinas.

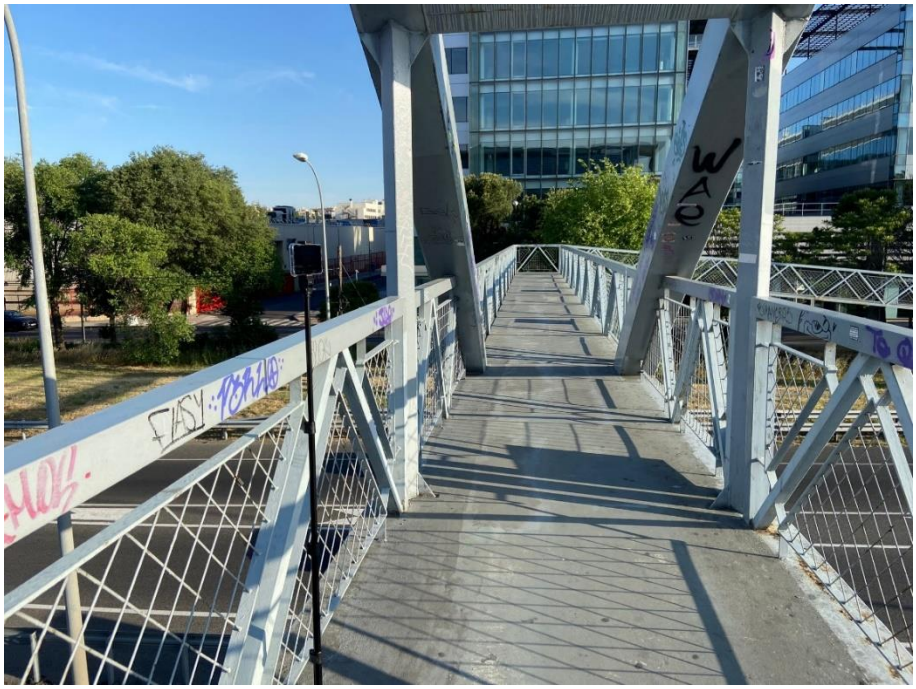

Figura 29. Distancia desde el punto de grabación a las oficinas.

3.1.5 Cálculo de efectividad del modelo

Una vez diseñado el modelo de detección de objetos, surgió la necesidad de definir una medida de efectividad que tenía el modelo sobre el número total y por clase de vehículos que circulaban por la vía de circulación. Para calcular la exactitud de cuántos han circulado, el alumno realiza un conteo manual a través del vídeo para tener un valor de referencia en el que se apoyará para calcular el porcentaje de acierto.

Junto al conteo manual se obtiene por medio del algoritmo el conteo realizado por la máquina. Una vez se hace uso de la matriz de confusión explicada en el apartado 4, extraemos los datos de dicha matriz y los introducimos en la siguiente ecuación (2). Estos datos serán los filtrados para evitar las casuísticas de error que puedan haberse producido durante el conteo automático de la máquina, como por ejemplo que un vehículo se cuente en una clase, aunque realmente pertenezca a otra distinta. Con ello, se calcula la efectividad total del modelo de clasificación multiclase.

$$Precisión\ modelo = \frac{a_{k,k}}{\sum_{i=1,j=k}^{N} a_{i,j}} \qquad (2)$$

- $a_{k,k}$ se refiere al valor k-sino de una fila y columna determinada de la matriz de confusión.
- $\sum_{i=1,j=k}^{N} a_{i,j}$, es la suma total de los valores correctamente clasificados y contados por el modelo.

Aplicamos la supervisión del alumno al modelo para poder comprobar qué coches ni identifica ni cuenta, los que identifica, pero no cuenta, los que cuenta bien, los que cuenta, pero con una clase errónea y después podremos definir cuál es el % de precisión de nuestro modelo.

4 Resultados

En este proyecto, se han utilizado los diferentes modelos pre-entrenados que ofrece YOLOv8 sobre un primer vídeo de test en la India y un vídeo grabado por el alumno en España. Se tiene esta diferenciación en cuenta ya que los vehículos que circulan por cada país no son exactamente iguales.

Se presenta una matriz de confusión, figura 31, con los resultados donde se tiene en cuenta una variedad de escenarios. Observa tanto los vehículos contados de manera global y por cada tipo de vehículo, como los vehículos que no se detectan (falsos negativos) y, por lo tanto, no se cuentan. Los vehículos que no han sido detectados en su categoría correctamente y contados para otra categoría diferente a la suya (falsos positivos).

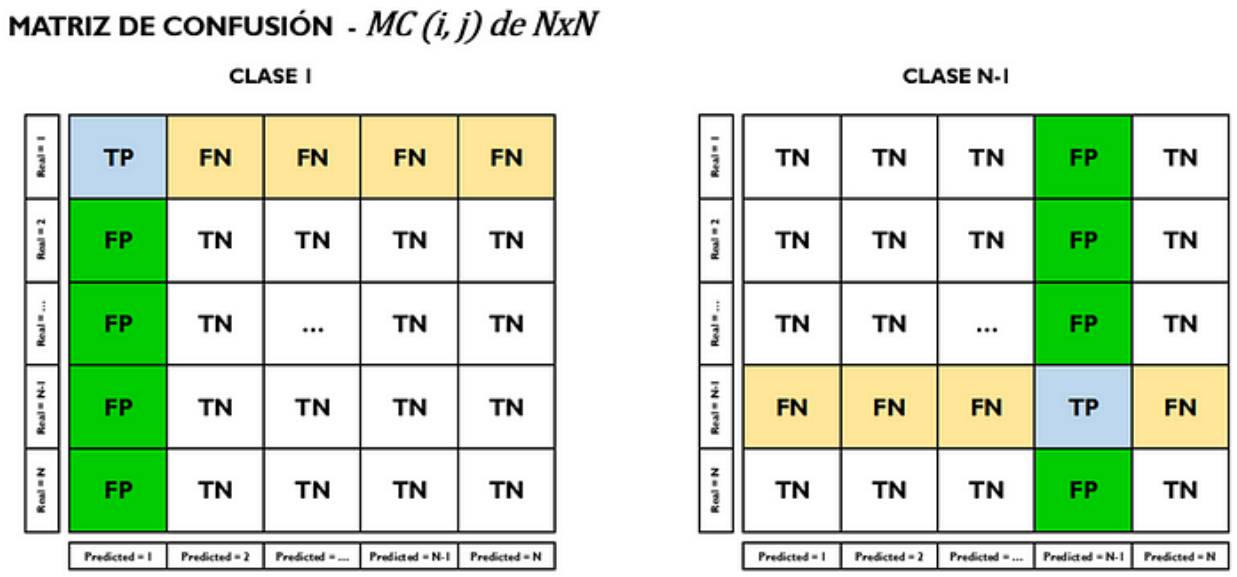

Figura. Matriz de confusión para el tratamiento de los resultados. [55]

Estos valores han sido comparados con el *ground-truth* realizado por el alumno, lo que quiere decir que se realizó un conteo manual de los vehículos que circulan por la carretera y de cada categoría para conocer el porcentaje de acierto de los modelos propuestos.

Para ello, a la hora de comparar los resultados, definimos una serie de abreviaturas que se usarán más adelante en la Tabla 2:

- GT_i donde $i = c, p, l$ indica la categoría de *ground-truth* donde i puede ser un turismo, un vehículo pesado o un vehículo ligero.
- N_i donde $i = c, p, l$ indica el valor total de vehículos donde i puede ser un turismo, un vehículo pesado o un vehículo ligero.

Tabla 2. Tabla genérica comparativa de los resultados del modelo y del alumno.

	Precisión coches	Precisión veh.pesados	Precisión veh.ligeros	Precisión modelo
Modelo N	GT_c / N_c	GT_p / N_p	GT_l / N_l	GT / N

Para conocer la precisión de nuestro modelo, se tendrá que comparar estos resultados con los genéricos obtenidos por el alumno con el sistema *ground-truth*. Dichos resultados estarán reflejados en la columna de **Total (realidad)** que se mostrará en la matriz de confusión de cada modelo. A su vez, en la fila de **Total (modelo)** se encontrarán los valores para el conteo que realiza nuestro modelo.

Para poder entender cada matriz de confusión, de color naranja se han coloreado las casillas donde se produce un error, de color azul son las casillas que ha acertado el modelo, y de color verde está representada la suma total de vehículos que ha contado el modelo.

4.1 Vídeo de test en India

En esta sección, se van a presentar los resultados obtenidos de los distintos pesos pre-entrenados de Yolov8 trabajando sobre el video obtenido desde Internet para el test del modelo y que ha sido grabado en la India.

De color azul se muestran los vehículos correctamente clasificados y contados y, después, de color verde está marcado el valor total de vehículos que ha contado nuestro modelo, haciendo la suma de todos los turismos, vehículos pesados y ligeros que ha contado.

4.1.1 Modelo N

Comparando los resultados obtenidos por el modelo, con los extraídos del método *"ground-truth"* del alumno, obtenemos unos valores de precisión como los que se muestran en la Tabla 3.

Tabla 3. Precisión del modelo N sobre cada clase y el total.

	Precisión coches (Pc)	Precisión veh.ligeros (P.l)	Precisión veh.pesados (P.p)	Precisión modelo (Pm)
Modelo N	71 / 71 = 100%	0 / 2 = 0%	1 / 2 = 50%	72 / 75 = 96%

En la tabla 4 se muestra, de manera más detallada, la precisión por cada tipo de vehículo en este modelo y la cantidad de vehículos mal categorizados por cada clase.

Tabla 4. Resultados del Modelo N sobre el vídeo de la India

Modelo / Realidad	Coches	Vehículos ligeros	Vehículos pesados	Total (Realidad)
Coches	71	0	0	71
Vehículos ligeros	0	0	0	2
Vehículos pesados	0	0	2 (-1)*	2
Total (modelo)	71	0	1	72 / 75**

***Nota:** Se cuentan dos pesados, pero es el mismo que se cuenta duplicado por lo que para no contaminar la precisión del modelo, se resta.

****Nota:** Como se puede apreciar en los resultados recogidos en la Tabla 4, circulan 2 vehículos pesados, pero solamente se cuenta uno, aunque se cuenta por duplicado, pero se reconoce el error de no detección de uno de ellos. En cuanto a turismos, todos los que circulan por la carretera han sido correctamente identificados y contados, sin embargo, de los vehículos ligeros, ninguno de los 2 se han identificado ni contado.

4.1.2 Modelo S

Comparando los resultados obtenidos por el modelo, con los resultados *"ground-truth"* del alumno, obtenemos unos resultados de precisión que se muestran en la Tabla 5:

Tabla 5. Precisión del modelo S sobre cada clase y el total.

	Precisión coches (Pc)	Precisión veh.ligeros (P.l)	Precisión veh.pesados (P.p)	Precisión modelo (Pm)
Modelo N	65 / 71 = 91.5%	0 / 2 = 0%	1 / 2 = 50%	66 / 75 = 88%

En la tabla 6 se muestra, de manera más detallada, la precisión por cada tipo de vehículo en este modelo y la cantidad de vehículos mal categorizados por cada clase.

Tabla 6. Resultados del Modelo S sobre el vídeo de la India

Modelo / Realidad	Coches	Vehículos ligeros	Vehículos pesados	Total (Realidad)
Coches	65	0	7(-1)*	71
Vehículos ligeros	0	0	0	2
Vehículos pesados	1	0	1	2
Total (modelo)	66	0	8	74 / 75**

***Nota:** Se cuentan 7 turismos como pesados, pero uno de ellos se cuenta doblemente por lo que, para no contaminar la precisión del modelo, se resta.

****Nota:** En los resultados recogidos en la Tabla 6, de los dos vehículos pesados que circulan, sólo cuenta 1 y los otros 7 son turismos o furgonetas que cuenta como vehículo pesado. De los vehículos ligeros, no cuenta ninguno y cuenta un vehículo pesado como turismo. Cabe destacar que, de los turismos que cuenta como pesados, uno lo cuenta doble.

4.1.3 Modelo M

Comparando los resultados obtenidos por el modelo, con los resultados *"ground-truth"* del alumno, obtenemos unos resultados de precisión que se muestran en la tabla 7:

Tabla 7. Precisión del modelo M sobre cada clase y el total.

	Precisión coches (Pc)	Precisión veh.ligeros (P.l)	Precisión veh.pesados (P.p)	Precisión modelo (Pm)
Modelo N	69 / 71 = 97.1%	0 / 2 = 0%	1 / 2 = 50%	71 / 75 = 94.6%

En la tabla 8 se muestra, de manera más detallada, la precisión por cada tipo de vehículo en este modelo y la cantidad de vehículos mal categorizados por cada clase.

Tabla 8. Resultados del Modelo M sobre el vídeo de la India

Modelo / Realidad	Coches	Vehículos ligeros	Vehículos pesados	Total (Realidad)
Coches	70(-1)*	0	2	71
Vehículos ligeros	0	0	0	2
Vehículos pesados	1	0	1	2
Total (modelo)	71	0	3	74/75**

***Nota:** De los turismos que se cuentan uno de ellos es duplicado, por lo que, para no contaminar la precisión del modelo, se resta.

****Nota:** En la Tabla 8 del presente modelo podemos ver que, a pesar de que circulan 2 vehículos pesados solamente se cuenta uno. De los vehículos ligeros, no cuenta ninguno y cuenta un vehículo pesado como turismo.

4.1.4 Modelo L

Comparando los resultados obtenidos por el modelo, con los resultados *"ground-truth"* del alumno, obtenemos unos resultados de precisión que se muestran en la tabla 9:

Tabla 9. Precisión del modelo L sobre cada clase y el total

	Precisión coches (Pc)	Precisión veh.ligeros (P.l)	Precisión veh.pesados (P.p)	Precisión modelo (Pm)
Modelo N	63 / 71 = 88.7%	1 / 2 = 50%	2 / 2 = 100%	66 / 75 = 88%

En la tabla 8 se muestra, de manera más detallada, la precisión por cada tipo de vehículo en este modelo y la cantidad de vehículos mal categorizados por cada clase.

Tabla 10. Resultados del Modelo L sobre el vídeo de la India

Modelo / Realidad	Coches	Vehículos ligeros	Vehículos pesados	Total (Realidad)
Coches	63	0	8	71
Vehículos ligeros	0	1	0	2
Vehículos pesados	0	0	2	2
Total (modelo)	63	1	10	74/75*

***Nota:** En la tabla 10 del presente modelo se identifica y cuenta ambos vehículos pesados y uno de los dos vehículos ligeros que circulan por la carretera. En cuanto a los turismos, clasifica 8 como si fueran vehículos pesados y el resto los identifica y clasifica correctamente.

4.2 Vídeo de test en España

En esta sección, se van a presentar los resultados obtenidos de los distintos pesos pre-entrenados de Yolov8 trabajando sobre el video grabado por el alumno expuesto en el punto 3.1.4.
Para el tratamiento de los resultados, se ha coloreado de color naranja los valores que contienen errores en la clasificación.

De color azul se muestran los vehículos correctamente clasificados y contados y, después, de color verde está marcado el valor total de vehículos que ha contado nuestro modelo, haciendo la suma de todos los turismos, vehículos pesados y ligeros que ha contado.

4.2.1 Modelo N

Comparando los resultados obtenidos por el modelo, con los resultados *"ground-truth"* del alumno, obtenemos unos resultados de precisión que se muestran en la tabla 11:

Tabla 11. Precisión del modelo N sobre cada clase y el total

	Precisión coches (Pc)	Precisión veh.ligeros (P.l)	Precisión veh.pesados (P.p)	Precisión modelo (Pm)
Modelo N	131 / 157 = 83.4%	0 / 5 = 0%	2 / 5 = 40%	133 / 167= 79.6%

En la tabla 12 se muestra, de manera más detallada, la precisión por cada tipo de vehículo en este modelo y la cantidad de vehículos mal categorizados por cada clase. En dicha tabla se ilustra cómo se identifica y se cuentan 2 vehículos pesados y uno de ellos lo cuenta doble de los 5 que circulan y ningún vehículo ligero de los 5 que pasan. En cuanto a los turismos, es capaz de identificar y clasificar correctamente a 135 de los 157 que cuenta.

Tabla 12. Resultados del Modelo N sobre el vídeo de España

Modelo / Realidad	Coches	Vehículos ligeros	Vehículos pesados	Total (Realidad)
Coches	135(-4)*	0	32(-6)*	157
Vehículos ligeros	0	0	0	5
Vehículos pesados	1	0	3(-1)*	5
Total (modelo)	136	0	35	171/ 167**

***Nota:** Se restarán 4 coches de los 135 bien clasificados y contados y 6 de los coches contados como vehículos pesados. Además, se resta 1 camión de los 3 contados ya que uno lo cuenta doble.

****Nota:** Al sumar los 135 coches que han circulado más los 32 coches que han sido contados como vehículos pesados obtenemos 167, lo cual es un desfase para tener en cuenta con respecto a los 157 que realmente hay. Este desfase proviene de que algunos de los coches que se cuentan como vehículos pesados, se cuentan doble y, además, 4 coches que se clasifican y cuenta bien como turismos,

corresponden a coches de un camión que los transporta pero que no han circulado realmente por la carretera.

4.2.2 Modelo S

Comparando los resultados obtenidos por el modelo, con los resultados *"ground-truth"* del alumno, obtenemos unos resultados de precisión que se muestran en la tabla 13:

Tabla 13. Precisión del modelo S sobre cada clase y el total

	Precisión coches (Pc)	Precisión veh.ligeros (P.l)	Precisión veh.pesados(P.p)	Precisión modelo (Pm)
Modelo S	143 / 157 = 91%	1 / 5 = 20%	3 / 5 = 60%	148 / 167= 88.6%

En la tabla 14 se muestra, de manera más detallada, la precisión por cada tipo de vehículo en este modelo y la cantidad de vehículos mal categorizados por cada clase. En ella se cuenta 1 vehículo ligero de los 5 que circulan por la carretera mientras que cuenta 3 de los 5 vehículos pesados. En cuanto a los turismos, es capaz de identificar y clasificar correctamente a 147 de los 166 que cuenta.

Tabla 14. Resultados del Modelo S sobre el vídeo de España

Modelo / Realidad	Coches	Vehículos ligeros	Vehículos pesados	Total (Realidad)
Coches	147(-4)*	0	19(-5)*	157
Vehículos ligeros	0	1	0	5
Vehículos pesados	0	0	3	5
Total (modelo)	147	1	22	170/167**

***Nota:** Se restarán 4 coches de los 147 bien clasificados y contados y 5 de los coches contados como vehículos pesados

****Nota:** Al sumar los 147 coches que han circulado más los 19 coches que han sido contados como vehículos pesados obtenemos 166, lo cual es un desfase para tener en cuenta con respecto a los 157 que realmente hay. Este desfase proviene de que algunos de los coches que se cuentan como vehículos pesados, se cuentan doble y, además, 4 coches que se clasifican y cuenta bien como turismos, corresponden a coches de un camión que los transporta pero que no han

circulado realmente por la carretera.

4.2.3 Modelo M

Comparando los resultados obtenidos por el modelo, con los resultados *"ground-truth"* del alumno, obtenemos unos resultados de precisión que se muestran en la tabla 15:

Tabla 15. Precisión del modelo M sobre cada clase y el total

	Precisión coches (Pc)	Precisión veh.pesados (P.p)	Precisión veh.ligeros (P.l)	Precisión modelo (Pm)
Modelo M	148 / 157 = 94.2%	3 / 5 = 60%	3 / 5 = 60%	154 / 167= 92.2%

En la tabla 16 se muestra, de manera más detallada, la precisión por cada tipo de vehículo en este modelo y la cantidad de vehículos mal categorizados por cada clase. Se puede visualizar cómo se identifican y cuentan 3 vehículos pesados y 3 vehículos ligeros que circulan por la carretera. En cuanto a los turismos, es capaz de identificar y clasificar correctamente a 153 de los 170 que cuenta.

Tabla 16. Resultados del Modelo M sobre el vídeo de España

Modelo / Realidad	Coches	Vehículos ligeros	Vehículos pesados	Total (Realidad)
Coches	153(5)*	0	17(8)*	157
Vehículos ligeros	0	3	0	5
Vehículos pesados	1	0	3	5
Total (modelo)	154	3	20	177/167**

***Nota:** Se restarán 5 coches de los 153 bien clasificados y contados y 8 de los coches contados como vehículos pesados.

****Nota:** Al sumar los 153 coches que han circulado más los 17 coches que han sido contados como vehículos pesados obtenemos 170, lo cual es un desfase para tener en cuenta con respecto a los 157 que realmente hay. Este desfase proviene de que algunos de los coches que se cuentan como vehículos pesados, se cuentan doble y, además, 5 coches que se clasifican y cuenta bien como turismos, corresponden a coches de un camión que los transporta pero que no han

circulado realmente por la carretera.

4.2.4 Modelo L

Comparando los resultados obtenidos por el modelo, con los resultados *"ground-truth"* del alumno, obtenemos unos resultados de precisión que se muestran en la tabla 17:

Tabla 17. Precisión del modelo L sobre cada clase y el total

	Precisión coches (Pc)	Precisión veh.ligeros (P.l)	Precisión veh.pesados (P.p)	Precisión modelo (Pm)
Modelo L	132 / 157= 84%	3 / 5 = 60%	3 / 5 = 60%	138 / 167= 82.6%

En la tabla 18 se muestra, de manera más detallada, la precisión por cada tipo de vehículo en este modelo y la cantidad de vehículos mal categorizados por cada clase. En este último caso, se identifican y cuentan 3 vehículos pesados y 3 vehículos ligeros que circulan por la carretera. En cuanto a los turismos, es capaz de identificar y clasificar correctamente a 138 de los 167 que cuenta.

Tabla 18. Resultados del Modelo L sobre el vídeo de España

Modelo / Realidad	Coches	Vehículos ligeros	Vehículos pesados	Total (Realidad)
Coches	138(6)*	0	29(4)*	157
Vehículos ligeros	0	3	0	5
Vehículos pesados	2	0	3	5
Total (modelo)	147	3	32	182/167**

***Nota:** Se restarán 6 coches de los 138 bien clasificados y contados y 4 de los coches contados como vehículos pesados.

****Nota:** Al sumar los 138 coches que han circulado más los 29 coches que han sido contados como vehículos pesados obtenemos 167, lo cual es un desfase para tener en cuenta con respecto a los 157 que realmente hay. Este desfase proviene de que algunos de los coches que se cuentan como vehículos pesados, se cuentan doble y, además, 5 coches que se clasifican y cuenta bien como turismos, corresponden a coches de un camión que los transporta pero que no han circulado realmente por la carretera.

4.3 Análisis de los resultados

La tabla 19 contiene las características adheridas a cada uno de los modelos, como el tiempo de inferencia entre fotograma y fotograma, el tiempo que tarda cada modelo en procesar el vídeo propuesto debido a la cantidad de parámetros con los que fue entrenado, etc. Con el análisis de estos parámetros de cada modelo, se comparará cuál modelo sería la mejor opción.

Tabla 19. Tabla comparativas con los parámetros de cada uno de los modelos pre-entrenados en España

	Precisión modelo	Inferencia (ms)	Tiempo de procesado	Nº de parámetros	Nº de capas ocultas
Modelo N	79.6%	11	6:20 minutos	3M	168
Modelo S	88.6%	19	7:43 minutos	11M	168
Modelo M	92.2%	41	9:09 minutos	25.8M	218
Modelo L	82.6%	58	11:22 minutos	43M	268

- La columna “Inferencia” es el tiempo requerido para realizar la inferencia o predicción utilizando el modelo, entre fotograma y fotograma.

- La columna “Tiempo de procesado” hace referencia al tiempo que ha tardado el modelo en analizar el vídeo por completo. Video el cual tiene una duración máxima de 3:30 minutos.

- Las dos siguientes columnas se refieren al número de parámetros que se han usado para entrenar la red neuronal y con cuantas capas ocultas.

5 Conclusiones

El presente proyecto demuestra la utilidad y efectividad de la combinación de técnicas de detección, seguimiento y conteo de objetos en la aplicación de la visión artificial. Se evaluaron diferentes categorías del modelo YOLOv8 para abordar el problema de multi-object counting (MOC) y multi-object tracking en tiempo real, obteniendo resultados satisfactorios en precisión y eficiencia.

El programa desarrollado puede detectar, seguir y contar los vehículos en un tramo específico de carretera a través de los frames de un video, clasificándolos en función de su categoría y recopilando datos que pueden automatizar y acelerar las mediciones acústicas necesarias en la carretera.

El resultado fue satisfactorio, con un porcentaje de acierto entre 80% y 90% en todos los modelos. Tal y como se ha mostrado en la Tabla 19, el modelo que presenta una mayor precisión, no solo en el conteo total de los vehículos sino también por cada clase y, además, una velocidad aceptable, es el modelo pre-entrenado de YOLOv8m.

En línea con esto, se destacó la necesidad de contar con videos colocados en una posición, ángulo y altura específicos para evitar oclusiones y sombras que intercedan en la precisión del modelo, y de realizar el conteo en un tramo horario significativo con alta cantidad de tráfico para comprobar su eficacia en entornos de alto flujo de tráfico. Por esta razón, se realizó el estudio a plena luz del día, evitando la noche debido a que la oscuridad podría opacar por completo el proyecto.

Asimismo, otro problema que puede surgir es el que ha sucedido en este mismo trabajo que es que circule un camión transportando vehículos ya que el modelo es capaz de detectarlos y contarlos, por lo que contaminaría la precisión y ha de tenerse en cuenta. Otra limitación sería que, al ser la línea dependiente del vídeo, debería crearse en código una nueva LOI en función al video que fuera a tratarse.

5.1 Futuras líneas de trabajo

Las futuras líneas de trabajo que se pueden abordar a partir de los resultados obtenidos en este proyecto incluyen la exploración e implementación de nuevos modelos de detección de objetos y seguimiento de objetos para mejorar la precisión y eficiencia del sistema, así como la implementación de técnicas de visión por computadora para mejorar la detección y clasificación de vehículos.

Se pueden estudiar alternativas para solucionar las limitaciones encontradas en el posicionamiento y orientación de las cámaras de video, como el uso de cámaras múltiples o drones para una mejor visibilidad y cobertura del área de interés. También sería interesante la exploración de aplicaciones de estas técnicas en otros ámbitos, como el control de calidad en la industria o la seguridad en espacios públicos.

En resumen, hay muchas posibilidades para seguir mejorando y aplicando las técnicas de visión artificial en el seguimiento y conteo de objetos, y se pueden explorar diversas líneas de trabajo para seguir avanzando en este campo.

5.2 Competencias adquiridas

Las competencias adquiridas con este trabajo de fin grado han sido amplias en el sector de la inteligencia artificial. Desde conocimientos teóricos sobre los fundamentos de las redes neuronales convolucionales, desarrollo de habilidades programando con el lenguaje Python para implementar y entrenar redes neuronales, procesamiento de bases de datos con imágenes con tareas como redimensionamiento, normalización y aumento de datos, entrenamiento y ajuste de modelos utilizando conjuntos de datos de entrenamiento y evaluando su rendimiento con set de datos de prueba, optimización de dichos modelos reajustando hiperparámetros, número de capas, tamaño de las imagen de entrada, etc. Además de nociones relevantes en cuanto al coste computacional que significa entrenar una red, y tiempos de procesamiento en función a qué se use para entrenarla, aprendiendo con ello a hacer uso de la tarjeta gráfica o del entorno colaborativo que ofrece Google llamado *Google Colab* para realizar las operaciones con

tensores de forma paralela y disminuir considerablemente el tiempo requerido para el proyecto.

En este sentido, se profundizó en el acercamiento a cómo programar una red neuronal y por qué Python es un lenguaje muy usado en este sector, gracias a librerías como *Tensorflow*, *Pytorch*, *Keras*, bibliotecas de visualización de datos como *NumPy* y *Matplotlib* o *Pandas* para procesado de *dataframes*.

Bibliografía

[1] J. C. P. Gallegos *et al*, "Inteligencia Artificial," 2014. DOI: 10.13140/2.1.3720.0960.

[2] W. S. Mcculloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bulletin of Mathematical Biology,* vol. 52, *(1-2),* pp. 99, 1990. DOI: 10.1016/s0092-8240(05)80006-0.

[3] A. M. Turing, "Computing machinery and intelligence," in *Readings in Cognitive Science*Anonymous Elsevier Inc, 1988, pp. 6-19.

[4] Frank Rosenblatt, “The perceptron a perceiving and recognizing automaton. Report” 1957, No. 85-460-1 .

[5] Rodríguez-Sánchez, Juan & Landassuri, Victor & Flores Albino, José, “Reconocimiento de patrones numéricos para vuelo controlado de un AR Drone utilizando redes neuronales artificiales.” Research in Computing Science. 107. 61-71. 10.13053/rcs-107-1-6. (2015)

[6] A. S. Florez Fuentes, R. Guzmán Cabrera, y E. Vargas Rodriguez, “Identificación automática de cáncer de piel aplicando Machine Learning”. RCTA, vol. 2, n.º 40, mayo 2023.

[7] R. A. Manjarrés-Betancur y M. M. Echeverri-Torres, “Asistente virtual académico utilizando tecnologías cognitivas de procesamiento de lenguaje natural”. Rev. politec., vol. 16, n.º 31, pp. 85–96, mayo 2020.

[8] Stuart J. Russell y Peter Norvig, “Inteligencia Artificial: Un enfoque moderno”, 2ª Edición, Prentice Hall. 2010.

[9] Andreas C. Müller & Sarah Guido, “Introduction to Machine Learning with Python”, 1ª Edición, O’Reilly Media. 2017.

[10] S. Maqbool, M. Khan, J. Tahir, A. Jalil, A. Ali and J. Ahmad, "Vehicle Detection, Tracking and Counting," 2018 IEEE 3rd International Conference on Signal and Image Processing (ICSIP), Shenzhen, China, 2018, pp. 126-132, DOI: 10.1109/SIPROCESS.2018.8600460.

[11] *Técnicas ML*. https://openwebinars.net/blog/modelos-de-machine-learning/

[12] *Cross Validation*. https://towardsdatascience.com/cross-validation-explained-evaluating-estimator-performance-e51e5430ff85

[13] *Aprendizaje por refuerzo*. https://pixelabs.es/machine-learning-3/

[14] Ian Goodfellow, Yashua Bengio and Aaron Courville, “Deep Learning”, MIT Press. 2021.

[15] Morris RG. D.O. Hebb, "The Organization of Behavior, Wiley: New York; 1949. Brain Res Bull. ;50(5-6):437." 1999. . DOI: 10.1016/s0361-9230(99)00182-3.

[16] Estudio de Inttensidad Media Diaria de vehículos (IMD). Comunidad de Madrid (2021) tráfico. https://www.comunidad.madrid/servicios/transporte/estudio-intensidad-media-diaria-vehiculos-imd

[17] Xu Bao et al, "Traffic Vehicle Counting in Jam Flow Conditions Using Low-Cost and Energy-Efficient Wireless Magnetic Sensors." 2016. DOI: 1424-8220/16/11/1868#.

[18] M. F. AbdelHaq and A. Salman, "Wireless Sensor Network for Traffic Monitoring", 2020 International Conference on Promising Electronic Technologies (ICPET), Jerusalem, Palestine, 2020, pp. 16-21, DOI: 10.1109/ICPET51420.2020.00012.

[19] *Estructura del ojo humano*. https://cienciainquieta.wordpress.com/2013/12/02/el-ojo-humano-como-funciona/.

[20] *Estructura de una red convolucional*. https://www.codificandobits.com/blog/redes-convolucionales-introduccion/.

[21] L. Bottou et al., "Comparison of classifier methods: a case study in handwritten digit recognition," Proceedings of the 12th IAPR International Conference on Pattern Recognition, Vol. 3 - Conference C: Signal Processing (Cat. No.94CH3440-5), Jerusalem, Israel, 1994, pp. 77-82 vol.2, DOI: 10.1109/ICPR.1994.576879.

[22] N. Srivastava et al, "Dropout: a simple way to prevent neural networks from overfitting," Journal of Machine Learning Research, vol. 15, pp. 1929, 2014. . DOI: 10.5555/2627435.2670313#d2877736e1.

[23] U. Muhammad, W. Wang, S. P. Chattha and S. Ali, "Pre-trained VGGNet Architecture for Remote-Sensing Image Scene Classification," pp. 1622-1627. 2018. DOI: 10.1109/ICPR.2018.8545591.

[24] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, “Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning”, AAAI, vol. 31, no. 1, Feb. 2017.

[25] Z. Wu, C. Shen, A. Hengel, "Wider or Deeper: Revisiting the ResNet Model for Visual Recognition" Volume 90 ,Pages 119-133, 2019. DOI: https://DOI.org/10.1016/j.patcog.2019.01.006

[26] M. Kim, Y. Kwon, J. Kim, and Y. Kim, “Image Classification of Parcel Boxes under the Underground Logistics System Using CNN MobileNet,” Applied Sciences, vol. 12, no. 7, p. 3337, Mar. 2022, DOI: 10.3390/app12073337.

[27] *Detección de vehículos*. https://pysource.com/2021/10/05/object-tracking-from-scratch-opencv-and-python/.

[28] Rani, P. & Jamiya S, Sri, “ShortYOLO-CSP: a decisive incremental improvement for real-time vehicle detection” Journal of Real-Time Image Processing. (2023). DOI: 10.1007/s11554-023-01256-0.

[29] *Ultralytics Yolov8 Docs*. https://docs.ultralytics.com/models/yolov8/.

[30] A. Bewley, Z. Ge, L. Ott, F. Ramos and B. Upcroft, "Simple online and realtime tracking", pp. 3464-3468, (2016) DOI: 10.1109/ICIP.2016.7533003.

[31] K. de Langis and J. Sattar, "Realtime Multi-Diver Tracking and Re-identification for Underwater Human-Robot Collaboration", pp. 11140-11146. (2020). DOI: 10.1109/ICRA40945.2020.9197308.

[32] Maher, A., Taha, H. & Zhang, B. “Realtime multi-aircraft tracking in aerial scene with deep orientation network”, J Real-Time Image Proc 15, 495–507 (2018). https://DOI.org/10.1007/s11554-018-0780-1

[33] A. Bewley, Z. Ge, L. Ott, F. Ramos and B. Upcroft, "Simple online and realtime tracking", pp. 3464-3468, (2016) DOI: 10.1109/ICIP.2016.7533003.

[34] N. Xiong, , J. He, , J. Park, , D. Cooley, , Y. Li, “A Neural Network Based Vehicle Classification System for Pervasive Smart Road Security”, 15, 5. 1119–1142, (2009). DOI: 10.3217/jucs-015-05-1119.

[35] Fazli, Saeid and Mohammadi, Shahram and Rahmani, Morteza, “Neural Network based Vehicle Classification for Intelligent Traffic Control”, Vol.3, No.3, May 2012

[36] A. Goyal and B. Verma, “A Neural Network based Approach for the Vehicle Classification”, pp. 226-231. (2007). DOI: 10.1109/CIISP.2007.369173.

[37] Michael S. Landy, Yoav Cohen, George Sperling, "HIPS: A unix-based image processing system. Computer Vision, Graphics, and Image Processing", Volume 25, Issue 3, pg. 331-347, (1984). DOI: 10.1016/0734-189X(84)90199-3

[38] P. M. Daigavane, P. R. Bajaj and M. B. Daigavane, "Vehicle Detection and Neural Network Application for Vehicle Classification", pp. 758-762. (2011). DOI: 10.1109/CICN.2011.168

[39] O. Barnich and M. Van Droogenbroeck, "ViBe: A Universal Background Subtraction Algorithm for Video Sequences", vol. 20, no. 6, pp. 1709-1724, June 2011. DOI: 10.1109/TIP.2010.2101613

[40] Bay, H., Tuytelaars, T., Van Gool, L, “SURF: Speeded Up Robust Features”, vol 3951, (2006). https://DOI.org/10.1007/11744023_32

[41] X. Xiang, M. Zhai, N. Lv, and A. El Saddik, “Vehicle Counting Based on Vehicle Detection and Tracking from Aerial Videos”, vol. 18, no. 8, p. 2560, Aug. 2018. DOI: 10.3390/s18082560

[42] S. Tas, O. Sari, Y. Dalveren, S. Pazar, A. Kara, and M. Derawi, “Deep Learning-Based Vehicle Classification for Low Quality Images”, Sensors, vol. 22, no. 13, p. 4740, Jun. 2022. DOI: 10.3390/s22134740

[43] Yousaf, Kanwal & Iftikhar, Arta & Javed, Ali, “Comparative Analysis of Automatic Vehicle Classification Techniques: A Survey”, (2012). DOI: 10.5815/ijigsp.2012.09.08

[44] Memon, Sheeraz & Bhatti, Sania & Ali, Liaquat & Talpur, Mir & Memon, Mohsin, “A Video based Vehicle Detection, Counting and Classification System”, (2018). DOI: 10. 34-41. 10.5815/ijigsp.2018.09.05

[45] Z. Kadim et al, "Real-time vehicle counting in complex scene for traffic flow estimation using multi-level convolutional neural network,", vol. 8, (75), pp. 338-351. (2021). DOI: 10.19101/IJATEE.2020.762128

[46] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection", pp. 779-788. (2016). DOI: 10.1109/CVPR.2016.91.

[47] Liu, W. et al., “SSD: Single Shot MultiBox Detector”, vol 9905. (2016). https://DOI.org/10.1007/978-3-319-46448-0_2

[48] Singh Chauhan, Mayank & Singh, Arshdeep & Khemka, Mansi & Prateek, Arneish & Sen, Rijurekha, “Embedded CNN based vehicle classification and counting in non-laned road traffic”, January 2019.

[49] Everingham, M., Eslami, S.M.A., Van Gool, L. et al., “The PASCAL Visual Object Classes Challenge: A Retrospective”, pg 98–136. (2015). https://DOI.org/10.1007/s11263-014-0733-5

[50] Geiger, Andreas & Lenz, P & Stiller, Christoph & Urtasun, Raquel, “Vision meets robotics: the KITTI dataset”, The International Journal of Robotics Research. 32. 1231-1237. (2013). DOI: 10.1177/0278364913491297.

[51] Biswas, Debojit et al., “An Automatic Car Counting System Using OverFeat Framework”, vol. 17,7 1535, 30 Jun. 2017. DOI: 10.3390/s17071535

[52] S. Sivaraman and M. M. Trivedi, "A General Active-Learning Framework for On-Road Vehicle Recognition and Tracking", vol. 11, no. 2, pp. 267-276, June 2010. DOI: 10.1109/TITS.2010.2040177

[53] I. O. De Oliveira, R. Laroca, D. Menotti, K. V. O. Fonseca and R. Minetto, "Vehicle-Rear: A New Dataset to Explore Feature Fusion for Vehicle Identification Using Convolutional

Neural Networks", vol. 9, pp. 101065-101077. (2021). DOI: 10.1109/ACCESS.2021.3097964

[54] *Web curso de computer vision*. https://www.computervision.zone/courses/object-detection-course/

[55] *Matriz de confusión*. https://wbarriosb.medium.com/calculando-la-precisi%C3%B3n-en-un-modelo-de-clasificaci%C3%B3n-multiclase-224d96f52043