



Grado en Ingeniería Electrónica Industrial y Automática  
2021-2022

*Trabajo Fin de Grado*

## “Detector de Matrículas en Entorno Interurbano”

---

Matthias Gdanietz De Diego

Tutor/es:

José María Armigol Morejo

Álvaro Ramajo Ballester

Defensa: 11/7/2022 | Aula: 2.3C02B



Esta obra se encuentra sujeta a la licencia Creative Commons **Reconocimiento – No Comercial – Sin Obra Derivada**





---

## RESUMEN

En este proyecto se pretende desarrollar un sistema OCR como fase final de un detector de matrículas en entorno interurbano, OCR viene del inglés: *Optical Character Recognition* traducido como “Reconocimiento óptico de caracteres”.

Para desarrollar un sistema reconocedor de caracteres, el proyecto se divide en varias partes. Primero la creación de un *dataset*, con matrículas de España, en los distintos tipos de formatos que existen para los vehículos.

Se etiqueta cada una de esas imágenes, para luego segmentar y crear una parte para entrenar la red neuronal y otra para la validación de la misma. Seguidamente, se hace el entrenamiento del modelo, para ajustar los pesos usando para ello YOLO v5. Se buscará el modelo con mejores resultados a partir de los modelos ya pre-entrenados que ofrece YOLO v5.

Finalmente, el modelo entrenado servirá para la creación de un código que ordene los caracteres segmentados encontrados en el input de la matrícula recortada y rectificada, para un correcto funcionamiento.

### Palabras Clave:

Visión por computador, Detector de matrículas, Deep Learning, OCR.



---

## ABSTRACT

This project aims to develop an OCR system as the final phase of a license plate detector in an interurban environment, OCR stands for *Optical Character Recognition*.

To develop a character recognition system, this project is divided into several parts. First, the creation of a dataset, with license plates of Spain, in the different types of formats that exist for vehicles.

Each of these images is labeled, and then segmented and created a part to train the neural network and another for the validation of the same. Next, a model will be trained to adjust the weights using YOLO v5. The model with the best results will be searched from the pre-trained models offered by YOLO v5.

Finally, the trained model will be used to create a code that orders the segmented characters found in the input of the trimmed and rectified license plate, for a correct operation.



---

## AGRADECIMIENTOS

Quiero agradecer a mi familia y a todos los que me han ido acompañando en este camino, sobre todo a mi madre que me ha apoyado y ha luchado contra el cáncer.

También quiero agradecer a mis tutores José María Armigol y Álvaro Ramajo que me han ayudado en la elección de TFG y en el desarrollo del proyecto, finalmente agradecer a la Universidad Carlos III por dotar de los recursos necesarios para la realización del mismo.

# ÍNDICE DE CONTENIDO:

<b>Resumen</b>	III
<b>Abstract</b>	IV
<b>Agradecimientos</b>	V
<b>Índice de Contenido</b>	VI
<b>Índice de Figuras</b>	VIII
<b>Índice de Tablas</b>	X
<b>Lista de Abreviaturas</b>	XI
<b>1. Introducción</b>	12
1.1. Introducción de contexto .....	12
1.2. Intereses y Motivación del TFG .....	13
1.3. Objetivos .....	13
1.4. Planificación .....	14
1.5. Estructura del proyecto .....	15
<b>2. Estado del Arte</b>	16
2.1. Recursos externos .....	16
2.1.1. Sistemas ANPR .....	16
2.1.2. Sistemas OCR .....	22
2.1.3. Sistemas actuales Deep Learning para ANPR .....	24
2.2. Marco Regulador .....	26
2.2.1. Normativa legal .....	26
2.2.2. Estándares Técnicos .....	28
<b>3. Desarrollo e Implementación</b>	29
3.1. Análisis del Problema .....	29
3.2. Tecnologías Empleadas .....	29
3.2.1. Hardware .....	29
3.2.2. Software .....	30
3.3. Funcionamiento YOLO V5 .....	31

3.4.	Etiquetado OCR .....	34
3.5.	Entrenamiento YOLO V5 .....	35
3.6.	Reconocimiento de caracteres .....	42
<b>4.</b>	<b>Resultados</b>	<b>46</b>
4.1.	Resultados obtenidos del entrenamiento de YOLO V5 .....	46
4.2.	Resultados obtenidos en el reconocimiento de caracteres .....	48
4.3.	Análisis de los resultados .....	52
<b>5.</b>	<b>Trabajos futuros y Conclusión</b>	<b>53</b>
5.1.	Trabajos futuros .....	53
5.2.	Conclusión .....	54
<b>6.</b>	<b>Entorno Socio-Económico</b>	<b>55</b>
6.1.	Presupuesto .....	55
6.2.	Impacto Socio-Económico .....	57
<b>Bibliografía</b>		<b>60</b>

---

# ÍNDICE DE FIGURAS:

Fig 1: Diagrama de Gantt.

Fig 2: Sistema Convencional ANPR - Tres Etapas.

Fig 3: Detección de bordes Canny.

Fig 4: Imagen convertida a método de Otsu.

Fig 5: Resultado Meanshift.

Fig 6: Mínimos Cuadrados.

Fig 7: OCR Implementación Hardware.

Fig 8: Estructura general CRNN para reconocimiento de texto.

Fig 9: Arquitectura EasyOCR.

Fig 10: YOLO.

Fig 11: Arquitectura Software.

Fig 12: Ejemplo de caja limitadora YOLO.

Fig 13: División de la imagen | Cuadros delimitado | Cuadro de mayor probabilidad.

Fig 14: Arquitectura YOLO v5.

Fig 15: Etiquetado usando Labelme.

Fig 16: Representación de clases y cuadros delimitadores YOLO v5.

Fig 17: Ejemplo de entrenamiento, con probabilidad de detección.

Fig 18: Modelos ya entrenados de YOLO v5.

Fig 19: Comparación de YOLO v5 con EfficientDet.

Fig 20: Resumen de características de los modelos de YOLO v5.

Fig 21: Intersection over union.

Fig 22: Matriz de confusión.

Fig 23: Resultados de detección de objetos con dataset en COCO.

Fig 24: Gráficas métricas Weight and Biases.

Fig 25: Valores de pérdida Train y Val.

Fig 26: data augmentation mosaico.

Fig 27: Evolve Hiperparametros.

Fig 28: Hiperparametros cambiados + en Default.

Fig 29: Detección y rectificación de matrícula.

Fig 30: Matrícula Escalada.

Fig 31: Padding de Matrícula.

Fig 32: Reconocimiento de caracteres segmentados.

Fig 33: Labels y Coordenadas.

Fig 34: Label correspondiente al carácter.

Fig 35: Matrículas de ciclomotor / motocicleta.

Fig 36: Matrícula Detectada.

Fig 37: Gráfica de los valores F1- score y confianza de las clases de Hyp Evolve 2.

Fig 38: Gráfica Instancias / clases.

Fig 39: Matriz de confusión entrenamiento.

## ÍNDICE DE TABLAS:

Tabla 4.1.1 - Cuadro de métricas BestEntrenamiento.pt

Tabla 4.2.1 - Matrículas Convencionales

Tabla 4.2.2 - Matrículas de vehículos especiales

Tabla 4.2.3 - Matrículas formato previo al 2000

Tabla 4.2.4 - Matrículas Motocicletas y Ciclomotores

Tabla 6.1.1 - Matriz coste recursos humanos

Tabla 6.1.2 - Matriz coste herramientas hardware

Tabla 6.1.3 - Matriz coste herramientas software

Tabla 6.1.4 - Matriz coste recursos externos

Tabla 6.1.5 - Presupuesto total



---

## LISTA DE ABREVIATURAS:

TFG: Trabajo final de Grado

OCR: Optical Character Recognition

ANPR: Automatic Number Plate Recognition

CNN: Convolution Neural Network

CCA: Análisis de Componentes Conectados

HSL: Hue, Saturation, Lightness

SVM: Support Vector Machine

R-CNN: Region Based Convolutional Neural Network

YOLO: You Only Look Once

IoU: Intersection over Union

# 1. Introducción

En este primer apartado de la memoria del TFG se va a presentar la finalidad del proyecto a desarrollar, incluyendo una justificación con diferentes ejemplos para comprender y poder completar el trabajo con la mejor resolución posible. Para ello será necesario establecer unos objetivos y finalmente hacer un resumen del contenido de la misma.

## 1.1. Introducción de contexto

Este trabajo de investigación, se ha realizado en el departamento de Ingeniería de Sistemas y Automática en la Universidad Carlos III de Madrid, en donde surge la idea de desarrollar un sistema de detección y reconocimiento de placas de matrícula de automóviles para ser usado en entornos interurbanos. Se entiende como matrícula a la placa de metal que llevan los vehículos en la parte frontal y trasera de la carrocería para poder identificarlos.

En la actualidad, los sistemas de detección de matrículas o ANPR (Automatic Number Plate Recognition) son sistemas de vigilancia de vehículos y sirven para el reconocimiento de matrículas.

Esto se logra mediante el uso de cámaras que capturan imágenes y se analizan vía ordenador, mediante el uso de técnicas de Visión por Computador, de esta forma, se puede obtener información de las imágenes, extrayendo características relevantes visuales.

El reconocimiento de matrículas usando estos sistemas de Visión por Computador, se han convertido en esta última década en un tema de bastante interés comercial, debido a la continua recopilación de datos, con múltiples aplicaciones como el acceso de aparcamientos o el control de peajes, etc. Pero sobre todo se ha convertido en una parte fundamental para el control del tráfico. Sin embargo, el continuo avance de las tecnologías relacionadas en campo de la Visión por Computador, han permitido poner en práctica, de nuevos sistemas para el reconocimiento, usando para ello tecnologías como Machine Learning o Deep Learning, debido a que cada vez son más asequibles.

Además, uno de los temas de investigación más actuales son los Vehículos Autónomos, este campo abre muchas posibilidades de cambiar la manera de transporte, eliminando la intervención humana, usando por ejemplo estos sistemas ANPR.

---

## 1.2. Intereses y motivación del TFG

Los principales objetivos de este proyecto son alcanzar mediante el uso de técnicas de Inteligencia Artificial y Visión por Computador, la identificación y caracterización de vehículos para la monitorización del tráfico. Para ello, se utilizará una cámara de alta resolución que permita captar con suficiente detalle las matrículas a través de un sistema situado en una infraestructura en un entorno interurbano.

Una herramienta que será de utilidad para la gestión de las smart cities, estableciendo una comunicación entre los vehículos y la infraestructura vial. Esto permitirá mejorar el flujo del tráfico, sincronizando elementos como semáforos y señalización, para así lograr un mayor control de la circulación, evitando así accidentes o caravanas.

En cuanto a intereses personales y motivación para la realización del proyecto, ha sido el enfoque que ha tenido las últimas asignaturas realizadas en la universidad, sobre, sistemas de percepción, sistemas en tiempo real, etc. De las cuales el aprendizaje ha sido gratificante, pero sobre todo, la curiosidad y las ganas de aprender más. Siendo esto un comienzo a quizás un trabajo futuro.

## 1.3. Objetivos

En cuanto a los objetivos, este trabajo está centrado en el desarrollo de un OCR para el sistema de detección de matrículas en entorno interurbano.

Por lo tanto, el objetivo principal de este trabajo, es el estudio y puesta en práctica de redes neuronales convolucionales usando para ello, YOLO v5, para el entrenamiento, siendo este un detector de objetos rápido y preciso.

Sin embargo, para lograr implementar un sistema ANPR con éxito se deben establecer unos objetivos intermedios:

1. Creación de un *dataset* de imágenes de vehículos que muestren la matrícula y además los caracteres, debe tener imágenes de todo tipo, en distintas condiciones de iluminación, perspectivas, etc.
2. Hacer un correcto etiquetado de las imágenes para lograr un entrenamiento de la red neuronal efectivo.
3. Entrenar un modelo de red neuronal usando para ello el *dataset* creado y el detector de objetos YOLO v5 para el entrenamiento y que dé como resultado la capacidad de reconocer caracteres, de una matrícula.

4. Ajustar los hiper parámetros del modelo para lograr mayor velocidad de entrenamiento y mejores resultados.
5. Implementar los pesos para poder ser usados.
6. Finalmente el objetivo final es a partir del input de una matrícula, cuya perspectiva habrá sido rectificada, lograr ordenar los caracteres segmentados, e imprimir por pantalla la matrícula del vehículo.

## 1.4. Planificación

Para cumplir con los objetivos propuestos, la planificación resultante queda representada en la siguiente gráfica de Gantt, por un lado tenemos todos los meses para completar de principio a fin del proyecto y por otro lado de forma ordenada la actividad.



*Fig 1: Diagrama de Gantt*

El tiempo total del proyecto es estimado a lo que consta de una jornada laboral semanal, haciendo un total aproximado de 525 horas.



---

## 1.5. Estructura del Proyecto

Para cumplir los requisitos en relación al contenido, estructura y calidad del documento, la memoria del TFG se va a componer de varios apartados. Cada uno de los apartados corresponderá a temas debidamente diferenciados, tratando de mantener un orden y la claridad necesaria para evitar saltos en la lectura del documento.

Comienza con la *introducción*, donde se hace una breve descripción para comprender en qué consiste este proyecto, los intereses y la motivación para la realización del mismo. Además de establecer los objetivos que se persiguen.

Tras la introducción se encuentra el apartado del *estado del arte*, que consiste en comentar el proceso de investigación, mostrando los diversos estudios encontrados sobre la temática del proyecto en cuestión, además se comenta el marco regulador.

En el apartado 3, se da a conocer el *desarrollo del proyecto*, aquí se definen los caminos seguidos para llevar a cabo la elaboración del trabajo, al igual que alternativas que se han tenido en cuenta. Este apartado incluye imágenes para un mayor seguimiento.

Se añade un apartado más para comentar los resultados obtenidos en las pruebas realizadas una vez desarrollado el proyecto, para obtener información relevante y poder proponer soluciones para trabajos futuros. Seguidamente se elaboran las *conclusiones* de la memoria y se comentan posibles trabajos futuros.

Finalmente se comenta el impacto socio-económico referente al trabajo realizado, incluyendo el presupuesto total.

## 2. Estado del Arte

### 2.1. Recursos externos

Al investigar para el desarrollo de un sistema ANPR es común encontrar publicaciones relacionadas con la implementación de estos sistemas.

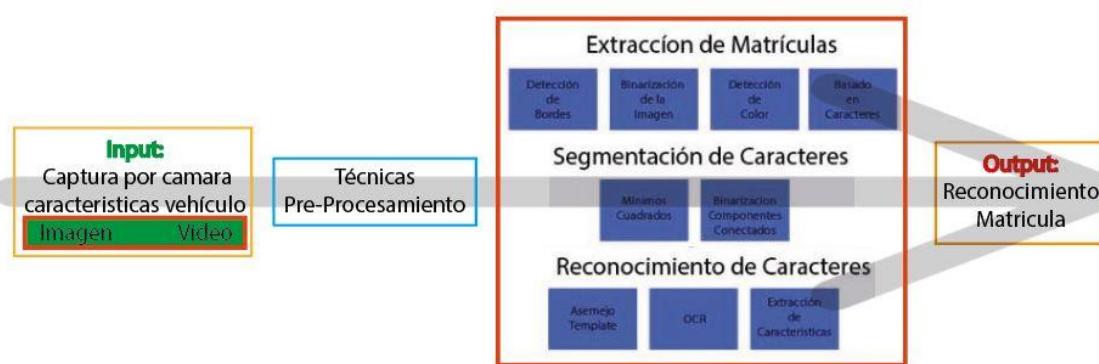
En dichos estudios se usa el campo de visión por computador o visión artificial para el procesamiento de imágenes, la extracción de información y características de ellas. Esto parece una tarea bastante sencilla para el ser humano, pero para una máquina es algo muy complejo. Sin embargo la capacidad humana de pensar de forma abstracta, ha permitido, que mediante simples operaciones matemáticas y usando las nuevas tecnologías, se logren cada vez más avances en el campo de la visión por computador.

Hoy en día, extraer información de las imágenes, como el grado de intensidad, la sobreexposición, son datos simples, que sirven para algoritmos más complejos como la detección de un rango de colores o la detección de bordes, que juntandolos pueden dar como resultado la capacidad a una máquina de incluso reconocer objetos.

#### 2.1.1. Sistemas ANPR

Para comenzar, ANPR es una abreviatura del inglés para *Automatic Number Plate Recognition* que corresponde a Detector de Matrículas Automático, estos sistemas a lo largo de los últimos años ha recibido varios nombres y acrónimos: NPR, ALPR, LPR.

Estos sistemas se desarrollan principalmente para la videovigilancia de vehículos, el acceso automático a parkings [1], el control de zonas restringidas, etc. Estos sistemas están compuestos por una cámara, que se encarga de obtener imágenes para su procesamiento mediante algoritmos, pasando por varias fases como se muestra en la *Fig 2*.



*Fig 2: Sistema Convencional ANPR - Tres Etapas*

A continuación se presentará un repositorio [2] sobre las distintas técnicas tradicionales para el procesamiento de imágenes usadas para cada una de las fases de un sistema ANPR. Estas técnicas se basan en las características de la matrícula:

→ **Extracción de la matrículas:**

La extracción de la matrícula es una de las partes más importantes de estos sistemas ANPR, ya que se establece el ratio de éxito a partir del total de matrículas detectadas en las imágenes de entrada. Es habitual instalar varias cámaras en cada línea de la carretera, pero cámaras más avanzadas son capaces de obtener resoluciones más altas y de múltiples carreteras.

En el caso de los modelos donde trabajan con varias cámaras en una carretera, requieren de más hardware y hace que los costes sean más altos, pero se obtienen las matrículas desde mejores ángulos. Sin embargo usar solamente una cámara de alta resolución situada en un poste fijo, reduce considerablemente el coste, pero tiene la desventaja de obtener las matrículas de los vehículos desde distintos ángulos, afectando directamente a la escala de los caracteres, o la deformación de la matrícula.

Otros factores que pueden dificultar la obtención de la matrícula, pueden ser por ejemplo, que la matrícula esté sucia de barro o rota, e incluso, fuera de la vista de la cámara. También afectan los factores medioambientales como la iluminación, el desenfoque de movimiento, los reflejos, la niebla. Todos estas condiciones hacen que la obtención de la matrícula eficientemente sea una tarea ardua.

Aun así existen múltiples algoritmos para la detección de matrículas, como por ejemplo la detección de formas geométricas, pero esto puede ser un problema, debido a la gran cantidad de cosas que se puede interpretar como una forma rectangular, por esa razón se usan distintos algoritmos para eliminar aquellas regiones que no interesan.

Por lo tanto, estos algoritmos deben ser robustos para diferenciar entre matrículas y otros objetos en la imagen.

Varias investigaciones y estudios usan estos diferentes algoritmos:

➤ Extracción de matrículas a partir de detección de bordes: La detección de bordes es un método importante para la extracción de características. Aplicar este método a imágenes complejas es difícil, ya que puede dar como resultado un límite de objeto con curvas no conectadas así que, trabajan la imagen antes de aplicar este método, pasándola a una escala de grises en muchos de los casos.

---

Existen distintos algoritmos de detección de bordes como Canny, Canny-Deriche, Diferencial, Sobel, Prewitt y Roberts.



*Fig 3: Detección de bordes Canny.*

La idea principal tras estos métodos es usar la forma rectangular que suelen tener las matrículas, para aplicar estos métodos [3] se escalan las imágenes a una relación de aspecto fija.

En la investigación [4-5] evalúan y ponen a prueba varios algoritmos comparando los resultados con un conjunto de datos propio. Uno de los métodos de extracción de matrículas que se evalúa se basa en la información del borde vertical, detectando los bordes verticales mediante el operador Sobel. La matrícula se localiza comparando las longitudes máxima y mínima con las aristas extraídas y eliminando las no deseadas.

Usando esta metodología el porcentaje total de extracción para 140 imágenes fue del 65.25%, que era mucho más bajo que el 99% que se había reportado en el informe [6]. Sin embargo, usando la información [7] del histograma de los bordes horizontales y verticales para la extracción de las matrículas, se testeó en 50 imágenes con cambios en la iluminación dando como resultado una precisión de extracción del 90%.

VEDA [8] es el algoritmo más robusto de detección de bordes para la extracción de matrículas. La precisión para un rango de 50 imágenes con cambios de iluminación es del 96%. Los bordes horizontales pueden dar diversos errores, por ejemplo, el parachoques del coche, este algoritmo, usa el método de bloques, teniendo en cuenta los posibles números de matrícula, estos son los bloques que tienen la magnitud de borde alta. Este método es independiente a la detección de los bordes de la matrícula y puede aplicarse para identificar a partir de imágenes poco claras. Se logra un acierto del 92.5% para el reconocimiento de caracteres, usando para ello 180 imágenes.

- Extracción de matrículas a partir de binarización de la imagen: En el procesamiento de imágenes binarias, la imagen se escanea y cada uno de los píxeles se etiquetan en componentes, basados en la conectividad de los píxeles utilizando el Análisis de Componentes Conectados (CCA) [9-10]. Luego para la extracción de la matrícula se utilizan medidas especiales que suelen ser la relación de aspecto y el área [11,12].

Se propone un método de dos pasos para la extracción de la matrícula, [9] para hacer frente a las diferentes condiciones de iluminación. El primer paso usa el método de Otsu's Threshold que es un método eficaz y sencillo de umbralización adaptativa.



*Fig 4: Imagen convertida a método de Otsu. [9]*

La imagen binarizada detecta las formas rectangulares, en ellas se utiliza el método CCA. El segundo paso es hacer la detección de bordes con el método de la curva cerrada para asegurar que la imagen generada es una matrícula. Con esto se consiguió un 96% de aciertos.

- Extracción de matrículas a partir del color: Las placas de las matrículas suelen ser diferentes al color del vehículo. Existen regiones en el mundo en donde la identificación de la matrícula va por colores, por lo tanto también se han realizado distintas investigaciones para la extracción de la matrícula mediante colores.

La idea principal que envuelve este método es que las matrículas tienen un color específico, y el contraste que crea con las letras hace posible sacar el contorno.

En el caso propuesto [13], se usan técnicas de extracción de color mediante rangos de HSL y no mediante escalas de grises como los métodos anteriores, este modelo está desarrollado para trabajar con matrículas chinas, por esa razón solo se incluyeron los colores negro, verde, blanco y rojo.

Luego en otros modelos [14] se aplican algoritmos de *meanshift* para segmentar los colores en posibles regiones y luego poder etiquetar como zona de interés, por ejemplo para establecer si es una matrícula o si se descarta. Se logró un 97% de precisión.



*Fig 5: Resultado Meanshift. [14]*

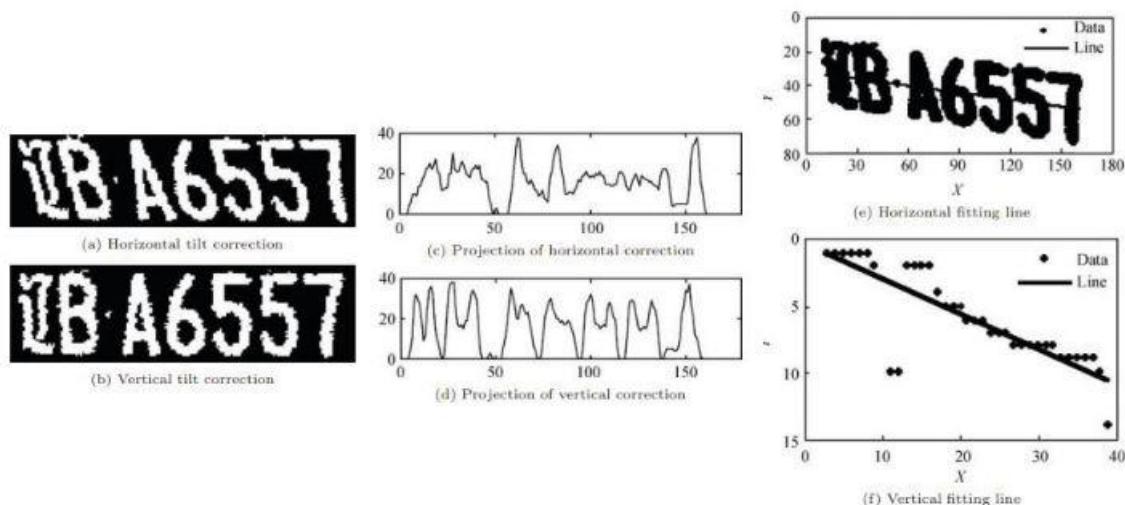
Este método de extracción de matrículas, tiene la ventaja de detectar matrículas que están deformadas, estropeadas o incluso inclinadas. Aun así también tiene desventajas como la necesidad de condiciones adecuadas de iluminación, ya que los modelos HLS son muy sensibles a estos cambios. También puede afectar el color de la carrocería, si es similar al de la matrícula.

- Extracción de matrículas basados en caracteres: Estos son métodos que detectan caracteres en la imagen y los localiza, considerando la región, como posible lugar de la placa. Para ello, se crea un método [15] que extrae las regiones similares a caracteres, en la imagen y las regiones extremas, suponiendo que pueden contener una placa de matrícula.

#### → **Segmentación de caracteres:**

La segmentación de la matrícula depende totalmente del éxito de la detección de la matrícula en la imagen. La placa aislada puede experimentar problemas como de contraste, condiciones variables de luminosidad o orientación de ángulos distintos.

Es por eso que se aplican técnicas de preprocessamiento dependiendo de las circunstancias de la matrícula, para luego poder hacer una segmentación de caracteres. Este paso se lleva a cabo bien en la fase de extracción o después de obtener una zona candidata aislada. Matrículas giradas se pueden arreglar usando mínimos cuadrados [16], que trata tanto las inclinaciones verticales como las horizontales. También se puede usar transformación bilineal [17] o métodos de ajuste de línea [18].



*Fig 6: Mínimos Cuadrados. [16]*

El segundo paso, es mejorar la imagen usando técnicas de eliminación de ruido, mejoras de contraste, ecualización del histograma y la transformación de la imagen a niveles de gris [19]. Facilitando la binarización de la imagen usando componentes conectados, [20] que permite etiquetar de forma sencilla los componentes con un 99.75%, estas son etiquetadas como imágenes binarias y se segmentan según el radio y el tamaño de los caracteres.

#### → Reconocimiento de caracteres:

La fase final de un ANPR es reconocer los caracteres segmentados. Estos pueden tener distintas formas o distintos tamaños debido a los zooms de la distancia de la cámara [21]. Los caracteres pueden estar alterados, girados o afectados por ruido.

Se han desarrollado distintos métodos para el reconocimiento de caracteres:

➤ Reconocimiento de caracteres por coincidencia de plantillas: El método más sencillo, es un método de correlación cruzada en la que la similitud del carácter extraído se compara con una plantilla de caracteres. Se elige el carácter que tenga mayor coincidencia dentro del conjunto de caracteres. Debido a los cambios de iluminación afecta directamente a la escala de grises, por esa razón estos métodos se usan en imágenes binarias.

Este método se ha usado para distintos modelos ANPR [22] con 1300 imágenes de un tamaño de 640x480, las imágenes están tomadas en condiciones adversas y logra un porcentaje de acierto del 92,12%, con una velocidad de procesamiento de 1.2s a 10 FPS

- Reconocimiento de caracteres mediante extracción de características: El método anterior envuelve cada uno de los píxeles, y requiere por tanto mayor tiempo de procesamiento. Este método [16] permite reducir ese tiempo de procesamiento eliminando los píxeles menos importantes, para ello se ha usado SVM [23], el vector de características, se produce mediante la proyección vertical y horizontal de los caracteres en binario. Cada una de las proyecciones se ve sometido a una cuantificación en cuatro niveles. Las transformaciones de cada matriz son emparejadas a una plantilla predefinida de caracteres.

Dentro del reconocimiento de caracteres se usan técnicas simples de procesamiento de imágenes, que requiere sobre todo de condiciones controladas, por ejemplo en la entrada o salida de un parking, donde el vehículo se encuentra detenido. Sin embargo existen sistemas más avanzados para el reconocimiento de caracteres, para sistemas ANPR con entornos no controlados.

### 2.1.2. Sistemas OCR

Dentro de los sistemas más avanzados de ANPR podemos encontrar que el reconocimiento de caracteres se realiza por OCR, desarrollado por primera vez en 1929 (patente de Gustav Tauschek), hoy en día esta tecnología ya logra un 99% de eficacia en determinadas condiciones de funcionamiento.

En el modelo propuesto implementando el OCR al hardware [24] se logra un 99,5% de acierto usando cámaras HD para 2790 caracteres testeados en tiempo real. Fue usado un algoritmo OCR para el reconocimiento de los caracteres de la matrícula.

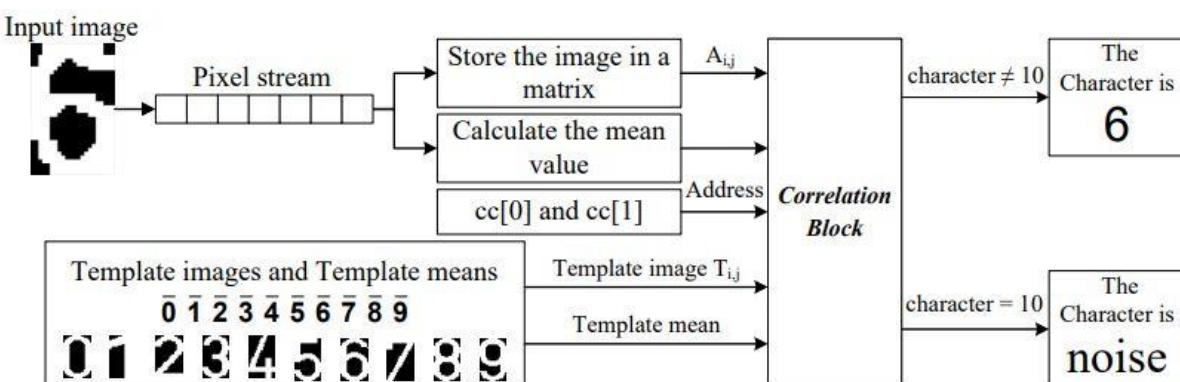


Fig 7: OCR Implementación Hardware. [24]

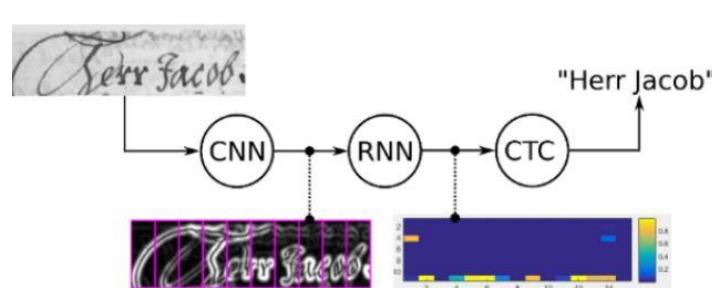
Viendo la eficacia de estos sistemas para el reconocimiento de caracteres de una matrícula, son algoritmos que suelen estar diseñados especialmente para el análisis de textos, extrayendo características geométricas de las letras y clasificándolas, aun así, hay otro

enfoque usando técnicas de inteligencia artificial y Machine Learning, de este concepto surgen distintos modelos OCR, algunos de ellos open Source como:

- GOCR: Es un sistema OCR bajo licencia pública, que convierte el texto de imágenes en archivos de texto.
- A9T9: También conocido como Free OCR para windows.

Aun así, los más empleados en sistemas ANPR son el Tesseract y el EasyOCR:

- Tesseract: Desarrollado entre 1984 y 1994, presentado en la prueba de OCR anual, logró unos buenos resultados, comparado con los modelos existentes en aquel momento. En 2015 Google tomó parte del proyecto y lo publicó como código abierto. Una de las características más sorprendentes es la capacidad de aprendizaje, tras someterlo a un proceso de entrenamiento, hoy en día Tesseract es el método más aceptado de OCR, capaz de reconocer más de 100 idiomas.  
Es un algoritmo que funciona paso por paso, empezando por un análisis de los componentes, para obtener los contornos de los caracteres, seguidamente se juntan los contornos cercanos en un mismo área para luego procesarlo, estas se organizan en líneas de texto y luego se descomponen en palabras. Luego se hace el primer intento de reconocimiento palabra por palabra y las que logre reconocer correctamente las introduce en el entrenador y hace una segunda pasada con ayuda del entrenador, finalmente extrae el texto digital.
- EasyOCR: Es un sistema OCR que usa Pytorch, [25] basado en el algoritmo de CRAFT para el reconocimiento de texto [26]. El modelo de reconocimiento es una red neuronal convolucional recurrente CRNN, que se forman de la combinación de dos redes neuronales importantes: la CNN seguida de la RNN. Compuesto por 3 componentes principales:
  - Extracción de características, usan actualmente ResNet
  - Etiquetado de secuencias (LSTM)
  - Decodificación CTC

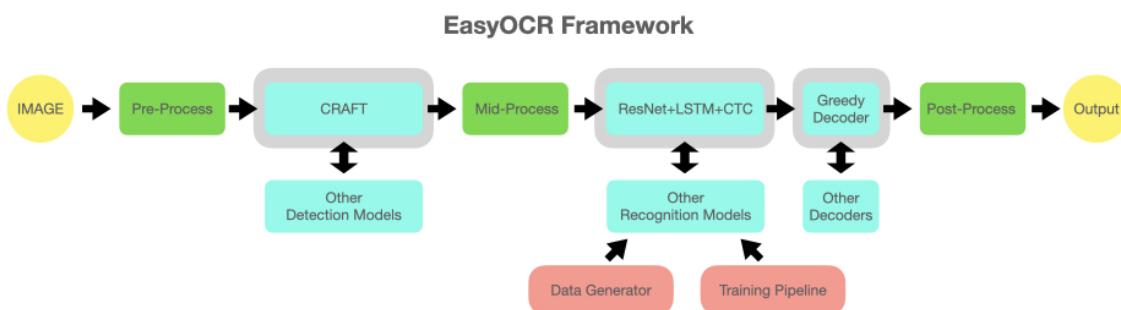


*Fig 8: Estructura general CRNN para reconocimiento de texto.*

Encontramos distintos parámetros importantes que se deben tener en cuenta para su correcto funcionamiento:

- “-image” : Imagen pre-procesada que contiene el texto para introducirlo al OCR.
- “-langs” : Códigos de idiomas, importante para reconocer distintos idiomas, al igual que Tesseract, por defecto se encuentra en inglés (en). Actualmente, EasyOCR es capaz de reconocer más de 80 idiomas diferentes
- “-gpu” : Algo que hace especial EasyOCR, es la capacidad de usar GPU o CPU. En caso de activar la opción GPU, usará CUDA y permite obtener resultados del OCR de forma más rápida.

La Arquitectura de esta red neuronal para el reconocimiento de texto se muestra más detallada en la *Fig 8*.



*Fig 9: Arquitectura EasyOCR.*

### 2.1.3. Sistemas Actuales Deep Learning para ANPR

En la actualidad, la tendencia es usar algoritmos de aprendizaje automatizado, en concreto aprendizaje profundo o Deep learning, basados en el uso de redes neuronales, para realizar la función de detectar objetos, varios estudios ya se han presentado para la detección de matrículas, dividiéndolos en dos grupos basándose en su enfoque:

1. Algoritmos basados en dos etapas: Están basados en clasificación, donde se extraen primero las características de interés y seleccionan la región de interés de la imagen. Seguidamente se clasifica la región para obtener la predicción final. Destaca la familia de algoritmos R-CNN [27]:
  - **R-CNN**: Es la primera versión de este algoritmo de detección de objetos que emplea una red de regiones propuestas, tiene poca precisión un 50% [28].
  - **Fast R-CNN**: Es una versión mejorada de la anterior que usa una red VGG16 que permitió mejorar la velocidad de entrenamiento y la precisión de la detección.

- **Faster R-CNN:** Es como la anterior una versión mejorada de Fast R-CNN que implementa una ayuda para predecir los límites de los objetos, aumentando la precisión.

- **Mask R-CNN:** Incorpora a Faster R-CNN una máscara de segmentación de imagen que ayuda a establecer el cuadro delimitador, consiguiendo un 50% más de precisión. Este algoritmo es implementado para la detección de matrículas [29].

2. Algoritmos basados en una etapa: Están basados en regresión, predicen clases y cuadros delimitadores:

- **SSD:** Este algoritmo establece puntuaciones a cada predicción de cada clase de todos los objetos, ajustando el cuadro delimitador a la que mejor se adapte a la forma, puede combinar predicciones de múltiples mapas de características, con distintas resoluciones. Es fácil de entrenar y obtiene buenos resultados de precisión [30].
- **YOLO:** es uno de los algoritmos más utilizados para la detección de objetos por su fiabilidad y precisión en las detecciones. Este algoritmo se ha ido actualizando y mejorando a lo largo de los años y es el preferido de muchos, presentada la primera versión en mayo del 2016 revolucionando el campo de visión artificial, aún así esta primera versión tenía limitaciones de resolución, no podía generalizar los objetos si tenía dimensiones distintas y tampoco detectaba objetos pequeños. En diciembre del mismo año sale la versión dos, corrigiendo muchos de esos errores, aumentando la precisión y la resolución de entrada a 416 x 416 píxeles, utilizando una arquitectura con Darknet-19. La tercera versión de YOLO salió en 2018, mejorando la velocidad y la precisión, debido al uso de la arquitectura Darknet-53 y predicciones de cajas, usando un sistema de extracción de características. Y mediante puntuaciones hace una predicción de las clases de cada objeto para establecer un cuadro delimitador.

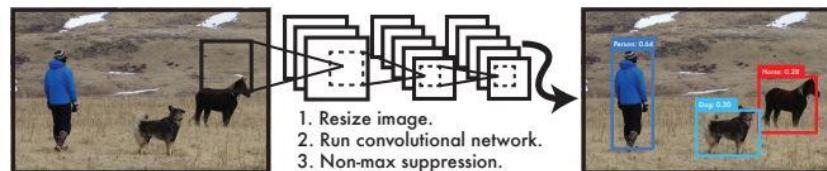


Fig 10: YOLO. [31]

La cuarta versión se presenta en 2020, superando a la anterior versión con creces, debido a que hace data augmentation al dataset para aumentar la cantidad de

datos en el entrenamiento, con mosaicos, etc. Además de permitir un entrenamiento con GPU.

Pocos meses después de YOLO v4 sale YOLO v5 con muchas ventajas al utilizar un framework de Pytorch, haciéndolo muy fácil de instalar y de entrenar. Con las distintas versiones se realizan varios estudios sobre ANPR basados en YOLO [32].

## 2.2. Marco Regulador

Uno de los sectores más amplios y necesitados de sistemas ANPR es el tráfico, con el objetivo de cumplir la ley y lograr una mayor seguridad frente a la conducción y en general en las vías públicas, se establecen distintas normativas para reducir los posibles accidentes, aún así hay mucha controversia en varios países sobre estos sistemas.

Por ejemplo, en países como Alemania, donde los vehículos son una parte fundamental de su cultura, no apoyan el uso de estos sistemas. En 2008, el tribunal constitucional federal de Alemania declaró que algunas leyes que permiten usar sistemas automatizados de reconocimiento de matrículas, violaban el derecho a la privacidad [33]. En concreto la retención de información, en este caso las matrículas.

Mientras que Inglaterra, pioneros en el uso de sistemas ANPR, han seguido desarrollando esta tecnología para fomentar la seguridad y ayudar a detectar la delincuencia, incluidos los grupos de crimen organizado y terroristas. Los datos de las matrículas se almacenan en el Centro Nacional de Datos ANPR durante dos años, y pueden ser usados como evidencia en investigaciones [34].

A continuación, se comentará la normativa legal en España sobre el uso de estos sistemas ANPR, la videovigilancia en carretera, el uso de cámaras, ley de protección de datos, etc.

### 2.2.1. Normativa Legal

En España hay distintas normativas para las distintas ideas que engloban los sistemas ANPR

#### **Normativa nacional sobre conducción automatizada**

Esta normativa se ha comenzado a establecer en varios Estados del mundo, no se tiene una referencia clara, debido a que se trata de algo bastante nuevo. Por lo tanto, en el caso de la Unión Europea se han ido elaborando para permitir la circulación de vehículos autónomos con el objetivo de hacer pruebas, por ejemplo en España. A su vez hay países como Alemania que han incluido en sus leyes internas el permiso de circulación de vehículos de los niveles 3, 4 y 5.

---

En España a pesar de formar parte de la Unión Europea no se encuentra vinculado al convenio de Viena que fue enmendado en 2014, pero sí al convenio de Ginebra que dice que el conductor es el que conduce el vehículo indicado en el Art 4. Este convenio además exige la presencia de un conductor en el vehículo, que deberá poder controlar el vehículo en todo momento señalado en los Arts 8 y 10 [35]. A pesar de no lograr los votos necesarios para enmendar el convenio en su momento, se permite que cada Estado pueda desarrollar su propia normativa en conducción autónoma según su jurisdicción.

A finales de 2015 la DGT aborda la cuestión mediante la Instrucción 15/V-113[36]. Donde se establecen las condiciones necesarias que debe tener el vehículo automatizado para poder realizar las pruebas de circulación en vías abiertas al tráfico, en este documento se establecen definiciones oficiales y tecnicismos sobre lo que es un vehículo autónomo. También indica sobre el tipo de responsabilidad civil ya que el seguro obligatorio no es el mismo que rige los vehículos convencionales.

### **Normativa nacional sobre la protección de datos**

La Ley Orgánica 3/2018, de 5 de diciembre, sobre la protección de datos personales [37] es un derecho fundamental protegido por el artículo 18.4 de la Constitución española, esta ley dictamina que limitará el uso de la informática para garantizar el honor, la intimidad personal y familiar de los ciudadanos y defender sus derechos.

### **Normativa nacional sobre la utilización de cámaras de videovigilancia**

Al tratarse de un proyecto donde la principal tarea es la detección de matrículas en entorno interurbano, nos lleva a informarnos sobre instalación y utilización de cámaras en zonas públicas.

Ley Orgánica 4/1997, de 4 de agosto [38] regula la utilización de videocámaras, por parte de los cuerpos de seguridad, el objetivo es preservar la convivencia ciudadana, el tratamiento automatizado de las imágenes y los sonidos no supondrá un perjuicio, a la Ley Orgánica 1/1982, de 5 de mayo [39] y por tanto se regirá por la Ley Orgánica 5/1992 de 29 octubre [40], de regulación del tratamiento automatizado de los datos de carácter personal.

Para la instalación de videocámaras de forma fija, deberá ser otorgada una autorización por la comunidad autónoma, cumpliendo los criterios presentados en el Artículo 4 [38], asegurar la protección de los edificios e instalaciones públicas, constar infracciones a la seguridad ciudadana y prevenir daños a personas y bienes.

En cuanto a la protección de espacios privados, hay alguna excepción para garantizar la seguridad, permitiendo la grabación de una porción de la vía pública, en caso de ser imprescindible [41].

## Normativa nacional DGT Radares

La DGT usa sistemas automatizados para los distintos sistemas de seguridad vial para preservar la seguridad de los ciudadanos y mantener las carreteras seguras.

Ley 18/2021, de 20 de diciembre [42] Prohíbe la instalación de inhibidores de radares o cualquier instrumento que sirva para interferir en el correcto funcionamiento del los sistemas de vigilancia del tráfico

### 2.2.2. Estandares Tecnicos

En este apartado se comentan los tipos de licencias que se han usado para el desarrollo del proyecto:

1. Python: Es usado para la programación de este proyecto, es un lenguaje orientado a objetos, de licencia de código abierto. [43]
2. OpenCV: Principal librería para visión computación, licencia para proyectos escolares BSD. [44]
3. Pytorch: Librería para Machine Learning y Deep Learning, licencia para proyectos escolares BSD. [45].
4. Weights and Biases: Licencia de estudiante. [46]
5. YOLO V5: Licencia GNU Public License, garantiza a los usuarios las cuatro libertades: ejecutar, estudiar, compartir y modificar el software. [47]
6. CUDA: Licencia GNU GPL v3. [48]
7. Visual Studio: editor de código de código fuente desarrollado por Microsoft, con licencia MIT.
8. Anaconda: Licencia pública GNU. [49]

### 3. Desarrollo e Implementación

Tras hacer un estudio de las soluciones planteadas en el estado del arte, es el momento de plantear un nuevo sistema para el reconocimiento de matrículas, en este apartado se expondrá los módulos que componen el desarrollo de este sistema.

#### 3.1. Análisis del problema

El título del tfg es el desarrollo de un detector de matrículas en entorno interurbano. Este trabajo está más centrado en la realización de un OCR tras la previa detección de la matrícula usando un modelo entrenado de Yolo v5 para realizar la detección de la matrícula.

La detección obtiene la matrícula y se aplican métodos de procesamiento de imagen para obtener una mejor imagen de la matrícula y seguidamente poder aplicar un OCR personalizado, pensado para el reconocimiento de los caracteres.

Hasta la fecha no se ha implementado ningún sistema ANPR usando YOLO v5 para hacer un OCR. En principio estos sistemas OCR, al ser implementados en los sistemas ANPR, consiguen buenos resultados. No obstante, no logran obtener un rendimiento favorable debido a que requiere de demasiados recursos y no cuentan con las ventajas que ofrece usar CUDA.

YOLO v5 es una tecnología de reconocimiento de objetos bastante nueva y poco trabajada, que ofrece unas prestaciones muy buenas según los estudios realizados, además es la más rápida y sencilla de entrenar, al estar implementada en Pytorch.

#### 3.2. Tecnologías empleadas

##### 3.2.1. Hardware

Al tratarse de un trabajo donde el entrenamiento de redes neuronales es una parte fundamental, este proceso de entrenamiento requiere bastantes recursos de ordenador, aun así gracias a los avances en este campo por parte de la empresa de Nvidia, se ha logrado poder reducir los tiempos de entrenamientos, gracias a su última gama de tarjetas gráficas.

Para la realización del proyecto, el hardware empleado ha sido un ordenador portátil MEDION Erazer P6689, con 8 gb de RAM ampliado a 16 gb, i7 de octava generación y gráfica GTX-1050 4 gb, usado para la programación del proyecto y para hacer las pruebas más simples. También se ha podido usar varios servidores proporcionados por la universidad Carlos III para el entrenamiento, con las siguientes especificaciones: Procesador AMD Ryzen 7 3700X 3.6GHz [53], con 32 Gb de RAM y gráfica RTX3060 con 12Gb.

### 3.2.2. Software

Para la realización del proyecto se ha utilizado tanto el sistema operativo de Windows 10 en un ordenador portátil, como la última versión estable de Ubuntu 20.04.4 para los servidores. En ambos casos se han instalado las herramientas necesarias para el funcionamiento de YOLOv5. Necesaria ha sido la última versión del toolkit CUDA [48] y los drivers de cuDNN para poder beneficiarse de las prestaciones que ofrece usar tarjetas gráficas de Nvidia.

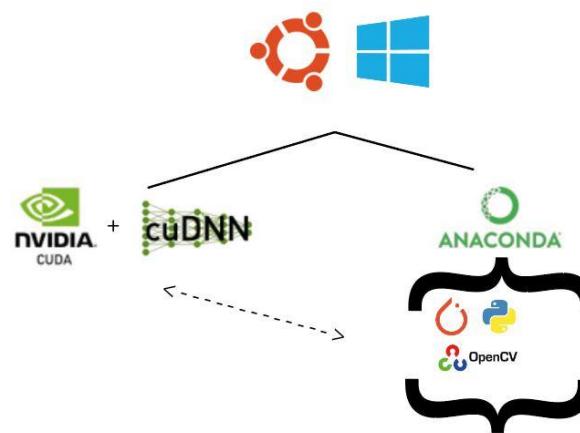


Fig 11: Arquitectura Software

Y para la gestión de las bibliotecas necesarias para la correcta ejecución de YOLOv5 se ha usado Anaconda por la facilidad de uso la interfaz, instalando todas las bibliotecas necesarias y Python como lenguaje de programación principal.

Por último se ha instalado también Pytorch para la compatibilidad con los repositorios necesarios.

- ❖ Pytorch: Es una librería Open Source basada en python, desarrollada para la realización de operaciones matemáticas mediante la programación de tensores, usado para Machine Learning y Deep Learning

También se han usado otras librerías como:

- ❖ OpenCV: Conjunto de bibliotecas para resolver los problemas de visión artificial, con esta librería se puede recortar imágenes, mejorar el brillo, contraste, segmentar imágenes y crear máscaras.
- ❖ numpy: Nos permite hacer operaciones matemáticas, una herramienta muy útil cuando se une con opencv, permitiendo crear vectores, y transformar los tensores en arrays.

---

Con estas librerías instaladas ya es posible usar todas las funciones que nos ofrece YOLO v5, aun así también requiere instalar los requisitos que establece en la página del repositorio Github, estos son el resto de librerías que usa para su funcionamiento.

Otros programas que se ha usado para el desarrollo del proyecto son los siguientes:

- ❖ Labelme: Es un programa de etiquetado de imágenes escrito en python, con una interfaz gráfica desarrollada en Qt creator, permite anotar imágenes mediante polígonos, rectángulos, círculos, etc.
- ❖ Visual Studio Code: Es un editor de código fuente.
- ❖ Weights and biases: Es una plataforma para Machine Learning para los desarrolladores, que ayuda a elaborar modelos de forma más rápida y cómoda. Permite un seguimiento de todas interpolaciones que se hagan con un dataset. También permite evaluar un modelo de entrenamiento y compararlo con otro para elegir el mejor.
- ❖ Anaconda: Es una distribución de Python que permite administrar paquetes y la instalación de librerías en entornos virtuales individuales, permitiendo el uso de esas dependencias de forma sencilla
- ❖ YOLO v5: Es un algoritmo de detección de objetos que divide las imágenes en un sistema de cuadrículas, donde en cada celda se detectan objetos.

### 3.3. Funcionamiento de YOLO v5

Una vez conocidas las tecnologías que hay que aplicar, es importante comprender el funcionamiento de YOLO v5. Esto hará más fácil la aplicación del mismo para usarlo en el proyecto.

Por lo tanto, para comprender YOLO v5 será de utilidad saber qué es YOLO. Este es un algoritmo de detección de objetos basado en regresión. Una vez introducidas las imágenes o videos a la red, YOLO completa la predicción de la clasificación y la información de la localización de los objetos según el cálculo de la función de pérdida.

En la *Fig 12* se ven las componentes de localización donde aparece la siguiente información:

- El centro del cuadrado ( $b_x, b_y$ )
- El ancho del cuadrado ( $b_w$ )
- El alto del cuadrado ( $b_h$ )
- La clase a la que pertenece el objeto ( $c$ )

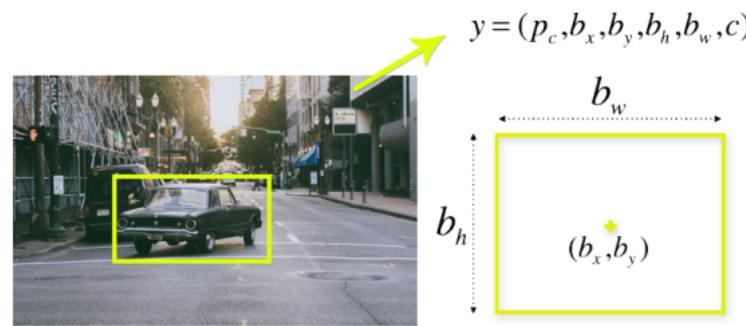


Fig 12: Ejemplo de caja limitadora YOLO.

Con estos parámetros se obtiene un valor de probabilidad ( $p_c$ ) que indica la confianza de que haya un objeto de dicha clase en la región definida.

YOLO usa diversas técnicas para ello:

1. Divide las imágenes en celdas
2. En cada celda, se usa la técnica de *sliding windows* en inglés que viene siendo, ventana deslizante [54] que busca los cuadros delimitadores de los objetos de las clases.
3. Estos comienzan a superponerse unos a otros, entonces utiliza la técnica de *non-max suppression*, que suprime todos los cuadros con baja probabilidad, dejando solo la probabilidad más alta. Hace la IoU obteniendo el cuadro delimitador con la probabilidad de clase más alta.

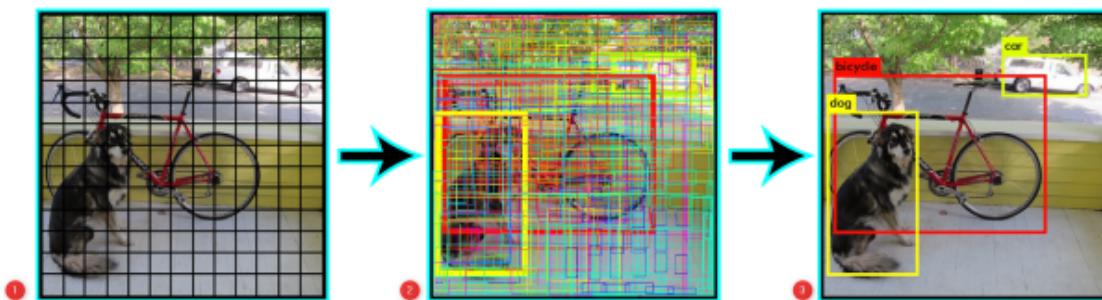


Fig 13: División de la imagen | Cuadros delimitado | Cuadro de mayor probabilidad.

YOLO v5 está basado en la arquitectura de detección de YOLO y utiliza una estrategia de optimización de algoritmos en el campo de CNN en los últimos años, como el autoaprendizaje de anclajes de cajas delimitadoras, aumento de datos en mosaico, redes parciales de etapas cruzadas. Estas son las encargadas para las distintas partes de su arquitectura.

## Arquitectura del Modelo

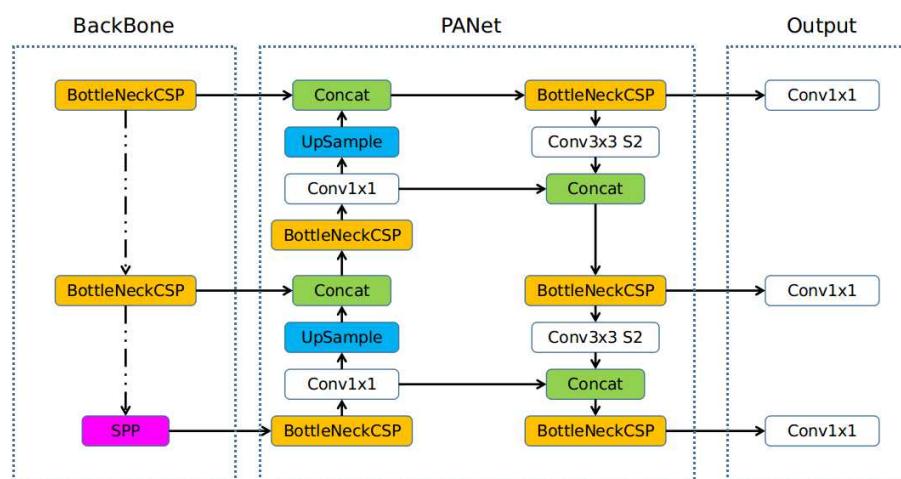
La arquitectura de YOLO v5 es un detector de objetos de solo una etapa, esto hace que tenga solo tres partes importantes más el *input* como se muestra en la *Fig 14*.

El *input* consiste principalmente en el preprocesamiento de datos, incluye además el aumento de los datos con mosaicos y el relleno adaptativo de imágenes. YOLO v5 para adaptarse a los diferentes conjuntos de datasets, integra un cálculo adaptativo del frame de anclaje, que se adapta a la entrada, para que de forma automática se ajuste al tamaño del frame inicial.

El *Backbone* se utiliza principalmente para la extracción de características importantes de la imagen de entrada. En YOLO v5 se usa como *Backbone* “CSP-Cross Stage Partial Network”. CSPNet ha mostrado una mejora en el tiempo de procesamiento con redes más profundas.

El *Neck* se utiliza para generar pirámides de características. Estas ayudan a los modelos a generalizar bien la escala de los objetos, ayudan a identificar el mismo objeto con diferentes tamaños y escalas. En YOLO v5, PANet es usado como *Neck* para obtener la pirámide de características.

El *Head* es la parte final de la detección, aplica cajas de anclaje a las características y genera vectores de salida con probabilidades del tipo de clase, además de puntuación de objetividad y un recuadro de delimitación. Este modelo es el mismo que se usa tanto para YOLO v3 como v4.





---

## Función de Optimización

Para la función de optimización tenemos dos opciones SGD y Adam. Para el entrenamiento por defecto usa SGD aunque puede ser cambiado a Adam como argumento.

## Función de Pérdidas

Para el cálculo de las pérdidas de probabilidad de la clase y para la puntuación del objeto se usa Binary Cross-Entropy con Logits Loss de la librería Pytorch.

### 3.4. Etiquetado OCR

Los modelos de YOLO v5, para ser entrenados, deben usar datos etiquetados, con el objetivo de aprender grupos de objetos de ellos, estos datos se llaman *datasets*. Existen múltiples bases de datos en internet, aunque para el propósito de este trabajo, no existía ninguno open source que se pudiera usar, así que, se optó por crear un Dataset personalizado, enfocado en el OCR, para tener una base de datos de todos los caracteres que están en uso en las matrículas españolas, un total de 36 caracteres, del 0-9 y de la A-Z sin contar la Ñ.

Al usar técnicas de Deep Learning, se necesita una gran cantidad de datos, para ello hay que recopilar una gran cantidad de imágenes, donde se puedan observar las características con las que queremos trabajar.

Por lo tanto para crear un *dataset*, hay que tener en cuenta algunos aspectos básicos:

- Las matrículas de los vehículos pueden sufrir variaciones, debido a la posición del vehículo, afectando al tamaño de los caracteres.
- También hay matrículas de distintos colores y también existen distintos estilos de matrículas que influyen en la colocación de los caracteres, estando en una o en varias líneas.
- Además, pueden afectar los cambios ambientales del entorno, como la iluminación o cambios meteorológicos como la lluvia, nieve, niebla, etc.
- Pueden estar colocados en ángulos y en condiciones complejas como suciedad, degradación, deformación, etc.

Teniendo en cuenta los distintos estados en los que se puede encontrar la matrícula, es importante que a la hora de recopilar las imágenes para crear el *dataset*, estos contengan gran cantidad de casos complejos, para luego poder entrenar con ellos y obtener los mejores resultados.

#### Crear Custom\_Dataset.yaml

En primer lugar, hay que crear el archivo donde se encontrará la configuración del *dataset* que usará YOLO v5 para entrenar, este archivo se organiza primero con la dirección donde se encuentra las imágenes etiquetadas ,luego el número de clases y por último una lista de las clases.

## Crear Etiquetas

Para crear etiquetas de imágenes, es necesario usar un programa, en este proyecto se usa el *labelme*. Con este programa, se etiquetó a mano un total de 2400 imágenes de vehículos, tomadas en distintas condiciones.

Se etiquetaba la matrícula y cada uno de los caracteres, usando polígonos y rectángulos respectivamente. Finalmente quedó un *dataset* pensado para la detección de la matrícula y otro para detectar caracteres.

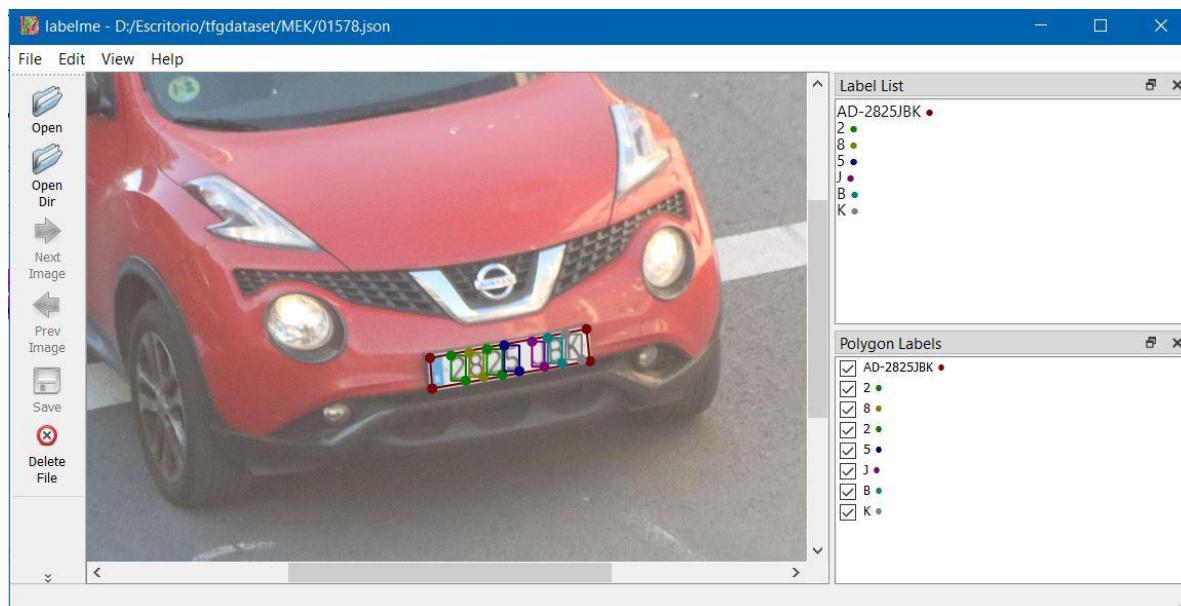


Fig 15: Etiquetado usando Labelme.

Con el proceso de etiquetado completado usando *labelme*, se encuentra en un formato con el que YOLO v5 no puede trabajar, así que se debe hacer una conversión del *dataset* para que lo pueda interpretar YOLO v5.



Fig 16: Representación de clases y cuadros delimitadores YOLO v5.

Para hacerlo hay que tener en cuenta el formato que necesita y el tipo de archivo que debe ser:

- Una fila por objeto.
- En cada fila el orden de los valores es: clase centro\_x centro\_y ancho altura
- Las coordenadas de la caja deben ser normalizadas, comprendidas entre 0 y 1. Siendo la esquina superior izquierda el (0,0), moviéndose hacia la derecha +X y hacia abajo +Y como se muestra en la *Fig 16*.
- La clase empieza en 0.
- El tipo de archivo es un \*.txt

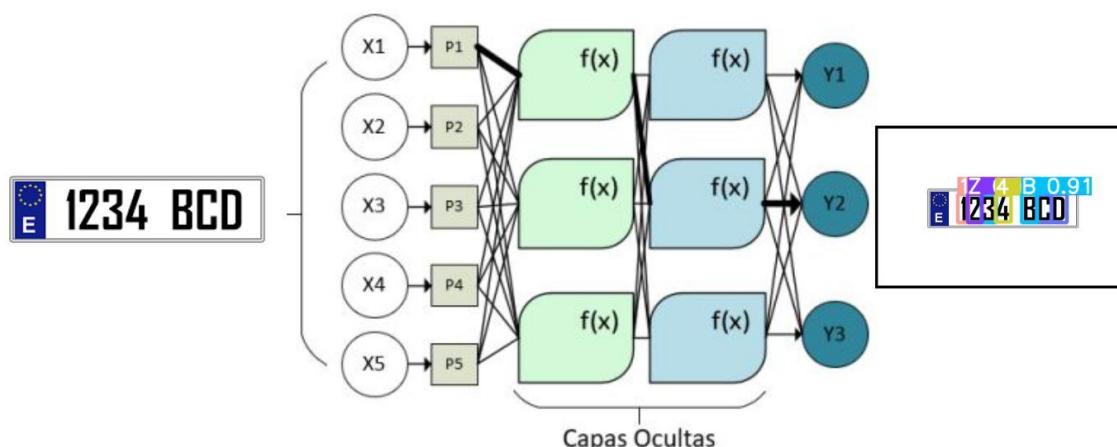
Comprendiendo el formato que debe seguir y tener las imágenes correctamente etiquetadas, es momento de entrenar YOLO v5.

Para ello, se divide de forma aleatoria el *dataset*. Para el entrenamiento se usan unas 1800, el 75% de las imágenes del *dataset* y para la validación unas 600 el 15%.

### 3.5. Entrenamiento YOLO v5

El entrenamiento de YOLO v5 consiste en adaptar los distintos pesos a la entrada de la red, de tal forma que las capas de salida se ajusten a los resultados a partir de los datos de los modelos proporcionados. Con el entrenamiento tratamos de dotar a la red neuronal de conocimiento para distinguir si el resultado de las operaciones son correctas, ajustando los valores de los pesos, la salida quedará mejor definida.

El posible resultado de la fase de entrenamiento tendrá una probabilidad sobre la certeza de la salida.



*Fig 17: Ejemplo de entrenamiento, con probabilidad de detección.*

Hacen falta muchas iteraciones para que la red neuronal sea capaz de identificar con precisión los caracteres, cuanto mayor sea el *dataset* con el que se entrena en la fase de entrenamiento se reducirá la probabilidad de error en el resultado.

En este proyecto se ha planteado la idea de usar *Transfer Learning*, para poder aprovechar las características aprendidas de un problema anteriormente resuelto, para evitar que los entrenamientos se alarguen demasiado. Esta técnica trata de usar una arquitectura de pesos que ya han sido entrenados a partir de un *dataset* con una gran cantidad de clases, en vez de utilizar pesos aleatorios

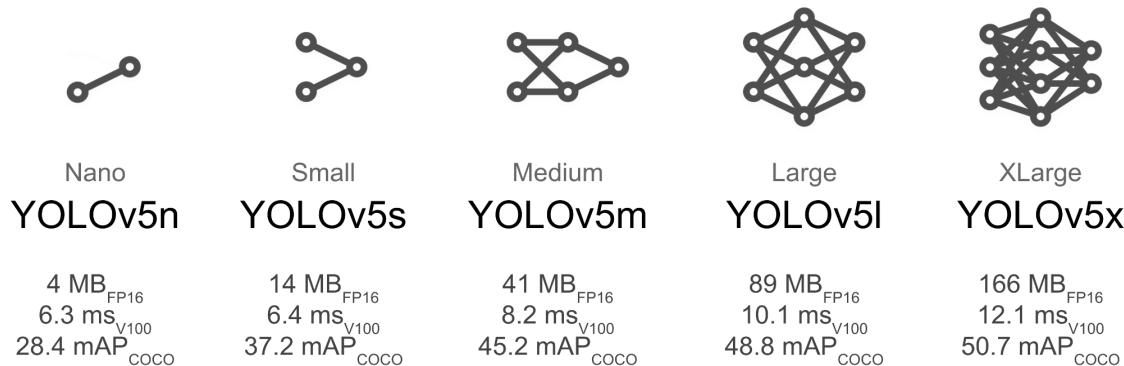


Fig 18: Modelos ya entrenados de YOLO V5.

YOLO v5 consta de cuatro modelos ya pre-entrenados YOLOv5s, YOLOv5m, YOLOv5l y YOLOv5x con el dataset de COCO [55], la diferencia entre ellos es el número de módulos de extracción de características y de núcleos de convolución en lugares específicos de la red.

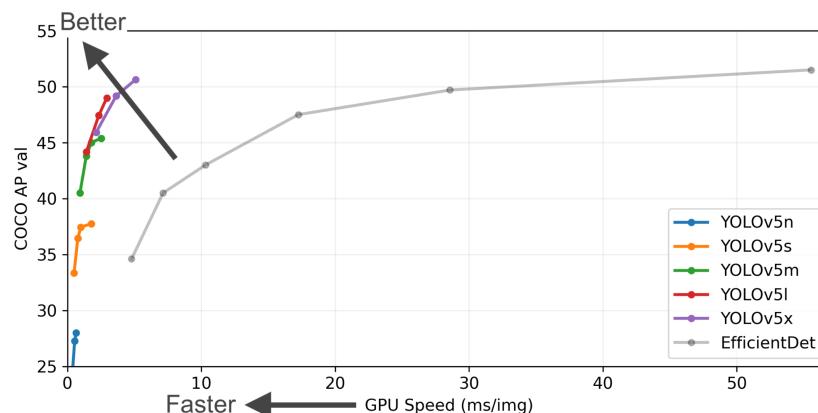


Fig 19: Comparación de YOLO V5 con EfficientDet.

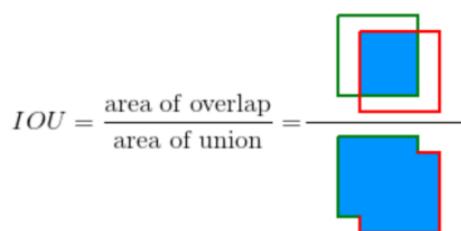
Model	size (pixels)	mAP <sup>val</sup> 0.5:0.95	mAP <sup>test</sup> 0.5:0.95	mAP <sup>val</sup> 0.5	Speed V100 (ms)	params (M)	FLOPs 640 (B)
YOLOv5s	640	36.7	36.7	55.4	2.0	7.3	17.0
YOLOv5m	640	44.5	44.5	63.1	2.7	21.4	51.3
YOLOv5l	640	48.2	48.2	66.9	3.8	47.0	115.4
YOLOv5x	640	50.4	50.4	68.8	6.1	87.7	218.8

Fig 20: Resumen de características de los modelos de YOLO V5.

Podemos encontrar tres columnas en la *Fig 20*, donde pone *mAP* (mean Average Precision) esta es la manera de cuantificar la precisión de un detector.

Para entender esto mejor se van a definir los siguientes conceptos:

- IoU: La intersección sobre unión, evalúa el solapamiento entre dos cuadros delimitadores.



*Fig 21: Intersection over union.*

Al ser un problema de clasificación se dan cuatro situaciones en cuanto a los resultados del algoritmo.

		Actual	
		Positive	Negative
Predicted	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

- Verdadero Positivo:  $\text{IoU} > \text{umbral}$
- Falso Positivo:  $\text{IoU} < \text{umbral}$
- Falso Negativo: Objeto no detectado
- Verdadero Negativo

*Fig 22: Matriz de confusión.*

- Precisión : Es el porcentaje de las predicciones que son correctas, esto se calcula:

$$\text{Precisión} = \frac{\text{Verdadero Positivo}}{\text{Verdadero Positivo} + \text{Falso Positivo}}$$

- Recall: Esto mide lo bien que encuentra los positivos reales.

$$\text{Recall} = \frac{\text{Verdadero Positivo}}{\text{Verdadero Positivo} + \text{Falso Negativo}}$$

Para determinar por tanto el valor de *mAP* durante el proceso de entrenamiento, se hace el promedio del Recall, como práctica habitual se pone como *mAP@0.5*, que significa que es la media ponderada con un IoU del 50% mientras que *mAP@0.5:0.95* indica que se han utilizado un rango de umbrales del 0,5 al 0,95 incrementeando con un paso del 0,05.

Tras hacer pruebas con los modelos pre-entrenados con los valores por defecto y usando para ello el dataset COCO.

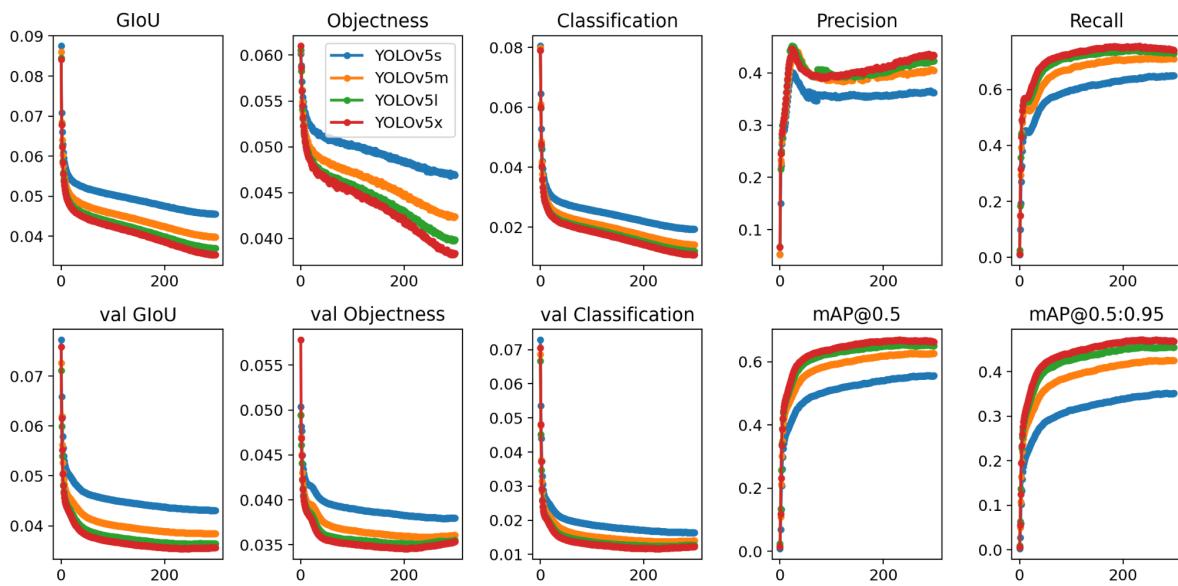


Fig 23: Resultados de detección de objetos con dataset en COCO. [55]

Se observa en la Fig 23, que usando YOLOv5s se logran resultados bastante aceptables en cuanto a rendimiento y velocidad de detección.

Teniendo claro las prestaciones, se usarán para el entrenamiento los servidores de la universidad, debido a que ofrece los recursos computacionales necesarios para lograr un entrenamiento efectivo.

Para empezar el entrenamiento inicial, se utilizan los hiperparametros por defecto y se hacen pruebas con distintos números de epochs y de batch-size con un tamaño de imagen de 640 píxeles, con esta resolución se consigue una detección correcta y un entrenamiento rápido.

El número de epochs es la cantidad de veces que el algoritmo de aprendizaje trabaja con el dataset de entrenamiento, y el batch-size es la cantidad de imágenes con las que trabaja el algoritmo a la vez.

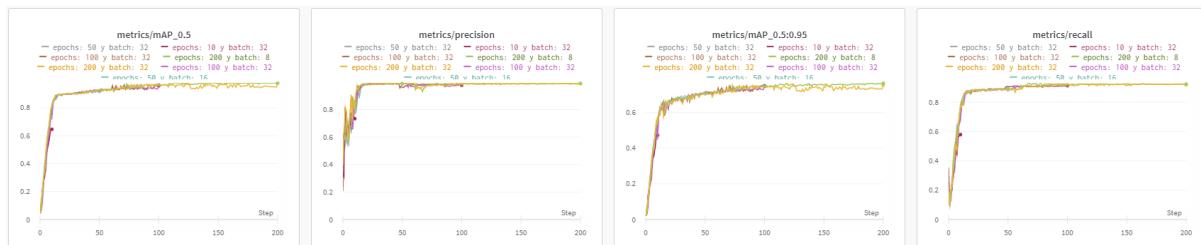


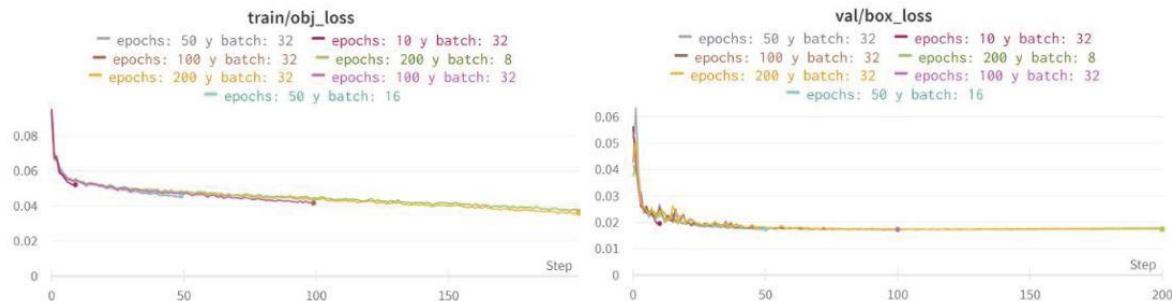
Fig 24: Gráficas métricas Weight and Biases.

En la Fig 24, se muestran distintos entrenamientos con valores de epochs y batch-size distintos, esto se hace para buscar el límite de overfitting o sobreentrenamiento. Aumentando

---

el número de epochs afinamos el entrenamiento y conseguimos los pesos del modelo que mejor resultado pueden dar.

Se puede observar también, en la siguiente *Fig 25*, las curvas de aprendizaje del modelo, que sirven para poder visualizar el rendimiento del aprendizaje, en cada época, es útil para revisar si durante el entrenamiento ha ocurrido algún problema como un sobreajuste.



*Fig 25: Valores de pérdida Train y Val*

Con estos valores de pérdida de validación y de entrenamiento, se puede cuantificar cómo de bien está prediciendo el modelo, en este caso se puede observar que es más fácil para el modelo hacer la validación debido al data augmentation que se produce en el entrenamiento, esto consiste en aumentar los datos para el entrenamiento, ayudando al modelo a distinguir las clases.



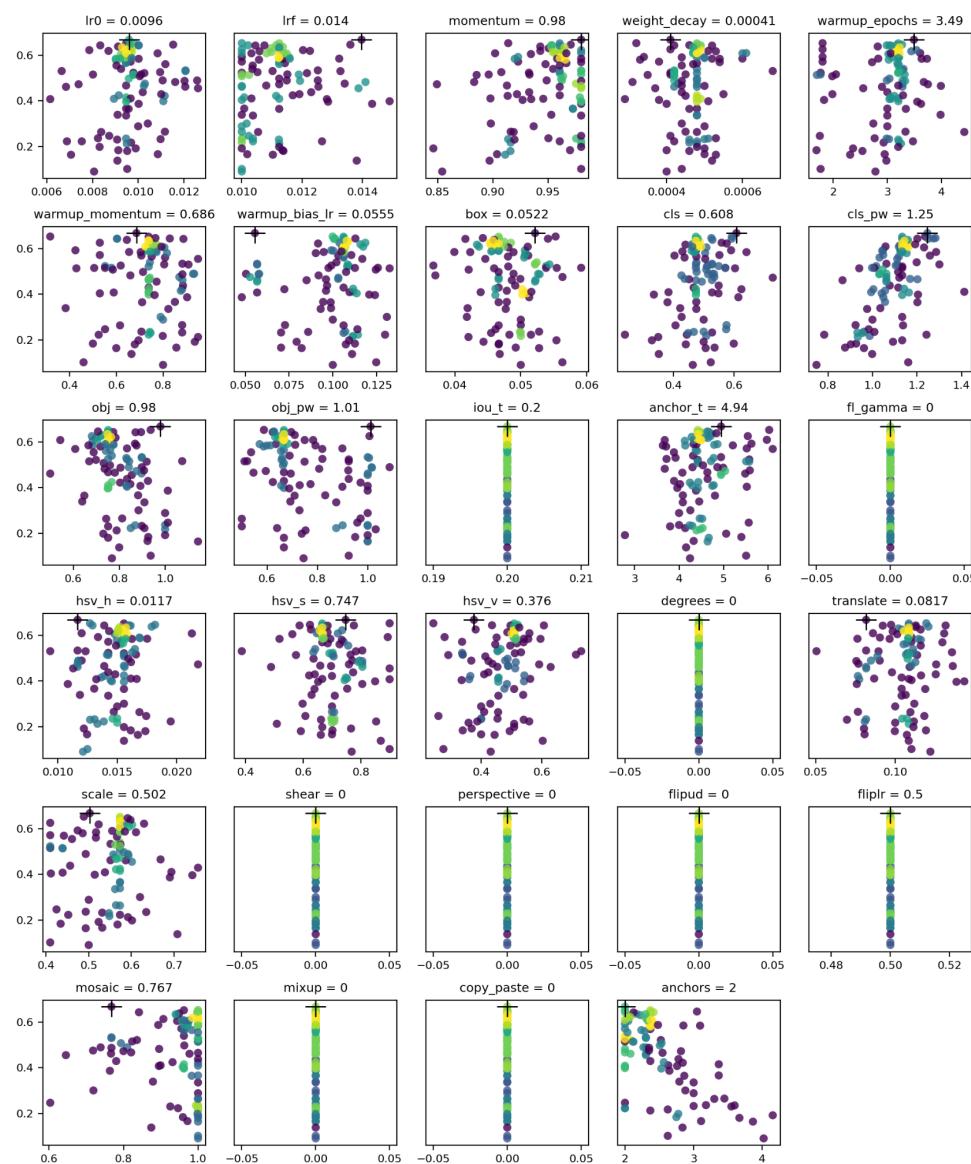
*Fig 26: data augmentation mosaico.*

De las gráficas anteriores se puede observar, que el entrenamiento llega a un punto donde se queda estancado y parece no aprender más, observando que el mejor resultado se da con 100 epochs y 32 de batch-size.

Ahora se propone el modificar los hiperparametros del modelo para ver si se puede mejorar el entrenamiento con 100 epochs y 32 de batch. Los hiperparametros son parámetros que se utilizan para controlar el proceso de aprendizaje, el objetivo es encontrar los parámetros que mejoren este proceso, encontrar estos valores puede ser un reto en muchos casos, aun así YOLO v5 permite cambiar estos 25 hiperparametros de forma automática usando la opción de *evolve* que va mutando los valores que mejores resultados den.

Este proceso de *evolve* lleva bastante tiempo de entrenamiento y requiere de bastantes recursos de ordenador ya que van cambiando estos parámetros, en cada generación.

Una vez que termina se obtiene un archivo *\*.yaml* con la mejor generación y una representación gráfica de los cambios que ha ido haciendo y cómo ha ido mutando, como se muestra en la *Fig 27* siguiente.



*Fig 27: Evolve Hiperparametros.*

Se hizo un evolve de 100 generaciones a 10 epochs y 32 de batch-size, tras obtener la mejor generación de entrenamiento, se entrena con esos parámetros a 100 epochs y 32 de batch-size, además se hizo un segundo entrenamiento con estos parámetros cambiando un parámetro manualmente del archivo `*.yaml` en concreto el `fliplr` esto indica la probabilidad con la que la imagen se volteea a la izquierda o a la derecha, se establece a cero porque es poco probable encontrar caracteres volteados. Para comparar los resultados de entrenamiento se usa *Weights and Biases* en la Fig 28.

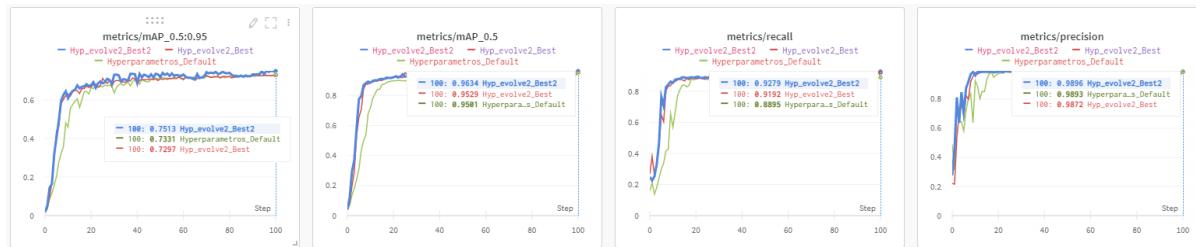


Fig 28: Hiperparametros cambiados + en Default.

En la figura se muestran las métricas de los tres entrenamientos realizados con los distintos valores de los hyperparametros, el mejor modelo servirá para el desarrollo del OCR.

### 3.6. Reconocimiento de caracteres

La segmentación de los caracteres y el reconocimiento, se logra con el entrenamiento de la red de YOLO v5, con el *dataset* creado de los números y caracteres de las matrículas etiquetadas. Este modelo entrenado se aplica en la entrada de la matrícula recortada y rectificada.



Fig 29: Detección y rectificación de matrícula.

---

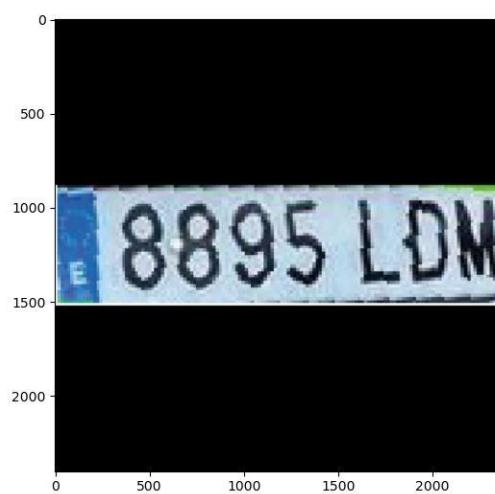
Por lo tanto, se recibe como *input* del programa, la imagen de la matrícula rectificada como se muestra en la *Fig 29*. Esta imagen se introduce al programa como lista de matrículas detectadas, de forma independiente y se utiliza la clase *DataLoader* de Pytorch, esta es la clase principal para la carga de datos, y se pasa por argumento un objeto *Dataset*.

Con este argumento *Dataset*, se puede introducir las imágenes de las matrículas y se les puede realizar transformaciones, usando funciones de Pytorch, como la normalización de la imagen, el escalado, etc. Aunque tratar de escalar la imagen a por ejemplo 640 x 640 provoca que se estire la imagen, haciendo que la detección de los caracteres sea incorrecta en varias ocasiones.



*Fig 30: Matrícula Escalada.*

La solución a este problema es llenar la zona de la escala de valores en cero, haciendo un padding o relleno.



*Fig 31: Padding de Matrícula.*

---

El *DataLoader*, puede definir el orden y la forma que se cargan los datos, además se carga un *batch* o un lote de imágenes que se cargan en RAM, y se pueden realizar en uno o varios procesos, donde ubica los tensores en memoria para facilitar la transferencia a la GPU.

Las imágenes de las matrículas se introducen en el modelo entrenado usando los pesos que mejores resultados habían obtenido, este se encargará de detectar los caracteres de forma independiente



*Fig 32: Reconocimiento de caracteres segmentados.*

La salida de la entrada de la imagen en el modelo, es una matriz de tensores que se puede separar en *labels* y en *coordenadas*. Los 4 primeros valores corresponden a las coordenadas  $x1, y1, x2$  y  $y2$  de los cuadros delimitadores y el último valor corresponde a la confianza de la predicción de la etiqueta.

```
labels: tensor([13.,  8.,  8., 22.,  9., 21.,  5.], device='cuda:0')
coordenadas: tensor([[0.73510, 0.10984, 0.83970, 0.61135, 0.91944],
[0.27025, 0.29367, 0.37409, 0.80119, 0.91703],
[0.15156, 0.32690, 0.26385, 0.85470, 0.90995],
[0.83298, 0.06554, 0.94967, 0.58006, 0.90943],
[0.37971, 0.25066, 0.48467, 0.75459, 0.90849],
[0.63135, 0.15022, 0.74595, 0.65573, 0.90777],
[0.48728, 0.20415, 0.59090, 0.71220, 0.90702]], device='cuda:0')
```

*Fig 33: Labels y Coordenadas.*

cada label corresponde a un carácter:

```
Label: tensor(13., device='cuda:0') Corresponde al carácter: D
Label: tensor(8., device='cuda:0') Corresponde al carácter: 8
Label: tensor(8., device='cuda:0') Corresponde al carácter: 8
Label: tensor(22., device='cuda:0') Corresponde al carácter: M
Label: tensor(9., device='cuda:0') Corresponde al carácter: 9
Label: tensor(21., device='cuda:0') Corresponde al carácter: L
Label: tensor(5., device='cuda:0') Corresponde al carácter: 5
```

*Fig 34: Label correspondiente al carácter.*

Ahora el problema está, en que los caracteres que ha detectado se encuentran desordenados como se muestra en la *Fig 34*. Por lo tanto, hay que ordenar los caracteres para que correspondan con la matrícula que se trata de detectar, para ello se usan las coordenadas de los cuadros delimitadores para saber la posición de cada carácter y se ordenan los caracteres de izquierda a derecha teniendo en cuenta primero los caracteres que se encuentran más cerca del  $(0,0)$  para poder hacer los casos donde las matrículas tienen un formato de más de una línea, como se muestra en la *Fig 35*. Que se consigue aumentando el margen de las alturas de cada uno de los caracteres detectados.



*Fig 35: Matrículas de ciclomotor / motocicleta.*

Finalmente se pueden unir todos los caracteres ordenados en un *String* e imprimir por pantalla el resultado de la matrícula.



*Fig 36: Matrícula Detectada.*

## 4. Resultados

En este apartado se describirán las pruebas realizadas con el método propuesto y los resultados, se dividen en dos partes:

- La primera parte, consiste en mostrar los distintos resultados de los entrenamientos, utilizando el *dataset* creado.
- La segunda parte, los resultados del reconocimiento de caracteres de la matrícula, sometiéndose a distintas pruebas.

### 4.1. Resultados obtenidos del entrenamiento de YOLO v5

La ejecución del código con el modelo entrenado usando el dataset creado, da como resultado la siguiente tabla de métricas:

TABLA 4.1.1 - CUADRO DE METRICAS BestEntrenamiento.pt				
Entrenamiento	Precisión	Recall	mAP@0.5	F1-score
Default	0.9893	0.8895	0.9501	0.9367
Hyp Evolve	0.9872	0.9192	0.9529	0.9519
Hyp Evolve 2	0.9896	0.9279	0.9634	0.9577

En la *Tabla 4.1.1* se ven las métricas del entrenamiento de este modelo, para las tres pruebas, cambiando los hiperparámetros, y con los parámetros por defecto. Comparando el *F1-score* calculado con la fórmula:

$$F1 = 2 * \frac{P * R}{P + R}$$

donde el valor más cercano a 1 es mejor.

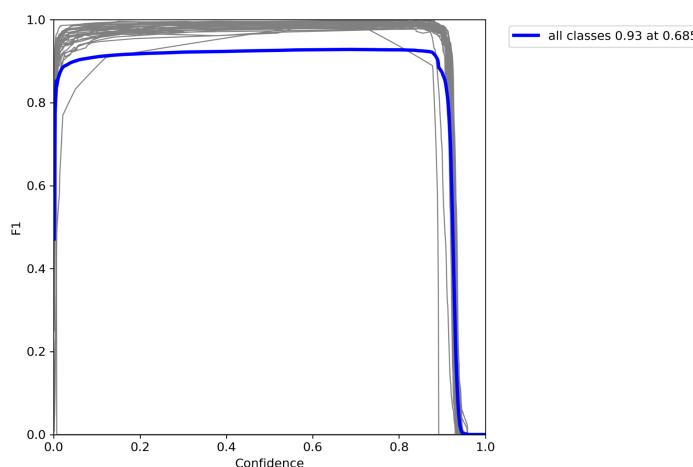
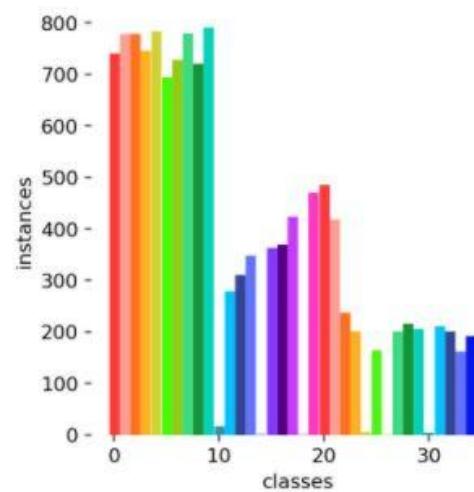


Fig 37: Gráfica de los valores F1-score y confianza de las clases de Hyp Evolve 2.

En la gráfica se pueden ver valores de la métrica y depende del valor de la confianza. Cada vez que el valor de confianza aumenta, el valor de F1 va aumentando hasta llegar a 0.5 y 0.6, cuyos valores son los que mejores resultados dan y luego vuelve a disminuir.

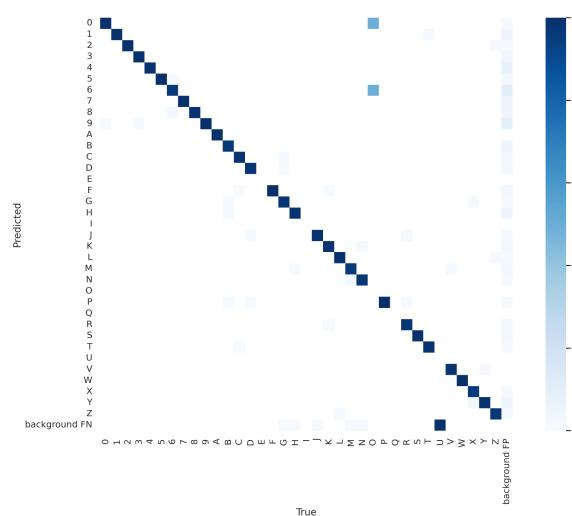
Analizando los resultados, se observa que los tres son bastante buenos, pero el que mejor resultado da es *Hyp Evolve 2*. A partir de ahora se hace referencia a este modelo, el cual en el entrenamiento nos ofrece información con respecto al *dataset*. Sirve para conocer en más detalle el proceso de entrenamiento y así pensar en cómo mejorar el *dataset* y a su vez el entrenamiento, para lograr una mejor detección.

Por lo tanto analizaremos la cantidad de etiquetas del dataset, representado en forma de gráfica, donde encontramos las 36 clases de los caracteres y la frecuencia absoluta.



*Fig 38: Gráfica Instancias / clases*

Donde se puede ver que algunos caracteres y n umeros escasean debido a la poca cantidad de ellas en el *dataset*. Tambi n el entrenamiento genera la siguiente matriz de confusi n:



*Fig 39: Matriz de confusión entrenamiento*

En la matriz de confusión, cada columna representa el número de predicciones de cada clase, mientras que en cada fila se representa las instancias en la clase real. Esto nos muestra los tipos de aciertos y errores que está teniendo el modelo en el momento de aprendizaje con los datos.

Se nota la escasez en algunas vocales donde el color de umbral cambia, como en la *A*, también se observa, por ejemplo que la *O* se parecen mucho al *0*, mostrando una clara falta de umbral en la posición. En cuanto a la *Q*, no existen matrículas con ese carácter y por tanto es un error al crear el *dataset* y no descarto como en el caso de la *N̄*.

Para mejorar por tanto el entrenamiento, se tendría que mejorar el *dataset* primero, completándolo con más datos, aumentando de esa forma la calidad, haciéndolo mejor para entrenar.

## 4.2. Resultados obtenidos en el reconocimiento de caracteres

Una vez tenemos el mejor modelo de entrenamiento escogido, es momento de hacer pruebas con distintos tipos de matrículas, para ello se usarán las matrículas obtenidas de la compra, ya que el modelo no ha sido ni entrenando ni validado con ellas y aporta por tanto imágenes nuevas para testear el modelo.

Para hacer las pruebas se introduce un dataset de imágenes, escogidas de forma aleatoria y algunas a propósito, con condiciones adversas como por ejemplo, deformación de la matrícula, iluminación complicada, etc. Y además se introducen al modelo distintas categorías de matrículas.

Comenzando con las matrículas que siguen el formato de hoy en dia:

TABLA 4.2.1 - MATRÍCULAS CONVENCIONALES		
Entrada	Salida por Terminal	Observaciones
	8844JSD	Detección <u>correcta</u> a pesar de la sombra y el parecido de la <i>D</i> con un <i>0</i> .
	0031KLZ	Detección <u>correcta</u> , a pesar de la sombra
	8133KXW	Detección <u>correcta</u>

	1346KZF	Detecta <u>correcta</u> , con deformación de la matrícula
	6739KNY	La <i>W</i> la interpretó por una <i>Y</i> debido a la sobreexposición de la letra en cuestión
	5858GFN	Detección <u>correcta</u> a pesar de sombra y el parecido de la <i>G</i> a un <i>O</i>

Tras realizar un total de 20 pruebas con matrículas convencionales, se puede observar que la detección es precisa al 95%, esto se debe a que en algunos casos no reconoce bien los caracteres por una sobreexposición de iluminación.

A su vez, podemos encontrar distintas matrículas hoy en día. Por ejemplo las que tienen distintos colores de fondo, como las que llevan taxis o vehículos de Uber, o los vehículos con seguros temporales.

TABLA 4.2.2 - MATRÍCULAS DE VEHÍCULOS ESPECIALES		
Entrada	Salida por Terminal	Observaciones
	7648JCY	Matrícula detectada de forma <u>correcta</u>
	5469JSG	Detección <u>Correcta</u>
	CD3248	No detecto el <i>0</i> threshold demasiado alto.
	HR8261BBT	Detecta la <i>R</i> como <i>H</i> y <i>R</i> Error en la IoU
	9560G6	No detecto correctamente la <i>C</i> ni el <i>I</i>

En este tipo de matrículas con distintos fondos de colores parece tener más problemas el modelo para predecir los caracteres, al menos en los casos de matrículas rojas y verdes. Sin embargo en matrículas azules sí que detecta con más precisión.

En cuanto a matrículas de vehículos que siguen el formato previo al 2000 como por ejemplo vehículos de coches clásicos, las pruebas realizadas son las siguientes:

TABLA 4.2.3 - MATRÍCULAS FORMATO PREVIO AL 2000		
Entrada	Salida por Terminal	Observaciones
AB·0032·T	AB0032T	Detección <u>correcta</u>
GC·1092·AW	GC1092AW	Detección <u>correcta</u>
M 4733 UJ	M4733J	No detectó la <i>U</i>
E M 3284 NU	M3284N	No detectó la <i>U</i>
CC·4380·M	CC4380M	Detección <u>correcta</u>

En cuanto a matrículas de este tipo, las vocales son las que más escasean en el dataset ya que se creó usando sobre todo matrículas convencionales y hay que tener en cuenta que ya no se usan vocales en las matrículas, por lo tanto, se obtiene una detección errónea en casos donde aparecen *U*, *O* que las interpreta como un *O*, aun así se observa que en matrículas donde hay una *U* y se muestra de forma clara, no parece reconocerla y tampoco la confunde. Aun así logra obtener la matrícula en la mayoría de los casos si la imagen que entra es nítida.

Por último, las matrículas con distintas formas rectangulares y con varias líneas de caracteres como se da el caso en ciclomotores y motocicletas *Fig 32*.

**TABLA 4.2.4 - MATRÍCULAS MOTOCICLETAS Y CICLOMOTORES**

Entrada	Salida por Terminal	Observaciones
	0425HTK	Detección <u>correcta</u> de la matrícula de la motocicleta
	3715BPK	Detección <u>correcta</u>
	BF2366	Error en la detección, umbral de reconocimiento alto por eso no detecta la <i>A</i> , confunde la <i>E</i> por <i>F</i>
	3504KRP	Detección <u>correcta</u> , con desenfoque
	C2790BLB	Detección <u>correcta</u>
	C7567BRS	Detección <u>correcta</u>
	C684BVG	Matricula ciclomotor, no detectada correctamente, falta el <i>O</i> y la <i>W</i> cambiada por <i>V</i>

Como se ha comentado anteriormente, las matrículas con vocales como la *A* o la *E*, las confunde o no las detecta en algunos casos. Aun así en el caso de las matrículas de ciclomotor, en el momento del etiquetado de las imágenes para la creación del *dataset* únicamente se etiquetaron con este formato un par de matrículas, y por esa razón se obtiene algún error, pero a pesar de ello logra un 90% de reconocimiento.

### 4.3. Análisis de los resultados

Después de observar los resultados obtenidos tras las pruebas realizadas, podemos llegar a la conclusión de una necesidad de aumentar el *dataset* del OCR, con una gran cantidad de vocales (*A*, *E*, *U*) para los casos con matrículas previas al 2000, en cuanto a los caracteres *I* y *O* son complicadas de diferenciar entre *I* y *O* respectivamente.

Además, habría que aumentar la variedad de matrículas de distintos colores, ya que cuando había matrículas rojas o verdes, la predicción en muchos de los casos era errónea, incluso teniendo caracteres que ya de por sí eran comunes en el *dataset*.

Sin embargo, el sistema OCR planteado es capaz de ordenar los caracteres detectados de la matrícula e imprimirlos por pantalla de forma correcta incluyendo las que tienen matrículas con saltos de línea. También es capaz de ordenar la matrícula con un margen de ángulo en la rectificación.

## 5. Trabajos futuros y Conclusión

### 5.1. Trabajos futuros

Tras completar el proyecto, como trabajo futuro se plantea la optimización y la mejora de la implementación, aumentando la tipología de matrículas.

Aun así, al comienzo de este trabajo se planteó la idea de utilizar este detector de matrículas para identificar vehículos usando una cámara Full HD [56], pensada para tráfico y hacer un mapeo 3D del entorno, mediante un lidar, para ello se tendría que hacer la instalación de un poste en una zona interurbana con la cámara y el lidar. Con el propósito de poder comunicar la infraestructura con los vehículos, para dar información de un entorno más amplio que lo que un coche autónomo podría obtener con tan solo las cámaras instaladas en el mismo.

Tambien salio la idea de hacer un etiquetado automático usando los modelos entrenados de la detección de matrículas y el OCR de este trabajo, ya que sería útil para mejorar el *dataset*, porque la cantidad inicial de imágenes eran 2400, de esas 2400 imágenes se obtuvieron gran cantidad de etiquetas de caracteres, pero como se ha podido observar, no son suficientes. La idea era introducir las 27.000 imágenes obtenidas mediante la compra [50], que incluyen un archivo XML donde se encuentra la matrícula, introduciendo estas imágenes en el modelo se obtendría la predicción de cada imagen y en un fichero \*.txt se escribirían los caracteres detectados siguiendo el formato de entrenamiento de YOLO v5 y luego la predicción de la matrícula se compararía con el XML. Las matrículas detectadas correctamente se separarían de las matrículas detectadas incorrectamente y se tendría que volver a etiquetar de forma manual, aun así la cantidad de etiquetas manuales se reduciría considerablemente usando este método.

Otro trabajo futuro, puede ser el desarrollo de este sistema usando el nuevo algoritmo de detección del 2021 YOLOR *You Only Learn One Representation* que se define como una red que codifica conocimiento implícito y explícito junto a la detección de objetos.

Finalmente quedaría la implementación de este sistema ANPR en un caso real, para hacer una validación más correcta.

## 5.2. Conclusión

En este proyecto, se ha presentado un sistema de reconocimiento de matrículas, aun así el trabajo se ha enfocado más en la parte del sistema OCR para el reconocimiento de caracteres de las matrículas, con la creación de un *dataset* propio y utilizando para el entrenamiento el algoritmo de YOLO v5, dando por sentado el cumplimiento de los objetivos propuestos inicialmente.

La detección de objetos basado en el uso de redes neuronales, ha tenido un gran avance durante estos últimos años de desarrollo, existiendo otros algoritmos de detección de objetos como YOLO, que seguramente también se obtendrían buenos resultados, aun así esta versión de YOLO, es de las más sencillas de implementar y entrenar al usar PyTorch.

Possiblemente uno de los temas a mejorar en el trabajo de investigación en este momento, sería el modelo OCR debido a la escasez de caracteres en distintas condiciones, poca luz, imágenes borrosas, etc. Para mejorar esto habría que aumentar la cantidad de datos, con imágenes donde se cumplan las condiciones donde el modelo falla y volver a entrenar el modelo.

Además para que este OCR funcione correctamente, es necesario que la matrícula que entra al modelo sea correcta, que esté bien preprocesado y rectificado, mientras se cumplan estas condiciones el sistema OCR podrá interpretar matrículas de distintos tamaños (ciclomotor , moto) y sacar un buen resultado.

Finalmente, la meta final de este trabajo es expandir el conocimiento y facilitar estas herramientas a la comunidad. Por ese motivo este trabajo, así como el software se ha publicado libremente en un repositorio público de GitHub. [57]

## 6. Entorno Socio-Económico

### 6.1. Presupuesto

Este trabajo se dividen los costes en función del personal para el desarrollo del proyecto, las herramientas usadas y recursos externos.

**Recursos humanos:** Dentro de este apartado se incluye el número de horas dedicadas a la investigación y al desarrollo de este proyecto. Al tratarse de un Trabajo de Fin de Grado, compuesto por un alumno, estimaremos el coste a partir de un presupuesto base normal de un ingeniero de la especialidad de electrónica industrial y automática en España. Siendo un salario anual de 33.862€. Partiendo, por tanto, de una jornada laboral completa de 8 horas, con 21 días laborables de trabajo al mes, se estima alrededor de 10 € la hora para un ingeniero junior.

Haciendo operaciones con estos datos en las horas totales trabajadas tanto para el gasto en la formación y en la realización del proyecto queda:

TABLA 6.1.1 - MATRIZ COSTES RECURSOS HUMANOS			
Puesto	Horas de Trabajo	Coste /h	Total
Autor	524	10	5240 €

Dentro de las herramientas usadas para el desarrollo del mismo se encuentran las herramientas hardware y las herramientas software:

- **Herramientas hardware:** Los equipos usados para la programación y la mayoría de pruebas realizadas para el funcionamiento, han sido con un portátil MEDION Erazer P6689 de un coste de unos 719,98€. También se utilizó un servidor de la universidad con un coste promedio de 1500 €

Para el cálculo del coste total se basará en el porcentaje de uso:

$$\% \text{ uso} = \frac{\text{Horas de uso}}{\text{Horas de vida útil}} * 100$$

Esto se multiplica por el precio total de adquisición, así sacamos el coste descontando su amortización. (Horas de uso: tiempo que ha sido usado/ Horas de vida útil: Tiempo total del tiempo que ha durado el proyecto)

## **TABLA 6.1.2 - MATRIZ COSTE HERRAMIENTAS HARDWARE**

Recurso	Precio	Horas de uso	Horas de vida útil	%	Subtotal
Medion ERAZER	719,98 €	525 h	~ 5040 h	10 %	75 €
Servidor	1500 €	300 h	~ 5040 h	5%	89 €
Coste Total					164 €

- **Herramientas software:** La totalidad del proyecto se ha planteado desde la idea del software libre y open source, por lo tanto los gastos en licencias son gratuitas.

### **TABLA 6.1.3 - MATRIZ COSTE HERRAMIENTAS SOFTWARE**

	Software	Versión	Precio
	Windows 10 Pro	21H1/19043.1416	Gratis
	Ubuntu	20.04.4 LTS	Gratis
	Python	3.9	Gratis
 OpenCV	Opencv	4.5.5	Gratis
	Visual Studio Code	1.66	Gratis
	Pytorch	1.11	Gratis
	Labelme	v5.0.1	Gratis
	CUDA toolkit	11.6.1	Gratis
Coste Total:			0 €

**Recursos externos:** Para la creación de un *dataset* era necesario la adquisición de una gran cantidad de imágenes, muchas de esas imágenes se obtuvieron de forma manual y por tanto se incluye en el coste de recursos humanos, aun así para tratar de mejorar el entrenamiento, aumentando la calidad del *dataset* se hizo una compra de unas 27.377 imágenes de placas de matrículas con sus respectivos XML.

TABLA 6.1.4 - MATRIZ COSTE RECURSOS EXTERNOS	
Platesmania [51]	70 €

Teniendo en cuenta estos datos se puede estimar un presupuesto total:

TABLA 6.1.5 - PRESUPUESTO TOTAL	
Recursos humanos	5240 €
Herramientas hardware	164 €
Herramientas software	0 €
Recursos externos	70 €
<b>TOTAL</b>	<b>5700 €</b>

## 6.2. Impacto Socio-Económico

En esta sección se comentará las consecuencias o causas de implementar un sistema de detección de matrículas, debido a que son muchas las perspectivas que uno puede tener sobre este tema, se supondrá la instalación de uno de estos sistemas de detección de matrículas en un entorno interurbano.

### Impacto Ético:

- Puede mejorar la seguridad de zonas de riesgo de accidentes.
- Permitirá el control del tráfico reduciendo atascos.
- Permitirá el seguimiento de coches en busca y captura por parte de la policía sin paralizar el tráfico.
- Un aumento de la seguridad y control de la población puede dar a entender que el individuo está siendo privado de la libertad. Este cambio radical en sus valores puede provocar cambios en las conductas de los individuos, dando como consecuencia la vandalización de estos sistemas, pintando o destruyendo las cámaras que forman estos sistemas ANPR.

- Otra cuestión que puede suponer controversia es la publicación de las matrículas en internet, parece estar mal visto hacer fotos de matrículas y ser vistas por las personas, sin embargo no es algo que sea ilegal, la información es de carácter público guardada por la DGT.
- En cuanto al sector de la conducción autónoma, se puede implementar este sistema para hacer un reconocimiento del entorno en donde se encuentra el vehículo, observando los vehículos que se encuentran alrededor, pudiendo identificarlos con un sistema ANPR y comunicárselo así al vehículo para tomar una decisión más precisa. Controlando así por ejemplo el tráfico en intersecciones.

### **Impacto Económico:**

En cuanto al impacto económico, es complicado determinar sobre sistemas de reconocimiento automática de matrículas. Las últimas investigaciones del mercado, [52] recopiladas en este informe de investigación, están compuestas por un análisis amplio del alcance del mercado con información detallada realizada desde el 2019 al 2029, se divide en la gestión del tráfico, la aplicación de la ley, en el cobro de peajes, etc.

Sin embargo, en el sector automovilístico, la asistencia a la conducción está cogiendo un papel importante, aun así, esta tecnología está reservada para los coches premium, pero el parlamento Europeo cifra en 61.000 millones de euros los beneficios que supondrán la movilidad autónoma para España, planteando numerosas mejoras, por ejemplo, en el campo de la Salud Pública, se logrará viajes en coche más seguros, liberando al conductor de momentos de estrés en circunstancias de tráfico congestionado. Según unos estudios realizados por la administración nacional de seguridad de tráfico en las carreteras de Estados Unidos, los costes sociales llegan a 900.000 millones de euros al año.

Además la continua necesidad de mejorar las tecnologías, para reducir riesgos de accidentes, abre las puertas a nuevos sectores en el mercado.



## Bibliografía

- [1] Anton Satria & Ariff Idris, "A Study of Car Park Control System Using Optical Character Recognition", International Conference on Computer and Electrical Engineering, 2008. [En línea]. Disponible en: URL Acceso: Abril 2022.
- [2] Chirag Patel, Dipti Shah & Atul Patel, "Automatic Number Plate Recognition System (ANPR): Survey", International Journal of Computer Applications, 2013.
- [3] Imran Shafiq Ahmad, Boubakeur, Pejman Habashi, William Anderson & Tarik Elamsy, "Automatic License Plate Recognition: A Comparative Study" IEEE International Symposium on Signal Processing and Information Technology (ISSPIT), 2015.
- [4] Danian Zheng, Yannan Zhao & Jiaxin Wang, "An efficient method of license plate location" Laboratory of Intelligent Technology and Systems, 2005.
- [5] M. Ashourian, N. Daneshmandpour, O. Sharifi Tehrani & P. Moallem, "Real Time Implementation of a License Plate Location Recognition System Based on Adaptive Morphology", IJE TRANSACTIONS B, 2013.
- [6] Zheng D., Zhao Y., "An efficient method of license plate location", 2005, [URL](#).
- [7] Syed Ali Haider & Khurran Khurshid, "An Implementable System for Detection and Recognition of License Plates in Pakistan", International Conference of Innovations in Electrical Engineering and Computational Technologies, 2017.
- [8] Ashwathy Dev, "A novel approach for car license plate detection based on vertical edges", International Conference on Advances in Computing and Communications (ICACC), 2015, [URL](#).
- [9] I. Slimani, A. Zaarane , A. Handoun"Vehicle License Plate Localization and Recognition System for intelligent Transportation Applications", International Conference on Control, Paris; France, 2019, [URL](#)
- [10] BF Wu & Sp Lin, "Extracting characters from real vehicle license plate out-of-doors", IET Comput, 2007.
- [11] N. Bellas, S.M. Chai, D. Linzmeier & M. Dwyer, "FpGA implementation of a license plate recognition SoC using automatically generated streaming accelerators", IEEE International Parallel & Distributed Processing Symposium, Greece, 2006, [URL](#).
- [12] H.H.P. Wu, H.H. Chen, R.J. Wu & D.F. Shen, "License plate extraction in low resolution video", International Conference on Pattern Recognition, Hong Kong, 2006, [URL](#).

- [13] X. Shi, W. Zhao & Y. Shen, “Automatic license plate recognition system based on color image processing”, In International Conference on Computational Science and its Applications, Singapore, 2005, pp.1159-1168.
- [14] Wenjing jia, Huafeng Zhang, Xiangjian He & Massimo Piccardi, “Mean Shift for Accurate License Plate Localization”, IEEE Conference on Intelligent Transportation Systems, Austria, 2005, [URL](#).
- [15] C. Anagnostopoulos, T. Alexandropoulos, S. Boutas, V. Loumos & E. Kayafas, “A template-guided approach to vehicle surveillance and access control” IEEE Conference on Advanced Video and Signal Based Surveillance, Italy, 2005, [URL](#).
- [16] M.S. Pan, J.B. Yan & Z.H. Xiao, “Vehicle license plate character segmentation”, International Journal of Automation and Computing, 2008, [URL](#)
- [17] X. Xu, Z. Wang, Y. Zhang & Y. Liang, “A method of multi-view vehicle license plates location based on rectangle features”, International Conference on Signal Processing, 2006.
- [18] M.S. Pan, J.B. Yan & Z.H. Xiao, “A new method for correcting vehicle license plate tilt”, International Journal of Automation and Computing, 2009.
- [19] J. Cao & L. Li, “Vehicle objects detection of video images based on gray-scale characteristics”, International Workshop on Education Technology and Computer Science, 2009.
- [20] O. Hommos, A. Al-Qahtani, A. Farhat, A. Al-Zawqari, “HD Qatari ANPR system”, International Conference on Industrial Informatics and Computers Systems (CIICS), United Arab Emirates, 2016.
- [21] Yungang Zhang & Changshui Zhang, “A New Algorithm for Character Segmentation of License Plate”, IEEEIV2003 Intelligent Vehicles Symposium, Columbus, 2003.
- [22] G.G. Desai & P.P. Bartakke, “Real-Time Implementation of Indian License Plate Recognition System”, IEEE Punecon, India, 2018.
- [23] M.S. Al-Shemarry & Y. Li, “Developing Learning-Based Preprocessing Methods for Detecting Complicated Vehicle License Plates” [URL](#).
- [24] A.A. Farhat, A. Al-Zawqari, A. Al-Qahtani, & A. Amira, “OCR-based hardware implementation for qatari number plate on the Zynq SoC”, IEEE-GCC Conference and Exhibition, Bahrain, 2017. [URL](#).
- [25] EasyOCR, 2021, URL: <https://github.com/JaideAI/EasyOCR>
- [26] CRAFT, Pytorch official text detection, URL: <https://github.com/clovaai/CRAFT-pytorch>

- [27] P. Shi & C. Zhao, “Review on deep based object detection”, International Conference on Intelligent Computing and Human-Computer Interaction (ICHCI), 2020.
- [28] R. Girshick, J. Donahue, T. Darrell & J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation”, 2014.
- [29] F. Song, L. Wu, G. Zheng, X. He, G. Wu & Y. Zhong, “Multisize plate detection algorithm based on improved mask rcnn”, IEEE International Conference on Smart Internet of Things, 2020.
- [30] Wei Liu, Dragomir Anguelov, Dumitru Erhan, C. Szegedy, S. Reed, Cheng-Yang Fu, “SSD: Single Shot MultiBox Detector”, arXiv:1512.02325v5, 2016, [URL](#).
- [31] J.Redmon, S.Divvala, R. Girshick & A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection”, arXiv:1506.02640v5, 2016, [URL](#)
- [32] Rayson Laroca, Luiz A Zanlorensi, Gabriel R, Eduardo Todt, William Roson Schwartz & David Menotti, “An Efficient and Layout-Independent Automatic License Plate Recognition System Based on the YOLO detector”, arXiv:19009.01754v4, 2021, [URL](#)
- [33] Das Bundesverfassungsgericht, Bverfg.de. 3 de noviembre de 2008.
- [34] “Reconocimiento automático de matrículas” Colegio de Policía 2013.
- [35] Consejo Federal suizo “Convenio de Ginebra” 19 de Septiembre de 1949
- [36] Instrucción 15/V-113: Autorización de pruebas o ensayos de investigación realizados con vehículos de conducción automatizada en vías abiertas al tráfico en general, Madrid, Dirección General de Tráfico, 2015.
- [37] Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales, («BOE» núm. 294 de 06/12/2018).
- [38] Ley Orgánica 4/1997, de 4 de agosto, por la que se regula la utilización de videocámaras por las Fuerzas y Cuerpos de Seguridad en lugares públicos. («BOE» núm. 186, de 5 de agosto de 1997).
- [39] Ley Orgánica 1/1982, de 5 de mayo, de protección civil del derecho al honor, a la intimidad personal y familiar y a la propia imagen. («BOE» núm. 115, de 14 de mayo de 1982).
- [40] Ley Orgánica 5/1992, de 29 de octubre, de regulación del tratamiento automatizado de los datos de carácter personal, («BOE» núm. 262, de 31 de octubre de 1992, páginas 37037 a 37045).

[41] Agencia Española de protección de Datos, *Guía sobre el uso de videocámaras para seguridad y otras finalidades*, [URL](#).

[42] Ley 18/2021, de 20 de diciembre, por la que se modifica el texto refundido de la Ley sobre Tráfico, Circulación de Vehículos a Motor y Seguridad Vial, aprobado por el Real Decreto Legislativo 6/2015, de 30 de octubre, en materia del permiso y licencia de conducción por puntos, («BOE» núm. 304, de 21 de diciembre de 2021, páginas 156147 a 156170)

[43] Python, disponible en: <https://www.python.org/>

[44] Opencv, disponible en: <https://opencv.org/>

[45] Pytorch, “From research to production”, disponible en: <https://pytorch.org/>

[46] Weights and Biases, “The developer-first MLOps platform”, disponible en: <https://wandb.ai/site>

[47] YOLOv5, disponible en: <https://github.com/ultralytics/yolov5>

[48] Nvidia Corporation, “CUDA Toolkit”, disponible en: <https://developer.nvidia.com/cudnn>

[49] Anaconda Inc, “Data science technology for groundbreaking research”, disponible en: <https://www.anaconda.com/>

[50] Jobteb, “Salario Ingeniero electronico industrial y automatización”, disponible en: [URL](#)

[51] Platesmania, “License plates of Spain”, disponible en: <https://platesmania.com/es/>

[52] Markey World ANPR (Automatic Number Plate Recognition) Cameras Market Research Report 2027 (Covering USA, Europe, China, Japan), [URL](#).

[53] Inc Advanced Micro Devices, “AMD Ryzen 7 3700X” disponible en: <https://www.amd.com/en/products/cpu/amd-ryzen-7-3700x>

[54] L. Peterson & B. Davie “Computer Networks: A Systems Approach”, Elsevier, 2012.

[55] COCO “Common Objects in Context” disponible en: <https://cocodataset.org/>

[56] Hanwha Techwin Europe, “XNO-6120R/FNP”, Camara de red IR tipo bullet, disponible en: <https://www.hanwha-security.eu/es/business-security-products/xno-6120rfnp/>

[57] Matthias Gdanietz de Diego, “Detector de Matrículas en entorno interurbano”, TFG, Madrid, 2022, disponible en: [Github](#)